FDSOI Design using Automated Standard-Cell-Grained Body Biasing

Dissertation der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Johannes Maximilian Kühn
aus Künzelsau

Tübingen
2016

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	24.02.2017
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Wolfgang Rosenstiel
2. Berichterstatter:	Prof. Dr. Hideharu Amano

Acknowledgments

First and foremost, I am deeply grateful for all the support and opportunities my advisors Professor Wolfgang Rosenstiel and Professor Hideharu Amano have bestowed upon me. They have been kind teachers in every sense of the word and allowed me to grow academically, but also as a person. I am also deeply thankful for their openness and their great support for this German-Japanese cooperation, which gave me ideal circumstances and opportunities to conduct my research. Closely related to this is also, of course, my sincere gratitude to Keio University, which repeatedly hosted me to conduct my research at Professor Amano's laboratory as well as to attend conferences. My sincere gratitude also extends to the State of Baden-Württemberg which made a scholarship available within the cooperative doctoral program "Entwurf und Architektur Eingebetter Systeme"[1] as well as the German Academic Exchange Service (DAAD) which complemented this scholarship during my stay at Keio University. Furthermore, I would also like to express my gratitude to the DFG Priority Program SPP1500 "Dependable Embedded Systems" as well as the ENIAC Things2DO[2] project within which I conducted parts of my research and from which I received generous support. On the Japanese side, this work was also partially supported by the "Ultra-Low Voltage Device Project" funded and supported by the Ministry of Economy, Trade and Industry (METI) and the New Energy and Industrial Technology Development Organization (NEDO). It was also partially supported by JSPS KAKENHI S Grant Number 25220002. Furthermore, I'd like to extend my sincere thanks to the VLSI Design and Education Center (VDEC) in Tokyo and Cadence, Mentor Graphics as well as Synopsys for supplying their EDA tools. I'd also like to extend my sincere gratitude to EDAUtils for allowing me to use their excellent Verilog parser.

All this, of course, could not be done without the support of my colleagues and friends on both sides of the world. First of all, I would like to thank my early mentors Thomas Schweizer and Sven Eisenhardt for introducing me to academia and for keeping me under their wing while transitioning into this then unknown world. Since they also introduced me to Professor Amano's work, I'm much indebted. Also, I'd like to thank Professor Oliver Bringmann for his advice and his kind support. It was always great fun to travel with you! I would also like to express my sincere gratitude to my dear friend Christoph Gerum for sharing his

[1] Design and Architecture of Embedded Systems
[2] THIN but Great Silicon to Design Objects

deeply respected views and knowledge on computer science. Many problems I encountered where solved thanks to him. Also, thanks for bearing the madness of sharing our 24/7 office ;). Furthermore, I would also like to thank Dustin Peterson with whom I co-authored a paper which served as one of the starting points for this thesis. Without his masterful programming skills, we definitely would not have made the deadline. Moving to the other side of the world, I wish to thank first of all my dear friends and colleagues Akram Ben Ahmed and Hayate Okuhara. Without their skill, herculean efforts and superhuman endurance, we could not have pulled off our test chip tapeout. It was always inspiring and motivating to work with you. Also, I would like to thank Toru Katagiri for developing the MuCCRA4 architecture, which in many respects served as a wonderful testbed for FDSOI technology. Concerning artistic inspiration, I'd also like to express my gratitude to the artist collective HGich.T for their refreshing creative genius. Since I started listening to your music, not a single paper has been rejected. Finally, there are the many colleagues and friends on both sides of the world who cannot all be named. Thus, I would like to humbly request your forgiveness for not listing you all with whom I have received the pleasure to work. I promise to make up for it in person.

Obviously, this thanks would not be complete without mentioning the closest people who had my back and gave me confidence throughout this endeavor. Thank you, Franziska, for always being there and for accompanying me on this path. Finally, I wish to express my sincere thanks to my parents, Marion and Detlef, who supported me from day one, allowing me to develop my curiosity freely.

Abstract

With the introduction of FDSOI processes at competitive technology nodes, body biasing on an unprecedented scale was made possible. Body biasing influences one of the central transistor characteristics, the threshold voltage. By being able to heighten or lower threshold voltage by more than $100mV$, the very physics of transistor switching can be manipulated at run time. Furthermore, as body biasing does not lead to different signal levels, it can be applied much more fine-grained than, e.g., DVFS. With the state of the art mainly focused on combinations of body biasing with DVFS, it has thus ignored granularities unfeasible for DVFS. This thesis fills this gap by proposing body bias domain partitioning techniques and for body bias domain partitionings thereby generated, algorithms that search for body bias assignments. Several different granularities ranging from entire cores to small groups of standard cells were examined using two principal approaches: Designer aided pre-partitioning based determination of body bias domains and a first-time, fully automatized, netlist based approach called domain candidate exploration. Both approaches operate along the lines of activation and timing of standard cell groups. These approaches were evaluated using the example of a Dynamically Reconfigurable Processor (DRP), a highly efficient category of reconfigurable architectures which consists of an array of processing elements and thus offers many opportunities for generalization towards many-core architectures. Finally, the proposed methods were validated by manufacturing a test-chip. Extensive simulation runs as well as the test-chip evaluation showed the validity of the proposed methods and indicated substantial improvements in energy efficiency compared to the state of the art. These improvements were accomplished by the fine-grained partitioning of the DRP design. This method allowed reducing dynamic power through supply voltage levels yielding higher clock frequencies using forward body biasing, while simultaneously reducing static power consumption in unused parts.

Zusammenfassung

Die Einführung von FDSOI Prozessen in gegenwärtigen Prozessgrößen ermöglichte die Nutzung von Substratvorspannung in nie zuvor dagewesenem Umfang. Substratvorspannung beeinflusst unter anderem eine zentrale Eigenschaft von Transistoren, die Schwellspannung. Mittels Substratvorspannung kann diese um mehr als $100mV$ erhöht oder gesenkt werden, was es ermöglicht, die schiere Physik des Schaltvorgangs zu manipulieren. Da weiterhin hiervon der Signalpegel der digitalen Signale unberührt bleibt, kann diese Technik auch in feineren Granularitäten angewendet werden, als z.B. Dynamische Spannungs- und Frequenz Anpassung (Engl. Dynamic Voltage and Frequency Scaling, Abk. *DVFS*). Da jedoch der Stand der Technik Substratvorspannung hauptsächlich in Kombinationen mit DVFS anwendet, werden feinere Granularitäten, welche für DVFS nicht mehr wirtschaftlich realisierbar sind, nicht berücksichtigt. Die vorliegende Arbeit schließt diese Lücke, indem sie Partitionierungsalgorithmen zur Unterteilung eines Entwurfs in Substratvorspannungsdomänen vorschlägt und für diese hierdurch unterteilten Domänen entsprechende Substratvorspannungen berechnet. Hierzu wurden verschiedene Granularitäten berücksichtigt, von ganzen Prozessorkernen bis hin zu kleinen Gruppen von Standardzellen. Diese Entwürfe wurden dann mit zwei verschiedenen Herangehensweisen unterteilt: Chipdesigner unterstützte, vorpartitionierungsbasierte Bestimmung von Substratvorspannungsdomänen, sowie ein erstmals vollautomatisierter, Netzlisten basierter Ansatz, in dieser Arbeit Domänen Kandidaten Exploration genannt. Beide Ansätze funktionieren nach dem Prinzip der Aktivierung, d.h. zu welchem Zeitpunkt welcher Teil des Entwurfs aktiv ist, sowie der Signallaufzeit durch die entsprechenden Entwurfsteile. Diese Ansätze wurden anhand des Beispiels Dynamisch Rekonfigurierbarer Prozessoren (DRP) evaluiert. DRPs stellen eine Klasse hocheffizienter rekonfigurierbarer Architekturen dar, welche hauptsächlich aus einem Feld von Rechenelementen besteht und dadurch auch zahlreiche Möglichkeiten zur Verallgemeinerung hinsichtlich Many-Core Architekturen zulässt. Schließlich wurden die vorgeschlagenen Methoden in einem Testchip validiert. Alle ermittelten Ergebnisse zeigen im Vergleich zum Stand der Technik drastische Verbesserungen der Energieeffizienz, welche durch die feingranulare Unterteilung in Substratvorspannungsdomänen erzielt wurde. Hierdurch konnten durch die Anwendung von Substratvorspannung höhere Taktfrequenzen bei gleicher Versorgungsspannung erzielt werden, während zeitgleich in zeitlich unkritischen oder ungenutzten Entwurfsteilen die statische Leistungsaufnahme minimiert wurde.

Contents

1 Introduction

The steady development of applications and technologies demand ever greater computing performance in unchanging form factors and power budgets. These demands put the semiconductor industry in a difficult position, wanting to answer customers' demands while traversing the thin line of the physically possible. This increase in performance demands is primarily realized by shrinking transistor geometries, a process commonly called technology scaling. Technology scaling, however, also requires to address many physical challenges that become ever more challenging, the smaller transistors become. Classical transistor architectures, that is bulk technology, already reached its physical limit at about $20nm$ as the electric fields controlling transistor functions were overlapping so much that it was no longer possible to manufacture a functioning and economically feasible transistor below this limit.
Thus, two alternatives were proposed and constitute to this date the only industrially used approaches to manufacture below the aforementioned limit. These two principal architectures are FinFET and Fully Depleted Silicon on Insulator technology (FDSOI). Both approaches aim to increase electric control of the transistor channel but differ mainly in the gate structure. FDSOI essentially remains a planar technology, while FinFET technology rotates the channel ninety degrees to form the channel in a vertical fin with the gate along the sides of the fin. While these architectures allow realizing functioning transistors below $20nm$, the continuing scaling itself leads to ever more complex physical challenges that ultimately impact the merit gained through increased scaling capabilities. For example, with decreasing channel length, so-called short channel effects and variability are exacerbated, leading to impaired static power consumption and the need for higher variability margins. To make matters worse, the innovations realized in such semiconductor technologies are often battery powered and thus not only suffer from the factually required peak performance but get increasingly drained by just keeping computational resources available, conducting very lowly demanding computations. Many approaches have been tried and employed to counter these effects, but in the end, these gains are immediately outweighed by further, ever denser integration.
One example for such approaches is big-little computing. Big-little computing is an architectural approach aiming to increase the energy efficiency for mission profiles where tasks with greatly differing performance requirements and temporally separated scheduling dates are present. The actual improvement is realized by fitting two different core types onto one die: Highly performant and power hungry cores and low- to medium performant (ultra) low-power cores. Depending on the tasks' requirements as well as the system's utilization, the most suiting cores are selected while the other cores are put into a sleep mode. While architectural approaches are charming as they usually work with any semiconductor technology, their effect is also limited to the architecture itself.

It does not change the physical process involved when realizing the actual computations in silicon. Power management approaches such as Dynamic Voltage Frequency Scaling (DVFS) aim to do precisely that by altering the operation conditions. By dynamically changing the supply voltage of specific cores, the physical operation conditions of the transistors in which the computation is conducted can be changed. Applied to the above example, this means that during periods where only little performance is required, the supply voltage is lowered, requiring subsequent frequency adjustments as timing degrades. DVFS allows not only to reduce static power consumption, but also dynamic power which is in a quadratic relationship with supply voltage. When high performance is required, DVFS can increase supply voltage and scale clock frequency along to meet the performance requirements. Both above-described approaches are usually also combined with power gating, which is a technique to cut off the supply voltage of defined chip areas physically. This is a very effective approach which however also requires significant extra hardware to manage the transitions from and to a sleep-mode in a well-defined manner and incurs considerable overheads. Nonetheless, all of these approaches do not change the actual physical process involved when transistors switch their state to realize computations.

In contrast to FinFETs, FDSOI, however, does not only allow to continue technology scaling but also to manipulate the transistor characteristics of even tiny clusters of transistors. This method is called body biasing, describing the application of a potential onto the electrically insulated transistor body. Body biasing acts similar to a second gate and thereby either heightens or lowers the threshold. This capability enables chips to alter one of the central transistor characteristics of said clusters at runtime as if the process technology was changed. Thus, when, e.g. only little performance is required, FDSOI can use body bias to increase threshold voltage, cutting leakage currents to below 10% of the original leakage, or, to decrease threshold voltage and therefore increase maximum clock frequency without changing the supply voltage. Body biasing can be realized in much finer granularities than, e.g., DVFS, as only the transistor bodies need to be electrically separated, with no need for costly level shifters and high performant voltage regulators or expensive synchronization hardware for multiple clock domains. This additional flexibility also allows utilizing the supply voltage range much more efficiently. As body bias's influence on timing is relative to the supply voltage, it can give enormous speed boosts at ultra-low supply voltages, making an extremely appealing high energy efficiency case. On the other hand, it can also be used to sharply reduce static power consumption at higher supply voltages without performance degradation.

Body biasing fundamentally offers a new degree of freedom for power management techniques but also confronts chip designers with additional complexity. As this additional degree of freedom can be managed similarly to DVFS, it will be readily integrated into existing power management solutions. This being said, there's nothing free in this world and neither is body biasing in FDSOI. On the contrary, if misused, it can make chip designs less efficient than not using body biasing at all. Beneficially leveraging this feature is a highly complex design step. While the physical implementation is straightforward, deciding what groups of transistors should be biased together, by how much and at

what time, is a very complicated step requiring design automation to allow FDSOI to be leveraged properly.

This thesis illustrates how to tackle these issues by proposing several approaches with increasing levels of automation. Thus, starting with intuitive ways to partition a design into body bias domains, general body bias domain partitioning guidelines are derived, evaluated and discussed. These partitioning methods escalate in granularity, starting with entire cores, moving to the particular case of DRP's regular array of processing elements before lifting granularity limits entirely through fine-grained body biasing. These intuitive methods are however hampered by the need for hand (pre-)partitioning. While experienced designers can identify very good partitioning schemes by hand, it is still considerable work. Furthermore, many optimizations destroy design hierarchies, leaving no human-readable structures. The destruction of design hierarchies, however, is integral to many optimization methods. Forcing the hierarchy to stay intact would prevent these methods from functioning correctly. Thus, after these intuitive approaches, a fully automated standard cell-based approach is proposed as an extension of the intuitive fine-grained approach, allowing body bias domain partitioning to consider partitionings on standard-cell granularity.
The thesis is structured as follows: After this introductory part, chapter 1 closes with a brief motivation. Then, chapter 2 starts with a background on available semiconductor technologies for digital designs with a specific focus on FDSOI technology as well as dynamically reconfigurable processor architectures, in particular, the MuCCRA architectures. This chapter is followed by the state of the art in chapter 3 considering all fields relevant to this thesis. This is, of course, semiconductor technologies, power management techniques and dynamically reconfigurable processors. Before moving on to the partitioning approaches, the actual problems this thesis aims to tackle are introduced in chapter 4. Finally, chapters 5 and 6 propose the evaluated body bias domain partitioning approaches. Chapter 5 considers intuitive, i.e., general body biasing domain partitioning approaches while 6 proposes an automatized method to partition highly optimized netlists into body bias domains. Before moving on to the actual evaluation results, chapter 7 introduces the test chip manufactured to provide the real chip evaluation part in chapter 8, the results chapter. This chapter is split into two parts: simulative results and results based on real chip measurements. Naturally, as simulations allow to freely exchange and modify the evaluated design and the evaluation conditions, real-chip measurements lack this flexibility and thus cannot cover all considered strategies presented in the simulative evaluation part. Finally, this thesis closes in chapter 9 with a brief conclusion as well as an outlook on possible future work.

Motivation

Apart from FinFET technology, FDSOI is the only remaining option to continue technology scaling. The actual miniaturization of transistors, however, created a problem of its own. The sheer physical dimensions no longer allow scaling all aspects accordingly. While single transistor energy efficiency still increases per node, it does not scale at

the same pace as the number of transistors increases [1]. This development leads to a new paradigm called dark silicon which ultimately also put a halt to straightforward multi-core scaling [2]. With FDSOI, this phenomenon can be significantly mitigated as body biasing allows to change the transistor characteristics to suit the power versus performance requirements dynamically at runtime. If e.g., no performance is required, i.e. if background tasks are executed, reverse body biasing can almost eliminate static power consumption. If low- to medium levels of performance is required, ultra-low supply voltages can provide sufficient performance with the option of additional speed boosts via forward body biasing. When combining these two aspects and applying it to entire designs in a fine-granular manner, both aspects can even be leveraged at the same time, leading to a highly efficient design style leveraging FDSOI's capabilities and thereby circumventing many of the limits imposed through the dark silicon phenomenon.

As the physical challenges associated with technology scaling continue to increase in complexity, the challenge also became an economic one. For decades, the cost per transistor decreased with the ability to fit ever more transistors onto the same area with relatively similar defect probability. Below the 28nm node, however, cost per transistor increases again as the sheer effort to manufacture a functioning unit as well as development cost increased so steeply, they cannot be compensated without a considerable price increase [3]. In this regard, again, FDSOI dominates in two respects. Firstly, it is planar technology requiring no complete switch of manufacturing and design methodology. Secondly, as hinted above, by being capable of providing more performance at lower supply voltages, its characteristics also resemble a lot more a further scaled node than in which it is manufactured. For example, 28nm FDSOI using body biasing exhibits similar power and performance levels as 22nm FinFET at the same supply voltage, still leaving much headroom for FDSOI's ultra-low supply voltage operation. Thus, FDSOI also possesses strategies to address many of the economic issues as well.

These capabilities have very practical implications. With smartphones inactive most of the day, battery charges typically last no longer than a single day. This can mainly be attributed to the many background tasks running periodically to allow the device to monitor its status, checking emails and staying informed in order to provide an actual "smart" user experience. Such tasks, however, do not require much performance, but still, with process technologies safe limits of low supply voltage operation, the lowest power operation point is still significantly above the requirements for such tasks. FDSOI, on the other hand, allows putting the smartphone into an ultra-low-power mode, adjusting the transistor characteristics to the actual computational demands. This could potentially give smartphone users days instead of close to a day on a single battery charge. However, FDSOI could not only make background tasks much more efficient. For normal operation, supply voltage could be significantly dropped as well, and thus be made a lot more efficient than in competing semiconductor technologies. These capabilities increase in attractiveness when considering wearable devices, where the battery life plays an essential role in the willingness of users to adopt new devices.

One of the killer applications, however, may be small, lightweight nodes as projected as part of the Internet of Things (IoT). While IoT is still picking up the pace, both its use-case and the potential benefit of FDSOI are self-evident. IoT lives on massive numbers of sensors and actors, where, e.g. smart environments such as sensor-equipped roads. The sheer number of nodes required as well as the difficulty in providing adequate power supplies to such devices call for ultra-low-power devices that still provide enough computational performance to conduct pre-processing on the spot.

In this specific respect, as well as regarding the question for suitable benchmarks, the focus moves onto DRPs. Both, to obtain meaningful evaluation results as well as to stress the suitability of FDSOI for extremely challenging ultra-low-power applications such as IoT nodes, the evaluated architecture should reflect this demanding profile. The DRP concept is per definition among the most energy efficient approaches to (embedded) computing conceivable. As a compromise between FPGA-like single bit configurability and CPUs' word grained operation, combined with an internal reconfiguration mechanism reminiscent of CPU instructions, DRPs operate close to an area and energy efficiency optimum. By restricting additional hardware to a bare minimum, power consumption is close to the physically required minimum. Coupled with a massively parallel computing approach employing up to hundreds of processing elements, computational performance can be scaled to a multitude of applications. This is first of all extremely appealing to ultra-low-power applications such as in the IoT, but also constitutes one of the most challenging benchmarks conceivable for any netlist-level power and energy efficiency optimization method.

All these advantages in mind, technologies rarely succeeded when they where hard to apply to real-life projects, where highly complex and challenging design methodologies, requiring weeks of training and designated engineers lead to a spike in labor costs. For now, the option of just moving to the next process node is still a valid alternative. Thus, FDSOI strongly depends on design automation to leverage its capabilities without causing higher labor costs than moving to the next node would. As semiconductor manufacturers have to put even more efforts into creating even smaller nodes, costs will have to be offloaded to the customer. At this point, it will be a decisive factor to have several options and thus, for the above described, present use-cases and for the nodes further down the road, it will be vital to be able to make the most of each node. For these reasons, this thesis makes a significant contribution by pointing out directions for the automatized utilization of FDSOI's greatly increased body biasing capabilities, allowing to make use of FDSOI's full capabilities.

2 Background

In this chapter, the technologies this thesis is based on are explained in detail, and the essential facts are summed up in each section. The chapter is split into two parts: Architecture and properties of FDSOI and Dynamically Reconfigurable Processors.

2.1 Architecture and properties of FDSOI

The principal structure of SOI transistors, or, in this thesis more specifically FDSOI transistors, are much closer to planar technologies as, e.g., FinFETs. Despite this, the SOI construction gives rise to particular effects that shall be elucidated coherently, moving from architecture to device physics, covering specific SOI effects and such that are shared with conventional transistors but behave differently due to the change in architecture. As a uniquely required step, body bias domain construction, as well as well construction and insulation details, are touched upon.

2.1.1 Architecture

As a typical textbook definition, Weste and Harris define SOI as a transistor design where source, drain and body regions are insulated from the rest of the silicon bulk using some electrically insulating material [4, p. 360].

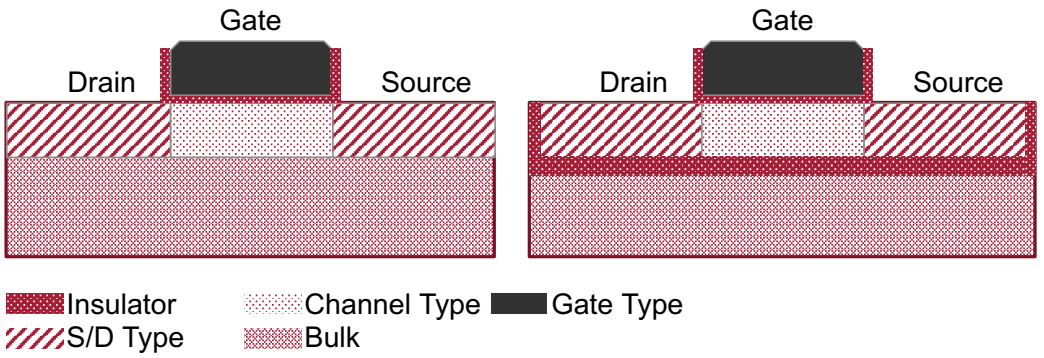


Figure 2.1: Principal bulk (left) versus SOI transistor architecture (right)

Fig. 2.1 illustrates this structure by comparing a simplified bulk against a SOI transistor. Both transistors insulate the gate structure from the channel to realize basic transistor functionality. In SOI transistors, however (Fig. 2.1, right), an additional layer of insulation prevents electrical interaction of the transistor structure with the bulk silicon below. While

this structure efficiently prevents interactions with bulk silicon, it does not prevent interaction with charge carriers within the channel. This difference leads to a differentiation between Partially Depleted (PDSOI) and Fully Depleted SOI (FDSOI) technologies.

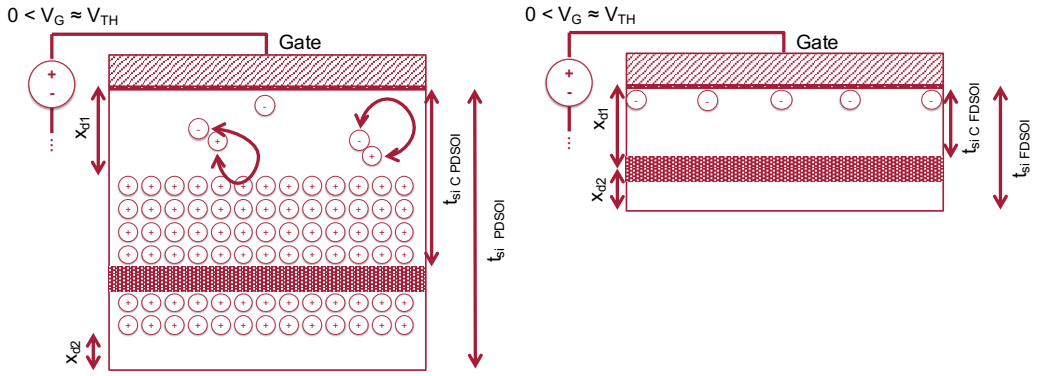


Figure 2.2: NMOS PDSOI with charge interaction (left) versus NMOS FDSOI (right), adapted from [4, 5]

If the sum of the depletion areas' depths x_{d1} and x_{d2} are less than the thickness of the silicon film in which the entire transistor is realized (e.g. $t_{siPDSOI}$ in case of PDSOI and $t_{siFDSOI}$ in case of FDSOI), the transistor is partially depleted, i.e. in this case PDSOI. If $x_{d1} + x_{d2} < t_{si}$, then parts of the transistor structure contains free charge carriers even at V_G close to $0V$ (Fig. 2.2 left) [5]. If on the other hand x_{d1} and x_{d2} are at least equal to the thickness of the entire silicon film in which the transistor is realized, no free charge carriers beyond those of the thermal voltage can be present at V_G close to $0V$. This means if $x_{d1} + x_{d2} \geq t_{si}$, then the transistor is realized in fully depleted technology, that is FDSOI (Fig. 2.2 right).

In PDSOI, charge can remain in the channel area and thus, body voltage and subsequently threshold voltage V_{TH} becomes dependent on the switching history [4, p. 362]. Thus, this effect is called history effect. This effect significantly adds to other sources of variation and leads to a mismatch of otherwise identical transistors [4, p. 362]. Furthermore, [4] cite the (partially) floating body as one mechanism for a parasitic bipolar transistor forming inside the PDSOI channel, which leads to additional unwanted current flow, so-called pass-gate leakage.

In FDSOI, however, the channel is fully depleted, i.e., there are virtually no charge carriers when the device is off, and no charge can remain in the channel between switching. In combination with the electrical insulation, this leads to full electrostatic control over the transistor channel. This furthers a major benefit of SOI and FDSOI technology in particular, which is potentially very low V_{TH}. The possibility to use fully undoped silicon as channel material, together with the SOI construction efficiently minimizes V_{TH} variability and thus requires significantly less guard-banding [4, 6, 7].

2.1.2 Device Physics

FDSOI technology exhibits peculiarities in regard to device physics. Most are shared with bulk transistors, although they differ significantly in extent while others are different altogether. Some of these effects are attributed to or exacerbated by the SOI construction and thus will be covered in the subsection on SOI Effects. The body effect is shared by all, bulk, PDSOI and FDSOI alike, although the range of voltages that can be applied as well as the incurred effect differ considerably. As it is the principal effect this thesis is exploiting, it is treated separately. Beforehand, principal (FDSOI) transistor physics will be introduced.

As hinted before, FDSOI transistors have two gate structures that control the transistor operation. The front gate is assumed to exert switching control, while the back gate can be used to manipulate transistor characteristics. Thus, there are also two surface potentials ϕ_{s1} (front gate) and ϕ_{s2} (back gate), one for each gate, given as

$$\phi_{s1} = (1 + \frac{C_{si}}{C_{ox1}}) = V_{G1} - \phi_{ms1} + \phi_{s2}\frac{C_{si}}{C_{ox1}} + \frac{Q'_d}{2C'_{ox1}} \tag{2.1}$$

$$\phi_{s2} = (1 + \frac{C_{si}}{C_{ox2}}) = V_{G2} - \phi_{ms2} + \phi_{s1}\frac{C_{si}}{C_{ox2}} + \frac{Q'_d}{2C'_{ox2}} \tag{2.2}$$

with ϕ_{ms1} and ϕ_{ms2} as the metal-semiconductor work function, Q_{ox1} and Q_{ox2} the gate oxide charges, C_{ox1} and C_{ox2} the gate capacitances as well as C'_{ox1} and C'_{ox2} the gate oxide capacitances per unit area, all for the front and back gate respectively. Furthermore, C_{si} denotes the depletion capacitances and Q'_d the depletion charge per unit area of fully depleted *Si* [8].

Equations 2.1 and 2.2 can thus be combined to get independent equations for ϕ_{s1} and ϕ_{s2} as shown in [5] as

$$\phi_{s1} = \left[V_{G1} - \phi_{ms1} + \frac{Q'_d}{2C'_{ox1}} + \frac{C_{si}C_{ox2}}{C_{ox1}(C_{ox2}+C_{si})}\left(V_{G2} - \phi_{ms2} + \frac{Q'_d}{2C'_{ox2}}\right)\right] \cdot \left[1 + \frac{C_{si}C_{ox2}}{C_{ox1}(C_{si}+C_{ox2})}\right]^{-1} \tag{2.3}$$

$$\phi_{s2} = \left[V_{G2} - \phi_{ms2} + \frac{Q'_d}{2C'_{ox2}} + \frac{C_{si}C_{ox1}}{C_{ox2}(C_{ox1}+C_{si})}\left(V_{G1} - \phi_{ms1} + \frac{Q'_d}{2C'_{ox1}}\right)\right] \cdot \left[1 + \frac{C_{si}C_{ox1}}{C_{ox2}(C_{si}+C_{ox1})}\right]^{-1} \tag{2.4}$$

. These two surface potentials control transistor operation, similar to transistors with only one gate and thus, assuming body biasing is omitted, also only one surface potential ϕ_{s1}.

Transistors have several different regions of operation as depicted in Eq. 2.5 [8].

$$\phi_{s1} = \begin{cases} 0 \cdots \phi_F & \text{depletion} \\ \phi_F \cdots 2\phi_F & \text{weak inversion} \\ 2\phi_F \cdots (2\phi_F + \Delta\phi) & \text{moderate inversion} \\ > (2\phi_F + \Delta\phi) & \text{strong inversion} \end{cases} \tag{2.5}$$

In the depletion region, which ranges from a surface potential at the front gate ϕ_{s1} of $0V$ to about the fermi potential, the transistor channel is depleted, i.e., almost no free charge carriers are present. Minority carriers are repelled, but there is not enough potential to free majority carriers to be present. Once ϕ_{s1} reaches levels above the fermi potential up to two times the fermi level, the potential is strong enough to free majority carriers, that is the channel is weakly enriched with the opposite kind of charge carriers than in the off-state, that is minority carriers. Hence this state is called weak inversion. This effect is strengthened when ϕ_{s1} is within two times fermi level and the latter plus the potential required to reach strong inversion. Strong inversion, i.e., ϕ_{s1} greater than the strong inversion threshold of $2\phi_F + \Delta\phi$ leads to the formation of an actual majority carrier channel.
In FDSOI transistors, however, there is not only one gate, but another one on the back-side of the transistor with surface potential ϕ_{s2}
In contrast to PDSOI transistors, both gates' voltages V_{G1} and V_{G2} are fully intertwined, as the channel is fully depleted. According to [5] and [9], gate voltages are given analogous to Eq. 2.1 through 2.4 as

$$V_{G1} = \phi_{ms1} - \frac{Q_{ox1}}{C_{ox1}} + \phi_{s1}\left(1 + \frac{C_{si}}{C_{ox1}}\right) - \phi_{s2}\frac{C_{si}}{C_{ox1}} - \frac{Q'_d}{2C'_{ox1}} \tag{2.6}$$

$$V_{G2} = \phi_{ms2} - \frac{Q_{ox2}}{C_{ox2}} + \phi_{s2}\left(1 + \frac{C_{si}}{C_{ox1}}\right) - \phi_{s1}\frac{C_{si}}{C_{ox2}} - \frac{Q'_d}{2C'_{ox2}} \tag{2.7}$$

[5, 9]. Therefore, the threshold voltage is also dependent on both gates. Using the definition of Eq. 2.5, if the front gate, the primary gate structure, reached $2\phi_F$ and the back gate is accumulated, threshold voltage V_{TH} at zero bias $V_{TH,zero}$ is defined according to [8] as

$$V_{TH,zero} = \phi_{ms1} + 2\phi_F\left(1 + \frac{C_{si}}{C_{ox1}}\right) + \frac{qN_{CH}t_{si}}{2C'_{ox1}} \tag{2.8}$$

and similarly

$$V_{G2,acc} = \phi_{ms2} - 2\phi_F\frac{C_{si}}{C_{ox2}} + \frac{qN_{CH}t_{si}}{C'_{ox2}} \tag{2.9}$$

with ϕ_F the fermi potential, electron charge q, dopant concentration in the channel N_{CH} and Si film thickness t_{si} and combined $qN_{CH}t_{si}$ as depletion charge. FDSOI and its two gates architecture thus enables an immediate effect on transistor characteristics. The major aspect this thesis focuses on is elaborated in *Body Effect and Body Biasing*. While body biasing is a very powerful and useful feature, the FDSOI architecture as well as leveraging body biasing may also cause unwanted side-effects as illustrated below.

2.1.2.1 I_{ON} Drive Current

An important measure for transistors is the size of the current the transistor is capable of driving. In bulk transistors, the current flowing through the drain electrode is given by

$$I_D = \mu_{eff} C_{ox} \frac{W}{L} \left[(V_G - V_{THS} V_{DS} - \frac{1}{2} n V_{DS}^2) \right] \tag{2.10}$$

and in saturation

$$I_{Dsat} = \frac{1}{2n} \mu_{eff} C_{ox} \frac{W}{L} (V_G - V_{THS})^2 \tag{2.11}$$

with V_{THS} as the body factor accounted threshold voltage defined as

$$V_{THS} = V_{TH0} + n V_S \tag{2.12}$$

where n is the body factor which is multiplied by the substrate voltage V_S [8]. In this respect, bulk and SOI transistors behave very similarly as this figure is mainly influenced by transistor geometry, potential strength and most importantly the threshold voltage which is modulated through the above introduced body biasing. Colinge further simplifies these equations for the depleted back interface case to

$$I_{Dsat} = \frac{1}{2n} \frac{W}{L} \mu_n C_{ox1} [V_{G1} - V_{TH}]^2 \tag{2.13}$$

where the body factor n in the back-gate accumulation case is

$$n_{acc} = 1 + \alpha = 1 + \frac{C_{si}}{C_{ox1}} \tag{2.14}$$

and in the fully depleted case

$$n_{fd} = 1 + \alpha = 1 + \frac{C_{si} C_{ox2}}{C_{ox1} (C_{si} + C_{ox2})} \tag{2.15}$$

[8]. Without restricting generality, this equation only considers the front gate V_{G1} while the back-gate or body bias influence can still be included through a modulation of the threshold voltage. Conclusively, equation 2.13 describes the relationship between threshold voltage V_{TH} and the maximum current the transistor is capable of driving, its saturation drain current I_{Dsat}. With decreasing V_{TH}, I_{Dsat} becomes larger as the $[V_{G1} - V_{TH}]$ term is getting closer to V_{G1}. Thus, if threshold voltage can be lowered, e.g. using forward body biasing, the transistor is capable of driving larger currents and inversely, if reverse body bias is applied, leading to an increased threshold voltage, the capability of driving currents decreases as well.

2.1.2.2 I_{OFF} and Leakage Effects

Closely related to the threshold voltage is sub-threshold leakage. It is the primary source of leakage in FDSOI and all other transistor types. This being said, apart from gate-tunneling

leakage, Gate Induced Drain Leakage (GIDL), as well as p-n junction leakage are suppressed in FDSOI [10]. Weste and Harris describe in [4] the body effect dependent sub-threshold leakage as

$$I_{DS} = I_{OFF} \cdot 10^{\frac{V_{GS}+\eta(V_{DS}-V_{DD})-k_\gamma V_{G2}}{S}} \left(1 - e^{\frac{-V_{DS}}{v_T}}\right), \tag{2.16}$$

with I_{OFF} as the sub-threshold current at $V_{GS} = 0V$ and $V_{DS} = V_{DD}$, the η term accounting for DIBL, k_γ a body effect coefficient, $V_{G2} = V_{SB}$, the back gate voltage which Weste and Harris refer to as V_{SB}, the source to bulk voltage and S the sub-threshold slope. Furthermore, gate-tunneling leakage is given as in an estimated form

$$I_{GATE} = WA \left(\frac{V_{DD}}{t_{ox}}\right)^2 e^{-B\frac{t_{ox}}{V_{DD}}} \tag{2.17}$$

where W is the gate width, as well as A and B are technology constants [4].

As Eq. 2.16 indicates the voltage of the back gate has an exponential effect on sub-threshold leakage. While this can be leveraged in some cases to reduce leakage, it also may have the opposite effect, exponentially increasing sub-threshold leakage. With sub-threshold already being one of the major problems in deep-submicron technologies, it is one of the limiting factors regarding body biasing.

Gate leakage on the other hand behaves virtually just as it does in bulk technologies as reflected in Eq. 2.17. Due to relatively large BOX thicknesses in the technologies covered in this thesis ($t_{ox} = t_{BOX} \geq 10nm$), tunneling through the back gate can be neglected for now.

2.1.2.3 SOI Construction Related Effects

While SOI architectures improve most aspects, the addition of insulation layers also leads to increased thermal insulation and thus exacerbates self-heating effects (SHE) [11].

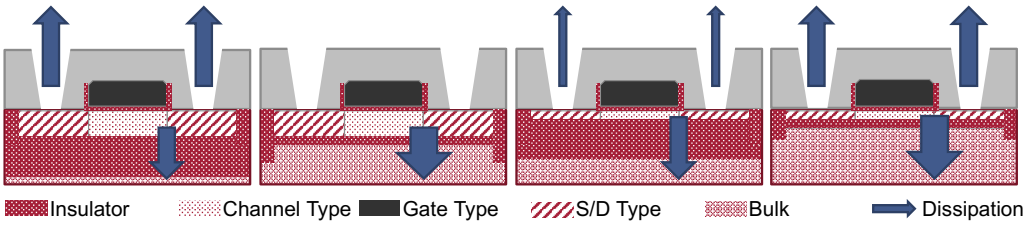


Figure 2.3: Schematic paths of heat dissipation in thick BOX (1), thin BOX (2), thin SOI (3) and UTBB SOI (4) MOSFETs, adapted from [11]

An increase in device temperature T_{DEV} directly translates to an increase in thermal resistance R_{TH} [11]. Takahashi et al. [11] showed that SHE is dependent on chip temperature and BOX and SOI thickness. Fig. 2.3 shows four different SOI architectures (1)-(4) and

their heat dissipation paths. Variant (1) with a thick BOX and not-thinned SOI dissipates generated heat through source and drain terminals and the substrate, (2) mainly through the substrate, (3) slightly via drain and source terminals while (4) can dissipate heat both strongly via the source and drain terminals as well as the substrate. According to [11], the ultra-thin body and BOX (UTBB) design is most efficient in mitigating SHE and therefore propose UTBB as improved SOI architecture.
While the added insulation does exacerbate SHE and thereby worsens variability, the SOI construction still severely mitigates variability. However, SHE needs to be reevaluated at smaller nodes. Ultimately, it is not a general SOI drawback but a side-effect of technology scaling which similarly affects bulk and (SOI) FinFET technologies [12].

2.1.2.4 Body Effect and Body Biasing

With the long history of body biasing, most descriptions of the body effect apply to bulk transistors. Kuroda and Sakurai give a detailed overview of mostly pre-SOI technologies exploiting body biasing in [13]. Similar to [4], Kuroda and Sakurai describe the body effect as

$$V_{TH} = V_{TH0} + \gamma \left\lfloor \sqrt{2\phi_b - V_{BS}} - \sqrt{2\phi_b} \right\rfloor, \tag{2.18}$$
$$\gamma = \frac{t_{ox}}{\varepsilon_{ox}} \sqrt{2\varepsilon_{si} q N_A},\ \phi_b = \frac{kT}{q} \ln\left(\frac{N_A}{N_i}\right),$$

with V_{BS} as substrate potential, V_{TH0} being the threshold voltage at zero bias $V_{BS} = 0V$, ϕ_b the surface potential at threshold, γ the body effect coefficient, ε_{ox} and ε_{si} the dielectrical constats for gate-oxide and silicon respectively, q the electronic charge, k the Boltzmann constant, temperature T and intrinsic doping N_i as well as N_A the channel doping. This equation, however, does not depict reality in FDSOI technologies. If a SOI transistor is manufactured in a triple well process, the transistor body is fully insulated from the transistor channel as well as the substrate. Thus, much larger potentials can be applied on the transistor body without causing any unwanted current flow.
Furthermore, in FDSOI, there are two actual gates, and due to the fully depleted construction, as noted above, their effects cannot be separated [9]. This means that any change in potential on one of the gates directly changes the potential required on the other gate to reach a certain channel state. Thus, the threshold voltage for the actual front gate is also highly dependent on the back gate. Following Eqs. 2.6 and 2.7, [5] summarizes the threshold voltage $V_{TH,FD}$ for fully depleted devices in dependence on the back gate, i.e. body bias, as

$$V_{TH,bias} = V_{TH0} - (n_{FD} - 1)(V_{G2} - V_{G2acc}), \tag{2.19}$$

whereas the body factor for fully depleted devices n_{FD} is given as

$$n_{FD} = 1 + \frac{C_{si} C_{ox2}}{C_{ox1}(C_{si} + C_{ox2})} \tag{2.20}$$

with the unaltered threshold voltage $V_{TH0} = V_{TH,zero}$ (see Eq.) the zero bias threshold volt-

age, body factor n_{FD}, the back gate voltage V_{G2} and the back gate voltage in accumulated state V_{G2acc}. This can be further broken down to

$$\Delta V_{TH,bias} = -(n_{FD}-1)\Delta V_{G2},$$

. This also corresponds to the experimentally validated observations of [14] that simplifies threshold voltage back gate dependency as a linear relationship equation

$$V_{TH,bias} = V_{TH0} - K_\gamma V_{G2}, \tag{2.21}$$

with an experimentally obtained approximation factor K_γ which should correspond to $(n_{FD}-1)$ of Eq. 2.19. The effect described in Eqs. 2.18 to 2.21 is also called body bias, i.e. the application of a potential onto the transistor body to alter the V_{TH} of, i.e. bias, a transistor.
The actual body bias has to be computed for both PMOS and NMOS separately, and in case on non-flip well architectures, it generally follows the scheme

$$\begin{aligned} V_{DD} &= V_{BP} + V_{BN} \\ V_{BP} &= V_{BN} - V_{DD} \end{aligned} \tag{2.22}$$

where V_{BN} denotes the body bias voltage for NMOS, where V_{BP} means the body bias voltage for PMOS. What is typically being referred to as body bias V_{BB} is the NMOS body bias voltage V_{BN}. So e.g. if $V_{BB} = x$ then $V_{BN} = x$ and $V_{BP} = x - V_{DD}$. If $V_{BB} = 0V$, it is called zero-bias. If $V_{BB} < 0V$, it is called Reverse Body Bias (RBB), whereas $V_{BB} > 0V$ is called forward body bias.
Combining Eq. 2.19 or Eq. 2.21 with 2.16, it can be observed that through body bias, i.e. the body effect via its influence on V_{TH} has an exponential relationship with leakage. Fig. 2.4 visualizes this relationship.
Fig. 2.4 depicts the relationship between body bias given as V_{BB}, denoting the NMOS body bias voltage, as well as leakage for three different FDSOI variants. The first two graphs depict STMicro's 28nm UTBB-FDSOI RVT (LR) and LVT (LL) flavors. Due to their difference in back-plane doping, RVT covers V_{BB} ranges from $-1.3V$ to $0.3V$ [1] and $-0.3V$ to $1.3V$[1] for LVT. Both LVT and RVT show an exponential response to V_{BB}. SOTB also displays exponential behavior but is more sensitive to changes in V_{BB}. Thus, leakage in- and decreases more steeply. For all variants, FBB exponentially increases leakage, while RBB exponentially decreases leakage.
In contrast, delays behave inversely. Weste and Harris derive from Sakurai and Newton's Alpha-Power law [15] the following delay scaling factor τ which is described as

$$\tau = k\frac{CV_{DD}}{(V_{DD}-V_{TH})^\alpha} \tag{2.23}$$

with k as a technology parameter, C as the driven capacitance and the Alpha-Power law parameter α [4]. Consequently, the relationship between V_{BB} and delay is given by com-

[1] Theoretically, the limit is $V_{BB,lim} = 300mV + V_{DD}/2$ for RVT and $-V_{BB,lim}$ for LVT [6]

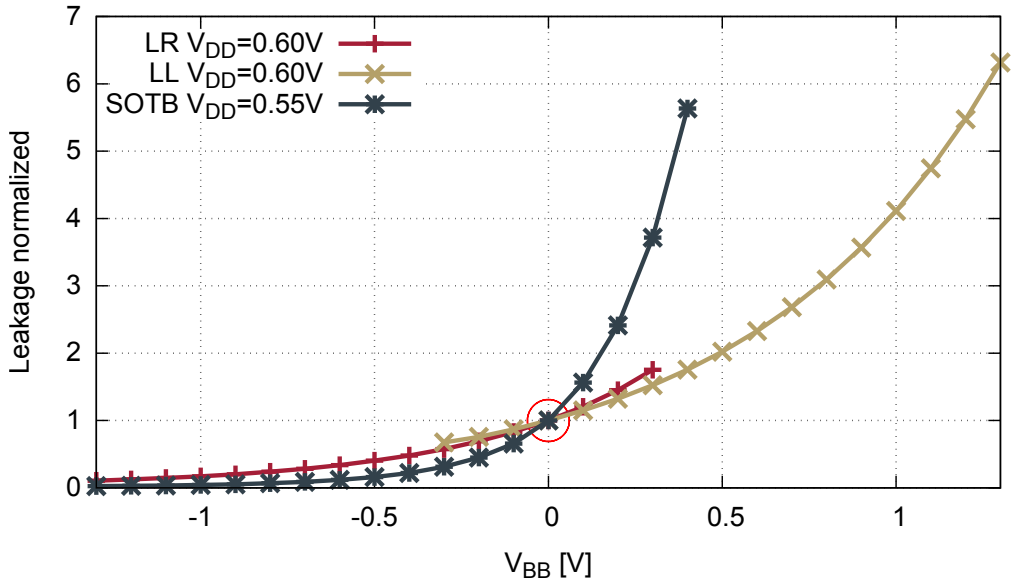


Figure 2.4: Normalized leakage by body bias for STMicro's 28nm FDSOI LVT (LL) and RVT (LR) flavor at $V_{DD} = 0.60V$ as well as an in-silicon measurement of Renesas' 65nm SOTB UM flavor at $V_{DD} = 0.55V$

bining Eq. 2.19 or Eq. 2.21 with Eq. 2.23. Fig. 2.5 visualizes this relationship of V_{BB} with switching delay for the same three FDSOI variants as above. While LVT and RVT display almost linear behavior, SOTB again exhibits higher sensitivity to V_{BB} as in Fig. 2.4.
Body bias' influence on delay is furthered by its impact on drive current. Thus, if FBB is applied and V_{TH} is subsequently lowered, the drain-source current I_{DS} should increase as well. Weste and Harris describe this relationship as

$$I_{DS} = k(V_{GS} - V_{TH}^{*})) \tag{2.24}$$

where k is a technology-specific factor and V_{TH}^{*} denotes the x-intercept of a linear-fit of a I_{DS} by $V_{DS} = V_{GS}$ plot [4].

Considering both Figs. 2.4 and 2.5, body bias offers a trade-off between leakage and delay. I.e. a decrease in delay can be achieved by allowing a leakage increase and inversely, a leakage decrease can be realized by allowing delays to increase.

2.1.3 Body Bias Domain construction

To supply different body biases to separate groups of standard cells, they have to be grouped into separate body bias domains. These groups' wells have to be electrically insulated from each other. To facilitate the electrical separation and to supply the respective body bias voltages to the appropriate well, tap-cells are inserted into each body bias domain at regular intervals (Fig. 2.6). In this case, tap-cells function entirely as well-connector,

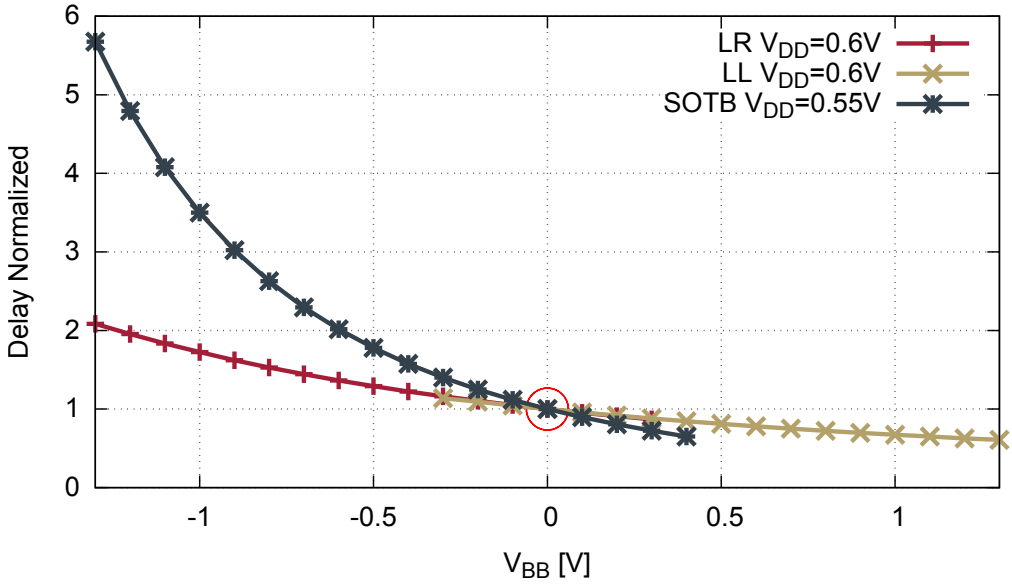


Figure 2.5: Normalized delay by body bias for STMicro's 28nm FDSOI LVT (LL) and RVT (LR) flavor at $V_{DD} = 0.60V$ as well as an in-silicon measurement of Renesas' 65nm SOTB UM flavor at $V_{DD} = 0.55V$

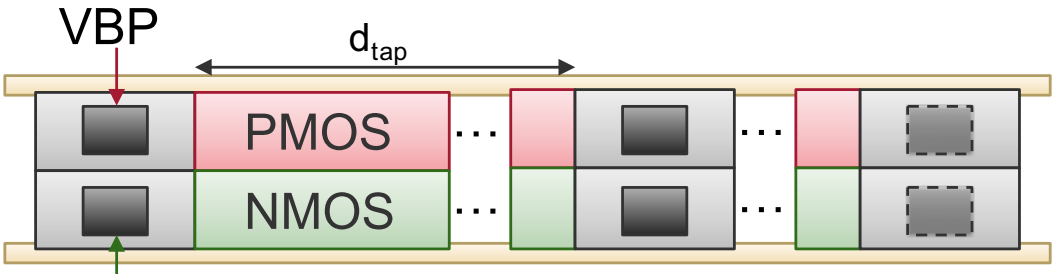


Figure 2.6: Top-view of a standard cell row with beginning tap-cell, at regular interval d_{tap} and with terminating tap-cell, adapted from [16]

connecting a supplied potential to the wells of a standard cell row. As visualized in Fig. 2.7, the potential is fed from the supply terminal to the standard cell row's wells. Such

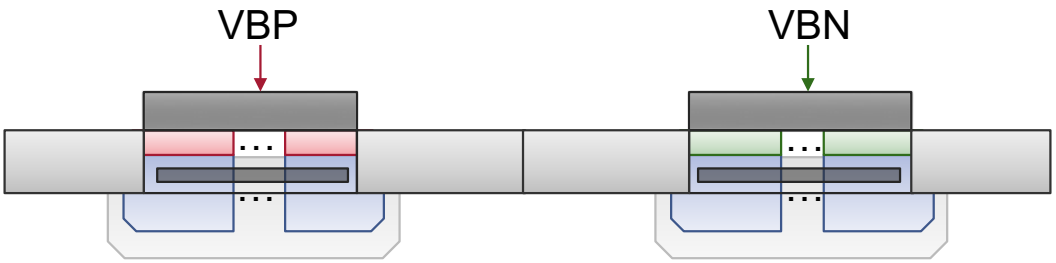


Figure 2.7: Frontal cross-cut of a standard cell row with beginning tap-cell, adapted from [16]

potentials are typically supplied using straps connected to a body bias supply mesh (Fig. 2.8). Most technologies also offer tap-cells to short V_{DD} and V_{SS}[2] to PMOS and NMOS respectively, in case body bias is not needed.

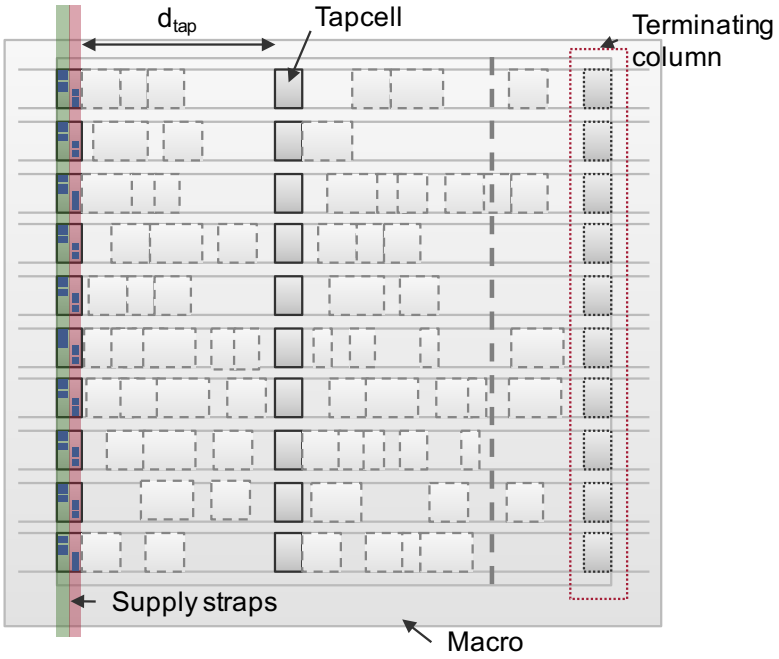


Figure 2.8: Exemplary body bias domain construction with a standard cell-based macro and body bias straps (green and red)

Furthermore, there are also differences concerning the delimitation of body bias domains. Within a standard cell row, a body bias domain can be ended by using terminating tap-cells, interrupting the well supply. Commonly, domains are delimited by placing them into a macro each, which are then physically separated by a separation margin.

[2]This depends on the employed FDSOI technology. There are also flavors that both require V_{SS} on both wells for zero-bias.

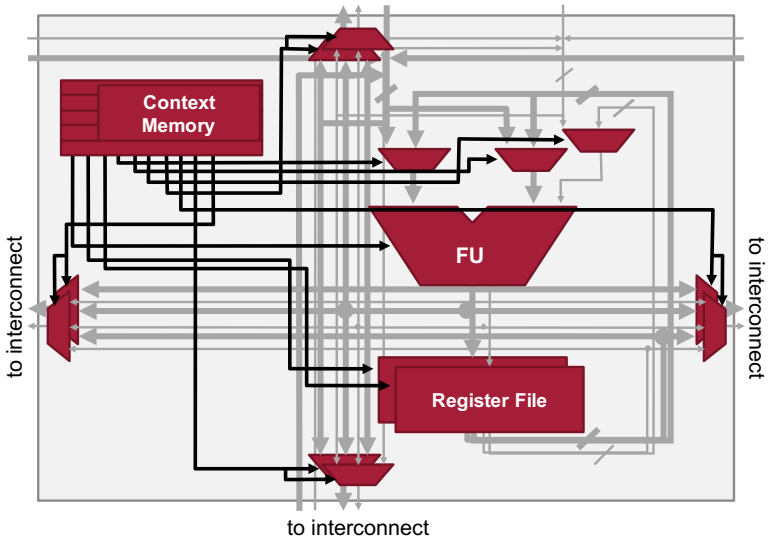


Figure 2.9: General structure of a Processing Element

2.2 Dynamically Reconfigurable Processors

Dynamically Reconfigurable Processors (DRP) are a subset of Coarse-Grained Reconfigurable Architectures (CGRA). CGRA are an architectural concept that consists of an array of interconnected Processing Elements (PEs) and offer certain reconfigurability. PEs operate on a coarse, word-grained granularity while each PE is assigned at least one configuration, called context in the CGRA domain. DRPs are often described as a step between Field Programmable Gate Arrays (FPGA) and general purpose CPUs (GPCPUs). DRPs combine concepts of both worlds: Actual architectural reconfigurability and cycle-by-cycle instruction-like reconfigurability. While the typical FPGA fabric realizes bit-grained reconfigurability with limited dynamic reconfiguration, DRPs realize cycle-by-cycle dynamic reconfigurability efficiently by operating on word granularity. Apart from special features each DRP architecture may implement, they largely differ in the number of contexts, i.e. "configuration slots" they can store on-chip, as well as the word size, i.e. granularity on which computations are conducted. In the following, the basic architectural traits of DRPs will be explained.

Like CGRAs, DRPs are comprised of an array of PEs. These PEs typically consist of routing elements (multiplexers or actual routing elements), some data manipulation units and registers. They are connected to each other using an interconnection network [17]. Such PEs consist of several components as visualized in Fig. 2.9. This representation is a reduction to the most general DRP traits, and thus, does not aim to capture all possible structures, but represents the most common case such as NEC's DRP [18], or, in the case of Oppold et al., the basic instance of the CRC scheme [19]. All the components are geared towards the architecture's word granularity, which refers to the number of bits in

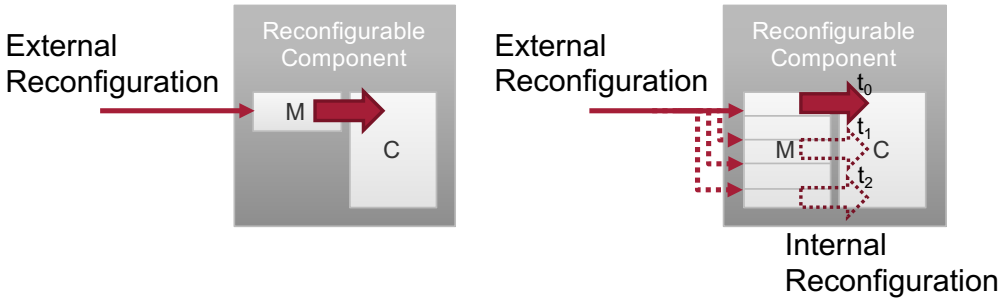


Figure 2.10: External (left) versus internal reconfiguration (right), visualized using an abstract representation of a reconfigurable component

one word of data. This being said, many DRPs also often accommodate operations on sub-word granularity, such as a half word or a single bit. Commonly, each PE contains at least one Functional Unit (FU), such as an ALU or even an FPU to manipulate data. Furthermore, a register file is usually included to store additional operands or intermediate results. Routing of data is facilitated by switching elements or is set statically. To select from those data sources, multiplexers are used. Depending on the PE design, additional registers may be introduced to create a pipelined structure. Furthermore, depending on the interconnect, there may be another multiplexing stage to select the output which is sent over the interconnect. All these components are controlled via the context memory contents, the actual configuration memory, on which the context pointer is set. Starting with the FU function, the internal data selection, register file read and write as well as the output destination and interconnect usage.

Deriving DRPs from CGRAs, the central defining architectural feature is so-called dynamic reconfiguration. To further distinguish (re)configuration concepts, two concepts of configuration delivery are introduced. Firstly, there is external reconfiguration which describes the most common reconfiguration process (Fig. 2.10 left). External reconfiguration means that the configuration memory (represented as M in Fig. 2.10) of a reconfigurable component is rewritten using external configuration data. As a popular example, FPGAs are externally reconfigured when writing a configuration bitstream via some interface (e.g., JTAG) to the configuration memory. The thereby stored configuration data then controls the actual functional component (represented as C in Fig. 2.10) using this memory.

Secondly, there is internal or dynamic reconfiguration (Fig. 2.10 right). Here, multiple configuration data are stored in the reconfigurable component using external reconfiguration. This allows faster and externally independent configuration delivery. Thus, without external intervention, configurations can be switched within a clock cycle or with a significantly smaller delay than e.g. in FPGAs. Following the example in Fig. 2.10, C is reconfigured at times t_0, t_1 and t_2 using the configuration data which has been stored previously using external reconfiguration.

The weaker definition of CGRAs also encompasses architectures that have configuration memory to hold precisely only one context, i.e., where one context is the entire configuration. This requires external reconfiguration every time the implemented function shall be

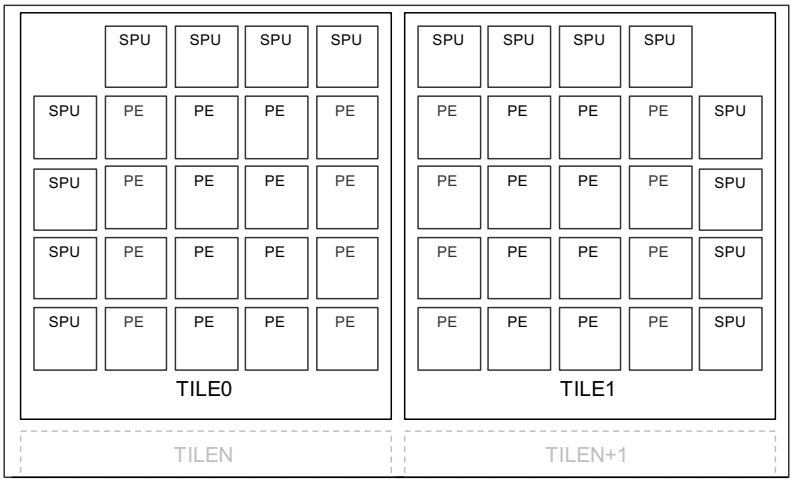


Figure 2.11: Hierarchical grouping of PEs and Special Purpose Units (SPU)

changed. With dynamic reconfiguration, the need for external reconfiguration can be limited and allows to implement more complex functionality on less area. This significantly increases area efficiency [17].
To accommodate as many PEs as possible, PEs usually are more light-weight than fully-fledged processor cores. To further this effort, certain functionality is offloaded to shared Special Purpose Units (SPU), which generally operate similarly to PEs on a context configuration basis and are mapped and configured similarly. For efficient SPU utilization, PEs are often grouped using another logical hierarchy level called tile as depicted in Fig. 2.11. Thus, all PEs in one tile share its SPUs. It is a broadly utilized concept and most often covers memory accesses, special computation functions, additional control logic, etc. Thus, while referring to this concept as SPU in this taxonomy, the term SPU is not restricted to a single type of special PE.

SPUs help to keep area requirements down by sharing functionality which is either temporally not used with such intensity, that each PE would need to be equipped with the unit's functionality, or spatially, e.g., to connect to memory outside the PE array to alleviate routing congestion and the interconnect topology complexity. The structure illustrated in Fig. 2.11 groups 16 PEs with 8 SPUs together to form one tile. As the SPUs are located on the outskirts of the tile, a prime example would be memory interface SPUs relaying and handling memory requests from the tile's PE array. The suitability of this arrangement for special, non-external memory dependent computation units largely depends on the interconnection network. However, previous implementations such as NEC's DRP used exactly such arrangement.

While the above structures describe the compute core, a DRP requires several additional components to operate. First of all, the DRP requires memory which can serve as input and output buffer, as well as intermediate storage. The above described SPUs access this

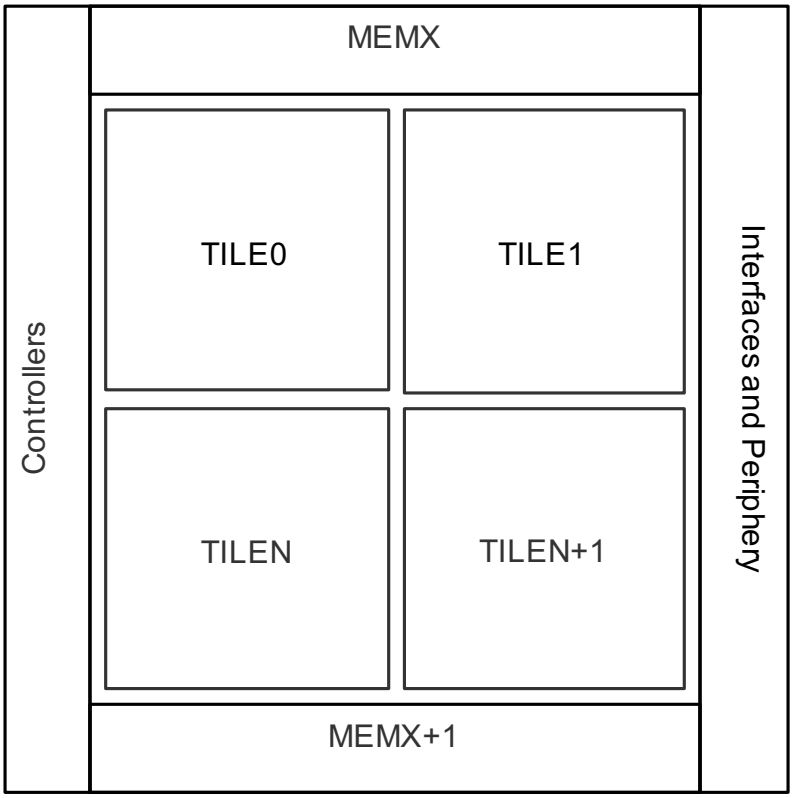


Figure 2.12: Entire DRP structure consisting of tiles, on-chip memory, controllers, periphery, and interfaces

memory. These memories may further be accessed by interface controllers which read from or write to the DRP's memory. Additionally, PEs, in general, do not contain control flow logic, but their program counter is centrally controlled using a finite state machine which may obtain control signals from the PE array. In case the architecture supports multi-threaded operation, the required control logic is thus also located in this execution controller. Furthermore, to facilitate reconfiguration efficiently, reconfiguration controllers are used to providing features such as configuration multicast or compressed configuration streams. The periphery of DRPs is mainly dependent on the degree of stand-alone operation and overall DRP complexity. Thus, such periphery may range from a non-volatile configuration memory to timers and interrupt controllers.

3 State of The Art

This state of the art is divided into three sections to reflect the current state of all areas that are touched upon in this thesis. First of all, FDSOI technologies are briefly introduced and compared to each other regarding geometry, process complexity, overall performance as well as benefits and drawbacks. To present a balanced view, the latest generation of FDSOI's main contender, so-called FinFET technology is presented as well.
To put such technology to good use, power management techniques, such as those this thesis proposes are utilized. Thus, the section on power management covers a broad selection of techniques and approaches on the latter, as well as a comparison, especially in regard to the contributions of this thesis.
Finally, the state of the art moves towards Dynamically Reconfigurable Architectures, the centrally utilized architecture of this thesis and approaches that exploit previously described power management techniques.

3.1 Process Technologies

3.1.1 Fully Depleted Silicon on Insulator

FDSOI technology is one of two remaining major approaches to tackle SCEs and other effects such as variability, hindering scalability. Currently, there are three major FDSOI technologies available: Renesas 65nm Silicon on thin BOX (SOTB), STMicroelectronics 28nm UTBB-FDSOI and Globalfoundries 22FDX 14nm front-end, 28nm metal-backend process.
SOTB has been first proposed in [20] along with the thereby enabled body biasing applications. Using such schemes, the drive current can be increased by 20% and great leakage reductions in standby modes can be achieved [20]. Furthermore, SRAM cells that are capable of exploiting body biasing are proposed. Continuing in [10], Ishigaki et al. give an extensive overview of the 65nm SOTB technology. SOTB currently is available in two flavors: Low-power (LP) and Low standby power (LSTP). Both flavors can co-integrate bulk structures in case non-SOI components need to be on the same die and both flavors are manufactured in a triple-well process. SOTB successfully suppresses SCEs via the BOX SOI structure and is free of self-heating [10]. Furthermore, the SOTB transistor architecture allows reducing variability by eliminating or at least reducing many variability sources such as random dopant fluctuation [21, 22] by employing an undoped channel region. Also concerning reliability, being a FDSOI technology, SOTB exhibits beneficial reliability properties when compared to bulk processes [23]. However, as these effects are often body bias dependent, this also leads to more complexity in reliability projections.

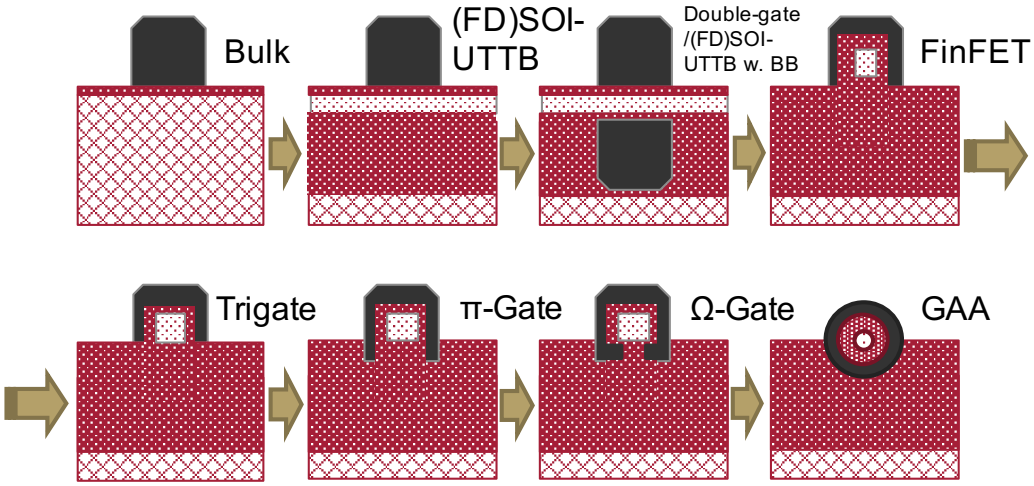


Figure 3.1: An illustration of the "FDSOI FinFET merge", describing how aspects of FDSOI and FinFETs combine in smaller geometries modeled after [32]

In smaller nodes, the Ultra Thin Body and BOX FDSOI technology developed by STMicroelectronics found widespread industrial adoption. In [24] Planes et al. introduced the technology with a focus on low-voltage computing. Due to the fully depleted channel region as well as the SOI construction, UTBB-FDSOI and derivative technologies offer high performance at low supply voltage, as well as high responsiveness to power management techniques [25]. UTBB-FDSOI comes in two principal flavors; both manufactured in triple well processes: Low Threshold Voltage (LVT) for high-performance applications, as well as Regular Threshold Voltage (RVT) for (ultra-)low-power applications [6]. The principal difference is in the back-plane doping and the gate-stack. Regarding reliability, Federspiel et al. demonstrated UTBB-FDSOI to be more resilient against all common effects in [26]. Globalfoundries 22FDX process has been announced in [27]. 22FDX shall be geared using four different flavors towards target domains. In the absence of official demonstration papers of Globalfoundries, publications by the entire SOI Consortium are cited. In [28], the technology outline is presented. Scaling from 28*nm* to 14*nm* in the front-end allows a delay reduction by 45%, 30% speed boost or a 55% power consumption reduction. Furthermore, body biasing capabilities are shown to retain their effect or even gain in effectiveness. Principal scalability has been shown to be feasible down to the 10nm node in [29], although further scaling is conceivable.

3.1.2 FinFET

FinFET tackles the SCE challenge by using a three-dimensional gate structure. Up-to-date processes by major semiconductor manufacturers, such as Globalfoundries (14*nm*), Samsung (14*nm*), Intel (14*nm*) [30] or TSCM (16*nm*) [31] are in mass production.
Reliability evaluations have been conducted and indicate similar results to FDSOI reliability evaluation. Despite continued scaling, BTI effects are not worsened [33]. While

FinFETs may have a more complex structure, variability is comparable to FDSOI [34] and has good prospects when employing SOI technology. In terms of principal scalability, TSMC reported the ramp-up of first commercial design in 10nm FinFET technology while 7nm FinFETs are announced to become available in 2018. FinFETs' further scalability is hardly a question of scalability, but rather terminology as Fig. 3.1 graphically illustrates. This figure visualizes the similarities of existing transistor architectures with proposed transistor architecture candidates for scaling below $7nm$ by illustrating the architecture evolution. This evolution indicates a convergence of architectural properties of FDSOI and FinFET technology.

3.2 Power Management

In the following, a broad selection of power management techniques is investigated and presented with two objectives in mind. First of all, this section shall allow a comparison of the state of the art with the approaches proposed in this thesis, and secondly, it should also elaborate on existing partitioning approaches for power management techniques.
However, to give such an overview of power management techniques and approaches that implement the latter in a logically structured manner, the entangled natures of these techniques only allow for compromises. This is then further obscured when considering the underlying partitioning techniques which may put power management techniques with seemingly different objectives into the same group. While, e.g., of course, the primary goal of multi-V_{TH} approaches is to improve static power consumption, it can also be used to optimize dynamic power consumption when reducing critical path delay with standard cells that employ a lower threshold voltage, consequently allowing to lower the supply voltage. Thus, it could be seen as both, a technique to lower static power consumption as well as dynamic power consumption. In regard to partitioning or rather selection of standard cells that should receive altered V_{TH} equivalents, it is e.g., identical to an unconstrained body bias partitioning problem with as many body bias levels as there are threshold voltage groups.

Thus, instead of categorizing these techniques by power management objective, they are loosely grouped, if at all, by the similarities of the underlying algorithmic problems. To highlight the objectives of the respective methods, they are categorized by their triangular relationship between static power consumption, dynamic power consumption, and energy efficiency. This relationship is illustrated in Fig. 3.2.
Starting with index 1, this marks the tradeoff between static and dynamic power consumption. This relationship in this state of the art refers to body biasing, where an increase in leakage can be traded for lower delays and vice versa. This, in turn, allows a circuit to reach a given clock frequency at a lower supply voltage level and thus possibly reduces dynamic power consumption. Whether or not this increases energy efficiency depends on the increase in static power consumption being less than the reduction in dynamic power consumption. Inversely, when reducing static power consumption at the cost of increased delay, the circuit potentially requires a higher supply voltage level for a given clock fre-

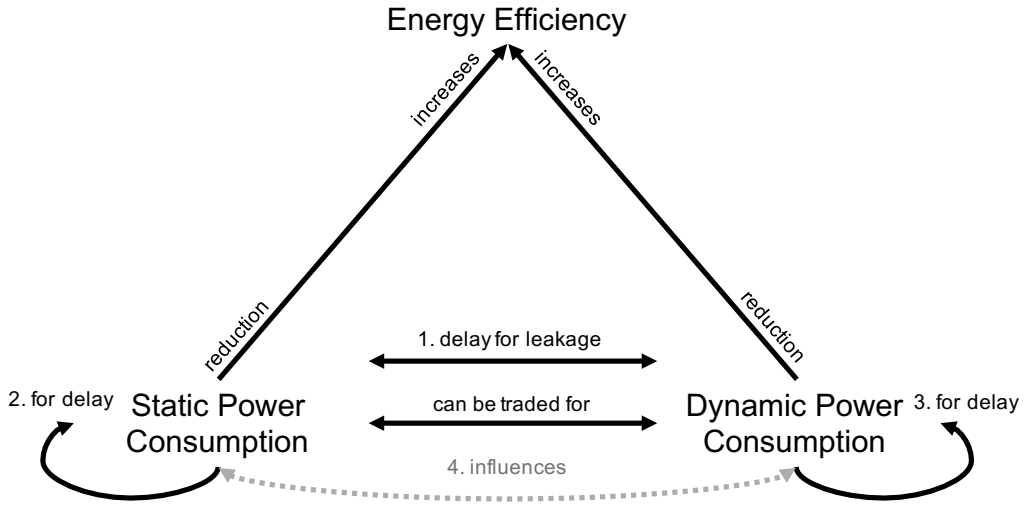


Figure 3.2: The triangular relationship between dynamic and static power consumption, as well as energy efficiency with categories of power management techniques indexed in numbers 1. to 4. accordingly.

quency and thus, is only sensible under certain conditions. Then there is the second relationship marked by index 2. This category fits multi-V_{TH} and multi-V_{DD} targeted for leakage reduction, power gating as well as pre-SOI body biasing. The relationship marked by index 3 applies to dynamic voltage (and frequency) scaling and clock gating.

Strictly speaking, multi-V_{TH} approaches are now also used in cutting-edge technologies to improve dynamic power similar to body biasing. To prevent the complexity of the proposed state of the art structure to raise further, such aspects will be ignored henceforth. Instead, a fourth relationship marked by index four is introduced which covers desired or unintended side-effects. For example, a change in supply voltage always affects static power consumption as well. However, as the supply voltage is usually primarily defined by the timing constraints, static power consumption is only spared a thought in this regard when the situation allows, e.g. in idle. Similarly, every power management technique requiring additional active components also increases static power consumption and incurs dynamic power consumption for its switching activity. These aspects, however, usually can be neglected as they are only employed when the benefit is significantly greater.

All these power management techniques and categories in mind, they increase energy efficiency if and only if the reduction on one side is greater than the increase on the other.

This triangular relationship categorizes the presented approaches as denoted in table 3.2. Considering these findings, there are three observations to be made. First, all power management techniques have side-effects, thus all techniques are in category 4. This category merely denotes that independent of the primary objective, other objectives are affected as well. This is a starch contrast to category 1 where such a tradeoff is actively sought through specified parameters. Secondly, if the power management method can be

Method	Scale Direction	Category
Pre-SOI Body Biasing	-	1, 2, 4
DVFS	up down	3, 4 3, 4
Clock Gating	-	3, 4
Power Gating	-	2, 4
Multi-V_{TH}	speed leakage	3, 4 2, 4
Multi-V_{DD}	-	3, 4
SOI Body biasing w. and wo. DVFS	up down	1, 2, 3, 4 1, 2, 3, 4

Table 3.1: Categorization of the power management techniques according to the triangular relationship in Fig. 3.2

used in opposite directions on a given scale, both directions need to be covered and usually result in different categories. Lastly, apart from category 1 representing the leakage/delay tradeoff exploited in body biasing, these approaches can be categorized along the lines of static power (2) and dynamic power optimization (3).

3.2.1 Pre-SOI Body Biasing

In regard to body biasing, the state of the art is split into two major categories. This section gives an excerpt of the state of the art on pre-SOI body biasing, that is the application of a potential on the transistor body in bulk transistors. While the principal effects are related to body biasing in FDSOI technologies, the actual physics differ considerably. Furthermore, body biasing in pre-SOI technologies is limited to such great extent, that it is used for entirely different purposes, such as mitigation of process variations. This of course also leads to a problem of nomenclature, as most people in the field associate this kind of pre-SOI body biasing with the general term "body biasing". To differentiate and to set both techniques apart, they are also treated separately, although they are obviously related.

In [35], Kobayashi and Sakurai propose a self-adjusting threshold voltage technique which utilizes body biasing to compensate for speed variation on a chip and thus allows to operate the chip at the ideal, non-variability affected frequency. To do this, they propose the so-called self-adjusting threshold-voltage schemes (SATS) which monitor the leakage current of a predetermined group in a constant feedback loop. Variation in leakage is assumed to correspond to the variation in threshold voltage. A sense stage thus measures leakage and then sets a body bias via a controllable voltage source which corresponds to the desired, lower threshold voltage. The voltage generator is realized using a charge pump design.

A similar, but substantially extended approach is proposed by Kuroda and Sakurai in [36]. They use the self-controlled capability to manipulate threshold voltage to both compensate for on-chip threshold voltage variation and to further boost transistor switching speed at low supply voltages. Here, the authors intentionally decrease supply voltage to reduce dynamic power consumption while they mitigate variability which has a specially high impact at low supply voltage. Furthermore, threshold voltage is further lowered to give a speed boost, reaching speed levels that would not otherwise be possible. This, of course, leads to increased leakage which is then again cut off by applying a reverse body bias in idle. Using this approach, the authors could reduce power consumption by 50% for a defined operation point.

Also targeting variability is Tschanz et al. in [37]. They propose a method called bidirectional adaptive body bias (ABB) which uses a self-controlled scheme similar to both [35] and [36]. Their approach utilizes both forward and reverse body biasing to compensate for die-to-die as well as intra-die variability. The authors target both timing as well as power constraints. Increased leakage is countered via reverse body biasing while timing issues are resolved by lowering threshold voltage via forward body bias. Timing problems are identified using a phase detector on the critical path while bias voltage generation is facilitated using analog multiplexers and externally supplied potentials. Using this approach, the number of chips not meeting the specification is reduced by a factor of $7\times$.

Martin et al. used one of the most popular applications of body biasing in the pre-SOI era in an actual product, the Transmeta Crusoe processor, and researched the tradeoffs for dynamic voltage scaling in combination with adaptive body biasing in [38]. Adaptive body biasing here is restricted to reverse body biasing, thus, the authors seek to find the most energy-efficient operation point in a tradeoff between dynamic power, influenced by dynamic voltage scaling, and static power using the exponential effect of body biasing on leakage. First, Martin et al. developed an analytical model using SPICE simulations and then using this model to derive optimized dynamic voltage scaling and adaptive body biasing strategies. Their evaluation on the Transmeta Crusoe processor, manufactured in a $180nm$ process, indicated an impressive 23% total power reduction, which is effect wise much closer to results of the FDSOI era.

In [39], Neau and Roy follow in the footsteps of [35–38] with their take on determining the optimal body bias to achieve compensation of variability as well as minimizing leakage. The authors analyze the impact of body biasing on all leakage components that were relevant at the time. Using this analysis, the authors model band-to-band and subthreshold leakage. They then define the leakage minimization goal as an equilibrium of band-to-band leakage and subthreshold leakage. A current mirror circuit is then used to determine the actual target body bias where this condition applies. These schemes are evaluated for $70nm$ and $50nm$ predictive models and achieve about 42% leakage reduction.

Teodorescu et al. present in [40] a fine-grained body biasing approach to mitigate process variations. They propose to partition a chip into body bias domains based on the inherent component structure, as timing variations are affecting component specific critical paths. The actual body biases are then chosen via a timing monitor and provided through a local bias generator.

Sathanur et al. propose a row based forward body bias assignment scheme to mitigate variability in [41]. By assigning forward body bias per row, Sathanur et al. are capable of avoiding the steep leakage increase as a side effect of using forward body bias on a chip granularity while keeping the physical implementation highly simple. The row based body bias assignment then works to reach overall timing while minimizing leakage. The actual row assignment algorithm benefits significantly from the row constraint and consists of two simple passes. First, a body bias is assigned to all rows so that the design meets timing. Once a body bias is found for which timing is met, the rows are sorted by timing criticality and the least timing critical rows are put on a lower- or zero body bias levels until timing is barely maintained.

The approaches presented in this section suffer from their pre-SOI intent where body bias could only be used in rudimentary forms due to physical limitations. While intriguing approaches are among them, like row-based or other clustering approaches, they are ultimately failing the reality of body biasing in SOI technologies where body biasing rivals supply voltage scaling. The central point is that they were not intended for any other use than variability mitigation or sleep current minimization. With much higher body biasing levels available, even a small group of standard cells can outweigh an entire row of standard cells at pre-SOI body biasing levels. Thus, while many methods are inspirational, the approaches cannot be transferred to SOI technologies.

3.2.2 Dynamic Voltage Frequency Scaling

Dynamic voltage frequency scaling (DVFS) is a method where the supply voltage as the primary variable is changed and then subsequently adjusts the clock frequency to meet the altered timings without timing violations. As it directly affects the operation conditions, it has a strong influence on the power versus performance ratio, drastically influencing both leakage and dynamic power consumption. However, it also has significant downsides and limitations. First of all, to conduct voltage scaling dynamically, significantly large voltage regulators are required. Furthermore, when two different voltage regions communicate through electrical wires, signals have to pass level shifters to account for the different signaling levels. Additionally, when using all degrees of freedom DVFS offers, signals between domains also have to pass different clock domains, requiring synchronization hardware such as phase locked loops. Thus, the hardware overhead is considerable, with its application warranted only on large granularities such as entire cores. Furthermore, with influence only on the operation condition of transistors and finite resolutions in its regulator components, it is far from an ideal solution as the following excerpt of the state of the art on DVFS shall show.

In [42], Isci et al. propose a workload-aware, global DVFS scheme under a fixed power budget. For such a system, they then consider different scenarios where the DVFS scheme is optimized towards prioritization of tasks, power per core balance or system throughput. The strategies are then evaluated for performance degradation, power budget utilization, and slowdown incurred by applying the proposed schemes. The presented results are of a more general nature, focussing on performance degradation over an oracle with perfect knowledge, i.e., optimal power strategies, as well as a comparison against chip-wide DVFS. In both respects, the approach performs very well with performance coming within 1% of the oracle for the throughput-oriented approach and clear superiority over chip-wide DVFS. With power versus performance tradeoffs having a different focus at the time of publication, the authors include their findings on power balancing, but fail to stress the implications that point to DVFS as a powerful tool to conduct such tradeoffs.

Herbert and Marculescu present in [43] the dynamics involved in DVFS strategy decisions and also propose a method to partition a processor into voltage as well as frequency domains. They evaluate three different DVFS algorithms: a utilization threshold based algorithm, a proportional-integral controller as well as a greedy algorithm searching for the minimal energy per throughput squared. The partitioning method, or more partitioning rule, merely prescribes what kind of component should be put into its own domain or multiple domains. The results are thorough and demonstrate that DVFS can also benefit from finer than core granularity partitionings. Furthermore, by putting the considered strategies into context with the evaluated benchmarks, they derive conclusions regarding suitability to benchmark classes. As an intriguing result, the authors conclude that the core-granular DVFS approaches often do not provide enough merit to justify the increased complexity. Furthermore, they continue by stating that the granularity on which DVFS is practiced to this date is insufficient and point out the energy efficiency gains that could be achieved when applying DVFS in a fine-granular manner.
Kim et al. concur with the savings that can be achieved by employing at least core granularity, but also further explore temporally fine-grained applications of DVFS realized using on-chip regulators [44]. This is especially interesting as it highlights the overheads associated with the different implementations of DVFS and thus illustrates the complexity and cost of efficient DVFS schemes. Kim et al. present an elaborate and fair assessment of the overheads power and area-wise, which is of great importance considering the hardware effort. Accounting for all overheads, the application dependent reduction in power consumption range from -7% for their worst-case scenario (`raytrace`), up to 21% for their best-case benchmark (`ocean`). Considering that the ideal DVFS savings are 2% and 24%, the proposed method is quite close to the optimum. An additional merit of the presented work is also an overhead evaluation regarding regulator implementation.
Yin et al. evaluate DVFS and its impact on a per-core basis for on-chip networks. In such designs, cores are interconnected using sophisticated interconnects and routers [45]. To mitigate the area overhead that would be occurred by placing a voltage generator for each router, the authors propose to use a power delivery mesh with specifically dimensioned transistors as voltage selectors. The actual DVFS scheme is then governed by the local

traffic status. For power consumption estimation, Yin et al. use the alpha-power model [15] by Sakurai and Newton. The power consumption reductions are cited to be between 45% and 60% with a roughly doubled latency increase. While the proposed ideas are solid, this work again points towards the complexity in realizing DVFS in a beneficial manner. First of all, offloading voltage generation may be sensible when requiring only few voltage levels, but aiming to use this setup in an extremely fine temporal granularity of a couple of nanoseconds contradicts to some extent the original intention of reducing the area overhead. Furthermore, more substantially, with all latencies almost doubled, other, probably more suiting techniques could be used as well.
In [46], Howard et al. propose a 48-Core x86 message passing processor design with 8 voltage and 28 frequency domains. Voltages are supplied from off-chip regulators ranging from 0 to $1.3V$ in $6.25mV$ steps. Each voltage domain can be put into a disabled retention mode at $0.7V$ or completely cut off if retention is not required. Communication is facilitated using a sophisticated 5-port virtual cut-through 2D mesh network, operating at a maximum frequency of $2GHz$ with 16-byte data links. The design allows for efficient power management schemes, allowing to adjust power and performance characteristics according to the momentary demands. Power consumption for the entire chip can be cut by 80% from normal operation down to a low power mode. Nevertheless, this naturally also goes along with significant performance cuts. To achieve the 80% reduction, clock frequency is reduced by a factor of $8\times$ from $1GHz$ to $125MHz$.
Cochran et al. focus in turn on DVFS strategies in [47]. They propose thread "packing", i.e., grouped execution of threads along with adjusted DVFS to meet a certain power budget, referred to as power cap in their work. Using performance counter data, power and temperature measurements along with multinomial logistic regression, thread packings and the corresponding optimal voltage and frequency settings are determined. The scheme is applied and evaluated on a real Intel Core i7 based server. In 82% of all executed tests, power consumption stayed in the power cap even without dedicated power consumption measurements. This strategy harmonizes threaded workload execution and DVFS control and thus gives an additional edge over thread-agnostic DVFS. This also allows to meet more power versus performance constraints and thus allows to leverage DVFS more efficiently with only a small software overhead.
Similarly, Kolpe et al. approach the voltage regulator overhead problem using software implemented DVFS strategies in [48]. By grouping several cores to one voltage frequency island, the number of required voltage regulators is reduced by the grouping factor. Then, by analyzing workloads, similar workloads are assigned a DVFS cluster. These schemes are evaluated in a simulated 16-core processor with evaluated cluster sizes from 1 core per voltage frequency island, i.e., per core DVFS, 4, 8 and finally 16 cores per cluster, i.e., chip grained DVFS. Then, for a given power constraint, the execution time required by each scheme is evaluated. The results indicate that while chip grained DVFS needs a relaxation of constraints, core clustering constitutes a valid and interesting method to reduce the hardware overheads usually associated with DVFS.

This excerpt of the state of the art on DVFS clearly shows that it is a very effective power management technique as it manipulates the operation point, but the selected works also

demonstrate that there is a considerable price to pay for in terms of overheads. Furthermore, in the case of DVFS as well, fine-grained applications indicate further directions for optimization but are ultimately hindered by the steep implementation overheads and the theoretical maximum power consumption reduction achievable through DVFS just not offering enough incentives to go for finer granularities. Using advanced techniques, workload scheduling and DVFS strategies, DVFS utilization efficiency can be optimized, however only towards an ideal DVFS strategy oracle. Furthermore, overheads associated with the implementation of DVFS can be reduced, but only at the expense of spatial or temporal flexibility, again resulting in larger power consumption for executing the same computations. Ultimately, DVFS is a highly efficient technique when the workload gives or is made to give enough leeway for the physical actuators to adjust the operating conditions.

3.2.3 Clock and Power Gating

In this subsection, approaches to leverage clock and power gating are presented. Clock gating is a method to cut off parts of the clock tree when the regions supplied by these parts are inactive, i.e., not used. This is realized using special standard cells for clock tree use which have increased blocking and driving capabilities to account for the strong drive power of clock tree cells, required for the large fan-out. By preventing the clock signal from entering inactive regions, considerable dynamic power savings can be achieved, while only static power consumption remains. Power gating is a continuation of the clock gating idea. Here, not only the clock signal but the entire power supply is cut off using power switches. Power gating, however, requires a lot more hardware to be implemented in a well-defined manner. It does not only need power switches, but also retention cells to retain some initial state into which the cut-off area is recalled when returning from its off state. Furthermore, even if no such state needs to be retained, the power gated region needs some kind of high resistance switch to prevent actual signals from entering and thus driving the cells within. Of particular interest for this thesis are the methods by which clock or power gating regions are defined.

Clock gating approaches presented in this section can be categorized into three different types: symbolic, analytical as well as architectural approaches. One of the classical examples of symbolic computing based clock gating is [49] by Benini and Micheli. Here, a symbolic, boolean representation of the circuit in question is used to determine the conditions, in which parts of the circuits are inactive. Using these conditions, appropriate logic determining the satisfaction of the condition together with a clock switch is inserted at the appropriate locations in the netlist. Analytic approaches ultimately also employ symbolic methods but are differentiated by analyzing the original RTL and then synthesizing a derived clock structure. For example, [50] and [51] do such analyses and then manipulate clock structure synthesis accordingly. In the case of [51], this is done implicitly by modifying the verilog source code which is then used for clock tree synthesis by a commercial layout tool. Many EDA tools also offer clock gating support as early as in [52], where the authors present the clock gating capabilities of Synopsys' Power Compiler. The last remaining clock gating approach category is also very intriguing. By

utilizing architectural information, Li et al. present a clock gating approach in [53]. In this work, the authors present a method how architectures such as in modern pipelined processors can be used to determine clock gated areas as well as their activation. As it is usually known a couple of clock cycles beforehand, whether certain parts of a pipeline stage or pipeline stages altogether are used or not, this can be leveraged to implement clock gating conditions and activation. All these presented approaches result in considerable dynamic power reductions and are thus well in the standard repertoire of low-power chip design.

Power gating, on the other hand, is not as straightforward due to the involved overheads, including those overheads occurring at runtime.
Usami and Okubo propose in [54] to use the clock gating capabilities to use this activation information to build power gating domains. This is done by traversing flip-flops with clock gating gates inputs and marking the combinational paths encountered. If other flip-flops without clock gating gates are encountered, they are marked using the original flip-flops id as well. This is done for all clock gating enabled flip-flops, while the markings of all the gates marked with more than one ID are concatenated. Then, power gating domains according to those labels are created, while those gates without any label are not power gated. For the actual power gating strategy, Usami and Okubo also analyze the overheads of deactivating and reactivating power gating domains and put areas to sleep only if the period of sleep is longer than the break-even point. As clock gating signals have been originally used, this is a critical step.
Leinweber and Bhunia propose hypergraph partitioning and shannon decomposition to determine power gating domains in [55]. In a first step, the mapped netlist is parsed and a hypergraph representation generated. Then, the resulting hypergraph is partitioned in a shannon expansion aware manner to allow shannon decomposition to exactly separate unrelated logic. In this step, the hypergraph is also coarsened on common activation conditions to simplify the structure, allowing also larger, complex designs to be partitioned. If these steps worked correctly, which is assessed using an introduced quality metric, shannon decomposition deals neatly with shared resource and defines non-overlapping power gating domains.
Sathanur et al. present two different approaches for power gating based on clustering functionality in rows in [56]. The optimal approach uses binary integer programming to assign standard cells to rows that are then power gated altogether, while the other, heuristic approach analyzes the layout-based row assignments of standard cells and ranks each row by timing criticality. Then, starting with the subset consisting of all rows, rows are dropped until power gating given rows do no longer lead to a timing violation. This, of course, requires the designs to be "pre-separated" using the row-based layout style. Despite this severe row clustering constraint, the power gating results get very close to normal, ideal power gating partitionings.
Saito et al. propose a fine-grained power gating approach based on the architectural structure of coarse-grained dynamically reconfigurable processors in [57]. Along with a high-speed power gating technique requiring only a couple of nanoseconds for wakeup, the authors also analyze the involved overheads and consider power gating only beyond the

appropriately chosen break-even points. By power gating PE components individually, leakage reduction is achieved even when parts of the PE are in use. As a nice plus, Saito et al. consider all overheads and present their achieved leakage reduction of 48% in the light of expended hardware overhead, which they measured to 9% in their real chip implementation.
Chen et al. propose a power gating design method for standard-cell like structured ASICs in [58]. To realize this, transistors are grouped into via-configurable logic blocks with power gating capabilities per such block. The proposed structured ASIC is comprised of both power gating enabled logic blocks as well as blocks without power gating capabilities. The actual assignment is then done using a standard synthesis tool while in the library information, power gating enabled logic blocks receive a smaller area, making the synthesis tool choose those over non-power gating enabled blocks once timing is met. Just like with clock gating, the actual power gating decision is offloaded to the synthesis tool.

In summary, clock and power gating rightfully belong to the standard repertoire of low-power design. While clock gating can be used almost free of side-effects, power gating requires significant consideration of tradeoffs due to its hardware overheads as well as its physical limitations. As circuits under power gating do not retain their state, additional precautions are required to send to and awake these circuits from their sleep state in a well-defined manner. This being said, power gating is the most efficient method to eliminate a component's power consumption without switching off the whole chip. However, the area overheads incurred by retention cells to retain the component's state, control logic as well as power switches require the power savings to be rather large to justify these efforts. i.e., fine-granular application is extremely complicated to achieve in an economical fashion.

3.2.4 Multi-V_{TH} and Multi-V_{DD} Approaches

Multi-threshold voltage standard cell design as well as multi-supply voltage design are two entirely static low-power design methods. Multi-threshold voltage (V_{TH}) standard cell libraries contain standard cells executing the same logical function, but utilize transistors with a different threshold voltage for their realization. In consequence, if transistors with higher threshold voltage are used, the standard cell is slower but also less leaky. In contrast, if transistors with lower threshold voltage are used, the standard cell incurs more leakage but delivers its outputs faster. The different threshold voltages are attained through changes in the transistor structure and thus are permanent and unchangeable after manufacturing. Other than this constraint, multi-threshold voltage designs only incur additional hardware overheads in accordance with the threshold voltage manipulation. If, e.g., different threshold voltages are realized using longer channel lengths, transistors become bigger and thus may require more area for their implementation. If, on the other hand, the gate work function is manipulated through the materials used, it may require an additional fabrication step and thereby also additional masks, but is then neutral in regard to area overheads.
Multi supply voltage (V_{DD}) design supplies clusters of standard cells with different supply voltages, thus attaining a similar speed and leakage tradeoff. Additionally, as supply

voltage is manipulated, dynamic power changes as well. Just like with DVFS however, different supply voltage domains require level shifters between such domains. This also leads to the above-mentioned constraint, i.e., that only clusters of standard cells are considered as otherwise the overhead would outweigh any benefit attained. Due to the limited applicability of sole supply voltage scaling, it has been rarely used. The overheads of supplying multiple supply voltages in combination with the required level shifters usually made static multi-supply voltage designs less appealing, in clear favor of dynamic voltage scaling or DVFS.

Multi-V_{TH} approaches have been used in a similar fashion as pre-SOI body biasing to achieve leakage and delay tradeoffs. In [59], Kim et al. use multi-V_{TH} techniques to minimize the power consumption incurred by leakage. To achieve this, the authors first model leakage and access time in dependence on threshold voltage V_{TH} and then use numerical solving techniques to minimize the leakage function constrained by the cache access time. This approach achieves significant leakage reductions, however, their peak reduction also requires adaption of the architecture to accommodate an additional cache level to mitigate the access time increases.
Gupta et al. propose to use fine-grained gate-length biases to offer a large variety of threshold voltages without the need for additional masks in [60, 61]. The authors focus on supplying additionally characterized fine-grained gate-length biased standard cell libraries which are then used by industry standard synthesis tools. Their approach includes the automatic adaption of the standard cell layout and thus constitutes a fully automatic multi-V_{TH} methodology. Like the early pre-SOI body biasing approaches, multi-V_{TH} is also used to mitigate sources of variability, such as [62].
In this approach, Calimera et al. use multi-V_{TH} cells to make designs less vulnerable. This is done by modeling the behavior of standard cells with different threshold voltage as well as their temperature dependent effects, such as temperature inversion. Then, the actual synthesis is conducted with slightly overconstrained timing which gives the critical path enough slack for temperature timing variation. The uncritical paths are then populated with high threshold voltage cells which are assumed to have a positive temperature inversion, and thus become less timing critical with increasing temperature. With the overconstrained critical path and the intended temperature inversion on all other paths, timing is kept even in worst-case corners.
For multi-V_{DD} approaches, Yeh et al. propose in [63] a simulated annealing based placement algorithm to cluster standard cells with equal intended supply voltage. These clusters are then supplied with different supply voltages, which minimize the use of level shifters only between those clusters. Their proposed approach clusters standard cells in rows while the availability of both supply voltages allows for mixed V_{DD} rows. The only constraint imposed on such rows is that one V_{DD} may be only on one of either side to minimize the inference of level-shifters.
Puri et al. present an interesting case-study in [64]. In this study, the authors explore the possibilities of combined multi-V_{DD} multi-V_{TH} design. While the authors acknowledge the theoretical nature of their study due to the additional complexity and overheads involved, their findings give a clear indication of the merits of multi-V_{DD} design in view of the over-

heads. The authors evaluate different optimization techniques and different numbers of threshold voltages in combination with single- or dual-V_{DD} design. In dual V_{DD} design, the design is partitioned into parts running at a lower supply voltage where the other parts operate at a higher supply voltage. While the target design in their study, a DSP processor, exhibits a potential 50% power consumption reduction, this is furthered by another 14% when combined with their dual-V_{DD} design method.
Li et al. propose to use programmable V_{DD} design for FPGA interconnects in [65]. As for uncritical paths, the interconnect is overdesigned, the authors propose to use multiple supply voltages to exploit the difference in timing criticality. To do this, three different supply levels are used: high V_{DD}, low V_{DD} and power gating. These different levels are then assigned by timing criticality or set to power gating when the interconnect part is unused. While it is evident that this method considerably reduces power consumption, the work falls short of considering the area overheads which cannot be neglected for such approaches.

In summary, while multi-V_{TH} approaches such as gate-length biasing found its way into standard low-power design, multi-V_{DD} as the works described in this selection of the state of the art, was virtually replaced by the more powerful DVFS schemes. While the merit of multi-V_{TH} design is indisputable, its static assignment limits the extent of potential power savings, as all paths and their V_{TH} had to be chosen in such manner, that timing is always kept, whether the path is used or not. Dynamic multi-V_{TH}, in this sense, is highly desirable, but physically impossible for these multi-V_{TH} approaches.

3.2.5 Body Biasing with DVFS and approaches solely focusing on Body Biasing in SOI Technologies

Most of the time, body biasing in SOI is combined with DVFS. This is not only sensible as both techniques, voltage scaling and body biasing effect the same central variable, clock frequency, but also allow them to be managed similarly. Body biasing can be easily integrated by not solely considering supply voltage in a DVFS controller, but rather a tuple consisting of both supply voltage and body bias. So it does not come as a surprise that most existing work considers body biasing in SOI transistors in conjunction with DVFS.

Such an approach is also proposed by Akgul et al. in [66]. The authors propose an adapted mode selection scheme for STMicroelectronics' 28nm UTBB-FDSOI process based on discretely convex sets of power modes. Each of the power modes is a tuple of frequency and power consumption. The space of power modes is then set up using supply voltage and body bias. Since the set of all power modes is not necessarily convex, the authors give instructions on creating a convex subset and also proof, that this process can be used to minimize power consumption for a given clock frequency. This method is then demonstrated using an oscillator implemented in STMicroelectronics' 28nm UTBB-FDSOI. In real chip evaluation, the authors achieved a 27.55% reduction in power consumption. To give the authors fair credit, it has to be noted that an oscillator does not represent an ideal test-case, but also indicates that there is much unused potential in cases where dynamic

power does not dominate as much as in ring oscillators.
In a related paper [67], Akgul et al. extended this approach in regard to two aspects: mathematical approach and evaluated design. The mathematical approach in this work is now explicitly based on piece-wise convex sets to optimize power consumption using three drivers: supply voltage, body bias and clock frequency. Then, for a given target frequency f_{target}, two modes are always considered: V_{DDi} as well as the next higher V_{DDi+1}. For these two modes, a power consumption curve is determined, i.e., power consumption is approximated and plotted for a given V_{DD} by sweeping over body bias. In a subsequent step, the piece-wise convex parts of each curve are determined and stored as piece-wise convex sets of each mode. If now the target frequency F_{target} can be reached using a power mode of the mode of the lower supply voltage V_{DDi}, this mode is chosen. If the mode cannot be reached, the piece-wise convex set of the next higher supply voltage V_{DDi+1} is searched, and its appropriate power mode is chosen. This approach is then evaluated using a DSP design where each PE is a DVFS and body bias domain. Obviously, the PEs considered in this DSP design are significantly large cores to offset the DVFS overheads. In total, despite the mathematical refinement, the authors determined a up to 17.31% improvement in power consumption over the FDSOI implementation not using the proposed combined DVFS body biasing method.
Chen et al. propose a power optimization technique for real-time multi-core SoCs with optimal DVFS and DPM in [68]. With the added complexity of real-time deadlines to be observed and the subsequent challenges for DVFS and DPM, Chen et al. resort to a variant of integer linear programming. Using this approach, optimal DVFS and body bias combinations are determined while observing the real-time aspects. Their approach yields a power consumption reduction of 10.5% and 8.9%, with peak reductions of 16.0% and 12.2% for 4- and 8-core systems respectively over all evaluated benchmarks while meeting all deadlines.
Jevtić et al. propose a combined per-core DVFS and body biasing scheme with the DVFS power management realized using switched-capacitor converters in [69]. The authors propose to let the switched capacitor voltage generator output to ripple and thus achieve higher DC-DC conversion efficiency. The ripple is the mitigated by frequency tracking the ripple using the processor core. This ripple spanning a considerable amount of time is compensated in the adapted DVFS scheme and its greatly enhanced body biasing range. Through body biasing, V_{DD} hopping requirements due to the ripple at lower V_{DD} is virtually overcome. By combining this approach with body biasing the authors realized an improvement in over all chip energy efficiency of up to 25% depending on the application.
One of the few approaches focusing on the optimal utilization of body biasing is [14] by Okuhara et al. The authors introduce a new, real-chip evaluation based theoretical model using a method for finding optimal supply and body bias voltages. The evaluations were conducted for Renesas 65nm SOTB FDSOI on a microcontroller design where the processor core is placed in a body bias domain other than the memories. Using the newly introduced model, the authors can make more applicable approximations for the examined FDSOI process and achieve a prediction accuracy ranging from 5.23% to 12.6% versus real chip observations. Essentially, the model and optimization method make sure that dynamic power consumption via supply voltage and static power via body biasing are kept

in an optimal tradeoff position.
Conti and Benini as well as Rossi and Benini developed the PULP platform geared toward ultra-low-power computing and presented two generations in [70, 71]. Each PULP chip features a SoC voltage domain and a cluster voltage domain with four OpenRISC cores, where each core can be biased individually. As an additional innovation, the body bias selection is realized using a voltage multiplexer referred to as BBMUX which selects body biases in a highly area-efficient manner from a global bias supply grid. While the first generation of PULP has been manufactured in STMicroelectronics UTBB-FDSOI RVT flavor, the second generation is manufactured using the LVT flavor, allowing for significantly increased forward body biasing. By exploiting body biasing on such a per-core basis, energy efficiency is improved well above 10% all over the supply voltage range, with peak efficiency improvements at low supply voltages of more than 30%.
Most influential for this thesis are the works by Hioki et al. on fine-grained body biasing enabled FPGAs as described in [72, 73]. In these works, Hioki et al. propose a FPGA structure partitioned into multiple, fine-grained body biasing domains. The FPGA is overlayed with a body bias supply grid where each body bias domain is equipped with small voltage selectors. In many respects, these studies are limit studies, i.e., they seek to define up to which granularity area overheads are incurred by fine body bias domain partitionings justified by the returns, i.e., performance improvement or leakage reduction. The test chip evaluated in [73] consists of 121 FPGA tiles, where each tile consists of 8 switch multiplexers, 12 input multiplexers, 24 local multiplexers, 4 look-up tables, a flip-flop set, a set of 2 to 1 multiplexers and pad output muxes. For each of these 57 elements in total, a body bias can be set individually as part of the configuration memory. With 121 tiles, this makes 6897 body bias domains in total. This, of course, also incurs a significant overhead on the $3.2mm \times 2.4mm$ die. That is 54% of the die is occupied with structures required for the implementation of the body bias domains and its supply structures. While such overheads, of course, are unacceptable for actual designs, Hioki et al. clearly illustrate the potential of body biasing in FDSOI technologies such as SOTB. The actual body bias levels are set by the EDA tool based on path timings.

Having summarized this selection from the state of the art on the utilization of body biasing in FDSOI, there are several issues with the presented works. First of all, the approaches are split in half between those that focus on the application of body biasing within the scope of DVFS [66–69] and those that focus solely on the optimized usage of body biasing [14, 70–73]. As noted on DVFS above, DVFS is severely limited in the manner it is applied due to the large overheads associated with the latter. Body biasing, on the other hand, can be realized on much finer granularities. First of all, body biasing generally does not need large voltage regulators as only a potential has to be generated, i.e., there is only current flow when charging the transistor bodies. However, no level shifters are required as signal levels are unchanged by body biasing. This additional degree of freedom is nonetheless completely ignored by the presented works with the exception of [72, 73]. Even in [14, 70, 71], granularities below core-granularity were not considered. In [72, 73], the authors went to great lengths to highlight the possibilities of extremely fine-grained body biasing, however, they only focussed on a specifically optimized FPGA design with

the goal of leakage minimization. Thus, also the field covered by this thesis, methods on optimized body bias domain partitioning and general, optimal body bias assignment methods is with the exception of a few pre-SOI works has been completely untouched to this day.

3.3 Dynamically Reconfigurable Processors

In this section, a brief overview of the state of the art on dynamically reconfigurable architectures will be presented. The first section presents a short, non-exhaustive overview of popular DRP architectures and applications and directions to which DRPs have been applied. In the second section, the MuCCRA DRPs will be introduced of which the most recent architecture has been used in the evaluations of this thesis.

3.3.1 DRPs and their Applications

Much of the principal groundwork concerning DRPs has been done around Masa Motomura who first introduced the NEC DRP-1 in [18]. DRP-1 featured all defining DRP concepts with PEs grouped into individual arrays, so-called tiles as illustrated in Fig. 3.3. Each tile is equipped with a state transition controller for execution control and memories as well as full-width multiplication units placed around the tile boundaries. Using this architecture, many different types of applications have been ported and successfully benchmarked, such as general stream applications [75] with another study on the thereby encountered peculiarities [76], IPsec cryptographic acceleration [77] or JPEG2000 as a graphics benchmark [78]. Furthermore, one of the major issues concerning DRPs, application mapping, has also been addressed by use of high-level synthesis [79]. Popular commercial architecture examples are also the ADRES architecture developed at IMEC [80] and the Samsung Reconfigurable Processor [81]. Both use the same principal VLIW processor PE/FU array integration as illustrated in Fig. 3.4. While their PEs are arranged in a two-dimensional array, a couple of specially equipped PEs (yellow) are reused to form a VLIW processor. The ADRES and SPR architectures are a mixture of DRP and VLIW processor with specially equipped, multi-functional PEs which can form a VLIW processor for designated tasks. Such special cases as found in architectures like the SRP or IMEC ADRES [80, 82–84] can also be treated similarly to an extension in the sense of Special Purpose Units as described in the background section. Another noteworthy architecture is the PACT XPP [85]. For all three architectures, ADRES, SRP and PACT XPP, a development environment including DRP mapping optimized compilers has been developed [82, 86].

Apart of such commercial variants, there are also academic DRPs such as the MuCCRA architectures introduced below. Among others, there is the CRC architecture template [19] for which also compilation technique studies and optimizations have been conducted [87, 88]. While the above architectures like the original NEC DRP aimed for generality within the scope of MIMD architectures, there are also more specialized

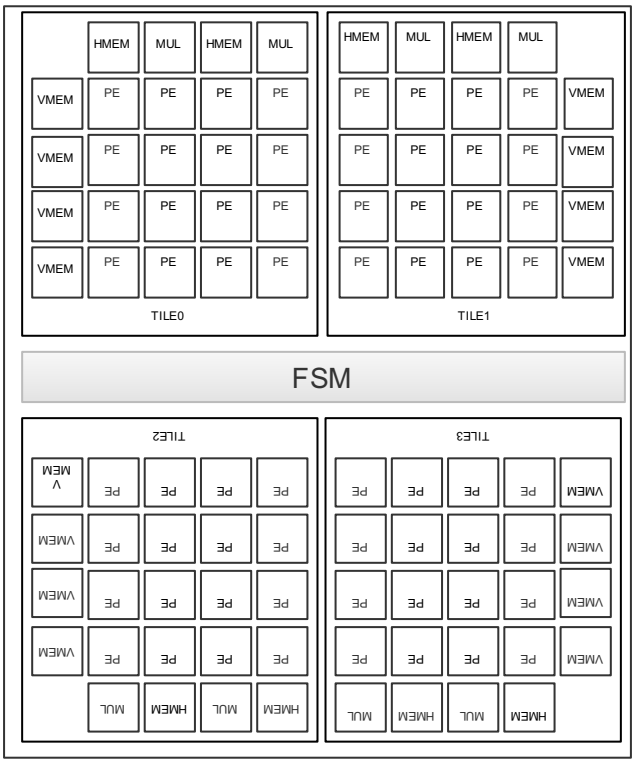


Figure 3.3: Illustration of NEC's DRP architecture style, modeled after [74]

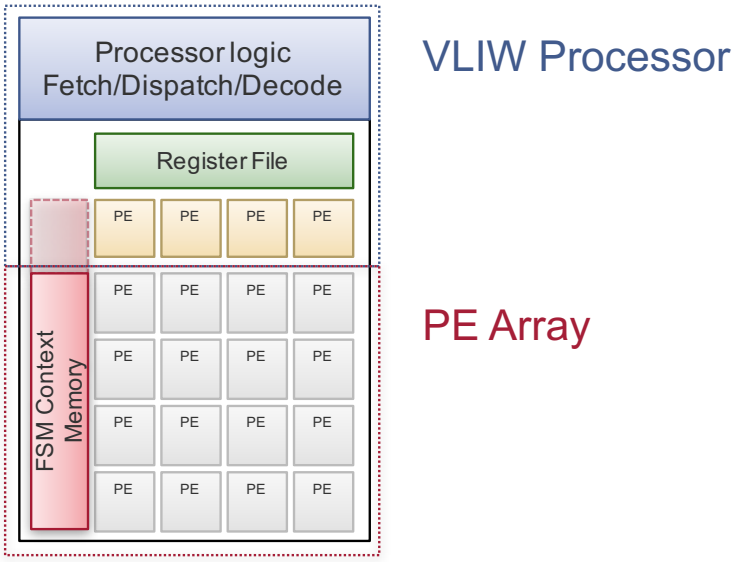


Figure 3.4: Illustration of IMEC's Architecture for Dynamically Reconfigurable Embedded System (ADRES) style DRP, modeled after [80] and [81]

variants of DRPs geared towards specific applications. For example, there are a couple of implementations featuring floating-point unit equipped PEs such as [89] or [90]. The obvious downside of such implementations is, of course, the thereby incurred area penalty of relatively large floating-point units per PE, while on the other hand, this returns superior throughput in floating-point use cases. Concerning this tradeoff, [83] examined multi-domain application mapping on DRPs and CGRAs. Another example for highly specialized applications of DRPs is [91], a DRP optimized for neural network applications.

As DRPs per construction traditionally aimed for energy efficiency, many studies using low-power techniques, such as those referenced in the section on MuCCRA DRPs, have been conducted. These studies however always suffered from the difficult area versus power consumption reduction trade-off DRPs had to face in light of the coarse, but compared to GPCPUs still considerably fine granularities. Most commercial architectures therefore also were rather conservative and pushed the low-power design aspect more into the software developer and compiler techniques domain. With most power optimization studies thus either exploiting the inherent energy efficiency of DRPs, there are no comparable studies on the exploitation of FDSOI through fine-grained partitioning of a DRP into body bias domains.

3.3.2 MuCCRA DRPs

Multi-Core Configurable Reconfigurable Architecture (MuCCRA) is a series of DRPs developed at the laboratory of Professor Hideharu Amano at Keio University, Japan. Among other projects, the MuCCRA architecture is a series continuous development and has thus

has been developed in a multitude of variants and implementations. Apart from the architectures, an ecosystem has been built around DRPs, ranging from software development tools, architectural templates to high-level synthesis. In the following section, the base architectures will be introduced, followed by noteworthy specialized versions and studies, closing with a brief glance on tooling.

Architectures and Implementations

MuCCRA-1 has been implemented on a 5mm square die in ROHM's 180nm process and featured sixteen 24 Bit PEs [92]. Each PE contained 64 context memory entries. MuCCRA-1 is a heterogeneous DRP architecture, as there is only one multiplier per PE row to reduce area consumption. Clock frequency ranges from 20 to 40MHz and is application dependent.
MuCCRA-2 is implemented using ASPLA's 90nm process on a 2.5mm square die and consists of sixteen 16 Bit PEs [92]. With MuCCRA-1 occupying much area for its memories, MuCCRA-2 was created to address this issue. This is achieved by sharing context memory between two PEs. Furthermore, the amount of local memory is reduced, which is compensated by increased routing capabilities.
Building on these experiments, MuCCRA-CUBE, a 3D-stacking MuCCRA chip has been developed [93]. Per die, there are sixteen 24 Bit PEs implemented in ASPLA's 90nm process on a 2.5mm by 5mm die. Each die is equipped with a through-chip interface based on inductive coupling with multiple channels. Using this method, 4 dies are stacked. Each PE is equipped with a multiplication unit and switching elements are extended to account for communication in three dimensions.
MuCCRA-3 is a consequent evolution of the previous MuCCRA DRPs, focussing especially on low-power applications [94]. It consists of a sixteen PEs array with a word side of 16 Bit each. It is implemented in Fujitsu's 65nm eShuttle process on a 4.2mm by 2.1mm die. One of the power optimizations is done by registering all PE outputs and thus reducing the amount of unnecessary switching caused by e.g. glitches. Consequently, the combination of operations, i.e., chaining, has been restricted as it couldn't be leveraged in previous versions.

3.3.2.1 The MuCCRA-4 Architecture

Building upon MuCCRA-3, the MuCCRA-4 architecture will be described in the following. MuCCRA-3 has been geared toward low-power computation [94]. To achieve this, several previously suggested low-power optimizations have been incorporated, e.g. [95]. Despite its low-power focus, each PE's ALU contains a full-width multiplication operation.
Fig. 3.5 depicts MuCCRA-3's PE array structure including its island style interconnect. The interconnection network is realized through switching elements (SE). Each SE has two channels. Each channel transfers 16 Bit plus 1 Bit carry data.
Likewise, the PEs operate on 16 Bit data and an additional carry bit. A part of the energy efficiency optimization has taken place inside PEs. The number of operations that can be

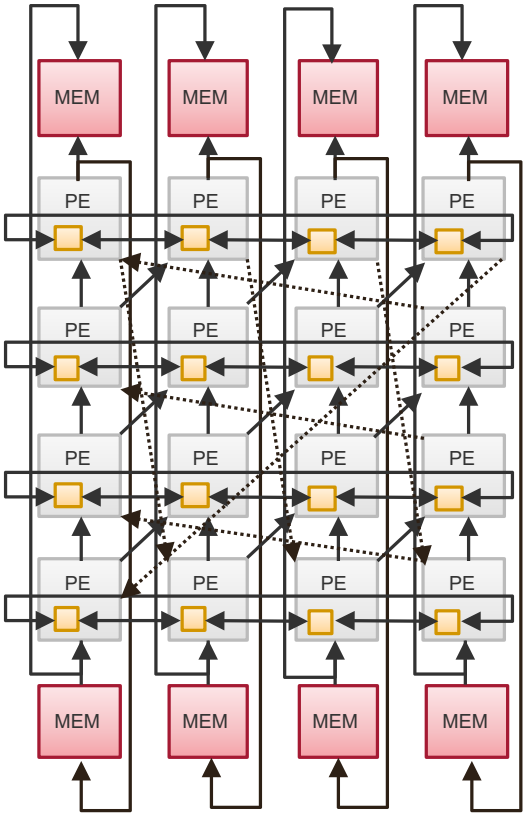


Figure 3.5: MuCCRA-3 DRP's PE array with island-style interconnect, modeled after [94]

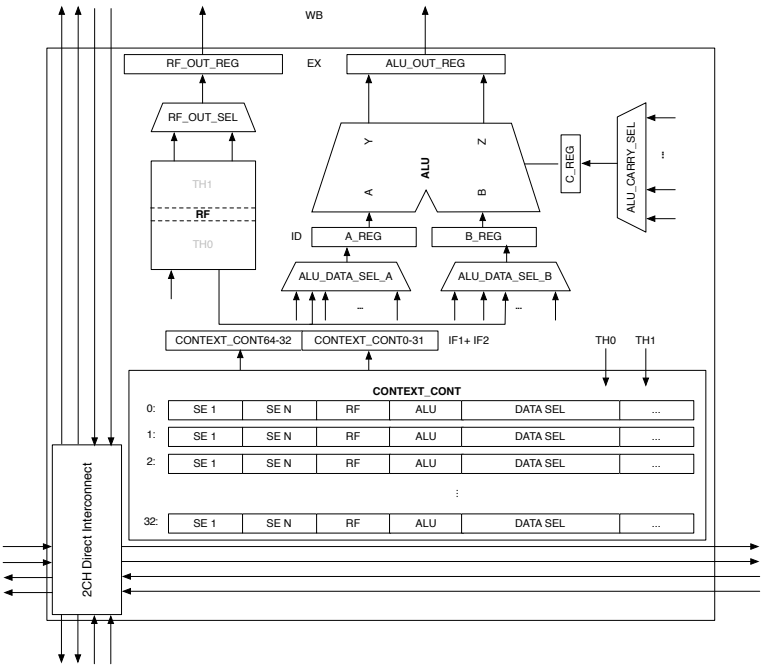


Figure 3.6: MuCCRA-4 PE featuring a direct interconnect with two channels in each direction, register file, ALU and data selection multiplexers as well as registers used for pipelining. Additionally, register file regions and thread pointers (`TH0` and `TH1`) to realize multithreading are added.

executed in a single PE has been significantly reduced to 13 basic operations. Complex, combined operations have been eliminated as they do not only heavily increase area and thus also static power, but can be rarely leveraged in applications and thus extend critical paths without actual benefit. By omitting such operations, more operations per second can be executed with less static power consumption. Even though the DRP is thereby in total more utilized, it increases energy efficiency [94].

MuCCRA-4 (Fig. 3.6) was designed building upon this strategy and alleviates one major design drawback of MuCCRA-3. Up to and including MuCCRA-3, each PE from input registers to output registers used to be a combinatorial net. All steps from switching context memory entries to data multiplexing, actual computation, and output multiplexing had to be done within a single clock cycle. Apart from resulting in much longer critical paths, this also led to higher switching activity.

Thus Katagiri et al. split the PE structure into five pipeline stages:

1. `IF1` Instruction fetch 1
2. `IF2` Instruction fetch 2
3. `ID` Instruction decode
4. `EX` Execute
5. `WB` Write-back

[96, 97]. The first two stages are introduced to cover instruction, i.e., configuration fetch

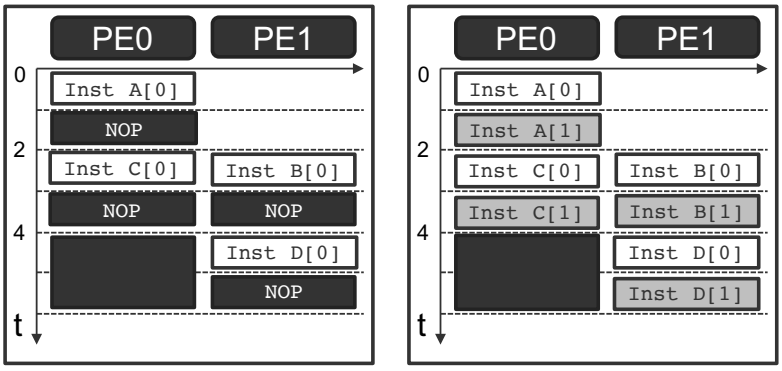


Figure 3.7: MuCCRA-4 pipeline-hazard caused by two staged instruction fetch (left) and solved pipeline-hazard by employing TVI (right)

from the context memories. At clock frequencies at the limit of a chosen operation point, memory accesses become timing critical. Especially when SRAMs are employed, accessing those memories limits attainable clock frequency. With very limited options to speed up this process, instruction fetch is split into two pipeline stages, doubling the time to read a configuration. In instruction decode, the previously read (`IF1` and `IF2`) configuration sets the data paths of the PE. This includes reading from registers as well as setting input multiplexers for the execution stage (`EX`). In the execution stage (`EX`), the actual computation is executed in the ALU. Additionally, the register file may be set to read out a register value as output. Both, the ALU and register file store the result of the instruction in an output register which is then used in the write back stage (`WB`). This stage is used to commit and send data on the interconnect.

This instruction fetch stage split, however, introduces another issue: By allowing two clock-cycles per fetch, a NOP is issued in the pipeline to compensate for the one clock-cycle delay before the instruction is completely fetched. This causes the pipeline to stall for a clock cycle whenever the PE switches contexts as visualized in Fig. 3.7. This not only affects a single PE but continues throughout the PE array and thus constitutes a major issue that had to be fixed in order to exploit the newly introduced pipeline structure. As DRPs usually execute loop kernels, this effect can be minimized by introducing a vector instruction [96, 97]. Here, the authors introduced the so-called tiny vector instruction (TVI).

The TVI exactly counters the stall induced by `IF1` and `IF2` by repeating the executed instruction twice, hence it operates on 2-tuples of data. The additional execution allows the one clock-cycle delay to be used by executing the same computation on another date. To facilitate this, the authors of [96] utilize loop-unrolling (Fig. 3.8).

As visualized in Fig. 3.8, the loop is unrolled by a factor of two, thus exactly fitting the vector size. In respectively marked loops, two data are always processed using the same operation. Thus, the pipeline-hazard is effectively solved and the speed gain achieved through pipelining of the PE can be utilized.

Furthermore, MuCCRA-4 also features multi-threading support. MuCCRA-4 supports the simultaneous execution of two threads. To realize this, the register file is split into two

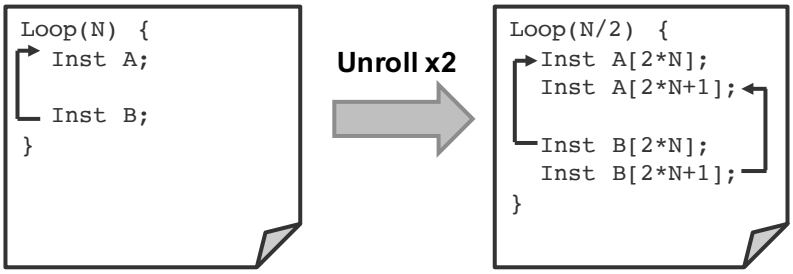


Figure 3.8: Regular execution (left) versus TVI realized through two-fold loop-unrolling (right)

halves when threaded execution is active. Thereby, register renaming is not required at the cost of a reduced number of available registers per thread. The target context during thread change is indicated by an additional thread pointer. Thus, each thread `TH0` and `TH1` has its own pointer. The selection is then done globally for all PEs through a control bit issued by the execution control.

3.4 Overview, Comparison and Contribution

In this state of the art several goals have been pursued. Apart from giving an overview on the latter and showing that no such approach, from either a technological or algorithmic point of view, as well as an architectural point of view, has been researched and evaluated before, two more central aspects have been covered:

1. Putting body biasing in SOI technologies into context with existing power management techniques, thereby allowing a fair comparison

2. Creating a survey on algorithms utilized to implement power management techniques that require some kind of partitioning and thereby indicating the research gap this thesis fills

With respect to the first aspect, it can be concluded, that there are many similarly powerful techniques available. As however, none of them make body biasing in SOI technologies redundant and as on the contrary, body biasing can build on them, enhance their effects or even replace some of them with additional benefits, it is safe to say that body biasing is among the key power management techniques. Whether considering DVFS or clock gating, body biasing complements these techniques in a highly beneficial manner. Power gating, on the other hand, could be replaced with a combination of clock gating and strong reverse body biasing, eradicating the need for long wake-up periods and retention cells. Similarly, in view of the strong effects of body biasing at low supply voltages, multi-V_{DD} design further lost in appeal. The case for multi-V_{TH} design, however, is very intriguing. Body biasing can be seen as a kind of dynamic multi-V_{TH} design where the threshold voltage can be set dynamically at runtime. This does not make actual multi-V_{TH} design superfluous but certainly, can replace it in many cases. Thus, body

Method	Scale Direction	Category	Dynamic	Static	Energy Efficiency	Area
Pre-SOI Body Biasing	-	1, 2	-	✓	- (✓)	-
DVFS	up down	3 3	✗✗ ✓	✗✗ ✓	✗ ✓	✗ ✓
Clock Gating	-	3	✓	- (✗)	✓	- (✗)
Power Gating	-	2	-	✓	✓	✗
Multi-V_{TH}	speed leakage	3 2	✓ -	✗ ✓	✓ ✓	- -
Multi-V_{DD}	-	3	✓	- (✓)	✓	✗
SOI Body biasing w. and wo. DVFS	up down	1, 2, 3 1, 2, 3	✓ ✓	✗✗ ✓	- ✓	✗ ✗
Proposed FGBB	up opt down	1,2 1,2 1,2	✓ - -	✓ ✓✓ ✓✓✓	✓✓ ✓ ✓	- (✗)

Table 3.2: Comparison of the presented power management techniques with the fine-grained body biasing method proposed in this thesis. Legend: ✓✓✓almost ideal, ✓✓very strong improvement, ✓significant improvement, - no improvement nor degradation, ✗degradation, ✗✗strong degradation, (✓/✗) tendency that virtually has no effect

biasing as considered in this thesis is a great extension of the power management repertoire.

In regard to the second aspect, this state of the art clearly indicates the lack of specialized body bias domain partitioning as well as a general method of optimal body bias assignment to such previously partitioned domains.

Also when comparing the key merit figures, static and dynamic power consumption, energy efficiency as well as area overhead, the comparison in table 3.4 further stresses the importance of filling this research gap to exploit such techniques.
This becomes most evident when comparing this thesis' approach to its main contender, the existing approaches exploiting SOI body biasing capabilities with or without DVFS. It might seem bizarre that it makes no difference whether DVFS is exploited or not, but as all approaches in the state of the art stay at core granularity, not further exploring finer granularities, all approaches suffer from either the exponential leakage increase when applying forward body biasing, and the exponential performance drop when applying reverse body biasing. With this thesis breaking with granularities formerly dictated by methods with comparatively huge overheads such as DVFS, fine granular body bias domain partitionings are capable of giving a forward body biasing speed boost while at the same time reducing overall leakage, and hence also constitute a significant improvement over the state of the art. Furthermore, the algorithms proposed in this thesis are also a step forward in the direction of automatization.

4 Problem Formulation

In this section, the underlying problems of leveraging body biasing in a hardware design is defined and elaborated. As introduced in the previous chapters, firstly, there is the need to partition a design into body bias domains during physical design, and secondly, the body bias that needs to be applied to the respective domains varies even within applications that are executed on the design. Thus, there are three major challenges:

1. Once the design is partitioned, it cannot be changed later on (physical partitioning)
2. The design space of possible body bias partitionings is huge (space of partitionings into body bias domains)
3. For a specific partitioning, different body bias assignments are required to achieve the desired result for different applications (body bias assignment and application dependency)

Problem 1. asks for a general way to partition a design into body bias domains. As this thesis aims to find a general partitioning method, no usage profiles are assumed. Every partitioning into body bias domains should be capable of executing applications without special restrictions. Problem 2. points towards the huge design space that is opened up by body biasing. Not only could virtually any standard cell be placed in a different body bias domain to constitute a different partitioning, but the body bias assignment for these domains further enlarge the possible design space by the number of possible body bias assignment combinations as well. This leads to the third problem, body bias assignment. Of all body bias assignments that do not violate timing, obviously, the leakage minimal solution should be used. In the following, these problems shall be put into a mathematical form.

4.1 Mathematical Definition

As described in 2.1.2, body biasing affects both leakage and delay, essentially reducing the body biasing problem to a leakage and delay trade-off. As there is no documented direct effect on dynamic power, the goal is to get the desired reduction of delays for the smallest leakage current possible. This intuitively leads to the formulation of the body biasing problem as leakage minimization. Therefore, for any given clock frequency at a given supply voltage, body biasing is used optimally if of all valid partitionings into body bias domains and their respective body bias assignments, the leakage minimal solution is chosen.

Consider a pipelined hardware design δ consisting of pipeline stages $\delta_k = (C, TP, V_d, V_b)$ with N components $C = \{c_1 \cdots c_N\}$, timing paths $TP = \{tp_1 \cdots tp_M\}$ where $\forall \{tp_i | tp_i \subseteq C\}$, a set of Q available supply voltages $V_d = \{V_{DD_1} \cdots V_{DD_Q}\}$ $[V]$ and a set of R available body biases $V_b = \{V_{BB_1} \cdots V_{BB_R}\}$ $[V]$. On such design δ, an application $A = (a_1 \cdots a_Z)$ is executed with instructions $a_i \subseteq TP$. Then there is a function $CL(p, c, V_{DD}, V_{BB})$ that determines for each domain p and the contained parts of component c, a specific supply voltage V_{DD} and body bias V_{BB} its respective leakage current. Then there is also $CD(p, c, V_{DD}, V_{BB}, a)$ which determines for each domain p and the parts of component c in it , at a given supply voltage V_{DD} and body bias V_{BB} the delay incurred when instruction a is executed. Each component is capable of executing at least one instruction. If the component is not involved in the realization of an instruction, its delay is zero.

Then, a valid body bias partitioning into k disjunct domains $P = (P_1 \cdots P_k), P_i \subseteq C$ with a body bias assignment $VBA = (VB_1 \cdots VB_k)$ with $VB_i \in V_b$ is valid, if

$$\forall a_g \in A \; \forall tp_i \in a_g \sum_{p_j \in P} \left(\sum_{c_k \in p_j} CD(p_k, V_{DD}, VB_j, a_g) \right) \leq t_{clk} \tag{4.1}$$

applies. Eq. 4.1 computes for each instruction used in the application executed on the design in question the delay incurred throughout the timing paths possibly leading through multiple domains and components. Of these components, however, only the actual delay, i.e., the parts of a component realizing instruction a_g is accounted. Thus if all instructions are executed in less or equal time than the clock period t_{clk}, timing is not violated, and the partitioning is valid.

Thus, of all valid partitionings and body bias assignments

$$I_{Leak} = \sum_{i=1}^{k} \left(\sum_{c_j \in p_i} CL(p_i, c_j, V_{DD}, VB_i) \right) \tag{4.2}$$

leakage current I_{Leak}, i.e., the sum of all components' leakage in all domains i at body bias VB_i should be minimal. For the exhaustiveness of this chapter, this problem definition can be extended to capture the aspects of the dynamic application of body biasing, which requires the following auxiliary definitions. Let $BC(p, C, \vec{V_{BB}})$ define the charge required to switch the body potential of the parts of C in p from $V_{BB,1}$ to $V_{BB,2}$ with $\vec{V_{BB}} = (V_{BB,1}\ V_{BB,2})$. Let additionally $BC(p, \vec{V_{BB}})$ define the transistor body charge required when switching the body bias of domain p from $V_{BB,1}$ to $V_{BB,2}$. Both $BC(p, C, \vec{V_{BB}})$ and $BC(p, \vec{V_{BB}})$ shall also account for the energy dissipated while charging and the loss due to supply effectiveness.

Then we can define the dynamic body biasing problem using

$$E_{Leak} = t_{clk} \cdot V_{DD} \cdot \sum_{a \in A} \sum_{p \in P} \sum_{c \in p} CL(p, c, V_{DD}, VB_{pa}) \tag{4.3}$$

with t_{clk} the clock cycle in seconds, supply voltage V_{DD}, instruction a of program A, domain

p of all domains P, parts of a component c in p and the body bias assigned to domain p at the execution of instruction a written as VB_{pa}. E_{Leak} thus is the leakage energy incurred through leakage throughout the execution of A with possibly changing body biases VB_{pa}. However, this is only part of the dissipated energy, as switching body biases also requires energy which is defined as

$$E_{BC} = \sum_{i=1}^{Z-1} \sum_{j=1}^{k} BC(p_j, (V_{BB,i}, V_{BB,i+1})) \tag{4.4}$$

the energy dissipated when switching body biases of k domains p_j during the execution of Z instructions. Then, the actual dynamic body biasing problem is defined as

$$\min E_{Total} = E_{Leak} + E_{BC} + E_{BC,static} \tag{4.5}$$

with $E_{BC,static}$ as the energy dissipated by the components supplying the charge required to switch the body bias in standby. Thus, the dynamic body biasing problem is the minimization of dissipated energy due to leakage currents and the energy required to change to affected transistor bodies throughout the execution of A. For a valid and realistic consideration of this problem, however, we have to consider the transition times required to facilitate the $\vec{V_{BB}}$ transition as well. With experimental data supplied by Okuhara and Amano in [14], it is clear that transition times of several hundred microseconds are beyond the temporal granularity focused on in this thesis for dynamically reconfigurable architectures. Thus, beyond this problem definition, dynamic body biasing will be only covered in a reduced form, called programmable body biasing, which will be introduced below.

4.2 Body Biasing Categories

Body biasing can be categorized by its temporal application, which in turn differs in versatility. Like multi-V_{TH} approaches, body biasing can be applied statically, fixed at design time. In the case of body biasing, however, the actual V_{TH} may still be changed depending on the supply of the body bias potential. Then there are two further overlapping ways to apply body biasing: programmable and dynamic body biasing. Both are comparable to DVFS, their actual distinction, however, is body biasing specific and thus shall be elaborated in detail in this section.

4.2.1 Static Body Biasing

With static body biasing, body bias domains are defined at design time like every other body biasing scheme as well. However, the body bias supply is physically connected to a single supply net, and thus it cannot be changed later. This means that all domains connected to this supply net will always have the same body bias applied. While it is still possible changing the potential on the supply net at the voltage source, this is not considered in this definition. Thus, static body biasing is defined as

$$\underset{A}{\forall}\ \underset{a\in A}{\forall}\ \underset{p\in P_{Static,i}}{\forall}\ :\quad VB_{pa} = V_{Supply,i} \tag{4.6}$$

with A the set of all applications, $P_{Static,i}$ the i-th set of static domains connected to a supply net supplying $V_{Supply,i}$. Thus, over all applications and the instructions executed therein, the body bias of the considered domains never changes.

4.2.2 Programmable Body Biasing

Opposed to static body biasing, programmable body biasing allows to change the bias of specific domains within certain time limits. This change of body bias is best characterized in reconfigurable architectures. When a reconfigurable architecture shall change its implemented function, this usually requires external reconfiguration which usually takes a period of the microseconds to milliseconds order of magnitude.
Similarly, programmable body biasing is defined as a change in body bias between two points in time t_i and t_{i+1} that conform to a lower bound $t_{Programmable}$. Thus, first of all, programmable body bias allows for different biases, however, within small applications or application blocks A_l, the bias may not change.

$$\underset{p\in P}{\forall}\ \underset{a_i\in A_l}{\forall}\ \underset{a_j\in A_l, i\neq j}{\forall}\ :\quad VB_{a_i p} = VB_{a_j p} \tag{4.7}$$

Thus Eq. 4.7 expresses, that all instructions a_i of an application A_l are executed under the same bias VB_{ap}. Between applications A_l and A_{l+1}, the bias may change if either one runs at least $t_{Programmable}$ seconds. Thus, between each application of a set of applications A, body biases may be changed if the following constraint

$$\underset{A_l\in A}{\forall}\ t_l - t_{l+1} \geq t_{Programmable} \tag{4.8}$$

is fulfilled. This allows a simplification in regard to body bias switching costs $E_{BC} + E_{BC,static}$ and gives a bound for $t_{Programmable}$. If $t_{Programmable}$ is large enough, then for switching between two body biases $V_{BB,l}$ and $V_{BB,l+1}$ between two applications

$$\Delta E_{Leak} = \Delta t \cdot \sum_{c\in p} \left(CL(p,c,V_{DD},V_{BB,l}) - CL(p,c,V_{DD},V_{BB,l+1})\right) \gg E_{BC} + E_{BC,Static} \tag{4.9}$$

ΔE_{Leak}, the leakage saving when actually switching body biases is much greater than the cost of switching. Furthermore, with at least $t_{Programmable}$ between each body bias switch, the standby cost $E_{BC,Static}$ can be significantly lowered. With regard to Eq. 4.9, E_{BC} and $E_{BC,Static}$ are thus neglected for programmable body biasing.

4.2.3 Dynamic Body Biasing

Dynamic body biasing allows changing body biases as long as the switching can be realized fast enough by the appropriate hardware. This also means that dynamic body biasing allows for switching strategies that are detrimental to power consumption. Thus, the main

challenge is to determine switching strategies that minimize power consumption as defined in Eq. 4.5. Thus, for any point in time t_l and t_{l+1}, it is worth to switch from bias $V_{BB,l}$ to $V_{BB,l+1}$ for domain p if

$$\Delta E_{Leak} > BC(p,(V_{BB,l},V_{BB,l+1})) \tag{4.10}$$

with ΔE_{Leak} defined as in 4.9 without further considering $E_{BC,Static}$ as for dynamic body biasing, voltage regulators realizing the bias switch are required in all cases. Thus, the charge required to change the body bias $BC(p,\vec{V_{BB}})$ is essential. Without this information disclosed by semiconductor companies and in light of the static scheduling of dynamically reconfigurable processors, this strategy is not further pursued in this thesis, but still remains an important definition to delimit the extent of programmable body biasing.

4.3 Partitioning Problem

Another problem of partitioning a design δ into body bias domains is the sheer possible number of partitionings. To elucidate this, Eq. 4.11

$$|\mathcal{P}_\delta| = S(N,k) \tag{4.11}$$

computes the cardinality of the set of all possible partitions $\mathcal{P}$ of a design δ composed of N individual components, partitioned into k non-empty subsets. This is done using Stirling numbers of second order which can be computed as

$$S(N,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}i^N \tag{4.12}$$

.

To find the leakage minimal partition $P \in \mathcal{P}$, all partitions have to be evaluated for the resulting leakage. While this problem cannot be solved differently, the search space can be significantly reduced by looking for a fitting reduction of N the number of components to be partitioned into body bias domains.

4.4 Optimization Target

The problems introduced in section 2.1.2, specifically sections 2.1.2.2 and 2.1.2.4, require considerable efforts to be solved. However, the benefits that can be achieved by solving these problems are at least equally desirable. The problems can be reduced to a leakage and delay trade-off, where reduced delay and thus higher achievable clock frequencies are attained by allowing a certain increase in leakage, i.e., static power consumption. Within limits, this trade-off is highly beneficial as it allows to virtually cut off leakage currents in idle and gives significant speed boosts without increasing supply voltage V_{DD}, which is in a quadratic relationship with dynamic power.

For reasons of simplicity, this thesis focuses on three corner cases for a given supply voltage V_{DD} and a respectively attainable maximum clock frequency F_{max}:

1. Max. RBB - Maximum reverse body bias with F_{max} scaled down to $F_{max,RBB}$
2. Low Power (LP) - F_{max} unchanged, optimized leakage
3. High Performance (HP) - F_{max} scaled to $F_{max,FBB}$ using forward body biasing

These corners are further separated into an idle mode and active modes. Idle modes aim to minimize leakage irrespective of other goals, while active modes seek to maximize energy efficiency by minimizing leakage. Max. RBB is an idle mode, thus lowers the clock frequency enough for all components to meet timing even at maximum reverse body bias. Any automatized method to determine body bias domains will report that it is not beneficial to partition a design into more than one domains when targeting Max RBB. Thus, to partition a design into body bias domains, this corner is unsuitable. Therefore, body bias domain partitionings are determined using one of the active corners, while for idle modes, all domains in the resulting partitioning are applied a maximum reverse body bias. In contrast to power gating, however, the design can still operate, although at lower clock frequencies. If this is not required, power consumption can be reduced to static power consumption at maximum reverse body bias, if clock gating is applied. This, however, is not covered in this thesis.
In contrast, active modes seek to minimize leakage while reaching a given clock frequency, thus maximizing energy efficiency. When such a mode is specified as the optimization target, the presented approaches will search for a leakage minimal distribution of components or standard cells into body bias domains with respective body bias assignments. This is done independently of the applications which may later be executed on the design. After partitioning, the resulting partitioned design can then be evaluated for specific benchmark applications.

5 General Body Bias Domain Partitioning Approaches

This chapter discusses general body bias domain partitioning principles and subsequently derived approaches not operating on standard cell granularity. This distinction is essential, as it allows to focus on the actual principles behind body bias domain partitioning and keeps combinatorial problem sizes within feasible limits. Thus, the following approaches operate on abstract components, e.g., adders, multipliers etc. These are in turn of course realized in standard cells. Using this abstraction, the approaches in this chapter need not examine every standard cell that realizes the respective functionality, but can treat these abstract components as clusters of standard cells, significantly reducing the problem size.

This restriction, of course, comes at a cost, as, first of all, it does not cover all conceivable body bias domain partitionings and it limits the degree of synthesis optimizations that can be applied to the component to be partitioned. In the following, basic partitioning principles are introduced which also justify the use of such abstraction, while core- and coarse-grained body biasing approaches introduce intuitive methods to partition a design using its inherent structure, with particular regard to dynamically reconfigurable processors. This, in turn, also motivates further, more fine-grained partitionings which are introduced at the end of this chapter. With fine-grained body biasing, the combinatorial k-subset approach is also introduced. This the basic algorithm to merge multiple components or domain candidates into a limited number of k body bias domains.

This chapter is based on [98–101], presented at the International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (ISHEART) in 2014, Design Automation and Test Conference in Europe (DATE) in 2015, Synthesis and System Integration of Mixed Information Technologies workshop (SASIMI) in 2015, and the IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips) in 2015 respectively.

5.1 Basic Partitioning Principles

All partitioning strategies are based on two principal characteristics: Timing and activation. Timing refers to the actual time a signal needs from input to output to reach a final state. As body bias affects timing, it determines how much reverse body biasing can be applied or how much forward body bias may be needed. Timing also concerns another aspect which becomes more apparent when examining partitioning on the standard cell level: if the application of body bias to one standard cell affects the timing of another standard cell, they are connected. Take, for example, an N-Bit adder as depicted in figure

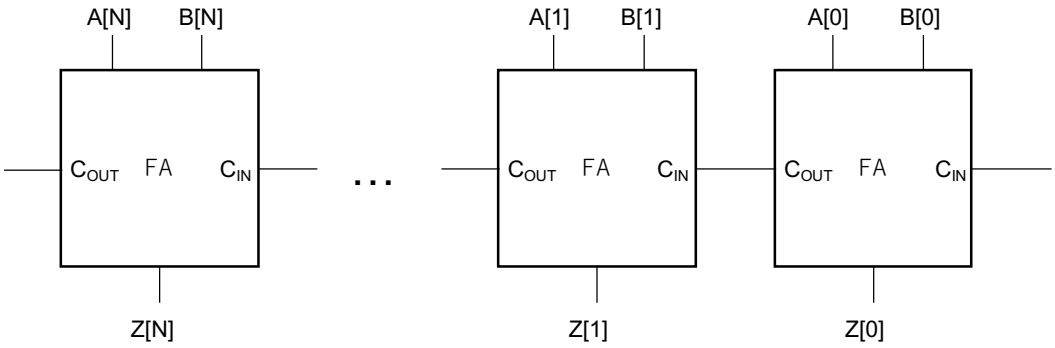


Figure 5.1: N-Bit adder comprised of N full-adder cells

5.1. The C_{Out} signal of the lower bit is connected to the C_{In} of the next higher bit's full-adder. The result $Z[k]$ of bit k is dependent on the $k-1$ bit's C_{Out} and thus also of every further lower bit's full-adders and their inputs. This also means that, if the addition in Fig. 5.1 should be sped up using body biasing, the whole component consisting of multiple components should be treated as one entity.

Activation, on the other hand, refers to activation patterns at different points in time. For example, if it can be determined that when component A is used, component B is always unused, it might make sense to put those two components into different BBDs.
The goal of each method should be to use forward body bias only when and wherever it is needed and to apply reverse body bias as broadly as timing allows.

5.2 Core-Grained Body Biasing

Core-grained body biasing is the most straightforward application in regard to partition granularity. A core denotes one functionally independent entity, i.e., a component which can operate independently once connected to the infrastructure required to execute its function, such as a bus, and once it has been able to receive the required data for execution. In GPCPUs, the actual CPU core plus required caches and its interconnect hardware would constitute such a core. In regard to DRPs, the whole DRP including IO controller, execution control data, and context memory represent a core. Furthermore, a core has only one clock domain. Thus, the characterizing aspect of core-grained body biasing is, that body biases have to be chosen in a way that all contained components can operate at a specified operating condition given as supply voltage V_{DD} and maximum clock frequency F_{max}. An obvious advantage is the reduced complexity of only one body bias domain. However, the applicability of reverse body biasing is limited to rare cases where only paths with positive timing slack are utilized. Furthermore, core-grained body biasing results in large leakage penalties as all components within a core, irrespective of necessity, are forward biased. As outlined in in the paragraph on leakage and its relationship with body biasing (2.1.2.2), this is limiting the application of forward body bias. With core-grained

body biasing, the threshold where forward body biasing becomes prohibitive, i.e., where supply voltage scaling is more efficient, is rather low.

In core-grained body bias, body bias domain partitioning is not required, as the core itself constitutes the body bias domain. Thus, to use body biasing, only the body bias voltage V_{BB} has to be computed, from which then the actual body potentials V_{BP} and V_{BN} are computed (see 2.1.2.4).

In DRPs, this means that the body bias has to be set according to the most timing critical operation conducted during the execution of an application in all of its PEs, controllers or other hardware components. Thus, the following algorithm is used for a given supply voltage V_{DD}, maximum clock frequency F_{max} and application $A = \{a_1 \cdots a_Z\}$ to find the minimal global $V_{BB,core}$ which does not violate timing:

Require: V_{DD}, F_{max}, VB, C
Ensure: $V_{BB,core}$ is the minimal V_{BB} not violating timing

```
1:  t_clk ← F_max^-1
2:  V_BB,core ← min(VB)
3:  for all c ∈ C do
4:      for all a ∈ A do
5:          timingViolation ← false
6:          for V_BB,c ← V_BB,core; V_BB,c ≤ max(VB); V_BB,c ← nextVBB(V_BB,c) do
7:              if CD(core, V_DD, V_BB,c, a) ≤ t_clk then
8:                  breakLoop
9:              else
10:                 if V_BB,c == max(VB) then
11:                     timingViolation ← true
12:                 end if
13:             end if
14:         end for
15:         if timingViolation == true then
16:             Fail
17:         end if
18:         if V_BB,c > V_BB,core then
19:             V_BB,core ← V_BB,c
20:         end if
21:     end for
22: end for
```

with $V_{BB,maxRBB}$ the maximum available reverse body bias. First, in line 2 $V_{BB,core}$ is set to maximum reverse body bias, that is the numerically minimal value as determined by $\min(VB)$, as the aim is to use as much reverse body bias and as little forward body bias as possible. Then, in lines 3 and 4, the algorithm iterates over all components c of C and all operations a of A. In line 5, a timing violation flag is declared, first assumed to be false. For each components' operations, the component timing CD is then checked at a body bias

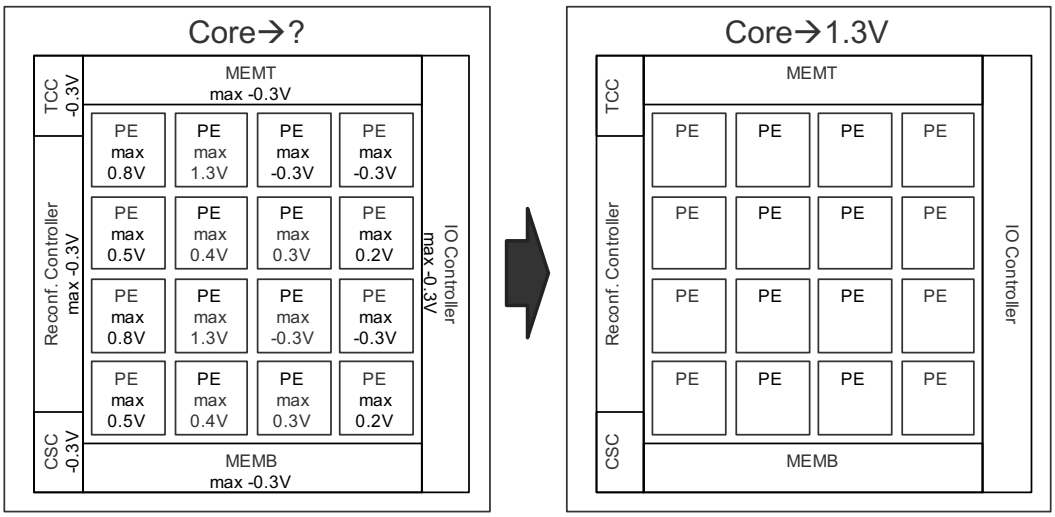


Figure 5.2: Core-grained body bias determination for a 4×4 DRP and maximum body bias per component incurred by the executed operations already determined

of $V_{BB,core}$. If once one previous operation of any component caused $V_{BB,core}$ to be set at any body bias greater than maximum reverse body bias, a body bias less than the $V_{BB,core}$ in the current iteration cannot be chosen without violating timing. However, if in line 7 the components' timing for the given operation is less than the clock period t_{clk}, timing is met. Thus, in line 8 the loop is broken, and the next operation a or the next component c can be examined. If, however, the timing is not met, first it has to be verified if the timing for the present component and operation can be met at all by checking if the current body bias $V_{BB,c}$ is less than the maximum available forward body bias. If it is, the next higher V_{BB} will be used to see if the timing is met. If it is not, timing cannot be met and the algorithm fails in line 16. If the algorithm hasn't failed yet, whether timing has been met yet or not, if $V_{BB,c}$ is greater than the global core body bias $V_{BB,core}$, it should be updated accordingly in line 19. Once the algorithm iterated through all components and its operations without failing, the global body bias $V_{BB,core}$ applied to the entire core is determined.

Fig. 5.2 visualizes the principle function of this algorithm. The left core depicts a DRP with maximal incurred body bias per component already determined (usually done in lines 4-7) for demonstration purposes. The goal of the entire algorithm is to find the most timing critical path and thus also the greatest body bias required in the entire core. In this example, the greatest body bias is a V_{BB} of $1.3V$ in the 2nd PE (counting from left to right, top to bottom) and in the 10th PE. Therefore, the entire core is assigned a body bias of $1.3V$ forward body bias (Fig. 5.2 right). However, as stated above, only two PEs need such strong forward body bias. This saving potential is leveraged using the following body biasing approaches.

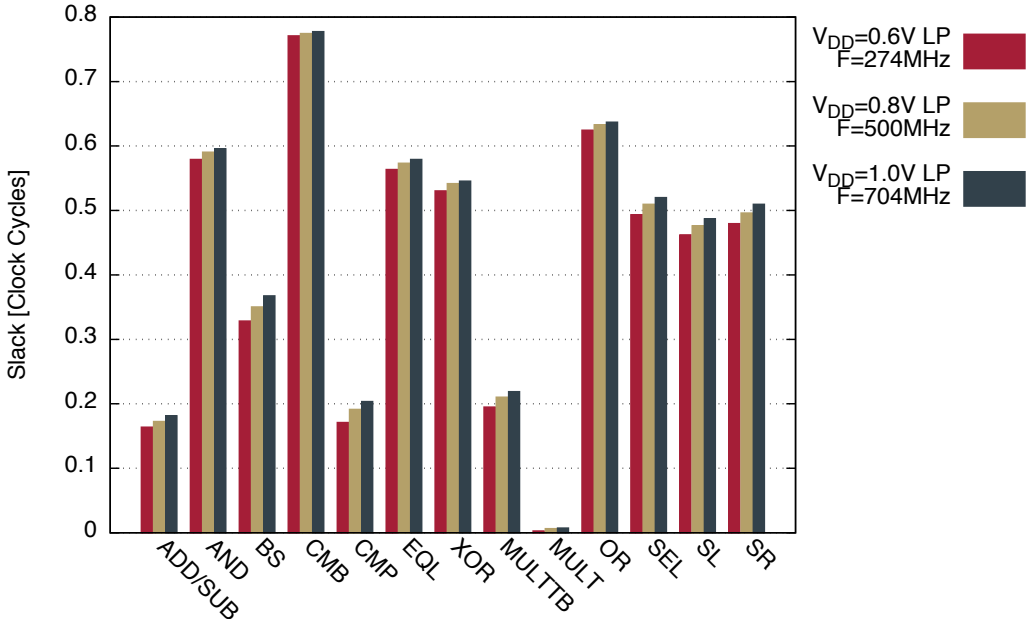


Figure 5.3: Operation dependent timing slack for STMicro's 28nm UTBB-FDSOI LVT flavor at supply voltages $0.6V$, $0.8V$ and $1.0V$ and the LP corner, reaching clock frequencies of $274MHz$, $500MHz$ and $704MHz$ without forward body biasing accordingly

5.3 Coarse-Grained Body Biasing

Coarse-grained body biasing is an evolution of core-grained body biasing by allowing much more locally focussed biasing. One of the characterizing aspects of coarse-grained body biasing is that, similar to core-grained body biasing, the components within one body bias domain must be able to execute the intended function with the required body bias applied. Thus, while components might depend on other components in different body bias domains to perform their function, timing dependencies must be restricted to the body bias domain, i.e. cross-domain timing dependencies are excluded.

In case of DRPs, an intuitive partitioning into body bias domains which observes this definition is to put each PE or groups of several PEs in one body bias domain. While a single PE or a group of PEs depends on additional functionality such as an execution controller to supply a context pointer, the actual functionality implemented in the PEs will be executed correctly if, under a given body bias, all used timing paths observe timing constraints. For this general definition, PEs are assumed to communicate via clocked registers. Chaining of several PEs to form a larger combinatorial circuit is omitted as well but will be treated as a special case below.

Like with core-grained body biasing, independent of the corner, leakage should be minimized by using only as much forward body bias as absolutely necessary. With the additional degree of freedom of coarse-grained body bias to have different body biases for

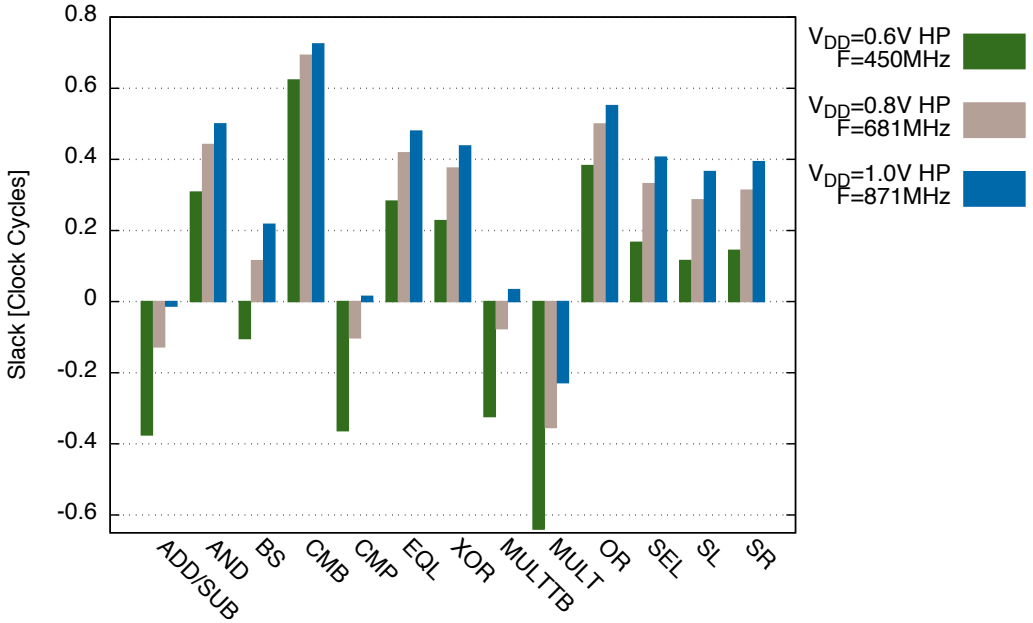


Figure 5.4: Operation dependent timing slack for STMicro's 28nm UTBB-FDSOI LVT flavor at supply voltages $0.6V$, $0.8V$ and $1.0V$ and the HP corner, reaching clock frequencies of $450MHz$, $581MHz$ and $871MHz$ with forward body biasing accordingly

each domain, the usage of reverse body bias should be maximized while forward body bias should be applied only wherever and only by as much as absolutely necessary.

The actual bias per domain, i.e., PE in the specific case of coarse-grained body biasing in DRPs, is determined based on application dependent timing slack. This timing slack is dependent on the corner the PE is being used in, thus Fig. 5.3 depicts the timing slack per ALU operation given for the LP corner, while Fig. 5.4. depicts the same for the HP corner. As can be seen, the HP corner targets much higher clock frequencies but then also incurs negative slack for some operations. Thus, positive slack can be used to apply reverse body biasing, while negative slack has to be compensated for using forward body biasing. This requires a relationship between body biasing and timing slack.

For this purpose, the function *RSF* (Required Slack Function) and function *PSF* (Per operation Slack Function) are introduced. *RSF* is defined per supply voltage V_{DD} and clock frequency F and returns per body bias voltage V_{BB} the required slack in nanoseconds that is required or can be compensated. For reverse body bias, positive timing slack is required which can then be exploited to reduce leakage currents, while negative timing slack, i.e. timing violations require forward body bias to speed-up the affected circuits and thus compensate for negative slack at the cost of increased leakage currents. *PSF*, on the other hand, takes the data plotted in Figs. 5.3 and 5.4 and returns per operation OP and clock frequency F the timing slack in nanoseconds.

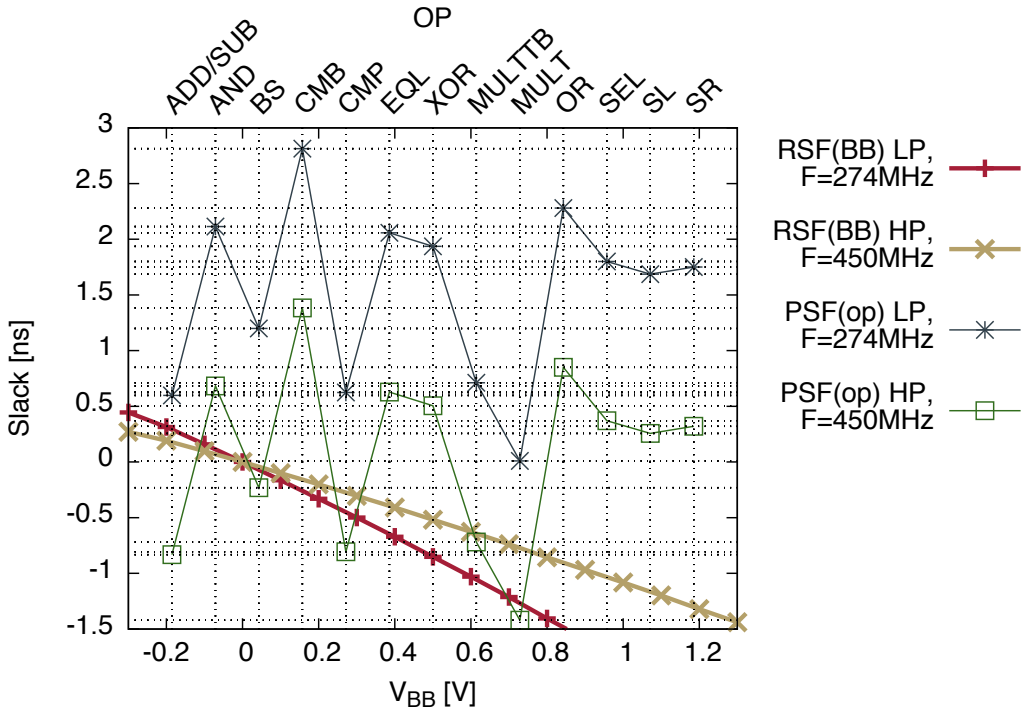


Figure 5.5: *RSF* and *PSF* functions for corners LP and HP at a supply voltage of $0.6V$.

Fig. 5.5 exemplarily shows both *RSF* and *PSF* for the LP and HP corners and their respective clock frequencies at a supply voltage of $V_{DD} = 0.6V$. To find the appropriate body bias per PE and application, the operations executed on each PE have to be examined, and the most timing critical operation executed, i.e., the operation with the smallest *PSF* op_{crit}, in the course of the application is noted. For this operation op_{crit} the intersection between $y = PSF(op_{crit})$ and *RSF* for a given corner is searched and the body bias required to avoid timing problems is found. Graphically put, for all used operations of a PE in an application, *PSF* has to be above *RSF* of the same corner. If this condition is met, timing is not violated.

This leads to the following algorithm which is executed for each PE's application mapping as visualized in Fig. 5.6:

Require: A per PE, VB, *RSF* and *PSF* for a given corner
Ensure: $V_{BB,PE}$ is assigned the leakage minimal body bias not violating timing or fail

```
1: for all PE do
2:     t_minslack ← ∞
3:     for all a ∈ A_PE do
4:         if PSF(a) < t_minslack then
5:             t_minslack ← PSF(a)
6:         end if
```

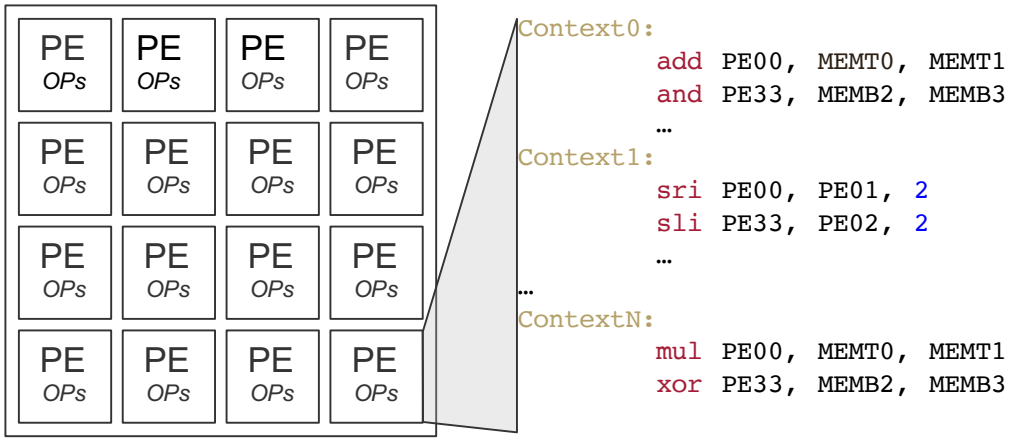


Figure 5.6: Per PE determination of the appropriate body bias using the application's instructions executed on the PE in question

7: **end for**
8: $V_{BB,PE} \leftarrow \text{solve}(RSF(X) \leq t_{minslack}, X)$
9: **if** $RSF(\min(VB)) < t_{minslack}$ **then**
10: $V_{BB,PE} \leftarrow \min(VB)$
11: **end if**
12: **if** $RSF(\max(VB)) > t_{minslack}$ **then**
13: Fail
14: **end if**
15: **end for**

where A_{PE} denotes the operations executed in a specific PE during the execution of application A and VB the set of available body biases. For each PE, in lines 1 to 7, the algorithm then searches the operation with the minimal slack and stores the actual minimal timing slack in $t_{minslack}$. Using this minimal timing slack, the appropriate body bias is searched by computing the intersection of RSF and $y = PSF(op_{crit}) = t_{minslack}$ in line 8. Lines 9 to 11 then check if the executed operations are timing uncritical enough for the strongest reverse body bias to be applied or if the PE is unused altogether ($t_{minslack} = \infty$). In this case, the strongest reverse body bias available ($V_{BB} = \min(VB)$) will be supplied. If, however, no body bias could compensate for $t_{minslack} < 0$, which is the case when $t_{minslack}$ is smaller, i.e., more negative than $RSF(\max(VB))$, then the algorithm should fail (lines 12-14). In simple terms, $RSF(\max(VB))$ denotes the maximum negative slack that can be compensated when using the maximum available forward body bias ($\max(VB)$).

5.4 Fine-Grained Body Biasing

Fine-grained body biasing is another evolution on core- and coarse-grained body biasing. Its principal difference is that with fine-grained body biasing, cross-domain timing dependencies are allowed. This enables designs to be partitioned in virtually every

```
1  module ALU (
2    INPUT_A,
3    INPUT_B,
4    INPUT_CARRY,
5    ALU_OP,
6    ALU_OUT
7  );
8  ...
9
10 always @(*) begin
11   case(ALU_OP)
12     `ADD  : ALU_OUT = INPUT_A + INPUT_B + INPUT_CARRY;
13     `MULT : ALU_OUT = INPUT_A * INPUT_B;
14     ...
15   endcase
16 end
17
18 endmodule
```

Figure 5.7: Example ALU before pre-partitioning into subcomponents with all functionality realized in one verilog module

possible way. With coarse-grained body biasing, the domain size is limited by the smallest timing independent components, i.e. in the case of DRPs the FU i.e., ALU, register-file etc. Fine-grained body biasing allows much smaller body bias domains as it does not restrict timing paths to a single domain. Thus, domains down to a single standard cell are conceivable, although they are not sensible in reality. The limiting problem with fine-grained body biasing is pre-partitioning and the thereby arising complexity. pre-partitioning defines the partitioning step of a component, i.e., a verilog module, into sub-components that form the original component. This is critical to make full use of the cross-domain resource sharing constraint elimination. While finer pre-partitionings into subcomponents possibly result in better domain partitionings, they are ultimately limited by the size of the partitioning problem. As explained in section 4.3, the partitioning problem quickly becomes too big to be solvable. This pre-partitioning is done by hand on the RTL into finer functional subcomponents. e.g., an ALU is split up by the functions it implements where each function constitutes a new subcomponent.

Consider the minimal verilog HDL example of an ALU in Fig. 5.7. In lines 12 and 13, addition and multiplication functionality is directly invoked from within the ALU module. Once the ALU is synthesized, the different functionality it implements can no longer be distinguished in the mapped netlist. Thus, to enable partitioning into body bias domains beyond the ALU module, it is split up into subcomponents. This can be done by identifying the different functions implemented or inferred in the HDL. In case of Fig. 5.7, there is an addition and a multiplication using data inputs `INPUT_A`, `INPUT_B` and `INPUT_CARRY`, and a multiplexer outputting the respective result depending on the value of `ALU_OP`. Each of these functionalities is then outsourced into separate HDL modules. Fig.

```
1  module ALU ( INPUT_A, INPUT_B, INPUT_CARRY, ALU_OP,
2    ALU_OUT );
3  ...
4
5  wire [`WORD_SIZE-1:0] ADD_OUT;
6  ADD add0 (.A(INPUT_A), .B(INPUT_B), .C(INPUT_CARRY),
7        .OUT(ADD_OUT));
8
9  wire [`WORD_SIZE-1:0] MULT_OUT;
10 MULT mult0 (.A(INPUT_A), .B(INPUT_B), .C(INPUT_CARRY),
11       .OUT(MULT_OUT));
12 ...
13 ALU_MUX mux (.ADD_OUT(ADD_OUT), .MULT_OUT(MULT_OUT), ...,
14        .ALU_OP(ALU_OP), .ALU_OUT(ALU_OUT));
15
16 endmodule
17
18 module MULT (A, B, C, OUT);
19 ...
20 assign OUT = A * B;
21 endmodule
22
23 module ADD (A, B, C, OUT);
24 ...
25 assign OUT = A + B + C;
26 endmodule
27 ...
28 module ALU_MUX (ADD_OUT, MULT_OUT, ..., ALU_OP, ALU_OUT);
29 ...
30 endmodule
```

Figure 5.8: Example ALU RTL pre-partitioning of Fig. 5.7 into subcomponents.

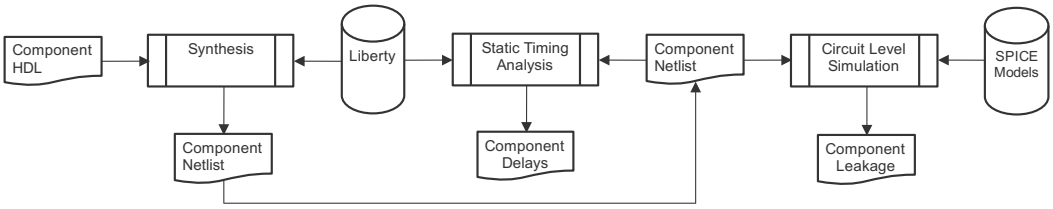


Figure 5.9: Characterization flow of pre-partitioned (sub)components

5.8 exemplarily demonstrates this outsourcing. New modules `ADD`, `MULT` and `ALU_MUX` are introduced and instantiated in place of the behavioral description found in Fig. 5.7. These (sub)components then can be synthesized and characterized individually to determine attributes, such as their timing and leakage. This process is illustrated in Fig. 5.9.
Each of these (sub)components is first synthesized using standard cell information provided by the foundry or the standard cell provider (labeled Liberty, format may differ). This yields a component netlist, i.e., a mapped standard cell netlist which is then used in a static timing analysis tool to determine its timings, i.e., the delays incurred by using the component. Furthermore, the mapped netlist is converted to a SPICE netlist to determine body bias dependent attributes such as leakage. While foundries began to include additional characterization points with body biases, it usually covers only between one and a few body biases. Thus, to cover a whole range of body biases, this additional step is required. Additional body bias dependent timing analysis is not conducted, as timings are scaled merely using normalized delay functions such as in section 2.1.2.4 Fig. 2.5.

The pre-partitioning and the characterization steps thus result in a detailed description of a component to be partitioned into body bias domains as well as detailed timing and leakage information for all the subcomponents. Using this information, body bias domain partitionings can be conducted. For this initial partitioning it is assumed that all components and all their timing paths have to be timing violation free, i.e., all components have to be usable. This is a constraint which is later on relaxed to require only those paths that are being used to be free of timing violations.
To approach the partitioning algorithmically, the set of timing paths TP introduced in chapter 4 has to be converted to a timing graph. This is done by creating a graph per timing path tp_i and then merging all such graphs with shared nodes. This is done by eliminating duplicate vertices, retaining one shared vertex and redirecting all edges which previously pointed from or to their respective duplicates from or to the now shared vertex as illustrated in Fig. 5.10.
Using this timing graph TG, the optimal partitioning into k_{best} body bias domains can be computed. In this step, the timings of (sub-)components are individually examined and the body bias for the target timing constraint t_{clk} computed. Since VB is a discrete set of the body bias voltages available, multiple (sub)components may be assigned the same body bias. These components are then merged to form a domain candidate. All these k_{best} domain candidates form their own body bias domain with the optimal body bias, that is,

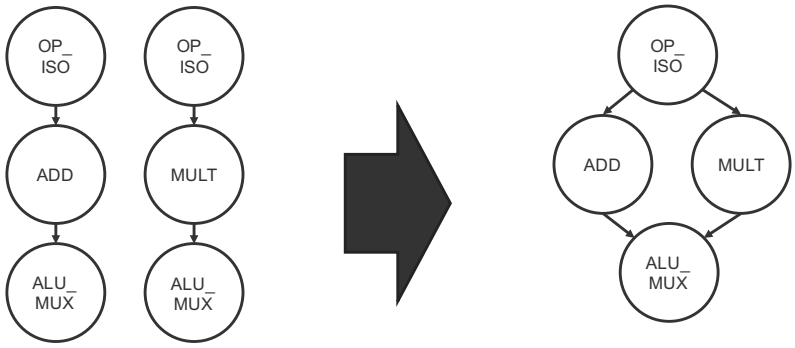


Figure 5.10: Conversion of the timing paths of example ALU of Fig. 5.8 to a timing graph with added operand isolation `OP_ISO` for better illustration.

timing is just met, applied. This, however, also raises the question how the optimal body bias assignment is found. In previous works such as [99] and [101] which build upon the latter, this is done by exhaustive search. As multiple body bias assignments on a timing path may observe timing constraints and thus are valid assignments, each valid assignment needs to be evaluated for leakage[1]. Then the leakage minimal assignment is chosen.

The following algorithm computes the k_{best} domain candidates and their optimal body bias assignment:

1: **for all** $G \in TG$ **do**
2: $minLeak \leftarrow \infty$, $best \leftarrow null$
3: $maxTP \leftarrow searchCriticalTimingPath(G, V_{BB} = 0V)$
4: **for all** $curVBA \in generateVBA(VB, maxTP)$ **do**
5: **if** $computeTiming(curVBA) \leq t_{clk}$ **then**
6: $leak \leftarrow computeLeakage(curVBA)$
7: **if** $leak < minLeak$ **then**
8: $minLeak \leftarrow leak$
9: $best \leftarrow curVBA$
10: **end if**
11: **end if**
12: **end for**
13: $VBA_G \leftarrow VBA_G \cup best$
14: **end for**
15: $VBA_G \leftarrow mergeEqualV_{BB}(VBA_G)$

For each timing graph G in TG, the minimal leakage body bias assignment *best* is computed. The body bias computation is based on the critical timing path in G, determined in line 3. Body bias assignments are defined as a tuple $VBA = (C_{DC}, V_{BB,DC})$, with C_{DC} as the set of components in the particular domain candidate and $V_{BB,DC}$ as the assigned body bias. The tuple incurring the least leakage while maintaining the timing constraint is then computed in lines 4 to 12 by iterating over all combinations of available body

[1] In chapter 6, an algorithm with polynomial runtime will be introduced.

biases VB applied to the critical timing path in $G\ maxTP$. Thus, every conceivable body bias assignment *curVBA* is checked. Of course, it only makes sense to proceed if this combination does not violate timing, thus, in line 5, the resulting timing is checked against timing constraint t_{clk}. If the combination does not violate timing and is thus valid, the resulting leakage for the combination is computed (line 6). If leakage has been reduced over previous iterations, then minimum leakage combination *best* with leakage *minLeak* is updated accordingly in lines 8 and 9. Once all valid combinations have been considered, the best body bias assignment *best* for *maxTP* is merged with the set of domain candidates VBA (line 13). Once for all $G \in TG$ the best body bias assignment has been found, VBA entries $VBA_i \in VBA$ and $VBA_j \in VBA$ are merged to a new tuple $(C_{DC,i} \cup C_{DC,j}, V_{BB})$, if V_{BB} of VBA_i and VBA_j are equal (line 15). Thus, after the completion of this algorithm, $k_{best} = |VBA|$ has been computed.

Usually, k_{best} is far bigger than the number of body bias domain k into which the hardware developer intends to partition a given design. Therefore, leakage increase over the optimal partitioning of k_{best} body bias domains or domain candidates is traded to compute a partitioning with only k body bias domains. In the following subsection, a general approach to find the leakage minimal distribution of k_{best} domain candidates into k body bias domains will be introduced.

5.4.1 Combinatorial k-Subset Approach

In the previous steps, the leakage optimal solution of k_{best} domain candidates has been computed. In most cases, however, a partitioning into k_{best} body bias domains would result in a prohibitive overhead. To reduce the incurred overheads to a tolerable minimum, k_{best} domain candidates are then merged into k body bias domains in such a fashion, that the leakage increase over the optimal partitioning into k_{best} domain candidates is minimal. To determine the leakage of a given partitioning into body bias domains, a two-step method is employed:

1. For a partitioning into k domains, determine the leakage optimal body bias assignment while maintaining timing constraints.
2. For this partitioning where each domain is assigned a body bias, compute the incurred leakage.

As the leakage optimal body bias assignment can only be known once a partitioning has been determined, an algorithm aiming to find the leakage optimal solution has to consider all possible combinations. Formally, the problem is thus approached by a method termed combinatorial k-subset approach. Using combinatoric methods, all possible combinations of k_{best} domain candidates into k body bias domains, i.e., subsets of the original k_{best} domain candidates $DCS = \{DC_1 \cdots DC_{k_{best}}\}$ are computed. Each of these combination $c = (sub_1 \cdots sub_k)$ is a k-tuple of non-empty subsets sub_i where

$$\bigcup_{i=1}^{k} sub_i = \delta = \{DC_1 \cdots DC_{k_{best}}\} \tag{5.1}$$

with the entire design δ which shall be comprised of k body bias domains, which in turn consist of domain candidates DC_i and

$$\bigwedge_{i=1}^{k} \bigwedge_{j=1, j\neq i}^{k} sub_i \cap sub_j = \emptyset \tag{5.2}$$

dictating two central partitioning requirements. Eq. 5.1 requires the whole design δ to be partitioned and Eq. 5.2 requiring an overlap free, i.e., physically manufacturable partitioning.

Together with the two-stepped evaluation approach, this leads to the following algorithm:

Require: Domain candidates $DCS = (DC_1 \cdots DC_{k_{best}})$, available body biases VB, satisfiable constraint t_{clk}

Ensure: Partitioning c_{final} is leakage minimal and obeys timing constraint t_{clk}

```
1:  c_final ← ∅
2:  minLeak ← ∞
3:  for all c ∈ generateSubSet(DCS, k) do
4:      VBA ← assignOptimalBodyBias(c, VB, t_clk)
5:      leak ← computeLeakage(c, VBA)
6:      if leak < minLeak then
7:          minLeak ← leak
8:          c_final ← c
9:      end if
10: end for
```

In line 3, a combinatoric generator function is used to generate all combinations, i.e., partitioning of k_{best} domain candidates in DCS into k non-empty subsets over which this for-loop iterates. Each of these combinations is then evaluated using the above-described two-step process. First, the leakage optimal body bias assignment for combination c is determined. This can be done as in section 5.4, using an exhaustive search, or by using an algorithm as introduced in the following chapter 6. This optimal body bias assignment is defined as the assignment of a body bias to each domain which results in the smallest leakage and still observes timing constraint t_{clk} (line 4). Then, leakage can be computed using the partitioning c and the body bias assignment VBA in line 5. Lines 6 to 9 then make sure that only the partitioning with the smallest resulting leakage remains in c_{final}. Additional checks have been omitted with the requirement that the timing constraint must be reachable using one of the body biases in VB.

Fig. 5.11 visualizes the algorithms' principle of function. First, there are four domain candidates (k_{best}) with their optimal body bias assigned, i.e., at this body bias, as much reverse body bias as possible and as little as necessary forward body bias is applied, without violating timing. Obviously, these domain candidates do not share a common body bias, in fact, all domains have different body biases. Thus, a tradeoff is sought by trying all combinations of these four domain candidates into two body bias domains. Fig. 5.11 gives an example for the evaluation of one such combination. To avoid timing violations,

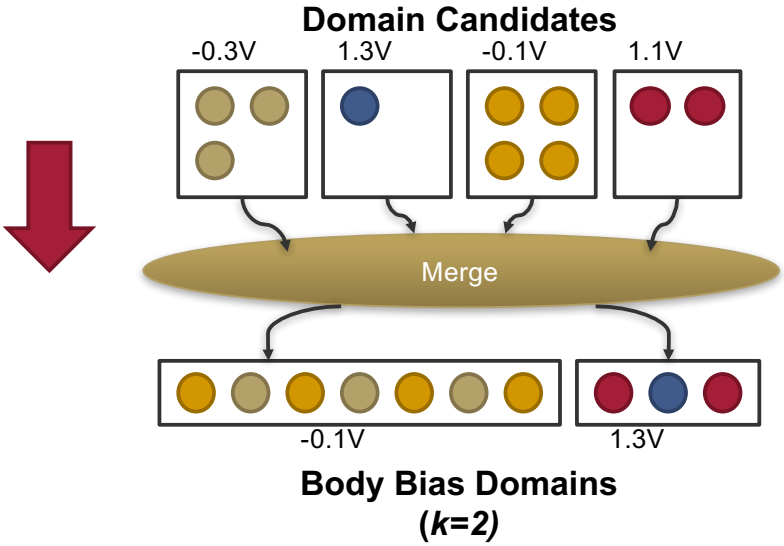


Figure 5.11: Visualization of the combinatorial k-subset approach for $k_{best} = 4$ and a target $k = 2$

the body bias has to be risen to the maximum body bias of all domain candidates that have been grouped together. For example, when grouping the blue dots domain candidate with the be.g., domain candidate, a minimum of $V_{BB} = -0.1V$ has to be applied, and similarly, for the orange-red combination, a minimum of $V_{BB} = 1.3V$ has to be applied. Such combination with the aforementioned body bias assignments is then evaluated for leakage. If it is the combination with the lowest leakage up till now, it is stored as prospective minimal leakage combination with the appropriate assignment. This step is repeated until all combinations have been evaluated.

5.5 Discussion

Depending on the component that shall be partitioned, the approaches introduced in this chapter has varying degrees of applicability and effect. It is always a question of temporal and spatial focus. If, e.g., the design in question is known to have balanced timing paths, i.e., no major differences in timing criticality and if all components are used at the same time, core-grained body biasing is an easy and straightforward approach to utilize one of FDSOI's key features: body biasing. This, however, is a rather strong assumption and in reality this being rarely the case, the penalties are severe.

In the case of DRPs, coarse-grained body biasing seems like a great match and indeed, it combines the simplicity of core-grained body bias with a much more local focus of body bias. However, especially in the case of DRPs, it is known in advance that only a couple of the supplied resources per PE are used per application. Among those PEs, the utilization also varies. While coarse-grained body bias in the way presented above, i.e., one body

bias domain per PE, is able to make use of the varying utilization, it always has to settle for the most critical path in use. With DRPs often consisting of large numbers of PEs, there is a lot unused potential that can be leveraged using more fine-grained approaches.

Fine-grained body biasing answers this demand by allowing virtually any granularity. However, the need for pre-partitioning by hand makes this partitioning approach very labor-intensive. Even for the examined DRP design, considerable efforts are required to enable the proposed analysis and partitioning. While one might argue, that complex components also require significant effort for physical design, increased design complexity also directly translates into significantly higher effort to pre-partition the design into domains candidates and also to me.g., the latter into body bias domains.

All efforts can be justified if the benefits outwe.g., the investment. However, ultimately all of the above-introduced approaches share another drawback: to analyze the mapped netlist of the design, strong optimizations, especially those that destroy the design hierarchy, cannot be used. If e.g., a timing analysis shall run on a PE to determine operation dependent slack of the ALU, the ALU needs to be clearly defined. Similarly, if components for fine-grained body biasing are pre-partitioned and subsequently merged, this also disables or severely limits opportunities for resource sharing or for that matter any optimization that requires taking the whole design into account, e.g., logical optimizations.

In sum, while leaving room for improvement, the three proposed approaches constitute concrete partitioning approaches, describing how to partition, what should be put into such a partition and what can be gained by that. Still, it falls short of the ability to analyze and partition highly optimized netlists. This shortcoming, however, can only be solved on the standard cell level for which the following chapter will propose a partitioning approach.

6 Standard-Cell-Grained Body Biasing and Automization through Domain Candidate Exploration

Highly optimized netlists represent a particular challenge as they reveal barely any structure that allows to draw conclusions regarding the central body bias partitioning principle of activation. Related to activation, timing poses another challenge, as the timing of certain paths determined by activation is required to know what to bias (activation), when (activation) and by how much (timing of the active path). While the results of highly optimized synthesis makes the initial situation slightly better, i.e., smaller resulting circuit size usually also means smaller leakage, it also introduces increased complexity at many other points. First of all, there needs to be a grouping mechanism that clusters standard cells to form domain candidates. In the previous sections 4.3 and 5.4.1, it has been shown that the computational complexity of the merging algorithm, e.g., the presented combinatoric k-subset approach, does not allow for standard-cell granularity due to its factorial growth in runtime. Since there is no other way around this issue, it is imperative to find methods to keep the number of domain candidates as low as possible. However, even accurately determining domain candidates in highly optimized netlists is a challenge, as synthesis optimizations seek to share resources wherever possible. With physical partitioning not allowing overlapping body bias domains, a way to determine unique physical partitionings is required. Even after having determined domain candidates, to be able to merge them, optimal body bias assignments across body bias domains are necessary to compute leakage and thus determine the optimal partitioning with optimal body bias assignments.

In this chapter, algorithms for the mentioned problems are proposed and discussed. All presented algorithms combined form the so-called Domain Candidate Exploration (DCE), which has been presented at the 53$^{\text{rd}}$ Design Automation Conference (DAC) in 2016 as publication [102], on which this chapter is also based on.

6.1 Methodology and Preliminaries

Again, a design δ with pipeline stages $\delta_1 \cdots \delta_k$ is assumed, where each pipeline stage is evaluated individually as the timing constraint has to be met within each stage. Additionally, memory components $M_1 \cdots M_t$ are explicitly considered as their activation, or rather usage patterns cannot be inferred without user-defined constraints. Thus, such components are declared with specified usage constraints as shall be later elaborated. For simplicity,

one concrete δ_k will be treated from here on, with a given V_{DD}, voltage regulators to supply $J = |VB|$ body biases and a minimal clock period of t_{clk}. Proceeding from these assumptions, three corners are defined in accordance with section 4.4:

1. $t_{max,RBB}$ minimizing leakage currents at the cost of reduced clock frequency
2. $t_{max,LP} = t_{clk}$ unchanged clock period while optimizing leakage wherever possible
3. $t_{max,HP}$ boosting attainable clock frequency using strong forward body biasing while minimizing leakage overheads

All the data required for the subsequent partitioning steps are obtained from technology-specific liberty files and SPICE simulations. The liberty files provide virtually all information for each standard cell, such as state dependent leakage, timing based on input transition and output capacitance, logical function per output pin, and the like. The SPICE simulations, on the other hand introduce the body bias dependent factors, such as its effect on delays (Fig. 2.5, p. 16) or leakage (Fig. 2.4, p. 15).

Nevertheless, liberty files are not supplied directly but first processed to contain simpler timing information from which worst-case timings are computed. In fully annotated liberty files, for each standard cell and each of its output pins, timings are given as input state dependent tables separately for rise and fall times. While this allows to compute very exact timings using fully-fledged, sophisticated timing solutions, in a proof of concept implementation this wealth of information cannot be leveraged. Thus, each standard cell's timing information is simplified to contain only one timing table per output pin. This timing table then contains the worst-case timings over all input states and rise as well as fall times. The same is done for leakage in case the liberty file contained state dependent leakage. In this case, of all states, always the worst-case leakage is used. The liberty files processed in this manner are then output together with all required information as JSON files, allowing straightforward object mapping. Similarly, using ring oscillator simulations, the effect of body bias on timings is measured by computing the normalized effect delay when applying forward or reverse body bias. Furthermore, the same is done for leakage through DC simulation of all available standard cells from which an average is then computed[1]. Delay and leakage values are normalized using their respective zero-bias values. Both of these measures are computed for each V_{DD} used. Temperature has always been set to the same temperature the utilized standard cell libraries were characterized at. Both measures are then also output as JSON files.

6.2 Determining Activation

To determine the activation of standard cells, that is when each respective standard cell is being utilized, the evaluated design δ_k is parsed and represented as a graph G as visualized

[1] Strictly speaking, it is not necessary to average leakage results over all standard cells, as body bias affects all transistors in the same manner and intensity.

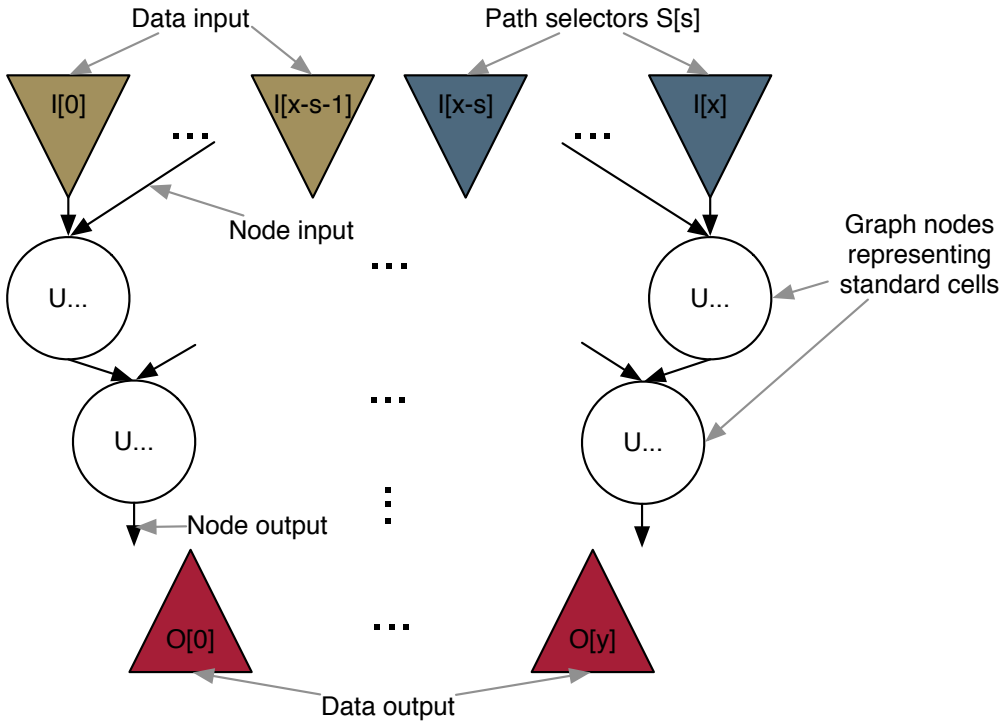


Figure 6.1: General structure of the graph used in DCE to analyze timing and determine activation

in Fig. 6.1, with input nodes $I[0]\cdots I[x]$, where s of these input nodes $S[s] \subset I[x]$ serve as so-called path selectors. Additionally, to those two types of input nodes, there are output nodes $O[0]\cdots O[y]$ as well as standard cell nodes U between input nodes $I[x]$ and $O[y]$ realizing the actual function. Path selectors are special input nodes such as e.g., the opcode input of ALUs, determining the function applied on $I[0]\cdots I[x-s]$ before outputting the result via $O[0]\cdots O[y]$.

First, the graph G is generated by parsing the mapped verilog netlist. Verilog netlist parsing is realized using the EDAUtils verilog parser [103]. The generated netlist structure is then traversed from inputs top-down through standard cells and their connections to each other down to output. Thereby, the graph structure G is built. After the graph has been built, it is annotated with standard cell information obtained from the liberty to JSON conversion.

To determine at what time, which standard cells are used for a given path selector, logical solving is employed. The principal idea is to first generate a boolean representation for each output node by traversing the realizing nodes top-down and expanding the logical equations using the cells' inputs to expand each standard cell outputs' equations. The thereby expanded output equations are then used as inputs of the cells further down the topology. To facilitate this expansion and the subsequent computations on boolean equations, a data structure which shall be called logic atom (LA) from here on is introduced.

There are two types of atoms: atomic atoms and composite atoms or molecules to stick with the analogy. Atomic atoms either consist of a boolean value

$$\alpha(\text{TRUE}|\text{FALSE}) \tag{6.1}$$

which is either TRUE or FALSE (Eq. 6.1) or

$$\alpha(VAR) \tag{6.2}$$

a variable name *VAR* (Eq. 6.2). Using such atomic atoms, composite atoms consisting of several atoms and operations can be realized. Thus

$$\alpha(\alpha \cdots \alpha, (\neg, \wedge, \vee)) \tag{6.3}$$

generally defines a composite atom as a two-tuple, consisting of a list of one or more atoms, atomic or composite, and an operation using which the list of atoms is combined. Valid operations are either negation $\neg$ which takes only a single atom as argument, logical and $\wedge$ as well as logical or $\vee$. The latter two take an arbitrary number of atoms greater or equal than two. Using this construct, any logical function can be realized. As an example, consider a 3-Input NAND as demonstrated in Fig. 6.2.

$$Z = \alpha(\alpha(\alpha(A), \neg), \alpha(\alpha(B), \neg), \alpha(\alpha(C), \neg), \vee)$$

Figure 6.2: Exemplary LA of 3-Input NAND with inputs *A*, *B* and *C*, as well as the resulting output *Z*.

To be able to work on standard cells, their logical function is first parsed and converted into a logic atom representation. Thus, the following algorithm converts the logical function string of the liberty library into a logic atom:

Require: Set of used standard cells $usedStdCells$ where each standard cell has a logical function string in DNF per output pin

Ensure: Each standard cell has a map, associating each output pin with the equivalent logic atom representation of its implemented function

1: **for all** $U \in usedStdCells$ **do**
2: $U.outputMap \leftarrow createMap(outputPin \leftarrow LA)$
3: **for all** $O \in U.outputPins$ **do**
4: $LA \leftarrow parseFunctionString(O.functionString)$
5: $U.outputMap.put(O \leftarrow LA)$
6: **end for**
7: **end for**

Line 1 iterates over all standard cells used in the graph *G* while for each standard cell, a map associating each output pin with the corresponding logic function, represented as logic atom is generated (lines 2-6). This is done by iterating over all output pins and generating a logic atom representation of the implemented function (lines 3-4). This

is done by parsing the logical function string obtained through the annotated liberty information. The parser that is written for DCE requires all function strings to be in disjunctive normal form (DNF). The thereby obtained logic atom *LA* is then stored in the map $U.outputMap$ of standard cell U, associated with logic atoms *LA* (line 5) as one standard cell may have multiple outputs.

With logic atom representations of the standard cells, the boolean equations for all output pins can be computed. To do this, the graph is traversed top-down, starting with the input nodes as denoted in the following algorithm:

Require: Input atoms $I[x]$ of G, logic atom representation of standard cells
Ensure: Of all nodes in G, their LAs are expanded

```
1:  worklist ← I[x]
2:  while worklist not empty do
3:      x ← removeFirst(worklist)
4:      for all v ∈ x.inputVar do
5:          (U, O) ← getInputCell(x.portMap, v)
6:          l ← U.outputMap.get(O)
7:          x.atoms ← substitute(x.atoms, v, l)
8:      end for
9:      worklist ← worklist ∪ x.toNodes
10: end while
```

This expansion problem can be easily solved using a work list approach. The work list is initialized with the input nodes $I[x]$ which are atomic logic atoms realizing the input variable names. As they are inputs themselves and not composite, they do not have the *inputVar* attribute and only the vertices to which their edges connect are added to the work list in line 9. If, however, the x which is removed from the work list in line (3) is a standard cell, it has the *inputVar* attribute. *inputVar* is a list of all input pins the standard cell has, internally represented as atomic variable logic atoms with the name of the corresponding input pin. For all these inputs (line 4), the connecting node is determined via the current standard cell x's portmap and the current variable v. The function *getInputCell* then retrieves the node as well as the responsible output pin from which the output is used as input (line 5). Now, using the previously created output maps, the correct logic atom is fetched by searching the $outputMap$ of U for the logic atom of output pin O (line 6). Line 7 then proceeds to the actual expansion step, where all occurrences of variable v are substituted using the previously determined input logic atom. After iterating over all input variables of the standard cells, the nodes reached by its outgoing edges are added to the work list in line 9 to make sure all nodes are reached. After the algorithm completes, the logical function of each output is then assigned as inputs to the nodes reached by the outgoing edge of the standard cell output.

These boolean equations are then mathematical representations of the realizing function with all input nodes the result may depend on, independent of the selected function. To determine the standard cells involved in the realization of a certain function per output,

specified by path selector $op = B_0 \cdots B_s$ with B being a boolean value for $S[0] \cdots S[s-1]$ respectively, op is inserted into the boolean equations. This is done by replacing all occurrences of $S[0] \cdots S[s-1]$ with logic atoms representing their respective boolean value in all output equations $O[0] \cdots O[y]$. Once this is done, the boolean equation can be reduced to only those parts that realize the functionality of opcode op by solving the equation. With actual input data $I[0] \cdots I[x-s-1]$ as unknown variables, this process only leaves the paths and their respective logic function in the outputs' equations that realize the functionality specified by op. The following pseudo code describes this algorithm:

Require: Opcode selector $S[s] = S[0] \cdots S[s-1] \subset I[x]$, logic atom representations of $O[y]$
Ensure: For each $O[y]$, only standard cells implementing the function of $opcode$ remain in $E[O[y], opcode]$ while the bijection $LA \leftrightarrow GN$ is maintained

```
1:  opcode[2^width(S[s])] ← generateOpSelArray(S[s])
2:  for all o ∈ O[y] do
3:      for all op ∈ opcode do
4:          E[o,op] ← clone(o)
5:          for all B ∈ op do
6:              origVarName ← getVarNameByPos(S[s],op,B)
7:              E[o,op] ← substitute(E[o,op],origVarName,B)
8:          end for
9:          E[o,op] ← solved(E[o,op])
10:     end for
11: end for
```

Line 1 first creates an array of all possible opcodes. Should one be unused, the ensuing process will just turn the output equation into a constant. Then, for all output equations o of outputs $O[y]$ and all operations op in the array of opcodes $opcode$, the output equations are copied. One copy for each output equation and operation is strictly required, as the following solving process is destructive, i.e., the data structure is irreversibly altered. With a safe copy, the path selector variables $S[0] \cdots S[s-1]$ are replaced with actual boolean values in $E[o,op]$, the boolean equation for output o when executing operation op. To do that, the opcode op is iterated bit by bit. To determine what should be replaced, the position of B in op can be used, as this position corresponds to the index in the array of path selector variables $S[s] = S[0] \cdots S[s-1]$. In this snippet of pseudocode, *getVarNameByPos* returns the variable in $S[s]$ specified by the index in op of B (line 6). With this known, all occurrences of the original variable name *origVarName* are replaced in $E[o,op]$ by B (line 7). Once all path selectors are replaced with concrete boolean values, the equation is solved in line 9. For this task, however, a commodity logic solver couldn't be employed, as all logic atoms have to maintain associated with a standard cell that physically realizes the logic function. This is done using the following simple rules and the logic atom structures:

$$X \wedge 1 \Rightarrow X,\ X \wedge 0 \Rightarrow 0 \quad \alpha(X,1,\wedge) \Rightarrow \alpha(X),\ \alpha(X,0,\wedge) \Rightarrow \alpha(0) \tag{6.4}$$

$$X \vee 1 \Rightarrow 1,\ X \vee 0 \Rightarrow X \quad \alpha(X,1,\vee) \Rightarrow \alpha(1)\ \alpha(X,0,\vee) \Rightarrow \alpha(X) \tag{6.5}$$

$$\neg 1 \Rightarrow 0,\ \neg 0 \Rightarrow 1 \quad \alpha(\alpha(1),\neg) \Rightarrow \alpha(0)\ \alpha(\alpha(0),\neg) \Rightarrow \alpha(1) \tag{6.6}$$

Each rule is given as standard boolean representation (left) and in the encapsulating logic atom representation used in DCE. Eq. 6.4 specifies the rules for logical and operations. If any part of a boolean equation X is combined with boolean true, then the true value may be removed without changing the equation. If the same is done with the boolean value false, then the whole clause can be reduced to false as logical and requires all operands to be true for the result to be not false. This condition is obviously no longer possible and the clause can be reduced to false. In the case of logical or, the rules are similar (Eq. 6.5). If any part X is combined in a logical or with the boolean true value, the whole clause can be reduced to true as for the result of a logical or to be true, only one expression needs to evaluate to true. Therefore, it is also not necessary to consider the other expressions. When combining a part X with boolean false, the false value may be eliminated as a result then solely depends on X. The most straightforward rule is, of course, the negation rule specified in Eq. 6.6. When represented as logic atoms, however, it reveals the inner workings of the solver. If a negation of a boolean value has not been executed, the negation logic atom "wraps" the boolean value to be negated. Once the negation is executed, i.e. true is inverted to false or vice versa, the wrapping atom realizing the negation is discarded. For simplicity, the solver supports only these three operations as specified in Eq. 6.4-6.6 and thus additionally requires all standard cells' logic equations to be in disjunctive normal form.

In a similar manner, blocks of inferred memories or, although not treated in this thesis, possibly also SRAM can be analyzed for activation. This is done by simply treating parts of the address like an opcode above. By doing so, the algorithm will determine which memory cells can be reached for a given part of the address. For example, if the memory should be partitioned into two domains or rather domain candidates at this stage, then the highest address bit is defined as path selector, resulting in an opcode array with two entries and two sets of output equations $E[o,0]$ and $E[o,1]$ respectively. This specification of the path selector, however, has to be done explicitly by the developer as it strongly depends on the usage profile of such components which cannot be determined by analysis of the hardware component.

If the result for a given output and opcode is not constant, i.e., if the result of the solving process is not true or false, the solver leaves the parts that determine the final result in the resulting equation together. All other in this opcode unused parts are eliminated. The parts that remain can still be associated with the standard cell that implements each part as each logic atom is marked with its implementing standard cell. This provides a bijection between logic atoms *LA* and graph nodes, i.e., standard cell nodes *GN*.

This bijection together with the solver results can now determine which standard cells are used in each opcode. Thus, for each opcode, all standard cells used for the logic atoms in the solved output equations are aggregated into sets indexed by opcode. This is described in the following algorithm:

Require: Bijection $ID_{stdcell} : LA \leftrightarrow GN$

Ensure: Set $A[s]$ contains active standard cells per opcode op
1: **for all** $op \in opcode$ **do**
2: **for all** $o \in O[y]$ **do**
3: **for all** $a \in E[o, op]$ **do**
4: $A[op] \leftarrow A[op] \cup ID_{stdcell}(a)$
5: **end for**
6: **end for**
7: **end for**

For each opcode, all previously solved output equations are considered (lines 1-2). Then, for each logic atom in the solved equations $E[o, op]$, the realizing standard cell is identified using the previously bijection, here used as function $ID_{stdcell} : LA \rightarrow GN$. The thus identified standard cells are then merged in the set $A[op]$. In principle, this set is a first variant of domain candidates. However, as phenomena such as resource sharing are not yet treated, the partitioning requirement

$$\underset{i \in opcode}{\forall} i \neq x : \; A[x] \cap A[i] = \emptyset \tag{6.7}$$

is not satisfied. Domains built from such quasi-domain candidates may not be manufacturable or might result in timing faults. Before resolving this issue in section 6.4, the second and for this step required principle, timing, will be treated.

6.3 Determining Timing Criticality

Along with activation, timing is a decisive factor when deciding what standard cells form a domain candidate, as timing ultimately determines the level of body bias for a given timing constraint. Thus, not only timing needs to be determined but also timing under body bias. As introduced in chapter 2, body biasing has a strong effect on the time a transistor requires to switch and thus body biasing influences overall timing. e.g., forward body biasing increases both I_{ON} and I_{OFF}, while reverse body biasing reduces both. Thus, depending on the output capacitance the respective standard cell has to charge, the input transitions of the following cells change with body bias and thus alter the timing of those cells as well.
Fig. 6.3 illustrates the general timing computation approach. First, the initial timing of the first cell is computed using a defined input transition and by computing the sum of the capacitances it drives on its outputs. Together with input transition, this will compute the cell's timing and the transition time for the connected cells. This is repeated until all cells on the path or those which exercise an influence on the timing path are computed. Algorithmically, this has been implemented using a worklist algorithm with an additional termination condition, specifying to terminate once the design graph has been traversed without any changes to the timing of one of its standard cell nodes.

To account for the effect of body biasing on timing, its influence has been approximated using ring oscillator experiments simulated in SPICE. From these simulation results, the

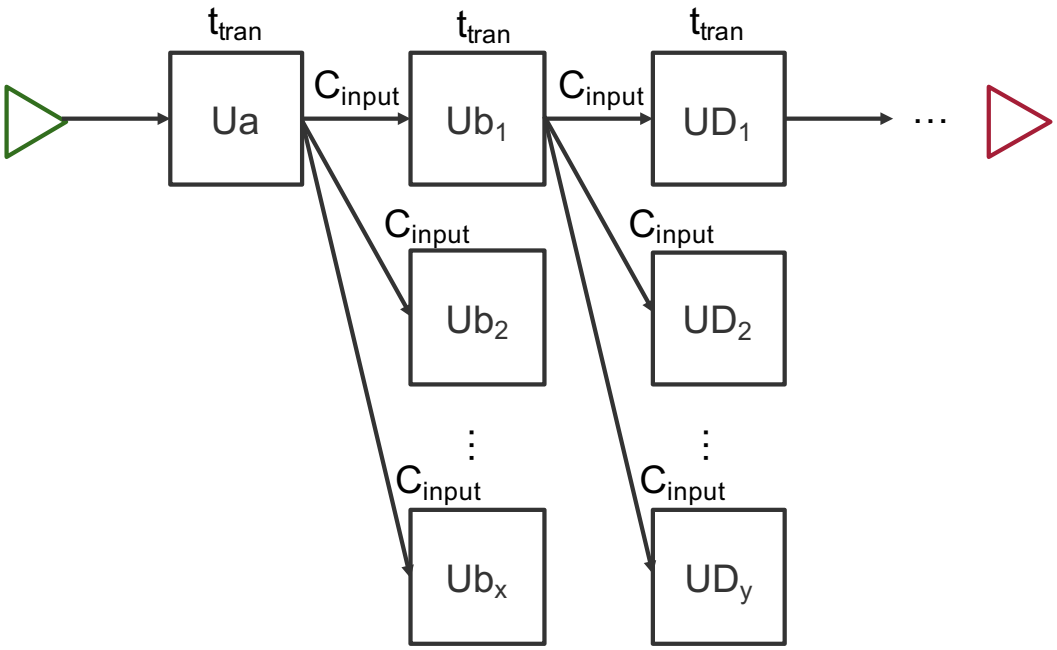


Figure 6.3: Exemplary timing path Ua to UD with depth D and different fanouts

oscillation frequencies $F_{osc,BB}$ are obtained by sampling the data signal in the oscillator. These frequencies are then normalized against $F_{osc,zero}$, the oscillation frequency at zero-bias and thus is directly proportional to the involved transistors' capability to drive current. This step provides scaling factors n_{BB}, expressing the frequency increase or decrease relative to zero-bias. The faster an inverter inverts an input signal, the stronger is its capability to drive the next inverter's input pin. Thus, while all capacitances stay the same, the resulting transition of a given standard cell time changes with body bias. Thus, transition time computation is conducted as

$$t_{tran,BB} = t_{tran} \cdot n_{BB} \tag{6.8}$$

with t_{tran} the originally resulting transition time, scaled using the appropriate body bias factor n_{BB}, i.e., the body bias factor corresponding to the V_{BB} used in the domain the standard cell is located. While this approximation might be slightly crude, it constitutes a good trade-off in regard to accuracy and suitability of implementation for a proof of concept. The actual timing annotation for design graph G is described using the algorithm below:

Require: List of timing LUTs $LUT[]$, graph G, active set $activeSet$, body bias factors n_{BB}, input nodes $I[x]$

Ensure: G is annotated with worst-case timings per node and per nodes' output pins

1: $worklist \leftarrow I[x]$
2: **while** $worklist \neq \emptyset$ **do**
3: $\quad curNode \leftarrow worklist.removeFirst$

```
4:    if curNode ∈ activeSet then
5:        if isInputNode(curNode) then
6:            worklist ← worklist ∪ curNode.outNodes
7:        else
8:            for out ∈ curNode.outputPins do
9:                t_timing ← LUTLookup(LUT, out.maxTimingTable, out.timingTemplate,
  curNode.t_timing,max, C_drive)
10:               t_timing,BB ← t_timing · n_BB
11:               if t_timing,bb > out.t_timing,max then
12:                   out.t_timing,max ← t_timing,bb
13:               end if
14:               if t_timing,bb > curnode.t_timing,max then
15:                   curnode.t_timing,max ← t_timing,bb
16:               end if
17:               t_tran ← LUTLookup(LUT, out.maxTranTable, out.tranTemplate,
  curNode.t_tran,max, C_drive)
18:               t_tran,BB ← t_tran · n_BB
19:               if t_tran,BB > curNode.t_tran,max then
20:                   curNode.t_tran,max ← t_tran,BB
21:                   worklist ← worklist ∪ curNode.outNodes
22:               end if
23:           end for
24:       end if
25:   end if
26: end while
```

Before getting into the algorithm's details, there are some assumptions which are set in the actual implementation. Initially, all target variables (e.g. $curNode.t_{tran,max}$, $curNode.t_{timing,max}$) that store the timing values are initialized with a value of minus infinity to ensure that all values are updated at least once. Line 1 fills the worklist initially with the input nodes $I[x]$ on which then the main while-loop iterates in line 2. The first element of the worklist is then removed and stored in the *curNode* variable (line 3). The check against the active set *activeSet* in line 4 allows the algorithm to be more general. Body bias domains are defined as disjunct sets of standard cells. If the timing of the entire design should be computed, *activeSet* is the set of vertexes of graph G. If, on the other hand, the timing of only certain body bias domains should be computed, *activeSet* should contain only vertexes, i.e., the nodes of the concerned domain(s). Following this check, the algorithm inquires whether *curNode* is an input node, i.e., not an actual standard cell, but an input of the design. If this is the case, then it also has no timing and no computation has to be executed. However, input nodes supply data input to standard cells, so all nodes reached by the input node's outgoing edges should be added to the worklist.

If the examined node *curNode* is a standard cell, then timings will be computed in lines 7 to 25. Timings are actually not computed per cell, but per output of each cell. Some

standard cells may have multiple outputs where each output may have a different timing, depending on the transition times of the cell's input, the capacitance it has to drive as well as the physical realization of its implemented function. Thus for each output pin *out* of all outputs, denoted by the set $curNode.outputPins$, a timing t_{timing} as well as resulting transition time t_{tran} is computed. The actual computation is realized as a lookup-table lookup. Thus, the lookup function $LUTLookup$ is called with the list of lookup tables LUT, the target value table $out.maxTimingTable$ or $out.maxTranTable$ in case of transition computation, the template name of the lookup table used to index the value table and the two values used to determine the index through the actual lookup table (lines 9 and 17). These two values are again pin-specific. Depending on the input variables, a different maximum transition time may apply. Obviously, the driving capacitance always depends solely on the input pins the output drives.

In both cases, timings and transitions are scaled using the respective body bias scaling factor n_{BB} used in the body bias domain to which the standard cell belongs. Once the resulting times have been scaled, they are examined to decide whether they are new maxima. If they are, the new maxima are noted (lines 11-13 and lines 19-22). Furthermore, in case of transition times, if the resulting transition time changed and reached a new maximum, the nodes reached by this output are added to the worklist for an update as their timing may change thereby as well. This method ensures that the while loop iterates until the true worst-case, timing of all standard cells is determined.

In order to find the critical path timing, now the previously computed standard cell timings need to be summed up while the critical path marks the path, whose endpoint has the greatest resulting timing.

6.4 Building Domain Candidates

With a first approximation of domain candidates from section 6.2 and body bias aware timing computation of the previous section 6.3, actual body bias domain candidates can be built. Domain candidates have the following characteristics:

1. A domain candidate contains components of equal timing criticality
2. Domain candidates fulfill the partitioning requirement, i.e., no overlaps
3. Implemented functionality may be distributed over multiple domain candidates

In the following, methods and definitions pertaining resource sharing among domain candidates, resolving of resource sharing induced overlaps and the determination of domain candidates based on activation and timing criticality equality will be introduced.

6.4.1 Resource Sharing and Cannibalization

In highly optimized netlists, resource sharing is a major difficulty in determining distinct body bias domains. In this approach, resource sharing problems are eliminated on domain

candidate level. The defining question addressed here is, if two prospective domain candidates share standard cells, which domain candidate shall receive the shared cells? Ultimately, for optimal results, this can be only answered per application. As a general rule, however, it is most sensible to keep timing critical domains as small as possible. Thereby, the exponential leakage increase under forward body biasing can be mitigated.

Thus, this thesis proposes to resolve resource sharing by using a method called cannibalization, i.e., the less timing critical domain candidate gets the standard cells of the more timing critical domain candidate. The following algorithm describes the procedure.

Require: Active sets $A[s]$, the timing of each set $timing(A[i])$

Ensure: The resulting set $C[s]$ conforms to partitioning requirement Eq. 6.7 and shared cells are in the prospective domain candidate with lower timing

1: $C[s] \leftarrow sortByTimingDesc(A[s])$
2: **for** $x = 0, x \leq s-1, x++$ **do**
3: **for** $y = x+1, y \leq s-1, y++$ **do**
4: $C[x] \leftarrow C[x] \setminus C[x] \cap C[y]$
5: **end for**
6: **end for**

First, $C[s]$ is assigned $A[s]$ sorted by descending timing (line 1). Thus, the sets of active cells per opcode are now sorted by timing instead of the numeral value of the opcode. Then each prospective domain candidate indexed by x is stripped of all cells that are shared by less timing critical prospective domain candidates indexed by y (line 3). This simple algorithm called cannibalization makes $C[s]$ conform to the partitioning requirement Eq. 6.7 and allows these sets of standard cells to be placed together with other prospective domain candidates to form distinct body bias domains.

6.4.2 Creating Domain Candidates

Now $C[s]$ is prepared to form actual domain candidates $DC[q]$. Up to this point, $C[s]$ is determined by activation, while only resource sharing conflicts have been resolved by the help of timing. However, as mentioned before, timing, i.e., timing criticality determines the required body bias. Thus, at this point, the timing information is unused, meaning that prospective domains that are equally timing critical and possibly even active at the same time may be grouped into different body bias domains. To resolve that, domain candidates of equal timing criticality are merged. The following algorithm describes the procedure:

Require: $C[s]$ with 2^{opcode} entries

Ensure: $DC[q]$ is composed of q distinct, non-empty sets of standard cells, defining domain candidates

1: **for all** $C \in C[s]$ **do**
2: **if** $C \neq \emptyset$ **then**
3: $DC[biasOf(C, t_{clk})] \leftarrow DC[biasOf(C), t_{clk}] \cup C$
4: **end if**
5: **end for**

By iterating over all sets C (line 1), that is per opcode one set which might be even left empty after cannibalization, it is first checked that C is not empty (line 2). Then, in line 3 the set of domain candidates DC is indexed using a helper function $biasOf$. This function checks for a given timing constraint t_{clk} which level of body bias needs to be applied. Since the set of available body bias is discrete, components of equal body bias will be collapsed into one domain candidate. Thus, the set of available body biases together with the target timing constraint give a discrete measure of timing criticality and furthermore, a definition of timing equality in respect to body biasing. That is, if two components are assigned the same body bias from a finite set of body biases, they are equal in regard to timing criticality.

6.5 Building Domains

Now that domain candidates are distinct without any overlapping cells, cross-domain resource sharing resolved and their number minimized, the set of domain candidates $DC[q]$ is optimized for usage with the previously introduced combinatorial k-subset approach. Previously, in fine-grained body biasing, components were pre-partitioned with possibly no intelligence at all. Thus, in the worst case, the number of domains used to artificially increase because of bad choices in pre-partitioning. DCE on the other hand determines domain candidates based on actual relations, activation, and timing. Thus, the number of domain candidates generated by DCE is greatly reduced, compared to the previous approach that didn't even attempt reducing the number of possible domain candidates. This is done while completely eliminating the need for manual pre-partitioning.

Furthermore, with the BBI based body bias assignment algorithm introduced in the following section, an exhaustive search through all possible body bias assignments of $|VB| = N$ to q domain candidates, resulting in N^q assignments to be evaluated can be omitted. Nevertheless, the merging problem ultimately can only be reduced in size through the DCE approach, as well as be mitigated in regard to additional combinations with the body bias assignment algorithm of the following section. The underlying problem still requires an exhaustive search over all possible combinations of q domain candidates into k body bias domains. Thus the algorithm of section 5.4.1 is modified, yielding the following algorithm:

Require: DCs $DC[q]$
Ensure: $DC[q]$ merged into k domains stored in A, with minimal leakage while observing timing constraint t_{clk}

1: $minLeak \leftarrow \infty$
2: $A \leftarrow \emptyset$
3: **for all** $part \in generateSubSet(DC[q], k)$ **do**
4: $\quad BB[k] \leftarrow assignBBByBBI(part, t_{clk})$
5: $\quad l \leftarrow computeLeakage(part, BB[k])$
6: $\quad$ **if** $l < minLeak$ **then**
7: $\quad\quad minLeak = l$
8: $\quad\quad A = part$
9: $\quad$ **end if**

10: **end for**

First of all, the variable storing the global minimum leakage *minLeak* is initialized with an infinite value, ensuring that the partitioning will be updated at least once. In the following line (line 2), the variable storing the domain partitioning information A is initialized, before the combinatorial subset generator function is called in the for-loop header. The function is the same as in section 5.4.1, generating a set of all valid partitionings of q domain candidates $DC[q]$ into k body bias domains. This allows the for-loop to iterate on them while conducting per partitioning evaluations. For each of these partitionings, the body bias impact based body bias assignment algorithm (see section 6.6) is called to determine optimal body bias assignments for a given timing constraint t_{clk} (line 4). With body bias assignments determined, leakage can be determined in line 5. If now the leakage l incurred by this particular partitioning and body bias assignment is less than the current minimum *minLeak* (line 6), it should be updated along with storing the present partitioning in A. Once the algorithm has iterated over all possible partitionings, the leakage optimal partitioning A into k body bias domains with leakage *minLeak* is determined.

6.6 Body Biasing Impact Metric and Optimal Body Bias Assignment

Once body bias domains have been determined, actual body biases have to be assigned in order to be able to use the design. This body bias assignment is timing driven, where the aim is to apply only as little forward body biasing as possible. With cross-domain resource sharing, however, a timing path may cross multiple body bias domains. The question is thus, which body bias domain should be biased by how much in order to meet timing? In the previous chapter, the fine-grained body biasing approach determined this by computing all possible assignments and the resulting leakages. From all these assignments, the leakage minimal one has been selected. To avoid the exponential complexity of this approach, a metric called body bias impact together with a polynomial optimal body biasing determination algorithm is introduced below.

When deciding to which body bias domain how much bias should be applied, the obvious choice is the domain which gives the largest timing improvement per leakage increase. This is a figure which can be obtained using the information derived from standard cell libraries and the body bias aware timing computation introduced in the sections above. Body bias impact (BBI) is defined as follows:

$$BBI = \frac{I_{leak}}{t_{crit}} \tag{6.9}$$

where I_{leak} is the leakage current per body bias domain and t_{crit} is the timing of the most critical path that is used in the concerned body bias domain. Thus, this is a measure of how much leakage is incurred per timing unit. The restriction on used paths is to keep the assignment algorithm as general as possible. In reconfigurable architectures such as DRPs,

it often can be ruled out with absolute certainty, that certain timing critical operations are not conducted. Thus the body bias assignment algorithm also does not need to account for those as well. Thereby derived assignments are of course application dependent. If such distinction is not desired or applicable, then the used timing paths set may just be ignored and the complete body bias domain with its overall critical paths is considered. To leverage the BBI information, the following algorithm is proposed:

Require: Affected domains $A = d[0] \cdots d[k]$, used tp_i, timing constraint t_{clk}
Ensure: BB assignment $VBA = \{V_{BB0}, \cdots, V_{BBn}\}$ to D so that leakage is minimal

1: $VBA = V_{BB0} \cdots V_{BBn} \leftarrow \min(VB)$
2: **while** $timing(p, VBA) > t_{clk}$ **do**
3: **for all** $d \in D$ **do**
4: $V_{BB,d\ sim} \leftarrow incBB(V_{BB,d})$
5: $I_{leak,d} \leftarrow computeLeakage(d, V_{BB,d\ sim})$
6: $t_{crit,d} \leftarrow computeTiming(d, tp_i, V_{BB,d\ sim})$
7: **if** $t_{crit,d} == 0$ **then**
8: $BBI[d] \leftarrow \infty$
9: **else**
10: $BBI[d] \leftarrow \frac{I_{leak,d}}{t_{crit,d}}$
11: **end if**
12: **end for**
13: $sortAsc(A, BBI)$
14: $sortAsc(VBA, BBI)$
15: $i \leftarrow 0$
16: forLoop:
17: **for** $; i \leq n; i++$ **do**
18: **if** $i == n$ **then**
19: Fail
20: **else**
21: **if** $VBA[i] < \max(VB)$ **then**
22: break forLoop
23: **end if**
24: **end if**
25: **end for**
26: $VBA[i] \leftarrow incBB(VBA[i])$
27: **end while**

First of all, the body bias assignment structure VBA is initialized with maximum reverse body bias denoted by $\min(VB)$ (line 1), i.e., the minimum available body bias voltage. Thereby it is ensured that only the minimum forward body bias reaching t_{clk}, and the maximum reverse body bias not violating t_{clk} will be used. Then, from line 2 on, the core part of the algorithm, the steady domain-by-domain BBI determined body bias increase is conducted until timing constraint t_{clk} has been met. Within the core part of the algorithm, body bias impact *BBI* for a simulated discrete body bias increase is computed. That means the body bias for each domain d out of D, a simulated body bias increase $V_{BB,d\ sim}$ is used

to compute leakage (line 5) and timing (line 6). Timing additionally uses the set of timing paths tp_i that are used in this particular domain. Consider e.g., full-adders that may be shared among multipliers and additions. Thus, for both the resources may be in the same domain, but the incurred timing depends on what paths are actually used, e.g. either the multiplier part is used, or the adder part is used, in which case the incurred timing usually is far less than in the latter.
Once both timing and leakage for the concerned domain with the simulated forward body bias is computed, body bias impact for the given domain $BBI[d]$ can be computed as well (lines 7-11). In case a domain is not involved in the realization of the concerned functionality, the incurred timing will be zero. Thus, in line 7, this condition is caught. If a domain is not involved, the domain's *BBI* is set to an infinite value which is why the domain will be omitted for body bias increases as the leakage penalty is infinite.

After the for-loop completes in line 12, the set of domains D and the set of body bias assignments VBA can be sorted by corresponding body bias impact in an nascending order (line 13 and 14). Now, in the set of domains at position 0, the domain with the lowest resulting body bias impact is located. However, depending on the number of previous iterations, it may not be possible to increase the body bias of the concerned domain in forward direction, as the maximum forward body bias which can be supplied may already have been reached. In this case, the algorithm should look for the domain with the next smallest resulting BBI. This is done in the loop from line 17 to 25. If, however, all domains are already at maximum forward body bias while timing is still not met, the timing goal can't be reached at the present supply voltage. In this case, the algorithm fails in line 19. If on the other hand, a domain with room for forward body bias has been found, the loop is broken (line 22) and the body bias of this domain is increased by one discrete step (line 26).

Using the given algorithm, always the domain's forward body bias is increased, which has the lowest BBI. This is due to the sorting of data structures by ascending BBI. Furthermore, with body bias impact BBI as a measure for how much leakage increase per timing improvement can be obtained, the algorithm always seeks the most leakage efficient way to improve timing. Thus, it makes use of body biasing in an optimal manner. Finally, as the algorithm starts out with maximum reverse body bias, leakage minimization using reverse body bias does not need to be covered separately, as the algorithm thereby always strives for optimal reverse body bias utilization as well.

6.7 Discussion

The approach presented in this chapter tackles the body bias domain partitioning problem in a fully automatized manner, thereby allowing to take even individual standard cells into account.
Similar to the pre-partitioned fine-grained approach, assumptions had to be made where in real-life concious choice may be a better option. Again, this concerns, in particular, the merging algorithm. For the leakage computation, the central optimization criterion, it is

assumed that all components need to be active, i.e., free of timing violations. Depending on the timing constraint, this may require considerable forward body biasing. While it could be argued that if the total leakage is optimized on the condition of all components being active, this will also reduce the leakage incurred when only requiring certain parts to be active. However, determination of the actual optimum partitioning will always depend on the applications that shall be run on the design in question, which again goes against generality. In addition, approximations, e.g., averaging leakage over all operations that should be executed in the design might not yield better results.

If for a design the application profiles are fully known, then the algorithms can specifically gear the body bias domain partitioning towards this profile. To do this, the leakage computation needs to be conducted per executed operation and then weighted according to its share in the total sum of operations. This allows an application specific fine-tuning of the body bias domain partitioning of a given design. For the sake of generalization, this has been omitted in this thesis by using the all-active assumption for all domain candidates while merging into actual body bias domains.

Another noteworthy aspect is the complexity of the underlying timing and leakage computations. As present EDA tools nor semiconductor manufacturers supply tools that can handle body biasing as a design variable, body bias aware timing estimation and leakage computation had to be custom implemented. To reduce complexity, e.g., the timing tables supplied by semiconductor manufacturers for timing computations have been pre-parsed to compute single worst-case timing tables. Among other simplifications, this is done by merging fall- and rise-dependent values by using the maximum of both. This, however, may lead to inaccuracies since it leads to a general overestimation of timing. Furthermore, to overcome the limited characterization supplied by semiconductor manufacturers, only zero-bias libraries were used while supplicating body bias dependent behavior using models obtained through SPICE simulations. These models mainly depend on the lowered V_{TH} as well as the higher I_{ON} characteristics. While many studies have reported similar effects on timing and leakage [6, 71, 104], more effort needs to be put into the computation of such values. Ultimately, until proper EDA tools become body bias aware, the thereby derived results should not be taken as the final statement on the matter. Thus, as cautious approach in this thesis, all computations were geared to overestimate at the disadvantage of the derived results. Thus, in regard to the results chapter documenting the evaluation of the presented approach, results will be given normalized to zero-bias results.

7 Test Chip Implementation

In preparation of the results chapter, an overview of the test-chip implementation and its evaluation goals will be given. The test-chip has been implemented in Renesas 65nm SOTB on a $3 \times 3mm$ die using a macro-based flow to implement the body bias domain partitioning obtained using the approaches described in chapter 5 and 6. The implemented architecture is the fully pipelined MuCCRA4 architecture proposed by Katagiri and Amano in [96, 97]. The implementation has been presented at COOLChips XIX in Yokohama, Japan in 2016 as publication [105].

The implementation environment is listed in table 7.1. For both logic and layout synthesis, Synopsys DesignCompiler and IC Compiler have been used. Additionally, Synopsys PrimeTime suite was utilized to run timing analysis and power estimations. For verification, Cadence IUS environment has been used for simulation (ncsim) on RTL, gate and layout level netlists as well as for waveform inspection. To perform low-level tests, Synopsys HSIM fast SPICE has been used together with Synopsys's CosmosScope to perform analyses on the simulation results. For the final touches and customizations on the chip-level layout, Cadence Virtuoso as part of the IC tool suite has been used. Finally, Mentor Graphics' Calibre was employed to perform DRC and LVS checks. The same tools were used for the sign-off of the final layout.

In the following, the macro-based implementation flow together with macro-level bias supply will be described. This is followed by a description of the body biasing schemes that can be realized using the previously realized body bias domain partitioning. This chapter closes with details on the bias supply method as well as the supply network used in this implementation.

Task	Manufacturer	Tool	Version
Logic synthesis	Synopsys	DesignCompiler	2012.06-SP5
Power & Timing Analysis	Synopsys	PrimeTime	2012.12-SP3
RTL/Gate/Layout level simulation	Cadence	ncsim (IUS)	10.20.131
Waveform analyzer	Cadence	SimVision (IUS)	10.20.131
SPICE level simulation	Synopsys	HSIM	2012.06-SP2
Waveform viewer (SPICE level)	Synopsys	CosmosScope	2013.12
Layout Synthesis	Synopsys	ICC	G-2012.06-SP5
Layout finalization and stream out	Cadence	Virtuoso (IC)	6.15
DRC/LVS	Mentor Graphics	Calibre	2013.2_18.13

Table 7.1: Implementation environment used during the tapeout of MuCCRA4-BB

7.1 Body Bias Domain Partitioning

To cover a variety of body biasing schemes and to back up the techniques as well as resulting schemes proposed in this thesis, a fine-grained approach with four body bias domains per PE has been implemented. Two domains are allocated for memories, while the first memory domain also serves as timing uncritical domain for all pipeline stages except for the `EX` stage that contains the ALU. The ALU is then split into two halves to accommodate for two categories of operations with different timing characteristics.
Fig. 7.1 gives a schematic representation of the partitioning into body bias domains. For more precise planning as well as for the tapeout including the body bias supply planning as part of power planning, a domain naming scheme has been introduced. As shall later be described, the design is partitioned into blocks, where the largest block is a column consisting of two PEs, upper and bottom PE. Then there are two memory and two ALU domains within the PEs. The naming scheme then first addresses the column, and then the PE as `CNL`, where `N` is either 0 or 1 (the array consists of 4 PEs) and `L` represents the location, `U` for upper or `B` for bottom.

This particular instance of MuCCRA4 is equipped with a memory fitting 32 contexts and a 8 entries register file per PE. The context memories are split in half into a lower part (`MEML`) and an upper half (`MEMH`). The lower half additionally contains the register-file as well as the data memory. This is done based on the principle of activation and timing as proposed in chapters 5 and 6 while restructuring the HDL with pre-partitioned modules according to the result of the fine-grained approach from section 5.4. When a single PE shall be used, some configuration is required, otherwise, it cannot be used at all. To allow for general operation, the register-file, internal data multiplexing structures, the interconnect and data memory is required. While the data memory is the biggest data structure, unlike the context memories, it is usually used as a whole to have sufficient data buffers. Thus, additional partitioning is not sensible here. For the register-file, on the other hand, partitioning would very well be a tempting application, especially in view of MuCCRA4's multithreading capabilities. However, unlike context memory which is just utilized incrementally as required, partitioning the register-file would introduce additional timing dependencies. With larger register-files, partitioning would be sensible as there is more room to prevent such dependencies, for an 8 entries file. However, the possible savings are somewhat limited compared to the overheads. Thus, in sum `MEML` is the essential body bias domain which has to be active to use the PE. This covers the activation side, but also in regard to timing, the aforementioned components are incredibly similar in this regard as well. Thus, this makes a sensible partitioning choice. For the first PE, that is `C0U`, following the above scheme, this idea is extended by grouping all global controllers into its `MEML` domain as well. In this case, however, this just means that the body bias supply of `C0U`'s `MEML` partition is reused for all standard cells placed outside PE areas. The upper half of the context memories are consequently put into a different body bias domain, `MEMH`. Many applications only consist of small loops and thus do not need more than 16 contexts. In this case, the partitioning of context memory into two domains allows to virtually cut off the upper half and thus increase energy efficiency.

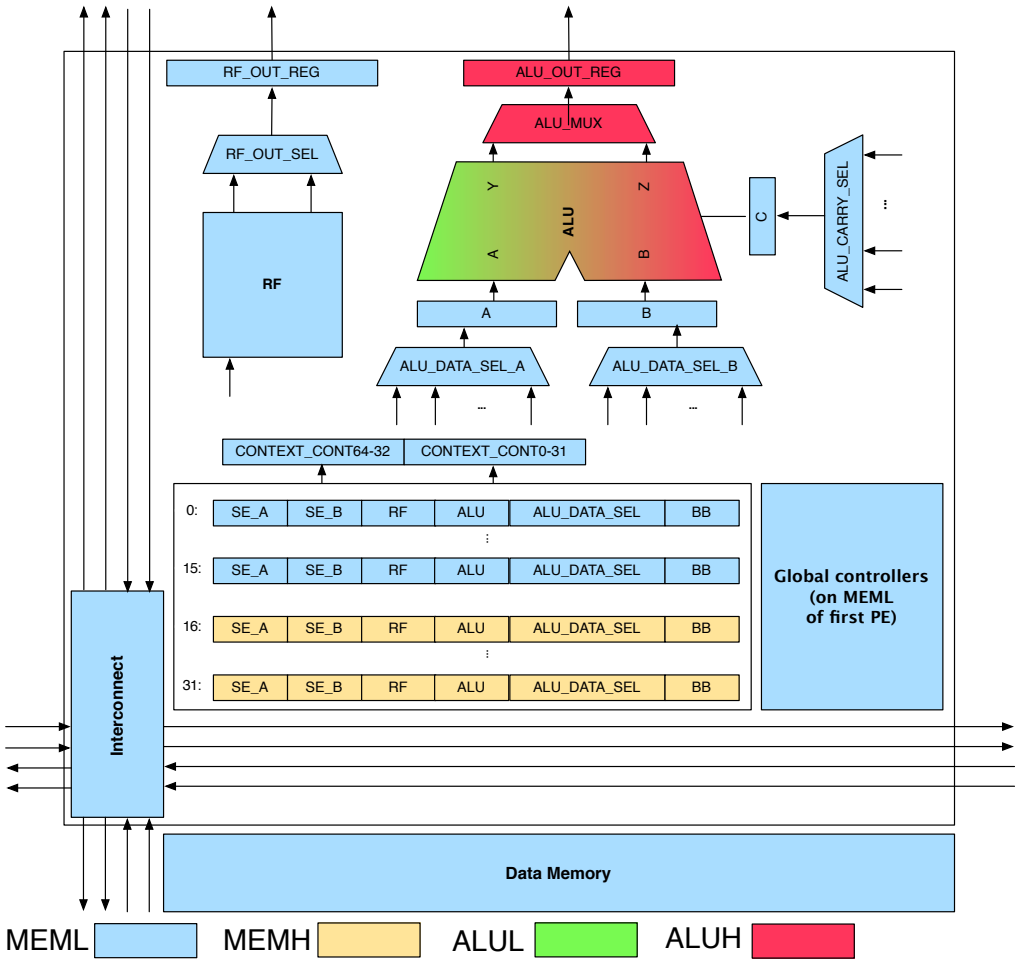


Figure 7.1: Body bias domain partitioning of a MuCCRA4 PE into four body bias domains ($k = 4$)

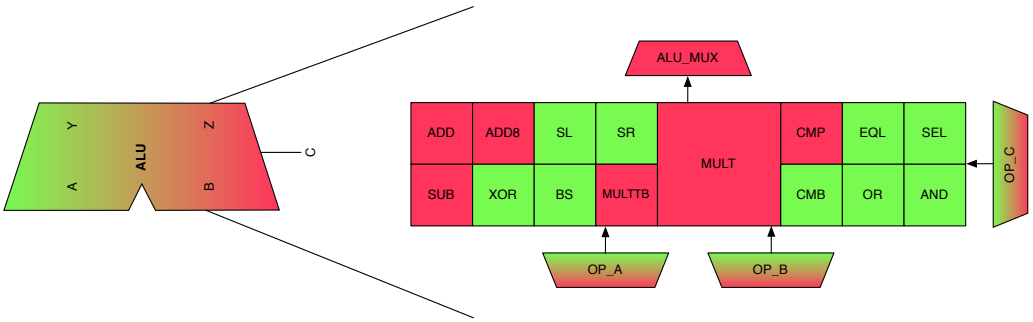


Figure 7.2: Body bias domain partitioning of the ALU into two body bias domains `ALUL` and `ALUH`

The central and most efficient partitioning is that of the EX stage. Here, the ALU and ALU multiplexers are partitioned into two domains ALUL and ALUH. The ALU is first analyzed using the approaches presented in chapter 5.4 as well as sections 5.4.1 6.2, 6.3 and 6.5. This yields a partitioning along the lines of logical and arithmetical functions of the ALU, as well as its output multiplexer functionality as visualized in 7.2. While logical functions are in general timing uncritical and thus put into ALUL, arithmetical functionality is put into ALUH. As last part on the critical path of the ALU, the output multiplexer is grouped into ALUH along with the arithmetical functions. Both the approaches described in chapters 5.4 and 6 led to the same partitioning.

In total, this partitioning results in the body bias domain table 7.2.

PE	Coord.	Domain Name	Type	Description
PE0	0,0	C0UMEML	MEML	Lower memory part including all global controllers
		C0UMEMH	MEMH	Upper part of the context memory
		C0UALUL	ALUL	ALU logical functions
		C0UALUH	ALUH	ALU arithmetic functions and output multiplexer
PE1	1,0	C1UMEML	MEML	Lower memory part
		C1UMEMH	MEMH	Upper part of the context memory
		C1UALUL	ALUL	ALU logical functions
		C1UALUH	ALUH	ALU arithmetic functions and output multiplexer
PE2	0,1	C0BMEML	MEML	Lower memory part
		C0BMEMH	MEMH	Upper part of the context memory
		C0BALUL	ALUL	ALU logical functions
		C0BALUH	ALUH	ALU arithmetic functions and output multiplexer
PE3	1,1	C1BMEML	MEML	Lower memory part
		C1BMEMH	MEMH	Upper part of the context memory
		C1BALUL	ALUL	ALU logical functions
		C1BALUH	ALUH	ALU arithmetic functions and output multiplexer

Table 7.2: Body Bias Domains of MuCCRA4-BB

7.2 Macro-based Body Bias Domain Implementation

To ease the implementation process and to realize the individual body bias domains, the major components of a PE have been implemented as macros. This allows to treat potential DRC errors more locally and is a proven way to reduce problem sizes. In this particular implementation, it also helps greatly to align body bias supplies as well. Starting with the innermost body bias domain, then continuing on the next hierarchical level surrounding the previous domain leaves an eye-shaped silhouette. Thus, we called this implementation style "eye-style" implementation.

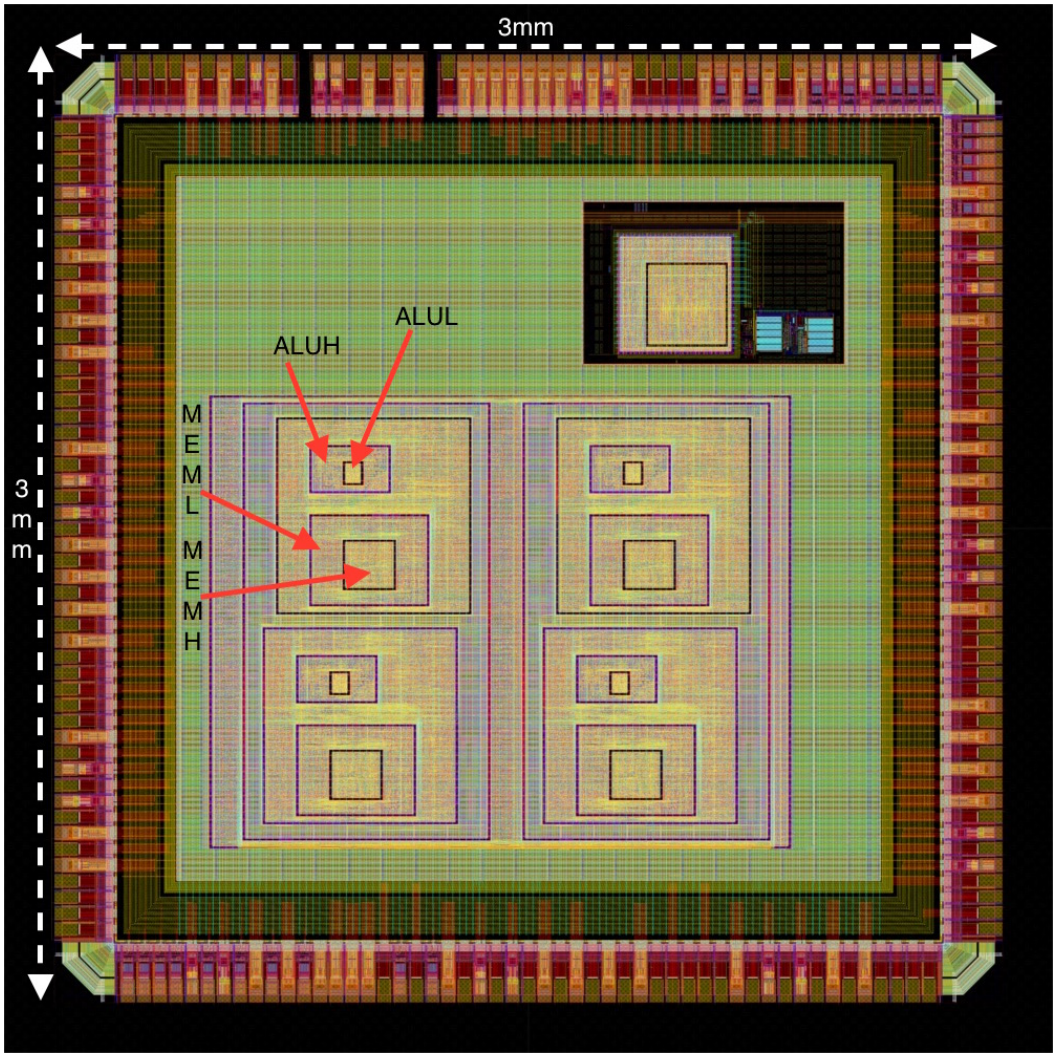
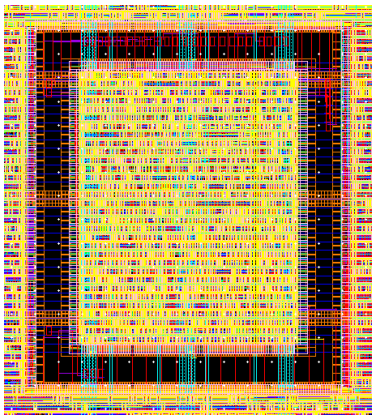


Figure 7.3: Layout of the MuCCRA4-BB test-chip with 16 body bias domain (4 per PE) and a body bias generator test-circuit in the top-right, manufactured in Renesas' 65nm SOTB

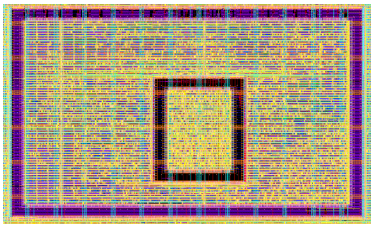
Fig. 7.3 depicts the layout of the taped out MuCCRA4-BB chip with annotated body bias domains MEML, MEMH, ALUL and ALUH in the C0U PE. The eye-style implementation process is visualized in Fig. 7.4. This particular implementation starts out with the innermost, usually smallest design part and then moves further outwards to construct the entire component. During each of the macro creation steps, tap-cells for the particular domain under construction need to be placed in defined x-intervals if standard-cells are placed during macro creation. This facilitates the body bias supply to the standard-cells' wells. Furthermore, when placing thereby created macros, a separation margin has to be added to insulate one body bias domain from the other.

Following Fig. 7.4, the ALU is built first. In this case, the innermost design part is the ALUL domain. For this domain, a macro is then built (Fig. 7.4 a)). Then, the surrounding ALU macro is built. Of course, this only makes sense when the engulfed macro is used as a part of the surrounding design, which obviously is the case here as ALUL contains the logical functions of the ALU. After building the ALU macro with the ALUH domain, this eye style macro can again be used as a macro (Fig. 7.4 b)) to instantiate an ALU in the next higher hierarchy level. In Fig. 7.4 c), the innermost macro is the upper half of context memories. This macro, in turn, is placed in the MEMORY macro along with the MEML domain for the lower half of context memories and the register file (Fig. 7.4 d)). Then, the ALU and MEMORY macro are combined in the PE macro, whereas the data memory is placed in the surrounding area with its tap-cells connected to the MEML domain supply of the PE (Fig. 7.4 e)). Therefore, despite not being placed in the same macro, by shorting the supplies, macros merge into the same domain. Two of these PEs are then placed to form a column of the PE array. As shown in Fig. 7.4 f), the two PE macros are placed with a slight x-offset. This is done to leave some space for the straps supplying the body bias. As both PE macros would have identical x-coordinates for their tap-cell rows, they also would have to be connected by a strap on the same x-coordinate. This situation is exacerbated as all macros also need to carry straps for all other domains that overlap on x-coordinates. Thus, a slight x-offset eases this task.

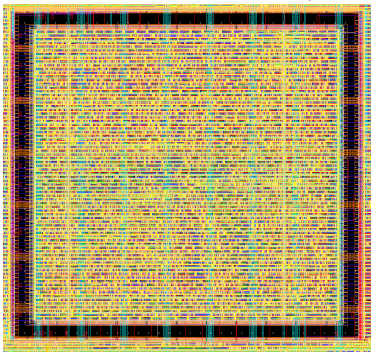
The resulting macro sizes are listed in Tab. 7.3. Ultimately, this implementation style is driven by routing and the need for fine-grained body bias domains specific to DRPs. To prevent massive routing overheads by placing macros alongside, the macros are placed where they are required, in this case, inside another macro.



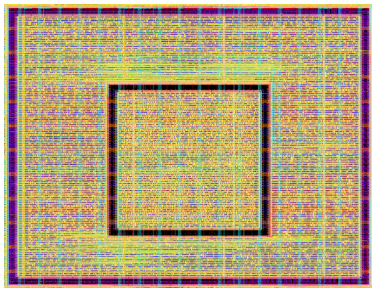
a) ALUL macro with domain ALUL



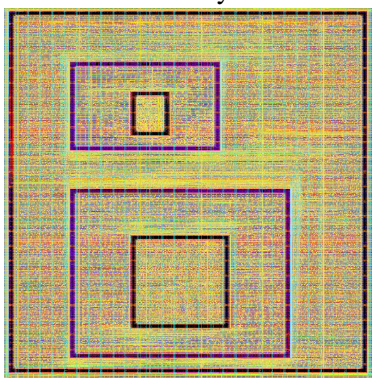
b) ALU macro with domains ALUH and ALUL in the eye



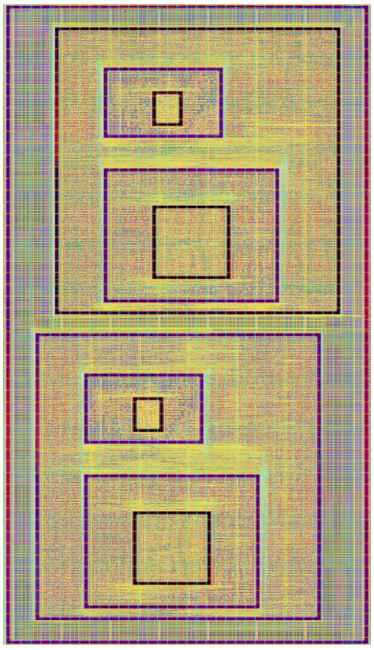
c) Macro MEMH with domain MEMH



d) Macro MEMORY with domains MEML and MEMH in the eye



e) Entire PE



f) PE Array Column

Figure 7.4: Eye-style implementation from ALUL into ALUH, MEMH into MEML, MEML and ALUH into a PE and finally PEs into a column

Name	Width [μm]	Height [μm]	Area [μm^2]
ALUL	50.18	61.2	3071.016
ALU	240.24	136.8	32864.832
MEMH	151.84	144.0	21864.96
MEMORY	360.1	273.6	98523.36
PE	595.14	601.2	357798.168
COL	1386.96	792.55	1099235.148

Table 7.3: Resulting MuCCRA4-BB macro sizes

7.3 Supported Body Biasing Schemes

The fine-grained implementation of body bias domains described above in sections 7.1 and 7.2 allows for the real-chip evaluation of core-, coarse- and fine-grained body biasing. As each PE is divided into four body bias domains, the extent of the partitioning is locally restricted to each PE. Thus, not only fine-grained schemes below PE granularity can be applied, but also coarse-, i.e., PE-grained body biasing, clusters of PEs as well as on the entire DRP, i.e., core-grained body biasing. This can be achieved by electrically shorting respective body bias domains' bias supplies. Table 7.4 gives an overview of the supported and sensible body bias domain partitionings. Again, sensibility here refers to the defined criteria for body bias domain partitioning, i.e., activation and timing. Of course, more than the schemes described in table 7.4 could be evaluated with this test-chip implementation, however, as there is no benefit to be expected, they were not included in this list. For instance, if one ALU domain and one memory domain would be shorted together, the chances are that one does not need to be active at all while the timing argument then finally rules out sensibility.

Name	#BBD per PE	Total #BBD	Description
FGBB	4	16	Context memory split in half, ALU partitioned into two domains
FGBB-H	2	8	Memory and ALU make up one domain each
CGBB	1	4	One domain per PE
CGBB-2	0.5	2	Two PEs forming a domain
GBB	0.25	1	Entire DRP forming one domain

Table 7.4: List of supported body bias domain partitionings for evaluation purposes

7.4 Bias Supply Network

All body bias domains receive their body bias potentials VBN and VBP from ring structures along the IO ring, which, in turn, are fed using power pins on the IO pad. Most IO pads on the left and right side are body bias supplies as depicted in the layout Fig. 7.3. As body biases are not multiplexed in this test-chip as in [70, 71], the body biases are supplied directly to the concerned domains. As described in the previous sections, tap-cells feed the body bias potential to the transistor wells. To enable the tap-cells to do this, each tap-cell has to be connected to the supplies for the NMOS (VBN) and PMOS (VBP) body bias supplies respectively. Therefore, for each row of tap cells, two vertical straps need to be drawn from supply structures. The body bias supply style used in this test-chip is visualized in Fig. 7.5.

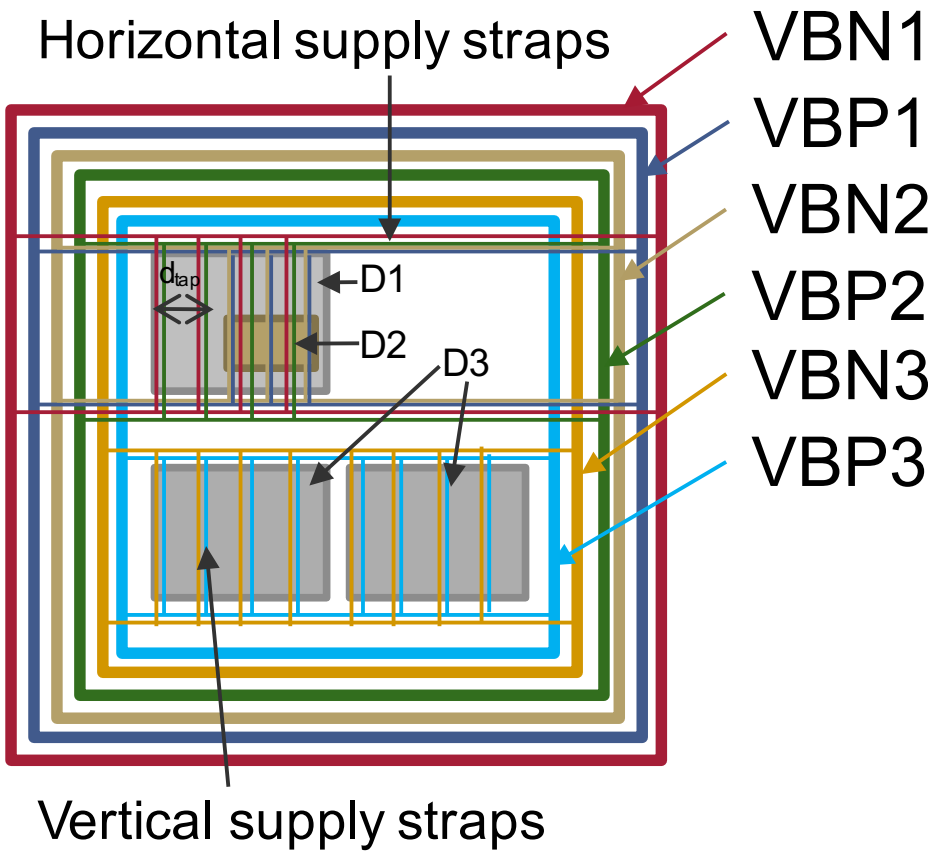


Figure 7.5: Cartoon representation of body bias supply mesh for three domains, $D2$ in $D1$ with overlapping straps and third domain $D3$

In this particular example, body bias supplies for three domains are drawn. As a realistic example close to the implementation style used in this tapeout, consider domains $D1$ and

$D2$. This is an example for the eye-style body bias domain construction where $D2$ is placed inside $D1$. $D3$, on the other hand, is an example for a regular, core-like body bias supply. In both cases, horizontal straps are first drawn from the body bias supply rings which are directly connected to the power pads supplying the bias potentials. Using these horizontal straps, vertical straps can be drawn in regular tap-cell intervals on the x-coordinate that match the body bias supply pins VBN and VBP of the tap-cells. Thus, in the case of $D1$ and $D2$, two pairs of body bias supplies are needed, totaling in four different potentials and eight horizontal straps to restrict the vertical straps to the area between the horizontal straps on top and below the macros of $D1$ and $D2$.
Furthermore, considering this example, $D1$ may have standard cells placed below and above the $D2$ macro. Thus, vertical straps need to be drawn through $D2$ as well, as straps always have to be drawn between two power structures of the same supply. For that reason, a horizontal strap is drawn from one side of the red ring to the opposite side and vertical straps are then drawn from one red horizontal strap to the red horizontal strap on the opposite side of the macro. Thus, this results in three requirements for the macro implementing $D2$:

1. Vertical straps for the body bias domains between horizontal supply straps that overlap on x-coordinates have to be drawn as well
2. Straps need to be aligned with the straps of those concerned domains
3. The macros or the straps of the placed macros need to be placed or aligned in a manner that they are not shorted or in conflict while maintaining the tap-cell interval

These rules are visualized in Fig. 7.5 and an example is given in Fig. 7.6. Reconsider $D2$ in $D1$. The body bias of $D1$ is supplied via $VBN1$ (red) and $VBP1$ (green), while the body bias of $D2$ is supplied via $VBN2$ (beige) and $VBP2$ (dark blue). In $D1$, there are first two pairs of $VBN1$ $VBP1$, while they are continued to be drawn as part of $D2$. Furthermore, the x-coordinates of the strap pairs in $D1$ and $D2$ are perfectly aligned, so they pass right through $D2$. Furthermore, rule 3 is also maintained as $D2$ is aligned in a manner that $VBN2$ and $VBP2$ are drawn in the specified interval d_{tap} without causing a conflict with $VBN1$ and $VBP1$.
Moving on to the example of the ALUL macro, these rules were adhered to in a similar manner. As macro ALUL is placed inside the ALU macro, not only supplies for body bias domain `ALUL` are required, but also those for `ALUH` as standard cells of the ALU macro in the `ALUH` domain may be placed around the ALUL macro. Additionally, as horizontal supplies are drawn above and below each PE, straps for domains `MEML` and `MEMH` have to be drawn as well, in case their x-coordinates overlap. In this case, they do overlap, and hence, despite only one tap-cell row is placed inside the ALUL macro, requiring one set of straps, the three other traps are drawn. Furthermore, these straps are aligned with the tap-cell interval of the other domains and are not causing any conflict, i.e. short circuits with other supplies.

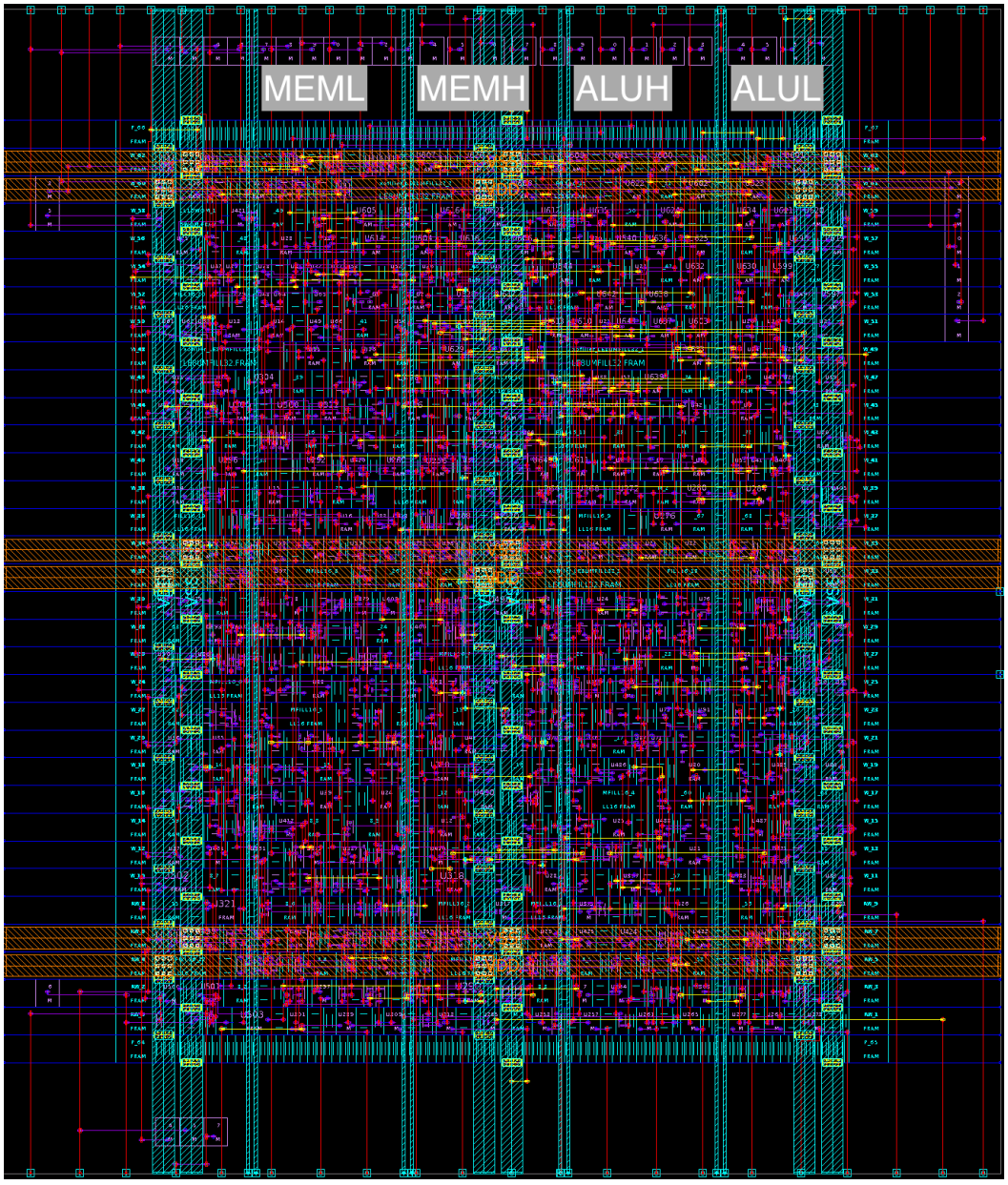


Figure 7.6: Schematic layout view of the `ALUL` partition, implementing logic function, with aligned straps for domains `MEML`, `MEMH` and `ALUH` of the body bias supply mesh passing through and only `ALUL` straps connecting to the proper tap-cells

7.5 Evaluation Environment

The returned dies were packaged and bonded in 208 Pin QFP packages and evaluated in an environment as illustrated in Fig. 7.7. The actual evaluation environment is displayed in the photograph of Fig. 7.8. The motherboard has been custom designed to fit the special requirements of the taped out chip. First and foremost, the chip requires a great number of power supplies to supply body bias voltages to the individual body bias domains. To facilitate this, six four channel power supply boards can be used on this motherboard, where each is controlled using a USB to UART bridge on the MicroZed board. The actual control is then exercised on a computer controlling the experiments. The actual stimulation and debugging are done using the MicroZed board which reconfigures the MuCCRA4-BB, starts the program execution as well as facilitates the readout. All data transfers to and from the MuCCRA4-BB are done using a simple custom serial protocol. For this purpose, the FPGA portion of the Zynq chip is utilized and programmed using Xilinx's Vivado 14 suite.

Additionally, for precision measurements and additional power supplies, an Agilent N6507A power analyzer, Agilent 34410A multimeters and Agilent E3631A power supplies were used. A typical experiment is set up by using the power analyzer to supply digital core V_{DD} as well as other critical supplies that need to be monitored for a detailed record of their power consumption, such as IO power supply and voltage level-shifter supplies. Body bias voltages are usually supplied using the programmable voltage generator boards which allows a temporal granularity of up to $1\mu s$. With each body bias domain requiring two supplied voltages, VBN and VBP, with a fully equipped motherboard, the setup can bias twelve body bias domains while the remaining four would have to be supplied using one of the lab power supplies. With the symmetry found in DRP applications, this is usually not required.

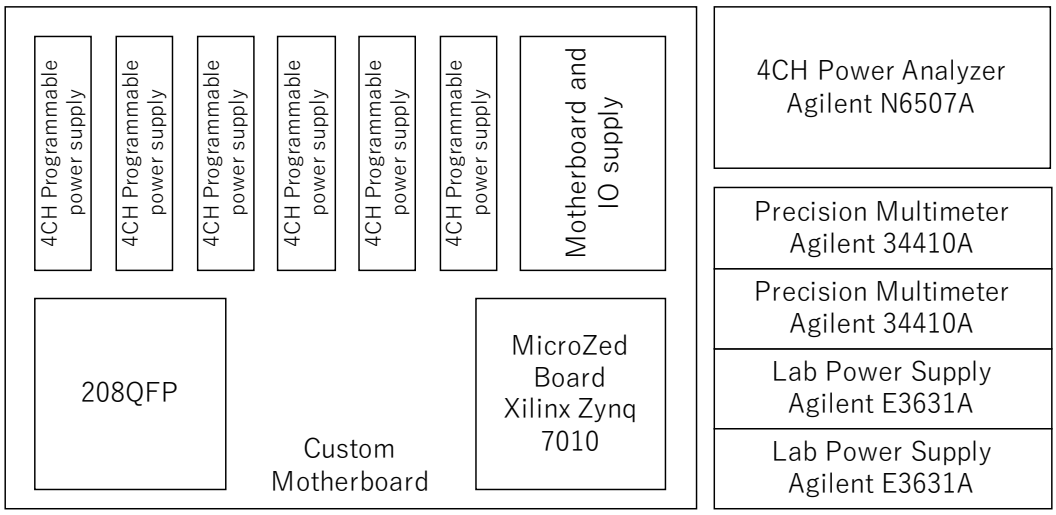


Figure 7.7: Evaluation environment consisting of the motherboard with the QFP208 socket TEG board, MicroZed Zynq board, and power supplies.

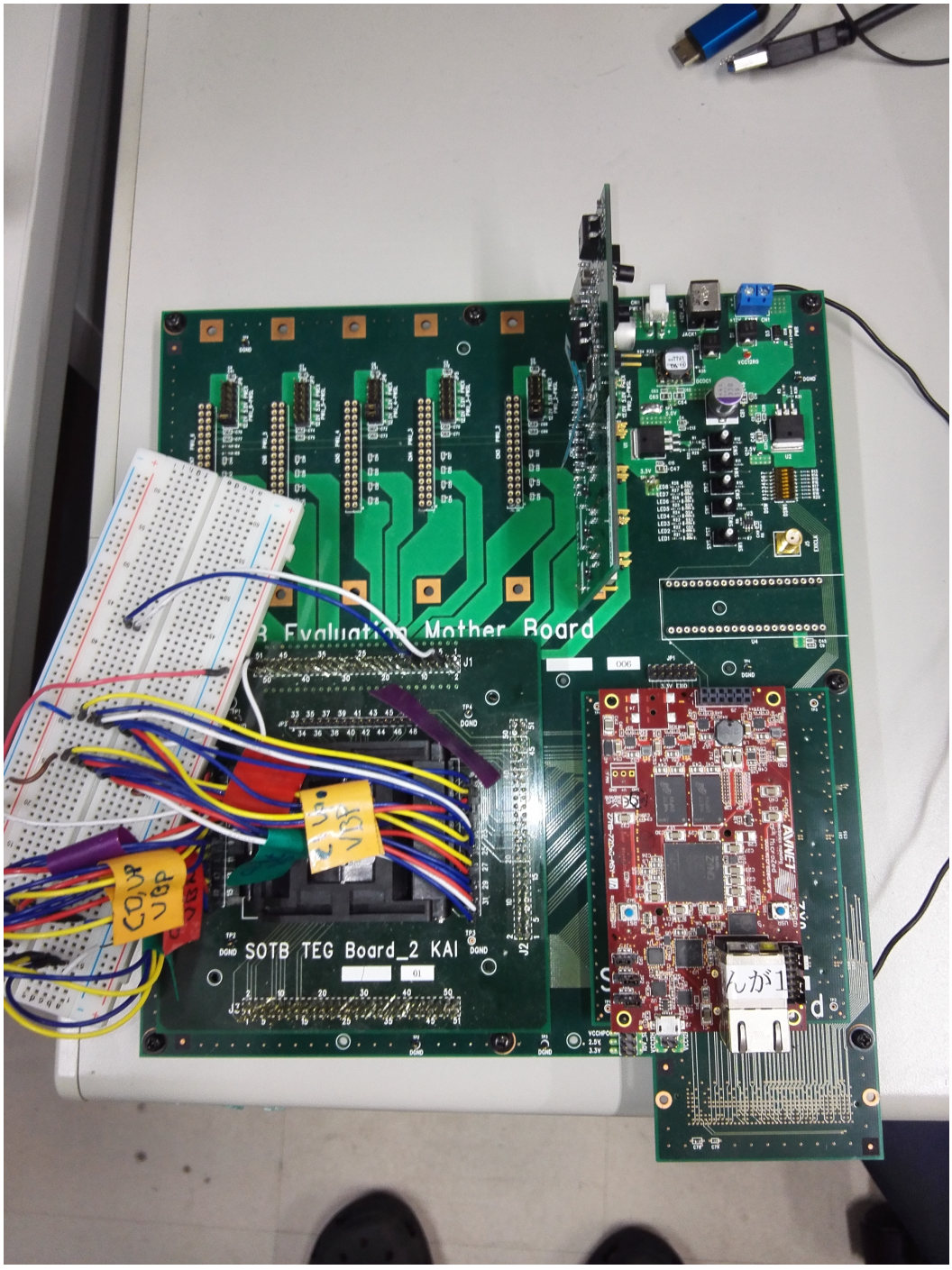


Figure 7.8: Photograph of the evaluation motherboard with one powerboard and a MicroZed board to control and coordinate experiments

8 Results

In this chapter, evaluations and the results for each proposed approach will be presented. The result chapter is split into two principal parts:

1. Results generated through simulation (Section 8.1)
2. Results generated through test-chip evaluation (Section 8.2)

whereas each of these parts is then further divided by the granularity of the evaluated approach. Each part will cover core-, coarse- and fine-grained body biasing, whereas in the simulative section standard-cell grained body biasing based on the DCE approach (chapter 6) shall be covered as well. After simulation and chip evaluation results, this chapter will close with a discussion of the results.

All simulative results are obtained using STMicroelectronic's 28nm UTBB-FDSOI in both RVT (regular threshold voltage) and LVT (low threshold voltage) flavors. Standard cell-grained body biasing results are additionally obtained in Renesas 65nm SOTB FDSOI as well. Test chip results obviously are only obtained in the technology used for manufacturing, which is Renesas 65nm SOTB.

The design used for evaluations is always the MuCCRA4 DRP architecture with varying array sizes. For results obtained through simulation, a 4×4 MuCCRA4 instance has been used, while the test-chip was manufactured with a 2×2 PE array, using the exact same architecture. These designs are then evaluated using the corners, i.e., design targets specified in section 4.4. Application specific approaches are evaluated for five different benchmark applications:

- An alpha blender `alpha`
- Discrete cosine transformation `DCT`
- A 24 tap finite impulse response filter `FIR`
- Sum of absolute differences `SAD`
- Sepia image filter `sepia`

These applications use a variety of operations and thus exploit different aspects of the MuCCRA4 DRP architecture. Depending on the particular instance used, the applications are mapped on either the 4×4 or the 2×2 variants.

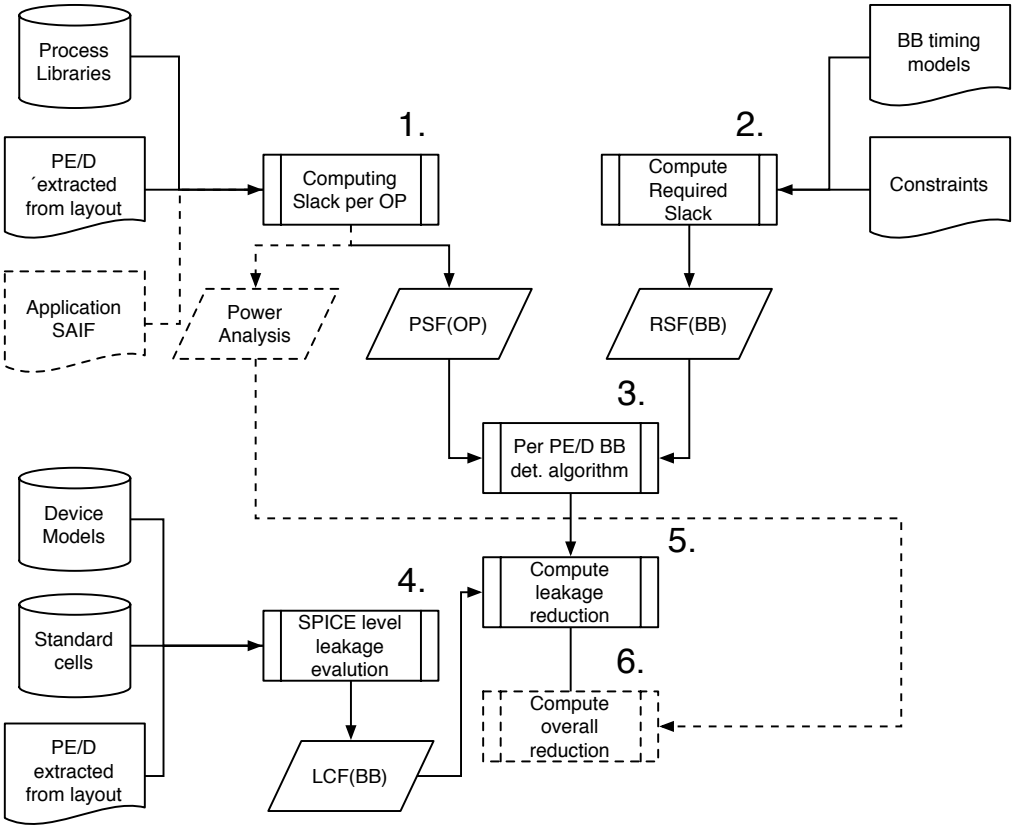


Figure 8.1: EDA tool-assisted evaluation flow for core-, coarse- and fine-grained body bias domain partitioning approaches as presented in chapter 5

8.1 Simulative Results

Simulation results have been obtained using two principal evaluation flows. The first flow is depicted in Fig. 8.1. This flow is characterized through the consequent usage of industry standard EDA tools. Such tools were at the time, and still are oblivious to freely ranging body biasing, i.e., tools only acknowledge body bias as an additional characterization point like another supply voltage or temperature. Thus, workarounds for this deficiency were required.

The evaluation for core-, coarse- and fine-grained body biasing approaches start out in 1. of Fig. 8.1 by determining the slack per operation or functionality using zero-bias characterized process libraries. This is done in an industry standard static timing analysis tool (Synopsys PrimeTime [106]) which is also used to run power analyses which may be later used for overall power reduction computation, which, however, is not strictly necessary for this evaluation. The primary result of this step is the per-op slack function $PSF(OP)$, giving timing slack as a function of the operation. In step 2., the amount of slack per body bias level is then computed using constraints such as the target clock

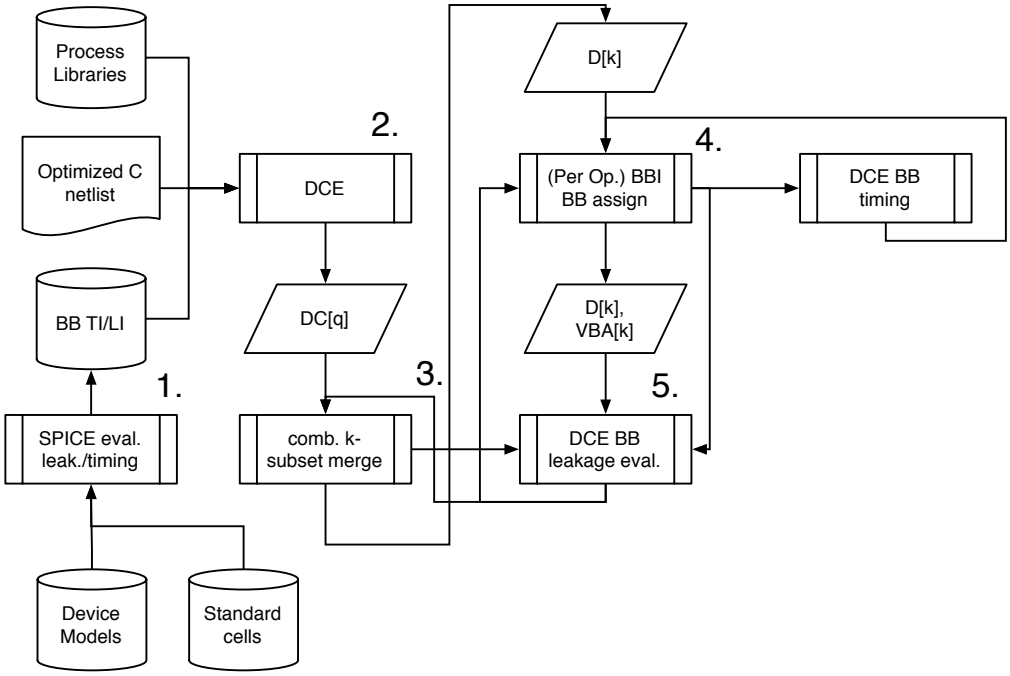


Figure 8.2: Domain Candidate Exploration based evaluation flow for standard-cell grained body bias domain partitioning, utilizing SPICE supplemented custom timing- and leakage computation

period t_{clk} and body bias ranges, as well as body bias timing models obtained through the evaluation of the effect of body biasing on ring oscillator frequencies. This then allows computing the required slack function per body bias level $RSF(BB)$. These two functions are used in step 3 for the algorithms proposed in sections 5.2, 5.3 and 5.4 to determine the required body bias levels to meet the timing constraint. Continuing in step 4., the PEs' or domains' netlists are evaluated for body bias dependent leakage. This has to be done by using SPICE simulators as only device level simulation allows for freely ranging body biases. Thus, the library characterization limitation has been overcome. These characterizations per PE or body bias domain (D) are then used in step 5. to compute the actually incurred leakage by adding the individual PEs or body bias domains' leakage currents at the specified body biases. In the optional step 6., the overall power consumption can be computed, if in step 1. dynamic power consumption has been computed as well.

The evaluation flow (Fig. 8.2) for the DCE standard-cell grained approach differs considerably, as DCE implements body bias aware timing and leakage computations. It is mostly independent of EDA tools after the required information has been obtained. This information gathering step is visualized as step 1. in Fig. 8.2. This step generates scaling factors for timing and leakage by evaluating ring oscillators for the timing impact, as well as standard-cells for the leakage impact of body biasing. These factors are normalized against zero-bias values and used in DCE subsequently. Step 2. then is the actual DCE

execution which results in the domain candidates $DC[q]$. Afterwards, they are merged in step 3. to form k body bias domains $D[k]$ using DCE's body bias aware leakage computation. Then, in step 4., similar to step 3. of Fig. 8.1, the body bias levels per domain are determined using the BBI based body bias determination algorithm. This results in a body bias assignment $VBA[k]$ per body bias domain $D[k]$, which is then used in step 5. to compute overall leakage. This is done per standard cell by using the leakage values of the standard-cell library, scaled with the body bias dependent leakage factor determined in step 1. of the respective body bias domain.

8.1.1 Core-Grained Body Biasing

Core-grained body biasing is limited in its extremely coarse application of body biasing. Thus, the effects that can be yielded are usually also accompanied by extreme side-effects. If e.g., speed is boosted with forward body bias, the leakage penalty is severe, while huge leakage reductions come with a reduction of maximum clock frequency by more than a factor of $1.5\times$.

Consider the results for STMicroelectronic's 28nm UTBB-FDSOI in Figs. 8.3 to 8.8. In the first three figures, RVT is evaluated for maximum reverse body bias, indicating the potential maximum leakage reduction at the cost of clock frequency, maximum forward body bias which is severely limited in RVT, as well as per application evaluations for the LP corner, i.e., reducing leakage while staying at the maximum clock frequency F_{max}. The maximum forward body bias, HP corner evaluation per application is omitted here, as the primary use case of RVT is low power applications with extended reverse body biasing capabilities.

As discussed in the background chapter, the effect of body biasing is relative to the supply voltage at which the transistors are operated. This relationship is also visible when comparing Fig. 8.3 to Fig. 8.4 and Fig. 8.5. The impact of maximum reverse body biasing on delays decreases with increasing supply voltage. This, however, should not give the impression that greater supply voltages should thus be used, on the contrary, leakage increases with supply voltage more than it could possibly be reduced with reverse body biasing if the maximum clock frequency is scaled up as well. Furthermore, depending on the most timing critical operation executed in the entire DRP, there are different levels of leakage reduction in the application dependent evaluation. While all applications but the `FIR` filter allow for significant leakage reduction, only at $V_{DD} = 1.0V$ a little leakage reduction is feasible as the timing impact decreases. In the case of `FIR`, the critical path, i.e., full-width multiplication is used, and therefore, with the aforementioned exception, leakage cannot be reduced. By leveraging application dependent timing slack, leakage can be significantly reduced, even when only applying body bias locally on an entire core.

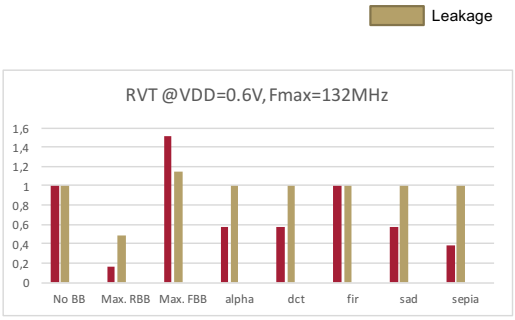


Figure 8.3: Core-grained evaluation of MuC-CRA4 in the RVT flavor at $V_{DD} = 0.6V$ for Max. RBB and LP corners by application

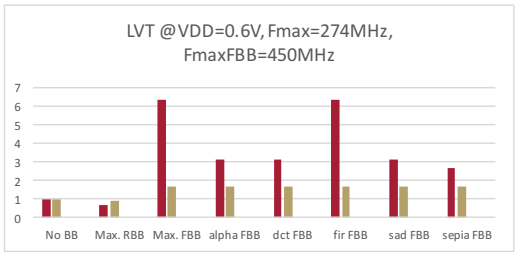


Figure 8.6: Core-grained evaluation of MuC-CRA4 in the LVT flavor at $V_{DD} = 0.6V$ for Max. RBB and HP corners by application

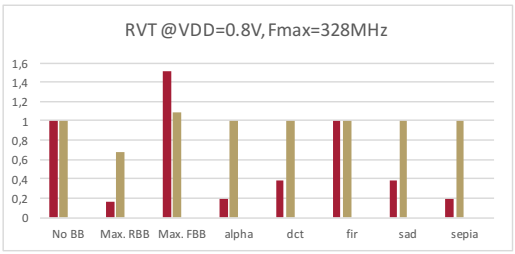


Figure 8.4: Core-grained evaluation of MuC-CRA4 in the RVT flavor at $V_{DD} = 0.8V$ for Max. RBB and LP corners by application

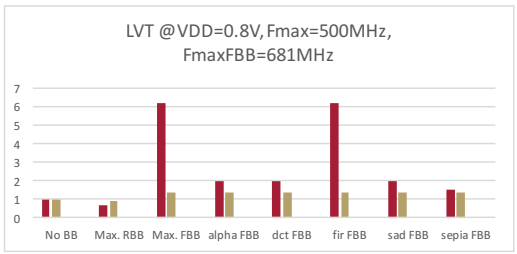


Figure 8.7: Core-grained evaluation of MuC-CRA4 in the LVT flavor at $V_{DD} = 0.8V$ for Max. RBB and HP corners by application

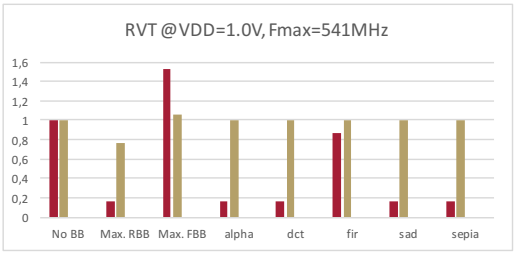


Figure 8.5: Core-grained evaluation of MuC-CRA4 in the RVT flavor at $V_{DD} = 1.0V$ for Max. RBB and LP corners by application

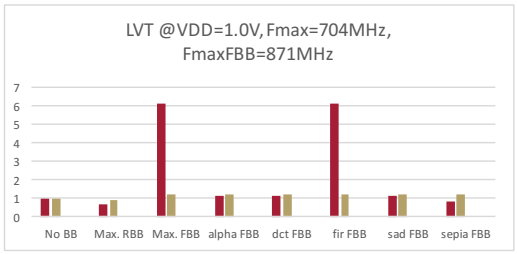


Figure 8.8: Core-grained evaluation of MuC-CRA4 in the LVT flavor at $V_{DD} = 1.0V$ for Max. RBB and HP corners by application

From the standpoint of performance and high energy efficiency, the LVT flavor is of greater interest than RVT. LVT has a lower threshold voltage and extended forward body bias capabilities. Similar to RVT, the focus in regard to the application evaluation is thus also put solely on the HP corner, omitting the LP corner which benefits mainly from reverse body biasing capabilities. LVT and its strong forward body biasing capabilities are predestined to be used for frequency scaling. Thus, Figs. 8.6 to 8.8 giving the core-grained results for the LVT flavor state two different clock frequencies:

- F_{max} the regular, unscaled maximum clock frequency

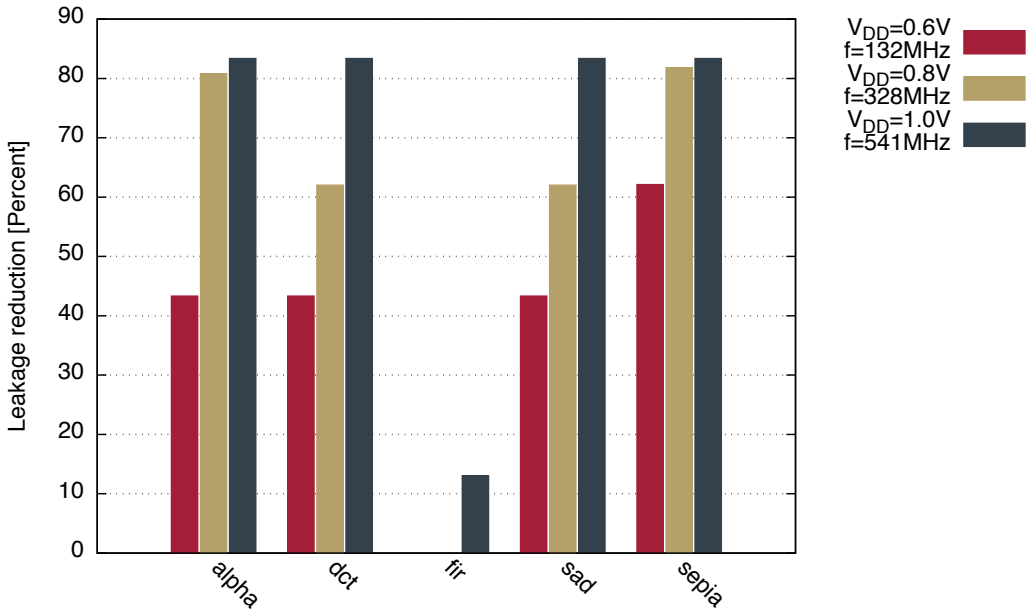


Figure 8.9: Leakage reduction obtained by leveraging application dependent timing slack on a core-grained granularity in STMicroelectronic's 28nm UTBB-FDSOI RVT flavor for supply voltages $V_{DD} = \{0.6V, 0.8V, 1.0V\}$ at their respective maximum clock frequencies.

- $F_{max,FBB}$ the maximum clock frequency scaled with the forward body bias boost

Opposite to RVT, LVT has only slight reverse body biasing capabilities and thus maximum reverse body bias' effect is limited for both leakage and delay. Furthermore, just as with LVT, the effect on delay is relative to supply voltage, decreasing with increasing V_{DD}. With forward body biasing a lot stronger than in RVT, the leakage penalty is also much more severe as visualized in the Max. FBB bars in Figs. 8.6 to 8.8. Thus, with leakage impact that strong, it is imperative to apply forward body bias in a more sensible manner. The first step towards this goal is to apply it according to the timing slack as visualized in the applications' bars. Only with the `FIR` filter, the leakage penalty could not be mitigated, while all other applications do not use such timing critical operations and thus do have less aggressive timing constraints to meet.

With all this being said, similar to the body bias and supply voltage relationship, LVT has, per construction, far greater leakage currents to begin with. Thus, forward body bias does not only increase leakage but also exacerbates a phenomenon which is considered problematic since at least 90*nm* nodes. This becomes most visible when putting the obtained results in a larger context. Consider, for example, the core-grained body bias leakage savings achieved using the RVT flavor (Fig. 8.9). Despite the biasing of an entire core, in RVT, leakage can be reduced even when there is very little slack to leverage.
Of course, as demonstrated in the `fir` case, when there is no slack to leverage, no leakage improvement can be obtained. This situation, however, presents itself only when running

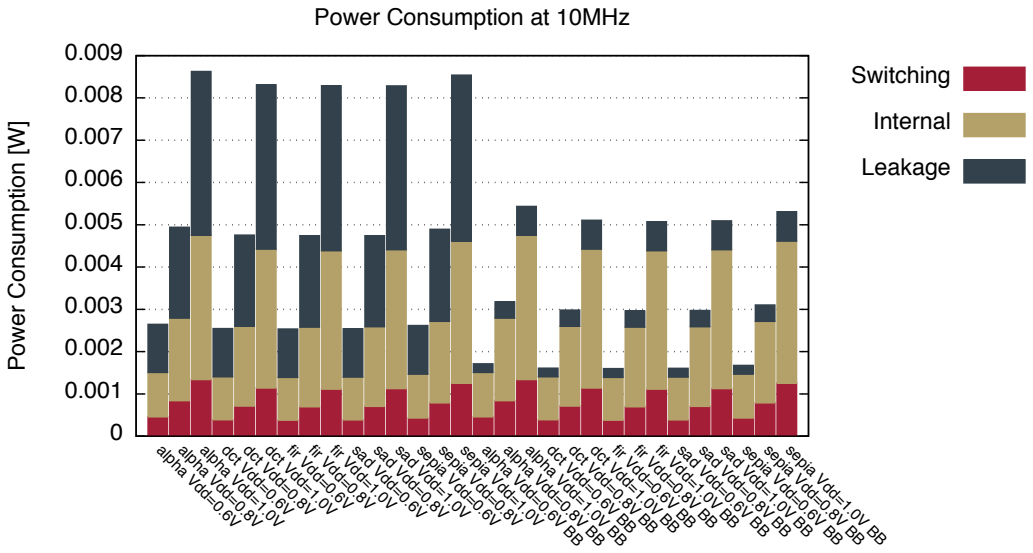


Figure 8.10: Total power reduction in a ultra-low-power scenario for the five benchmark applications executed at $F = 10MHz$ and evaluated for supply voltages $V_{DD} = \{0.6V, 0.8V, 1.0V\}$ without body biasing (left half) and with the proposed reverse body bias application (right half)

close to or right at the maximum clock frequency F_{max}. Consider an example for an ultra-low-power execution mode as depicted in Fig. 8.10, displaying total power consumption for the benchmark applications at supply voltages $0.6V$ to $1.0V$, with no body biasing on the left and with reverse body biasing on the right side.

It is immediately visible that in such modes, even in the RVT flavor, leakage may comprise more than 30% or more of total power consumption. Especially for battery powered or always-on devices, the utilization of this additional degree of power management is imperative. Considering LVT in a broader context, the focus switches to energy efficiency. Fig. 8.11 depicts total power consumption split up by switching, internal and leakage power consumption with an additional graph overlayed for energy delay product (EDP) for two modes: zero-bias unscaled (zero) and clock frequency scaled using core-grained, i.e., global body bias (GBB). Results are further ordered by application, where each application block is structured by pairs of zero-bias and core/global grained body bias (GBB) with ascending supply voltage. For each operation point, the maximum clock frequency is utilized. This means that e.g., at $0.6V$ without body bias, the maximum achievable clock frequency is $274MHz$, while with body bias it is $450MHz$ for this design and is thus used for the presented power evaluation. See Tab. 8.1 for a list of supply voltages and the attainable maximum clock frequencies without and with maximum forward body bias.

This representation allows to directly compare the power consumption of forward body bias scaled modes to supply voltage scaled modes. This comparison is especially of interest as scaling with forward body bias omits the quadratic relationship between supply voltage and power consumption, and thus may be a very energy efficient frequency scaling alternative to supply voltage scaling. However, the exponential increase in leakage

Supply Voltage V_{DD}	Max. Frequency	Max. Frequency with max. FBB
$0.6V$	$274MHz$	$450MHz$
$0.8V$	$500MHz$	$681MHz$
$1.0V$	$704MHz$	$871MHz$

Table 8.1: Maximum clock frequency unscaled and scaled with maximal forward body bias for the MuCCRA4 architecture implemented in STMicroelectronic's 28nm UTBB-FDSOI LVT flavor

Supply Voltage V_{DD}	Max. Frequency
$0.6V$	$132MHz$
$0.8V$	$328MHz$
$1.0V$	$541MHz$

Table 8.2: Maximum clock frequency for the MuCCRA4 architecture implemented in STMicroelectronic's 28nm UTBB-FDSOI RVT flavor

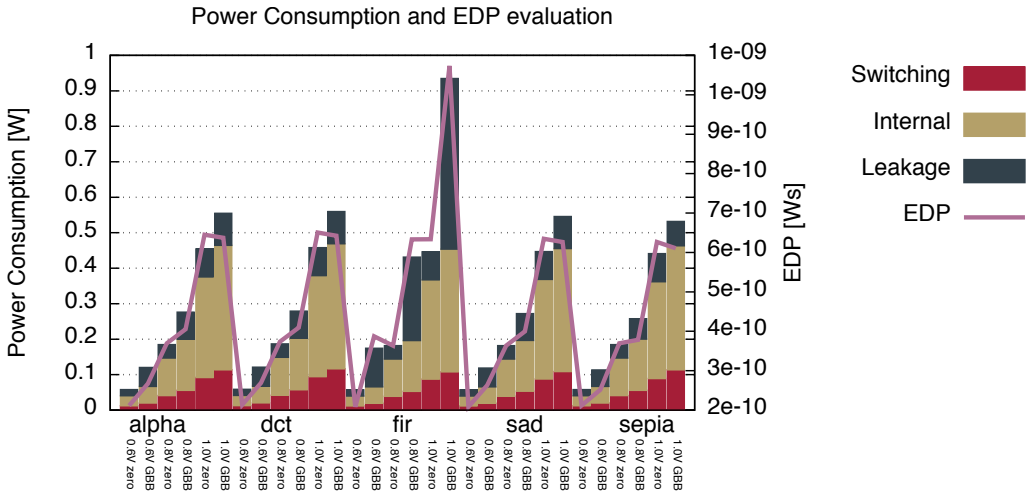


Figure 8.11: Total power consumption and EDP evaluation for five different benchmarks for $V_{DD} = \{0.6V, 0.8V, 1.0V\}$, without (zero) and with core-, i.e., global grained body bias (GBB) set based on application dependent slack at their respective maximum clock frequencies

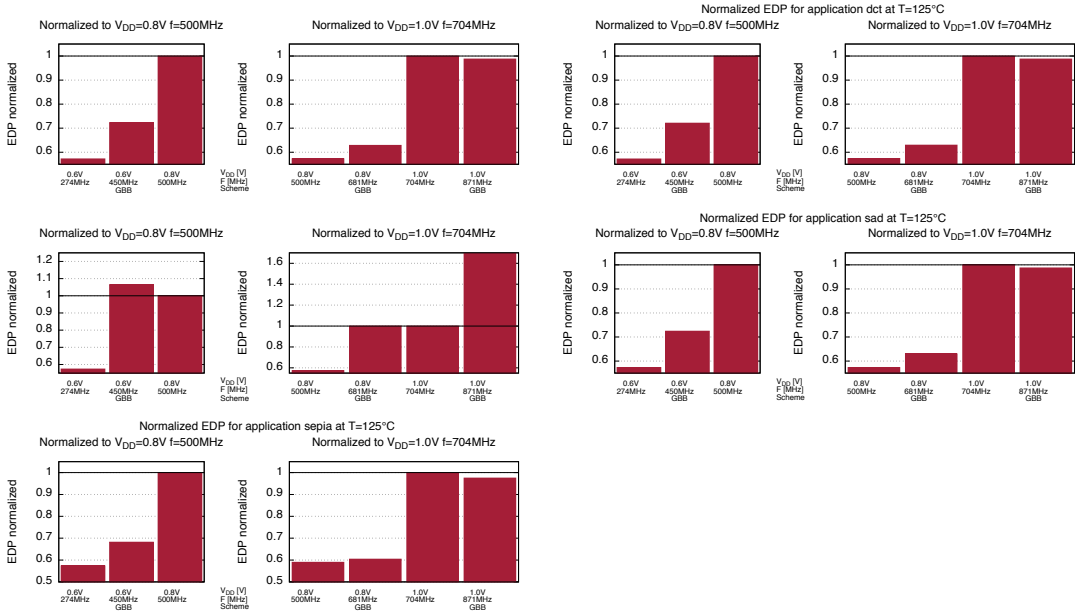


Figure 8.12: Core-grained/global body bias EDP evaluation of MuCCRA4 in the LVT flavor at $V_{DD} = \{0.6V, 0.8V, 1.0V\}$ in a clock frequency scenario (HP corner) where all results are normalized to the EDP of the next higher supply voltage scaled mode

leaves a lot of room for improvement. Consider e.g. $V_{DD} = 0.6V$. Throughout all applications, while the body bias scaled variant is still $50MHz$ slower than the next supply voltage scaled variant, it has better energy efficiency in all applications but the `fir` filter, constituting the worst-case application. The same trend also applies to greater supply voltages $0.8V$ and $1.0V$, but the decrease in energy efficiency continues.

Consider the EDP evaluations for the five benchmark applications in Fig. 8.12. As displayed in Fig. 8.11, the total power consumption of the forward body bias scaled variants for all applications were less than those of the supply voltage scaled variants. However, as the maximum clock frequency of the supply voltage scaled variants is greater, this also has an impact on the resulting EDPs. In fact, for all but the `fir` filter, EDPs are significantly improved when scaling using forward body biasing. However, as soon as one PE needs to use its critical path, which is generally assumed that it does, energy efficiency is worse than supply voltage scaled equivalents.

These results highlight several points. Even though in the case of RVT, the effect of reverse body biasing on timing with increasing V_{DD} diminishes, leakage increases stronger with V_{DD} than the levels of reverse body biasing can be increased without violating timing, assuming maximum clock frequency is scaled along. Furthermore, both leakage and energy efficiency wise, a chip should be operated at the lowest V_{DD} possible. In modes where low clock frequencies are tolerable, reverse body biasing helps tremendously in realizing

ultra-low-power modes. For such applications, core-grained body biasing is appealing, as at low clock frequencies, all components can tolerate strong reverse body biasing since slack is available in abundance. Forward body biasing especially helps with reaching considerably high clock frequencies even at low supply voltages. At the evaluated V_{DD} of $0.6V$, it increases the maximum feasible clock frequency by $1.64\times$. However, as leakage rises exponentially, core-grained application is not focused enough. With a single PE utilizing the critical paths, all other PEs and components are set on full forward body bias as well and thus add to the leakage penalty. In this case, energy efficiency drops below its supply voltage scaling counterpart. Thus, body bias granularity is a determining factor in leveraging body biasing.

8.1.2 Coarse-Grained Body Biasing

Coarse-grained body biasing is one step further towards locally focused body biasing. As shown in the previous section on core-grained body biasing, the leakage penalty incurred by applying forward body biasing on an entire DRP often outweighs the dynamic savings realized by enabling greater clock frequencies at lower supply voltages. Furthermore, even though core-grained body biasing is suitable for low clock frequency ultra-low-power modes, the capability to restrict leakage to a minimum is desirable in any mode. However, unless body biasing can be applied more locally, this can only be achieved by reducing clock frequency.

Coarse-grained body biasing allows to determine the leakage-speed trade-off per PE and thus is far more locally focused than core-grained body bias. In the following coarse-grained body bias simulation studies, a 4×4 PE array has been evaluated. As a first step, body bias assignments per PE are computed based on the application, each PE's application mapping, and the intended strategy. In case of the RVT flavor, the LP corner has been used, i.e., staying at F_{max} the maximal clock frequency and applying reverse body bias per PE as application slack allows. The maximum reverse body bias corner does not need any further evaluation, as it is identical to the figure given in 8.1.1. Applying maximum reverse body bias to all components is exactly the definition of the core-grained maximum reverse body bias. LVT is similarly evaluated, although, for the HP corner, the clock frequency is scaled F_{max} is scaled with maximum forward body bias to $F_{max,BB}$ and applying forward body bias as needed to meet timing based on application slack. In case of coarse-grained body biasing, e.g., an unused PE or a PE merely used for routing may be set to reverse body bias through the ability to focus body bias on a PE granularity. Again, like with RVT, maximum reverse body bias or maximum forward body bias needs no further consideration as this has been covered in section 8.1.1 as well.

As a first step, the body bias per application and PE is determined using the algorithms presented in section 5. Then for the design in the RVT flavor, the LP corner is applied using the unchanged maximum clock frequencies as denoted in Tab. 8.2. This results in the per application and per PE body bias assignment as visualized in Fig. 8.13.
Examining Fig. 8.13, it visualizes that reverse body bias can be applied on each PE which

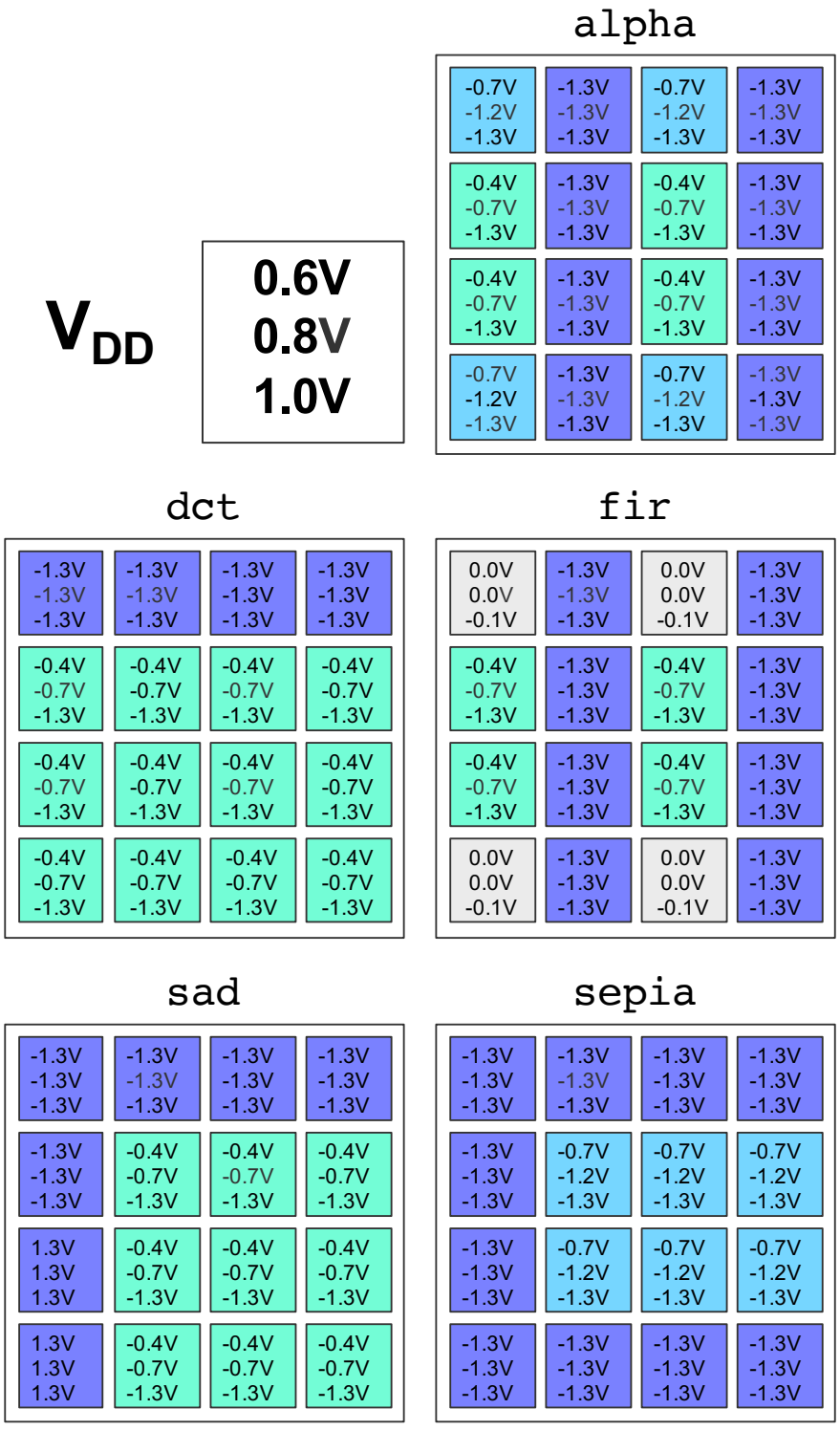


Figure 8.13: Body bias assignments for the RVT flavor using the LP corner for $V_{DD} = \{0.6V, 0.8V, 1.0V\}$

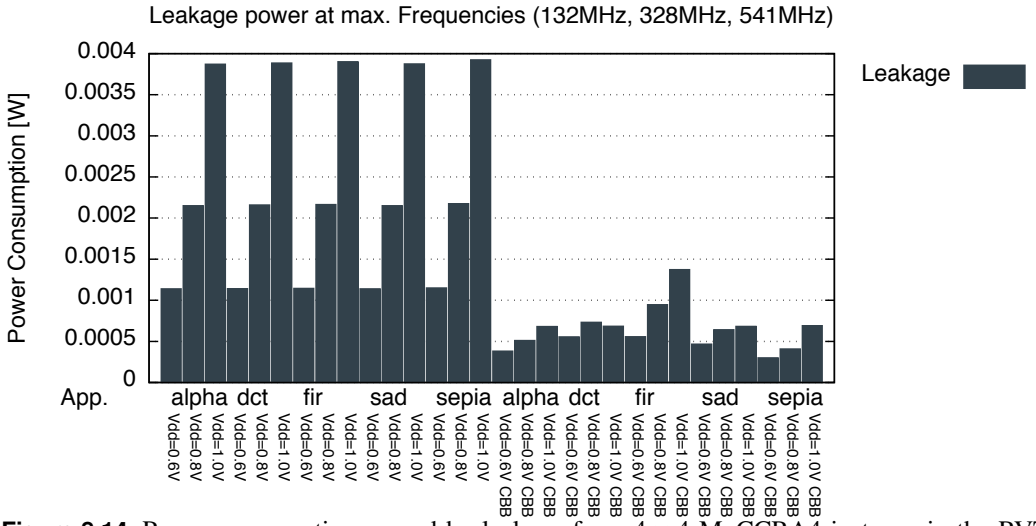


Figure 8.14: Power consumption caused by leakage for a 4×4 MuCCRA4 instance in the RVT flavor without (left) and with body biasing assignments of Fig. applied (right)

is not using the most timing critical full-width multiplication operation. Depending on timing criticality, increasingly greater levels of reverse body biasing can be applied. Most arithmetic functions like addition or half-width multiplication allow for considerable bias, whereas PEs using logical functions, PEs that are used for routing purposes only and of course inactive PEs' leakage can be cut using maximum reverse body bias. This results in considerably reduced leakage despite staying at the maximum clock frequency. Consider Fig. 8.14. It depicts the power consumption incurred through leakage without reverse body biasing (Fig. 8.14 left) and with reverse body biasing applied according to the assignments specified in Fig. 8.13 (Fig. 8.14 right). This clearly shows a strong cut in leakage power.
However, Fig. 8.15 demonstrates that MuCCRA4 implemented in the RVT flavor at maximum clock frequency mainly consumes power through dynamic power dissipation, while only a little portion is accounting for leakage. This little portion is also further contained using reverse body biasing, but may not justify the efforts of coarse-grained body biasing.

The usage of strong forward body bias in the LVT flavor, however, stands in stark contrast to these findings as the following will elucidate. For the LVT flavor, the HP corner is used again, i.e., scaling clock frequency using forward body bias. Again, body bias assignments are determined using the algorithms presented in chapter 5. The result is visualized in Fig. 8.16.
Inverse to the RVT assignments where reverse body bias is applied wherever application slack allows doing so, the algorithms now check how much forward body bias is necessary to be applied in order to reach the timing goals. Again, this is done per supply voltage V_{DD}. At first glance, Fig. 8.16 shows two major differences: both reverse, as well as forward body bias, is applied and inverse body biases are applied to each PE. For example, what previously couldn't take any reverse body bias for timing reasons is now applied maximum forward bias as $F_{max,BB}$, the scaled maximal clock frequency, is determined

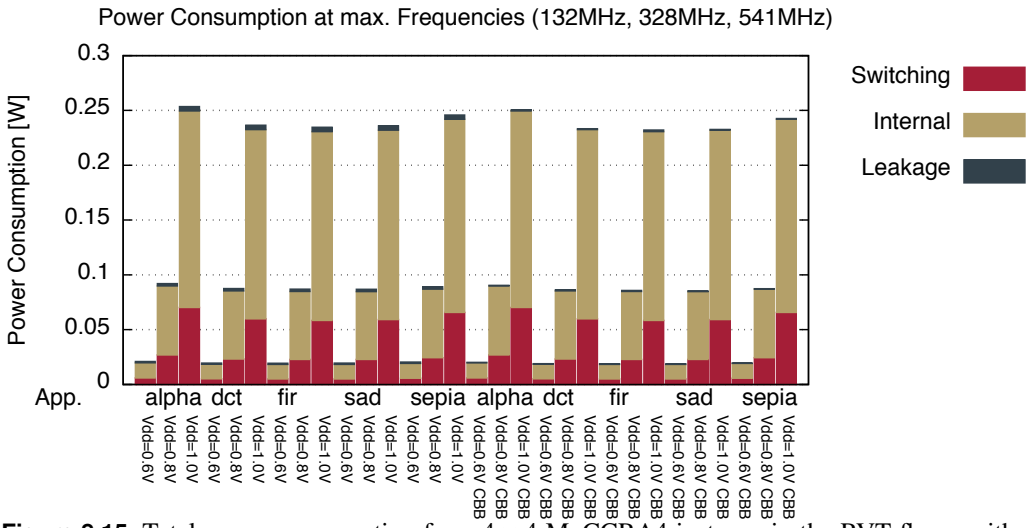


Figure 8.15: Total power consumption for a 4×4 MuCCRA4 instance in the RVT flavor without (left) and with body biasing assignments of Fig. applied (right)

by the critical path timing under maximum forward body bias (Fig. 8.16 `FIR`). Aside of the worst-case application `FIR`, moderate levels of forward body bias are applied. Again, noteworthy are the PEs that only implement routing, timing uncritical logic operations or those which are unused. These PEs are assigned the maximum reverse body bias the LVT flavor can provide and thus significantly add to the overall leakage reduction achieved by coarse-grained body biasing. This, of course, is also reflected in the power figures given in Figs. 8.17 and 8.18.

Consider Fig. 8.17 depicting the incurred leakage per application and body biasing scheme. The results are first ordered by application and supply voltage. Thus, first for all applications and a given body biasing scheme, e.g., zero meaning no body bias, incurred leakage is depicted per application ordered by supply voltage. Once for all applications and one scheme leakage results for all supply voltages are plotted, the next application scheme, in this case, coarse-grained body biasing, is plotted. Finally, for comparison, the leakage results of core-grained body biasing are added. Not only does the figure show that despite considerably increasing clock frequency, the incurred leakage is at about the level of the zero-bias result with one step increased supply voltage V_{DD}. Furthermore, it clearly shows that coarse-grained body biasing is capable of strongly mitigating the leakage penalty of core-grained body biasing in the `FIR` filter.

To properly assess the total power consumption results depicted in Fig. 8.18, they are now sorted in a different fashion: Now all results are grouped solely by application and within such application group, results are divided into sub-groups of supply voltage. Within these sub-groups, results are ordered by body bias application scheme, i.e., first zero bias, coarse-grained body bias (CBB) and lastly core-grained, global body bias (GBB). While facilitating a direct comparison of all schemes, this chart also allows comparing

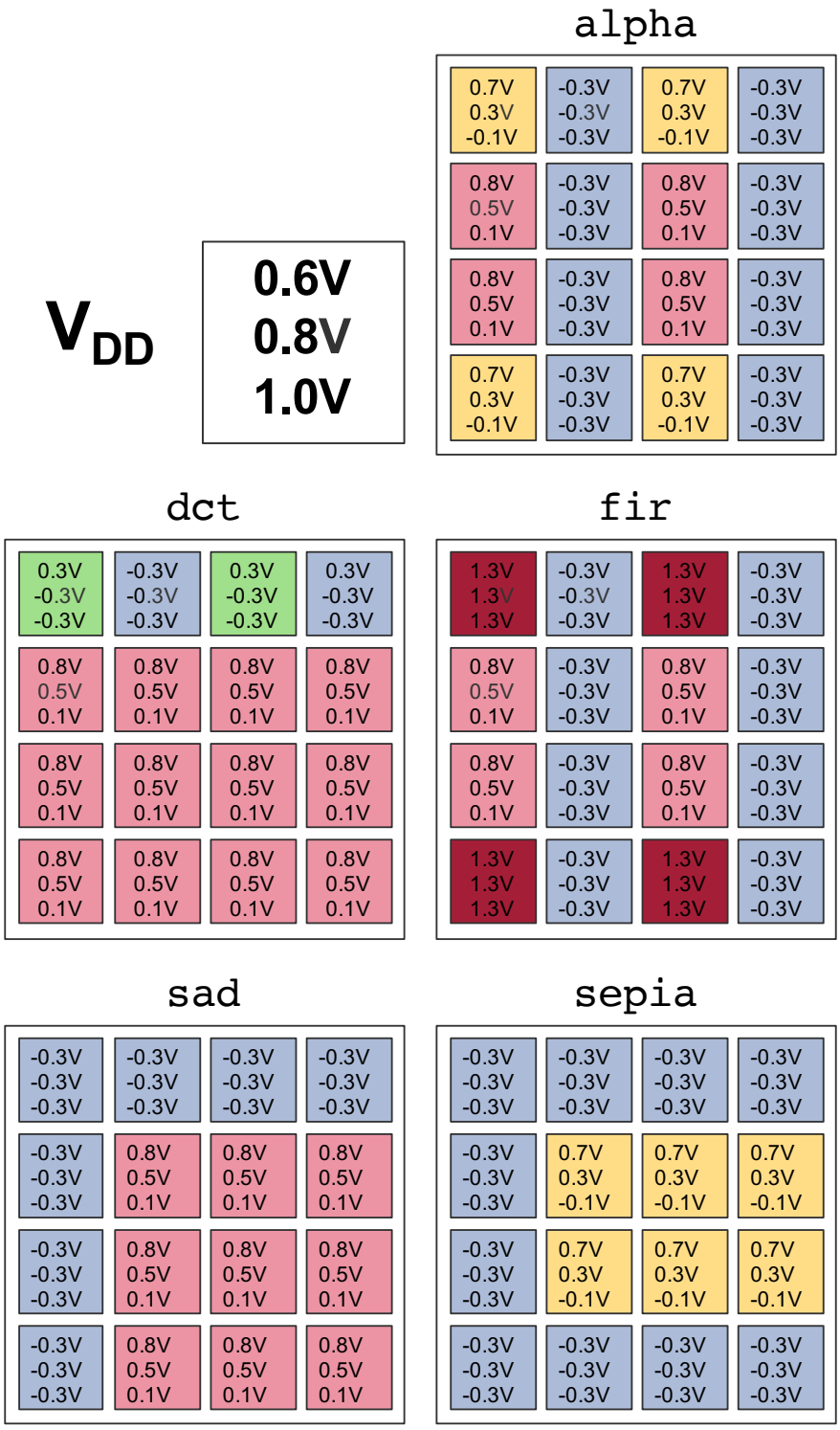


Figure 8.16: Body bias assignments for the LVT flavor using the HP corner for $V_{DD} = \{0.6V, 0.8V, 1.0V\}$

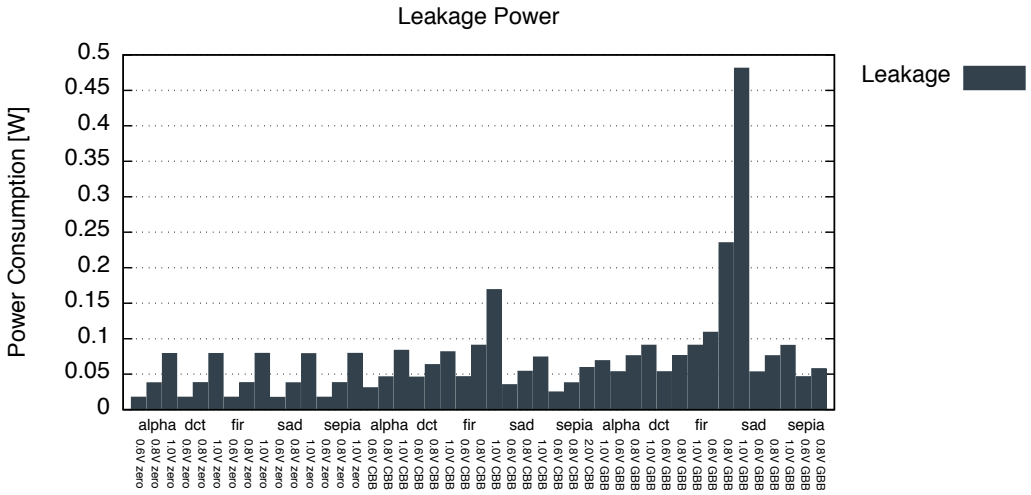


Figure 8.17: Power consumption caused by leakage for a 4×4 MuCCRA4 instance in the LVT flavor without (left, zero) and with coarse-grained body biasing assignments of Fig. 8.16 applied (middle, CBB) and core-grained global body biasing (right, GBB)

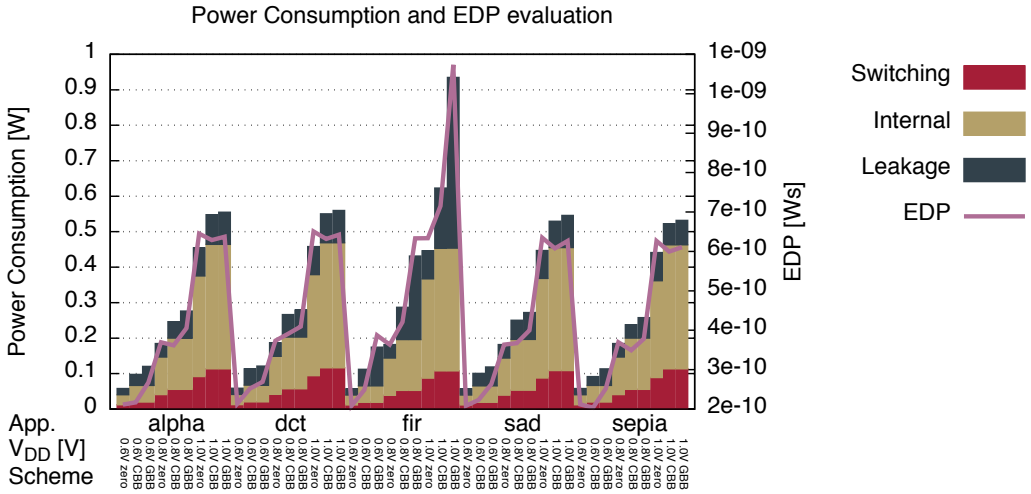


Figure 8.18: Total power consumption for a 4×4 MuCCRA4 instance in the LVT flavor grouping by application, within each application group by V_{DD} and body bias application scheme (zero, coarse-grained CBB, core-grained global body bias GBB), with coarse-grained body bias applying the body biasing assignments of Fig. 8.16

voltage scaling to body bias based frequency scaling in a more comprehensible manner. As overlay the EDP of all results is plotted on top of the bars. It is observable in Fig. 8.18 that leakage makes up for a significant part of total power consumption. Furthermore, Fig. 8.18 also indicates that coarse-grained application of coarse-grained body biasing yields better energy efficiency than scaling to the next higher supply voltage level for all applications, including the worst case application `FIR`. In case of the `sepia` application, the EDP of the coarse-grained body bias frequency scaled variant at $V_{DD} = 0.6V$ yields a better EDP than the original unscaled result. This is because the local assignment of reverse body bias can even result in a leakage reduction compared to the original zero-bias variant at the same V_{DD}.

Thus, these results highlight four points:

- Reverse body biasing focused strategies may consider core-grained body biasing
- Forward body biasing greatly benefits from coarse-grained body biasing
- Coarse-grained body biasing exhibits better energy efficiency in all cases compared to voltage scaling
- There is a lot of unused potentials to be leveraged by better partitioning methods and algorithms

Results on coarse-grained body biasing in processes and flavors with mostly reverse body bias capability may not benefit as much by focussing the bias more locally. This, however, depends strongly on the intended purpose and the process used for the implementation. As shall later be shown in chip evaluation results, processes that have both strong forward and strong reverse body biasing capabilities do benefit from more fine-grained approaches. For the sole purpose of realizing ultra-low-power modes, the overheads may not be justified. For forward body bias oriented flavors, however, the results clearly showed the superiority of coarse-grained body biasing over the core-grained method. Throughout all evaluated applications, it resulted in better energy efficiency, while also hinting the unused potential that can be leveraged with better, more fine-grained body biasing.

8.1.3 Fine-Grained Body Biasing

As the previous sections indicated, coarse-grained body biasing still leaves room for improvement. For this reason, the fine-grained body biasing approach proposed in section 5.4 were applied to the MuCCRA4 design. It has then been pre-partitioned on PE level into individual domain candidates. e.g., the register file, context memory, sub-components of the ALU (adder, multiplier, multiplexer etc.), etc. each constitute a domain candidate. Memories have not further been divided for this evaluation as this requires assumptions on memory usage to be made, which for this simulative study has been omitted.

The aforementioned hand defined domain candidates were then merged using the merging algorithms specified in sections 5.4.1 for a target number of body bias domains per PE

k ranging from 1 to 4. Obviously, $k = 1$ just means coarse-grained body biasing, as all (sub-)components of a PE are put into one body bias domain. Using the thereby gathered body bias domain partitionings, different temporal body biasing schemes, as specified in section 4.2, are evaluated. These schemes are static, i.e., hard-wired body bias assignment, programmable, meaning body biases may be changed per application and dynamic, i.e., switched at any given point in time. If increasing the number of target body bias domains didn't yield any leakage improvement during merging, this number of target body bias domains and above will be ignored. e.g., if the figures display a k ranging from 1 to 3, this means 4 body bias domains per PE didn't yield any improvement for the given design and temporal scheme.

The body bias domain partitioning for a target of $k = 3$ body bias domains generated in this step and used for most evaluations in this subsection is displayed in Fig. 8.19. Using this partitioning, the evaluation of power consumption for programmable body biasing is conducted, resulting in Fig. 8.20.

As this graph shows, there are no data points for $k = 4$, which means that for programmable body biasing, the partitioning visualized in Fig. 8.20 yields maximum leakage reduction. This, of course, is also due to the limited resolution of the domain candidates as they have to be pre-partitioned by hand. However, for no body bias indicated by the letter N and k ranging from 1 to 3, programmable body biasing shows a steady reduction in the leakage penalty incurred by the forward body bias used for frequency scaling. By splitting up PEs into several body bias domain, the components requiring strong forward body bias can be separated from those that do not need such body bias. On the contrary, some might even tolerate reverse body bias. Thus, increasing numbers of body bias domains allow for more separation of such parts. Furthermore, for all applications, EDP is now lower than unscaled variants on the same supply voltage level when k is at least $k = 2$. This means that with fine-grained body biasing, it is in the evaluated architecture always more energy efficient to use forward body bias to scale clock frequency than increasing the supply voltage by a discrete step as done in the evaluation.

Putting the leakage reduction into concrete numbers, Figs. 8.21 and 8.22 visualize the achieved leakage reduction of programmable body biasing per number of body bias domains per PE. Obviously, these figures are application dependent and thus, the best-case application incurs less reduction than the worst-case application. However, the best-case application, of course, starts out with less leakage to begin with. For application `sepia`, a solid 20-50% leakage reduction is achieved, while for `fir`, leakage is cut by more than half over coarse-grained body biasing ($k = 1$).
The evaluation for dynamic body biasing is more of a theoretical nature as switching body biases on a small time-scale is a multifactorial challenge. It is mainly dependent on the charge required to change the body bias from a given potential to the next, as well as the transistor bodies' *RC* characteristics. For this evaluation, a fictive substrate charge equivalent to charging transistor terminals is assumed. Both in terms of resistance R and capacitance C, this is highly underestimated. This being said, it ultimately only changes the

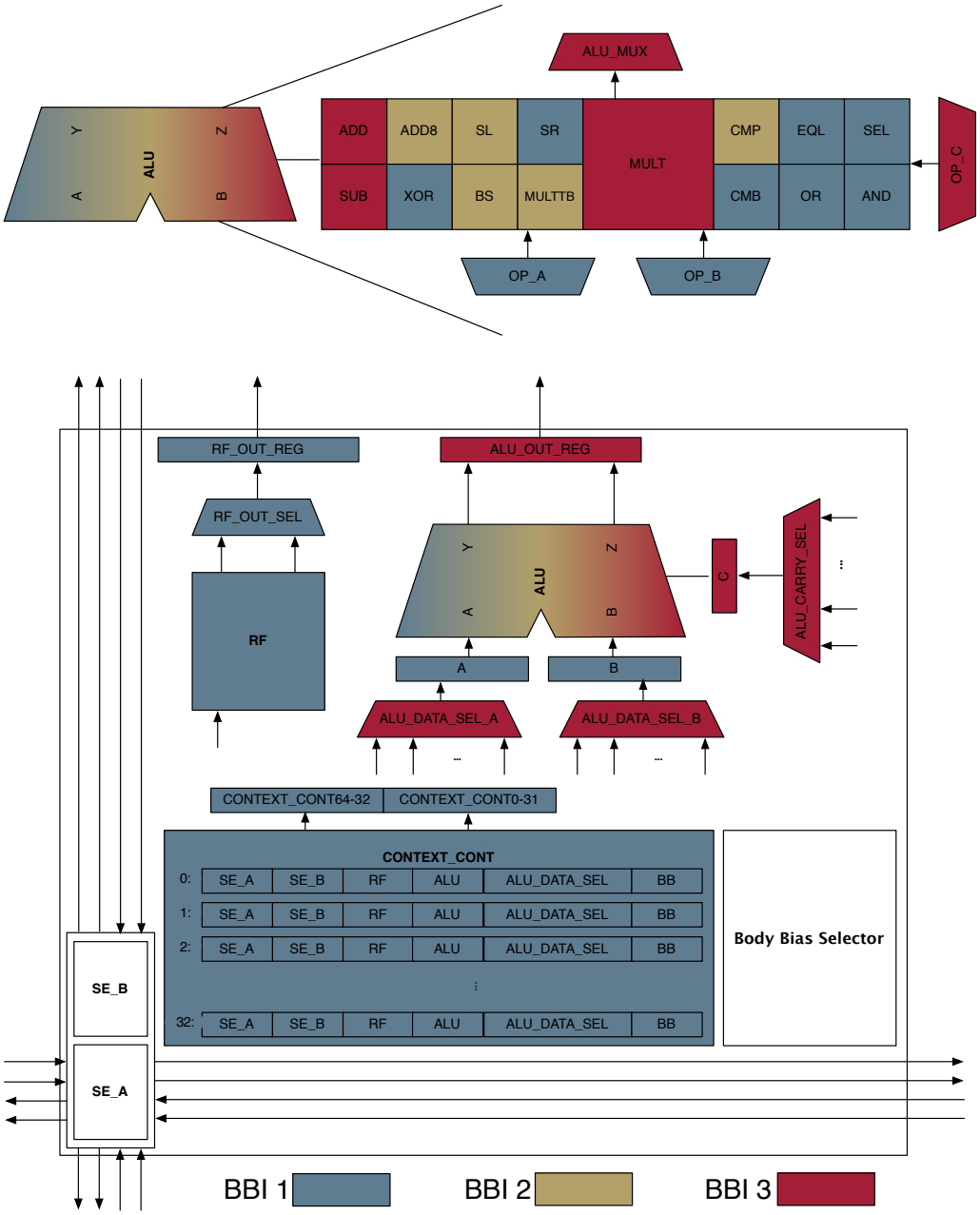


Figure 8.19: Partitioning into body bias domains generated through pre-partitioning assisted fine-grained body bias flow with combinatoric *k*-subset merging.

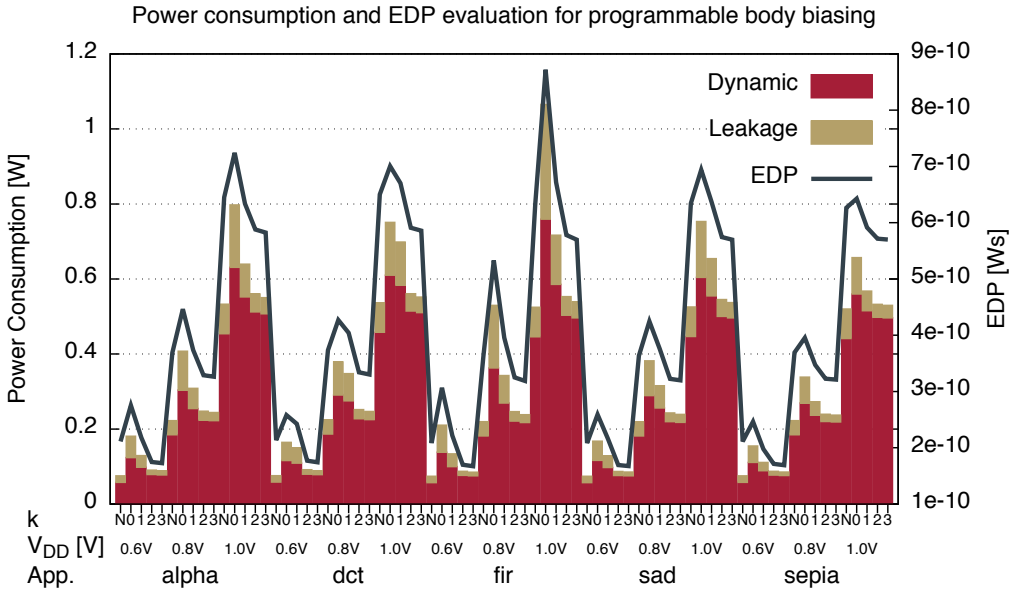


Figure 8.20: Total power consumption for a 4×4 PEs MuCCRA4 instance implemented in STMicroelectronic's 28nm UTBB-FDSOI LVT flavor in the body bias frequency scaling scenario (corner HP), split by dynamic and static power (leakage) with the respective EDP per bar as overlay as a measure of energy efficiency

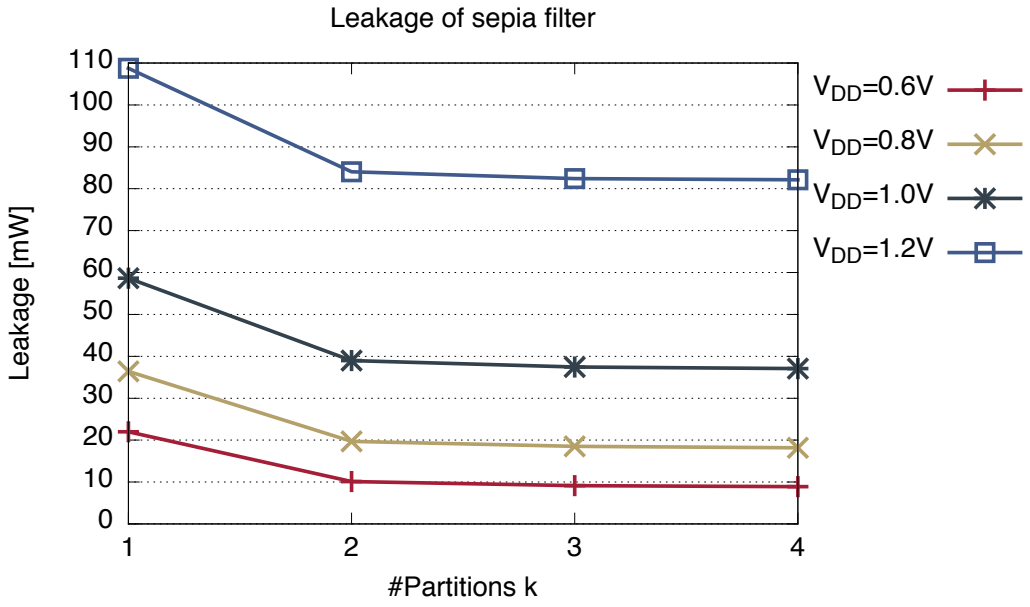


Figure 8.21: Leakage body biasing at $V_{DD} = \{0.6V, 0.8V, 1.0V, 1.2V\}$ for programmable body biasing in best-case application `sepia` and increasing numbers of body bias domains per PE

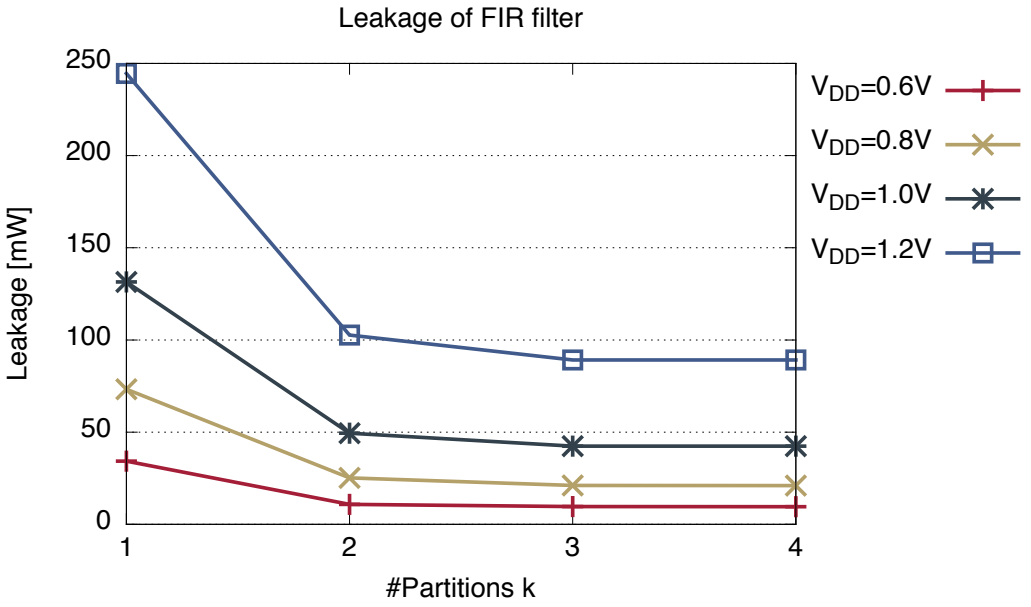


Figure 8.22: Leakage reduction over core-grained body biasing at $V_{DD} = \{0.6V, 0.8V, 1.0V, 1.2V\}$ for programmable body biasing in worst-case application `sepia` and increasing numbers of body bias domains per PE

period required to stay at a given body bias level to amortize the switching costs. Therefore, based on the operations executed in the applications, the time spent executing the respective operation, time-dependent body bias assignments were computed using the approaches presented in sections 5.4.1 and 6.6.

This results in body bias levels depicted in Fig. 8.23. The body bias is only changed, when the leakage reduction or the mitigated leakage penalty that is saved by switching body bias outweighs the switching cost. This computation, however, neither includes static power consumption of the realizing regulators, nor does it include the efficiency loss of the mentioned regulators. Thus, it is not expected for dynamic body biasing to yield a benefit at this time scale. This hypothesis is confirmed by the overall leakage reduction of the presented body biasing schemes. Consider Fig. 8.24.

Fig. 8.24 depicts the leakage reduction of programmable and dynamic body biasing with static, i.e., hard-wired coarse-grained body biasing as baseline. Here again, the worst-case application `fir` benefits much more from body biasing application schemes. In this case, fine-grained leakage reduction amounts to almost 90% while for `sepia`, the best-case, gets barely close to a 50% reduction in leakage. This being said, applications are similar in regard to the effect of the first additional body bias domain. Depending on the design and the different need of its components for different levels of body bias, this effect varies. For the relatively small design of one PE, there are two major categories: 1. arithmetic functions and 2. logical functions and inferred memories. While the first category can generate paths of considerable length and timing criticality, the second category is

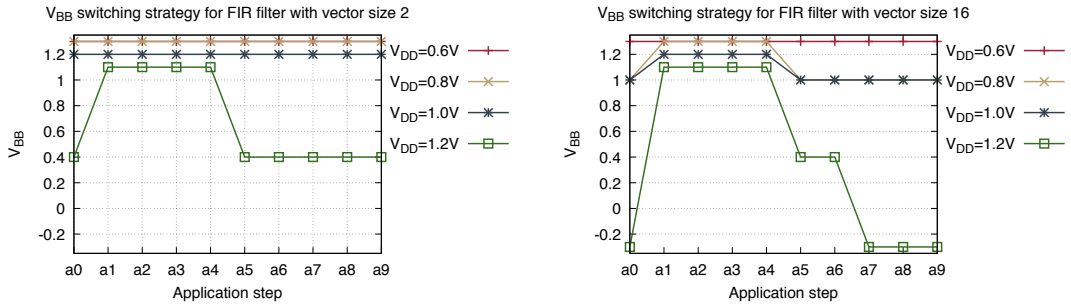


Figure 8.23: Dynamic switching strategy for application `fir` depending on vector instruction size, 2 (top) and 16 (bottom)

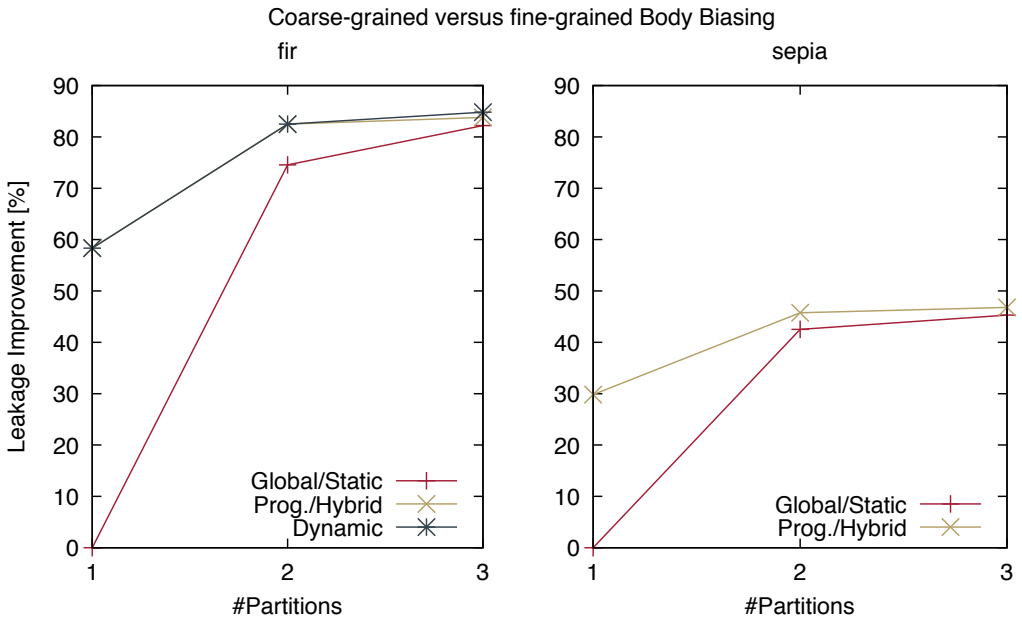


Figure 8.24: Overall leakage reduction of static coarse-grained body biasing ($k = 1$) over programmable and dynamic body biasing

uncritical in comparison.

The fine-grained results hint towards the limitations of body biasing. Increasingly fine-grained body biasing yields benefits only up to a certain point, which is defined by the number of components with highly different timing criticality. This, again, points towards the last remaining optimization potential: standard-cell grained body bias domain partitioning. In this pre-partitioning fine-grained body biasing approach, domain candidates are defined on HDL module level. In reality, ideal domain candidates are most likely not defined along the lines of component borders. Especially if greater levels of synthesis optimization are desired, effects such as resource sharing need to be accounted as well. With the pre-partitioning based domain candidate definition, not only is this clearly not feasible but also hinders synthesis optimization. Another limitation concerning dynamic body biasing is the temporal granularity. Dynamic body biasing is only beneficial beyond a threshold period for which the body bias remains unchanged. While there are, of course, very potent regulators e.g., for DVFS, one of the intriguing aspects of body biasing is the extremely low demands on drive current. Furthermore, the resistance characteristics of the transistor wells constitute another research challenge, ultimately giving a physical lower bound.

8.1.4 Standard-Cell Grained Body Biasing

To alleviate the shortcomings of all previous approaches, the DCE standard-cell grained body biasing approach has been proposed in chapter 6. Analyzing timing and activation in highly optimized standard-cell netlists does not only eliminate the issue of limited synthesis optimization but also removes the need for pre-partitioning into domain candidates by hand. This, however, also requires complex computations like timing and leakage analysis usually conducted by sophisticated EDA tools. In this thesis' case, rudimentary algorithms implementing such analyses have been implemented. However, although they are expected to be consistent for the comparison, they do not need to be exact at this point. For this reason, all results in this section are given relative to the zero bias (No BB) baseline.

To generate the following results, the target components, an ALU and a register-file, were synthesized using process specific libraries and Synopsys Design Compiler [107] with ultra optimizations and ungrouping enabled. The ultra feature is the synthesis feature with the highest level of optimization, thus leaving one single highly optimized netlist per target design. This netlist is then used in DCE together with process specific libraries and SPICE augmented body biasing data to identify possible domain candidates. Once resource sharing effects have been taken care of, these domain candidates can be merged using the optimized combinatoric k-subset merging algorithm described in section 6.5. The actual evaluation is then also conducted by DCE using its leakage and timing computation functionality. For evaluation, two modes are considered: static body biasing and dynamic body biasing. These two modes constitute two extremes: worst-case and best-case application. With static body biasing, it is assumed that the body bias is hard-wired and

thus cannot be changed while dynamic, in this case, assumes that body bias could be switched at any point in time. The latter mode's results are thus generated by averaging the leakage incurred when activating each operation group. An operation group denotes a set of operations which result in the same body bias assignment to body bias domains when activated. Thus, evaluation only needs to be conducted for each of such groups.

For a better distinction between results, the following nomenclature is used:

1. No BB - Zero bias
2. Max. RBB, maximum reverse body bias with reduced maximum clock frequency
3. *N kXM* Normal corner *N*, i.e., unchanged maximum clock frequency
4. *F kXM* Forward body biasing scaling corner *F*, with increased maximum clock frequency

where *kX* means the number of body bias domains *k* is $k = X$ and *M* is either *s* static body bias assignment, or, *d* dynamic assignment. For example, $Fk3d$ describes a result for the forward body biasing scaling corner with 3 body bias domains per evaluated component, with the evaluation conducted for dynamic body bias assignment.

For STMicroelectronic's 28nm UTBB-FDSOI, the target designs were evaluated at both supply voltages $V_{DD} = 0.6V$ and $V_{DD} = 1.0V$. Consider the results for $V_{DD} = 0.6V$ first in Fig. 8.25. As first observation, the flavors have opposite body bias capabilities. While RVT has an extended reverse body bias range and only little forward body bias, LVT has an extended forward body bias range with limited reverse body bias capabilities. Thus, the differences between LVT and RVT flavors in the Max. RBB corner, as well as all other results that benefit from one direction, can be accounted to the flavor difference.
An interesting phenomenon becomes visible when considering the static results. Independent of the corner, both *N* and *F* now show reactions to increased numbers of body bias domains *k*. This is due to cross-domain resources sharing effects. As with static body biasing, the body bias cannot be changed temporally, it is a fixed assignment that ensures that all components are always functional. As the critical paths cross all body bias domains, none of the assigned body biases could be decreased without violating timing. While resource sharing is, in principle, a good optimization, it can hinder proper utilization with body biasing. In the dynamic case, however, significant leakage reductions can still be achieved by adjusting the body bias depending on the operation groups executed during certain periods. In the worst case, dynamic body biasing is equal to static body biasing.
For a supply voltage of $V_{DD} = 1.0V$, results are fairly similar as depicted in Fig. 8.26. It corresponds to the reduced impact of body bias at higher supply voltages as discussed in previous chapters. As the components are not synthesized anew for greater supply voltages, the graphs exhibit the same effects as in Fig. 8.25. Static body biasing is still hindered by cross-domain resource sharing, while dynamic body biasing leverages the different body bias requirements of different operation groups.

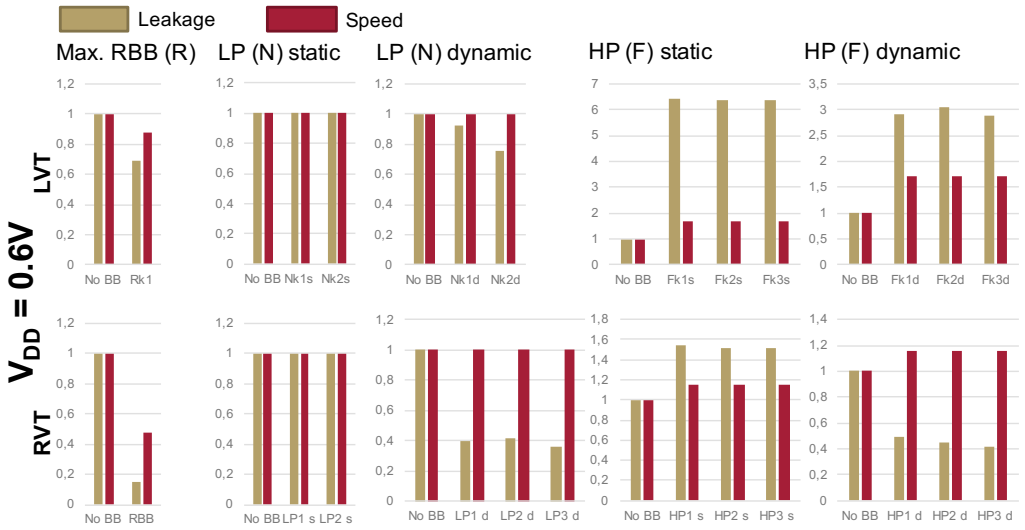


Figure 8.25: Normalized leakage and speed evaluation of the DCE based body bias domain partitionings for STMicroelectronic's 28nm UTBB-FDSOI flavors LVT (top) and RVT (bottom) for $V_{DD} = 0.6V$

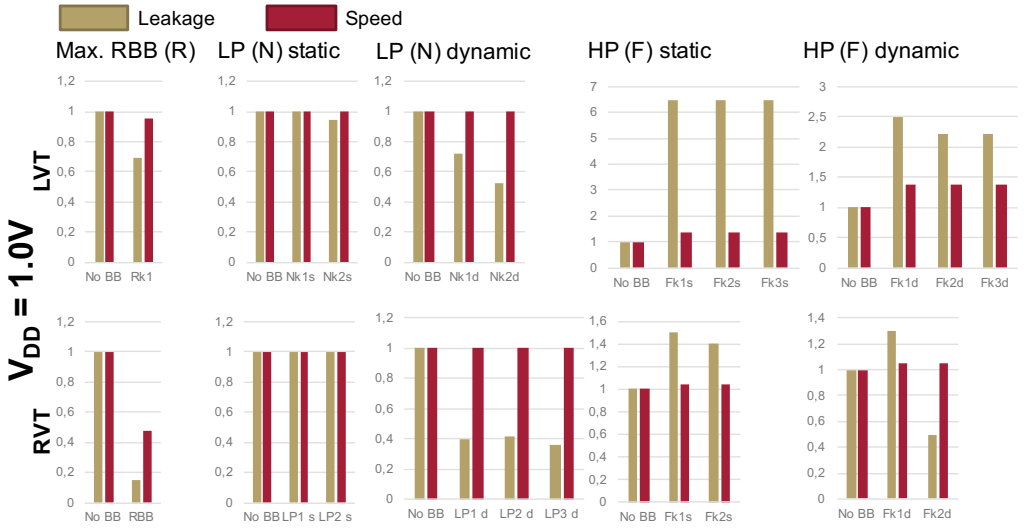


Figure 8.26: Normalized leakage and speed evaluation of the DCE based body bias domain partitionings for STMicroelectronic's 28nm UTBB-FDSOI flavors LVT (top) and RVT (bottom) for $V_{DD} = 1.0V$

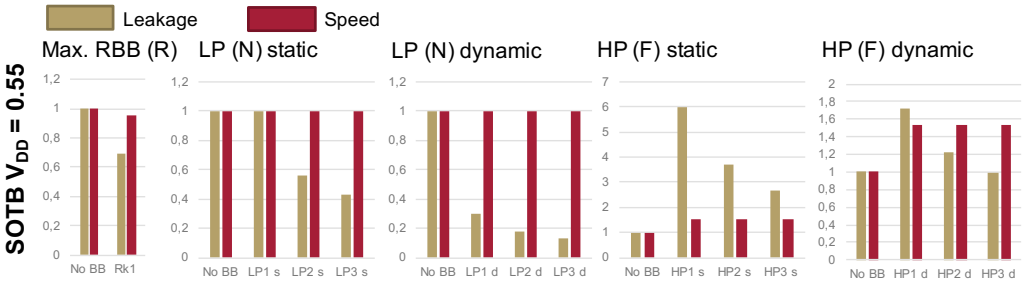


Figure 8.27: Normalized leakage and speed evaluation of the DCE based body bias domain partitionings for Renesas' 65nm SOTB FDSOI for $V_{DD} = 0.55V$

Renesas' SOTB, on the other hand, does not use as complex library characterizations than the processes above. This, as well as other reasons, lead to less resource sharing and thus also allows to leverage body bias domain partitioning to a far greater degree. Consider the results visualized in Fig. 8.27. First of all, SOTB and its very thin buried oxide enable the back gate to exercise greater control on the transistor channel, thus leading to strong Max. RBB results. This also drags down the speed figure more than e.g. with STMicroelectronic's process at similar biases. With less resource sharing, it can be observed that increasing numbers of body bias domains k reduce the incurred leakage considerably.

With domain candidates with different body bias requirements no longer being dependent on each other, more suitable body biases can be statically assigned to the individual domains. This results in leakage reductions, which are then furthered using dynamic body biasing. With the forward body biasing scaling corner F, the results are even more drastic. Even though the clock frequency is scaled using forward body bias, leakage is already below the original level when allowing more than one body bias domain. If dynamic body biasing is employed as well, the leakage reduction almost reaches the levels of the unscaled variants. This result, however, is to be taken with a word of caution, that dynamic body biasing is an unanswered research challenge of its own, and thus, while theoretically possible, this thesis does not claim this to be feasible with the proposed methods.

Another worthwhile endeavor is also to partition components which haven't been considered before, such as memories. In this thesis, only inferred memories like flip-flop or latch standard cells are considered. In many designs, such components make up for a large portion of chip area. Thus, in case the memories are unused, they could just be deactivated to reduce unnecessary static power consumption. DCE handles such memories similar to the ALU designs above, by treating the memory's address like an opcode. In this case, the highest address bit is specified as opcode and thereby partitions the memory into a lower half and an upper half. As this partitioning is application dependent, the designer has to specify such attributes by hand.

Consider Fig. 8.28. It clearly shows the expected response when half or all of the memory

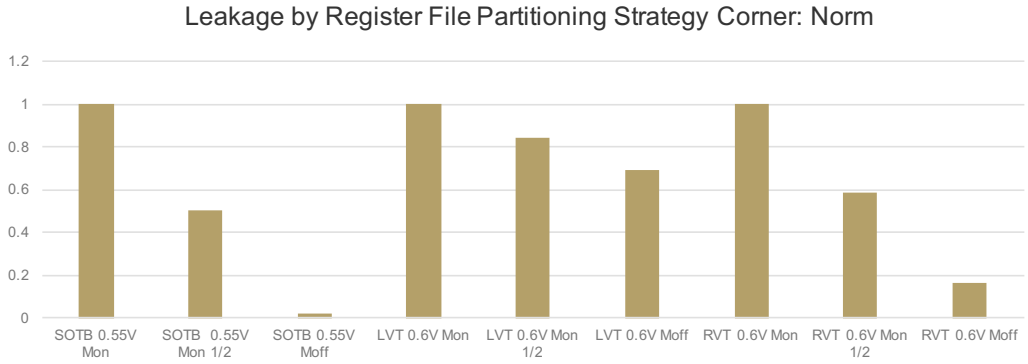


Figure 8.28: Normalized leakage for memory partitioning at LP corner using STMicroelectronic's 28nm UTBB-FDSOI and Renesas' 65nm SOTB at $V_{DD} = 0.6V$ and $V_{DD} = 0.55V$ respectively

is deactivated using maximum available reverse body biasing. The responses are thus mainly dependent on the processes' capabilities to exercise reverse body biases. While SOTB with its strong body effect exhibits the greatest reductions, it is closely followed by the RVT flavor of STMicroelectronics. Lastly, the LVT flavor, not having strong reverse body biasing capabilities still exhibits considerable reductions of about 25% when switched off completely.

In summary, standard cell-grained body biasing is a two-fold refinement on fine-grained body biasing. Not only is partitioning granularity no longer limited by component boundaries, but also eliminates the need for pre-partitioning by hand. By first identifying domain candidates on standard cell basis, it allows standard cell granularity with often even less computational overhead during the merging phase, as many domain candidates can be collapsed into a few. However, there are also downsides which will have to be addressed if body bias domain partitioning should become a standard process when designing with FDSOI. The greatest hindrance as demonstrated in Figs. 8.25 and 8.26, is cross-domain resource sharing which can be resolved by e.g., replicating shared resources. Heavy resource sharing can ultimately reduce or eliminate the benefits of additional body bias domains.

8.2 Chip Measurements

In this section, the actual chip measurements will be presented. They were obtained using a power analyzer attached to the digital core V_{DD} and measuring the incurred static current while changing the body bias of respective domains using FPGA controlled voltage regulators. By shorting body bias domains of PEs or entire PEs altogether, different granularities can be evaluated. e.g., if fine-grained body biasing is evaluated, each domain is biased individually, while for coarse-grained body biasing all body bias domains of one PE are shorted together to form one body bias domain. The design, as well as the body bias domain partitioning, has been implemented as illustrated in chapter 7.

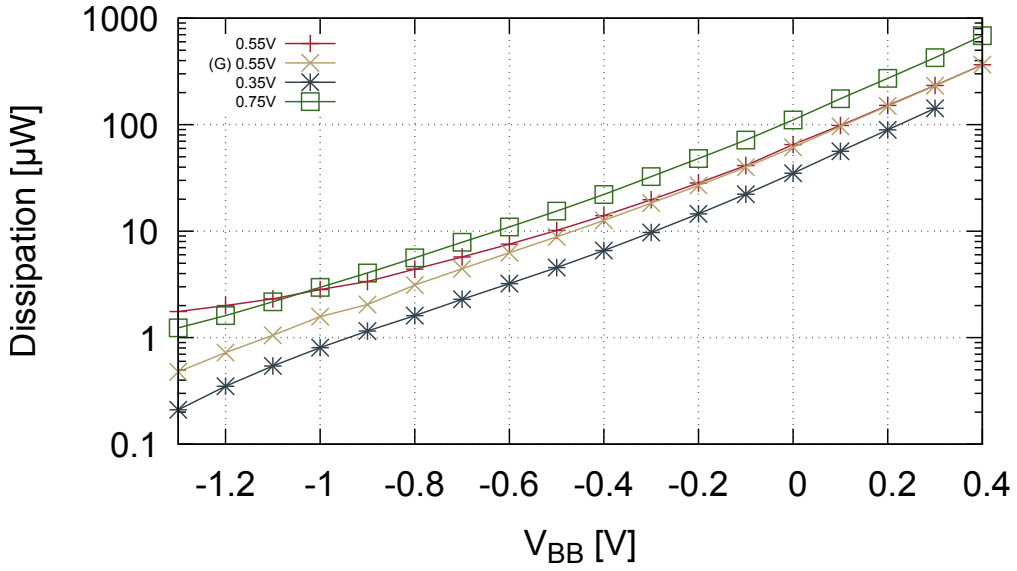


Figure 8.29: Static power consumption of the digital core at $V_{DD} = \{0.35V, 0.55V, 0.75\}$ where $V_{DD} = 0.55V$ is split into all evaluated chips and a good lot (G)

The chip leakage evaluation is conducted by first estimating the PE's timing at zero bias using the standard cell libraries supplied with the PDK. Using this timing estimate together with further SPICE simulations gives an accurate description of the effects of body biasing on the timing of each PE. Using these body bias augmented timing estimates, the body bias assignments are computed as described in chapter 5 and 6 for all covered granularities.

8.2.1 Core-Grained Body Biasing

Core-grained body biasing, or in this particular case, whole chip body biasing shortens all body bias domains to one big VBN and VBP net, thus acting like one single body bias domain. Fig. 8.29 explicitly depicts the incurred static power consumption instead of the leakage current to differentiate between the supply voltages V_{DD}. The y-scale is a logarithmic scale as body bias influences leakage and static power consumption in an exponential manner.

The results presented in fig. 8.29 were obtained using different numbers of samples, where $V_{DD} = 0.55V$ is the one with the most chips sampled. For this supply voltage, a total of 6 chips were evaluated in detail. From this group, a good lot in regard to leakage has been derived, comprising 4 chips. In the results, their average static power dissipation is visualized as (G) $V_{DD} = 0.55V$. From this good lot, one chip was taken to evaluate the design at two more supply voltages $V_{DD} = 0.35V$ and $V_{DD} = 0.75V$. For all evaluations, a mostly exponential behavior can be observed. At maximum reverse body bias, the $V_{DD} = 0.35V$ measurement reaches about $200nW$ while the good lot of

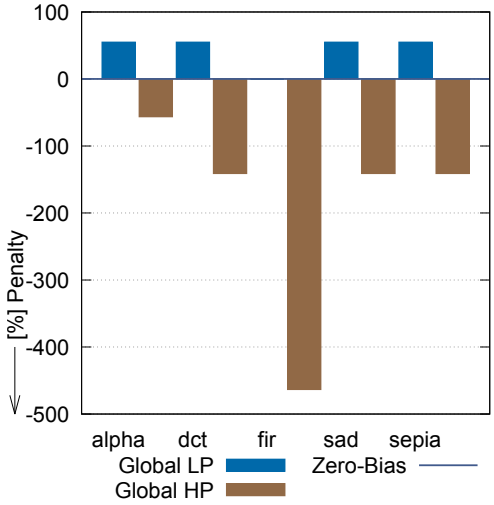


Figure 8.30: Leakage reduction or penalty respectively for global body biasing applied to five benchmark applications at $V_{DD} = 0.55V$

$V_{DD} = 0.55V$ reaches the respectable $500nW$ range. The regular $V_{DD} = 0.55V$ group with the bad chips, however, does not respond well to reverse body bias and even worsens the $V_{DD} = 0.55V$ results beyond the $V_{DD} = 0.75V$ data points. The measurements for $V_{DD} = 0.35V$ and $V_{DD} = 0.75V$ behave as expected, slightly translated below and above the good lot measurements of $V_{DD} = 0.55V$. Thus, the effectiveness of body biasing on a whole chip level, i.e., core granularity, is concluded.

Now applying these core-grained capabilities to application specific body biasing schemes, Fig. 8.30 depicts the leakage penalties (negative values) and reductions (positive values) achieved by core-grained body biasing for the entire chip.
The results clearly show that when staying at the regular, unscaled maximum clock frequency (LP), leakage can be reduced and in the worst-case, it does not yield any reduction at all over zero-bias. For the forward body biasing frequency scaling case (HP) however, the strong need for more fine-grained schemes becomes obvious. Without the ability to focus on forward body bias which leads to an exponential leakage increase, the leakage penalty becomes intolerable such as with the `fir` filter.

8.2.2 Coarse-Grained Body Biasing

As the previous simulative result sections have shown, coarse-grained body bias constitutes a significant step forward over core-grained, or in this case chip-grained body bias. Coarse-grained body bias allows to focus the application specific body bias per PE and thus to achieve more leakage reduction in the LP corner and greatly reduced leakage penalties in the HP corner as depicted in Fig. 8.31. For the LP corner, now all applications, even the `fir` filter shows leakage reductions achieved by disabling unused processing elements using

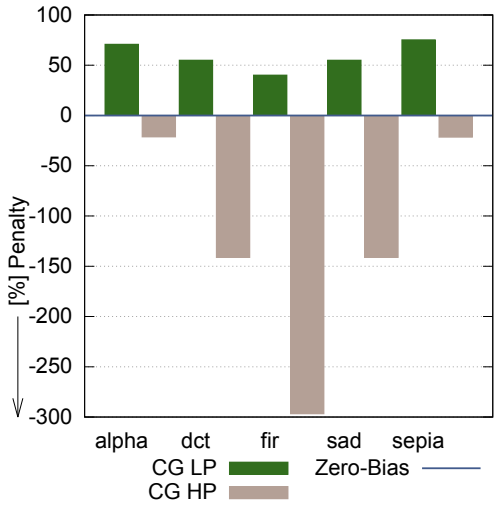


Figure 8.31: Leakage reduction or penalty respectively for coarse-grained body biasing applied to five benchmark applications at $V_{DD} = 0.55V$

strong reverse body bias, while the PEs running the expensive multiplications continue working at zero-bias. For the HP corner, on the other hand, the leakage penalty is almost halved compared to core-grained body biasing. While this is still not ideal, it constitutes a major improvement.

8.2.3 Fine-Grained Body Biasing

Ultimately, if body bias should be used in the most beneficial manner, forward body bias needs to be restricted to only those components that cannot work properly without and reverse body bias spread to all components that can take it in light of the timing constraints. Fine-grained body bias is a method to get very close to this ideal, as the following results in Figs. 8.32 to 8.36 will show. All individual leakage measurements indicate not only the attained leakage, but also the temperature at the time of measurement. As the results are obtained from a lot of six chips, all measurements including the chip case temperature are averaged. For brevity, the presented measurements are always measurements of leakage with body bias applied to a given domain type (e.g., MEML, MEMH, ALUL, ALUH), with the exception of the first PE's lower memory domain, containing all DRP controllers (`C0UMEML`). Thus, for MEMH, ALUL and ALUH, the leakage reading depicts the leakage incurred by biasing all four domains of the type in the DRP.

Fig. 8.32 depicts the incurred leakage for the `C0UMEML` domains as well as the three remaining MEML domains in the DRP. It also shows that despite `C0UMEML` containing all controllers, memories, i.e., the `MEMLx3` reading dominates all other leakage sources. This can be seen by the extraordinary increase in leakage when applying forward body bias on `MEMLx3` or by the strong reduction in leakage when applying reverse body biasing.

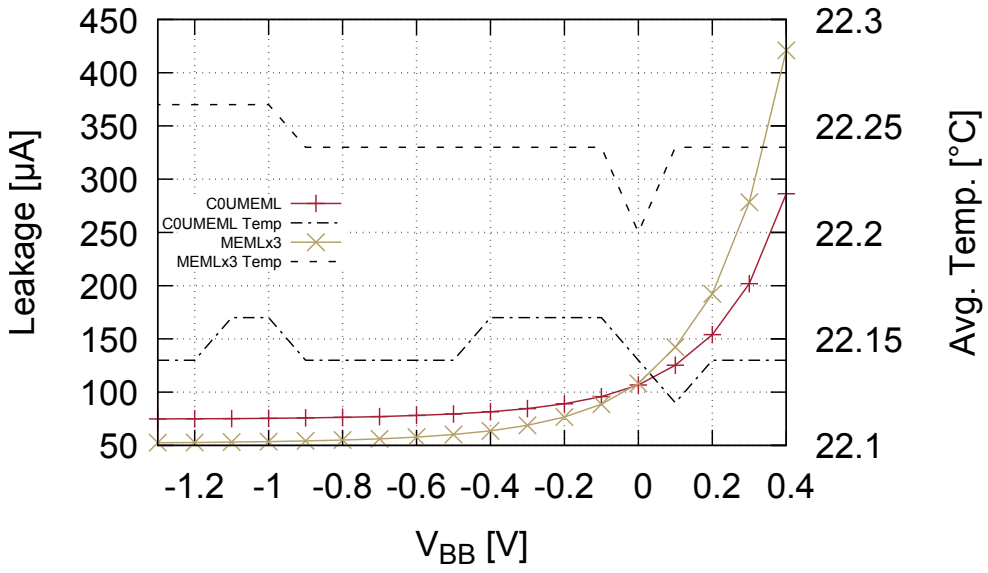


Figure 8.32: Static current and case temperature while changing the body bias of the three MEML domains as well as the `C0UMEML` domain at a supply voltage of $V_{DD} = 0.55V$

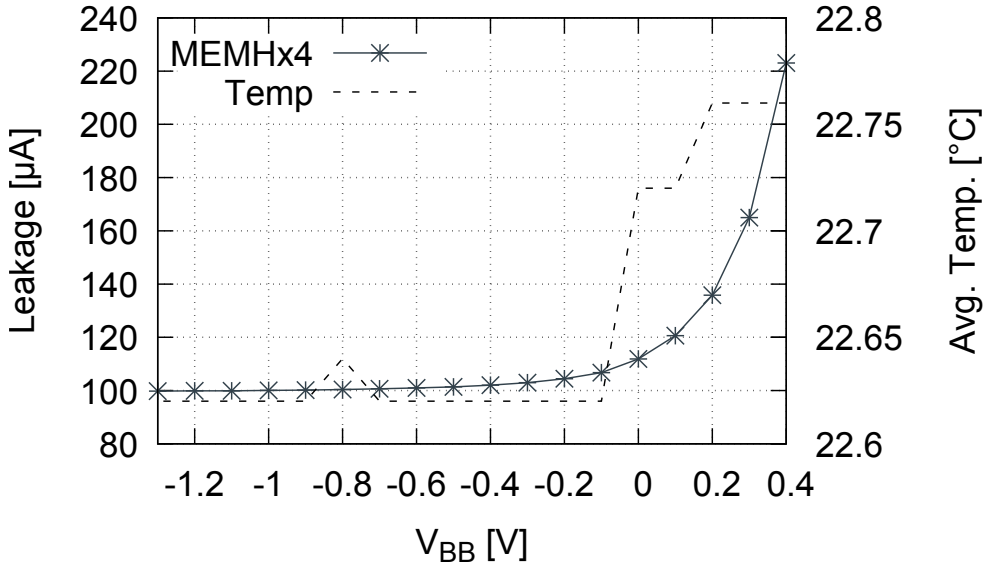


Figure 8.33: Static current and case temperature while changing the body bias of the MEMH domains at a supply voltage of $V_{DD} = 0.55V$

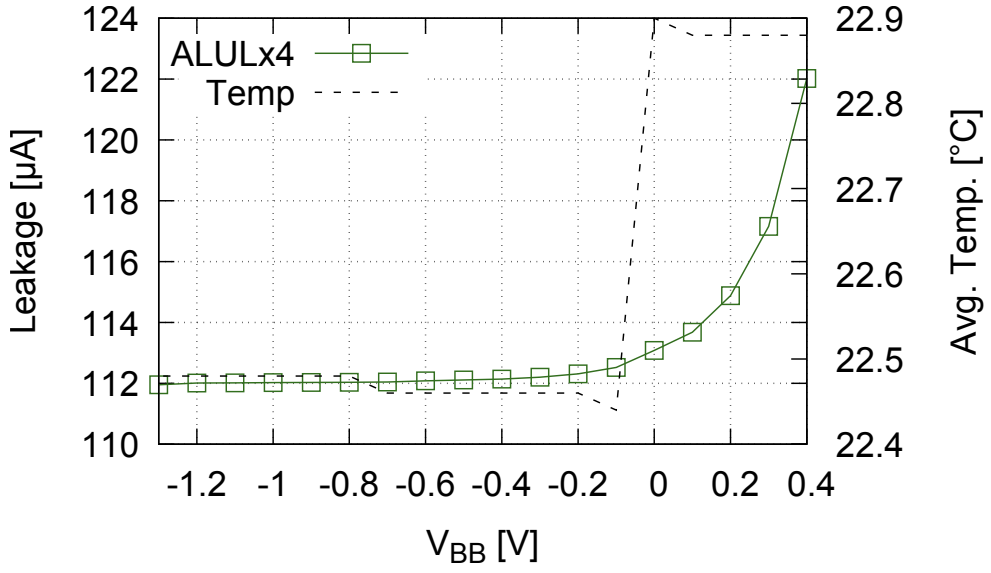


Figure 8.34: Static current and case temperature while changing the body bias of the ALUL domains at a supply voltage of $V_{DD} = 0.55V$

MEMH is the second largest domain type and thus also incurs significant leakage in foward direction and also allows for considerable leakage reductions when applying reverse body bias. ALUL, on the other hand, is the smallest domain type and can be seen as a mix of research interest in extremely fine body bias domains and the attempt to keep the critical body bias domain ALUH as small as possible. As ALUH is the only domain which is exhibited to maximum forward body bias, it is vital to keep it as small as possible. In this regard, ALUH may be considered the most important or most critical body bias domain type. It contains all arithmetic operations per PE and thus will require the maximum forward body bias. ALUH is restricted to the bare minimum it requires while also facilitating some resource sharing opportunities among arithmetic operations while at the same time omitting any actual cross-domain resource sharing. Despite all this, as ALUH also contains the output multiplexer as part of the critical path, ALUL and ALUH do have some cross-domain resource sharing. However, the operations in ALUL are never timing critical enough to cause an actual cross-domain resource sharing conflict.

This fine-grained body biasing scheme now allows restricting the forward body bias to wherever necessary, while allowing reverse body bias in large unrequired areas such as the upper half of the context memory etc. This, of course, is reflected in the leakage reductions over all corners and applications as depicted in Fig. 8.36.

Even when scaling clock frequency using forward body bias, leakage can be considerably reduced, as only a comparatively tiny part of the chip needs a strong forward body bias.

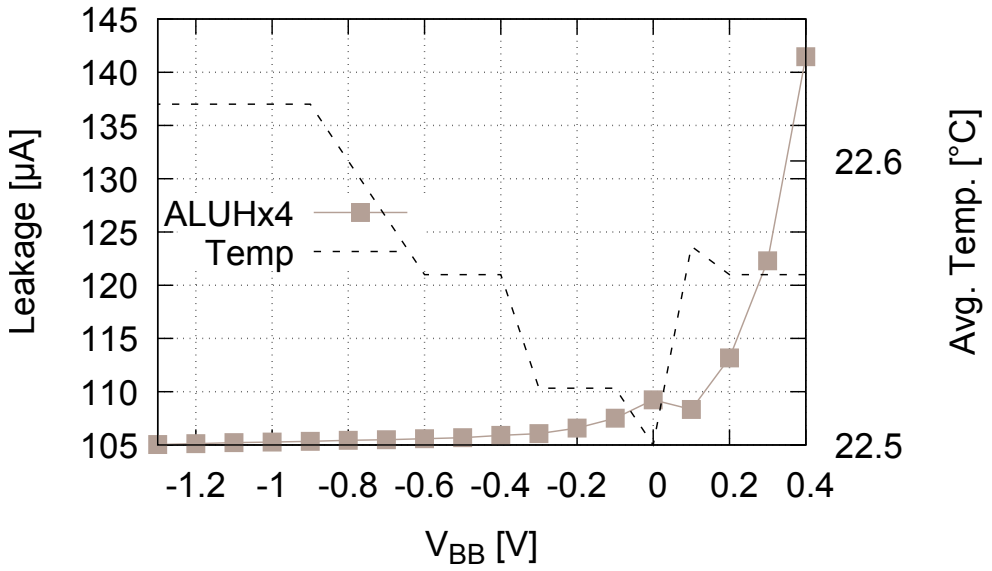


Figure 8.35: Static current and case temperature while changing the body bias of the ALUH domains at a supply voltage of $V_{DD} = 0.55V$

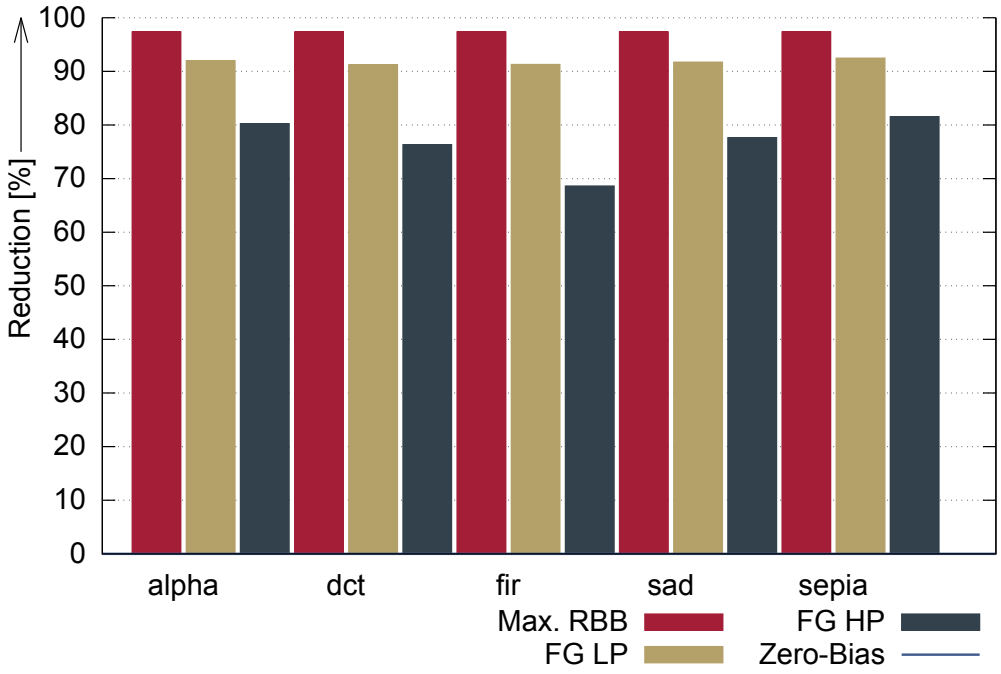


Figure 8.36: Fine-grained body biasing (FG) for the low power corner (LP) as well as the high performance, frequency scaling corner (HP) versus maximum reverse body biasing as well as zero-bias as baseline at 0% reduction

Corner	F_{max}	$F_{max,BB}/F_{max}$
Max. RBB	$F_{max,RBB} = 12.77MHz$	0.18
LP	$F_{max} = 72.55MHz$	1
HP	$F_{max,FBB} = 111.58MHz$	1.54

Table 8.3: Maximum clock frequencies at $V_{DD} = 0.55V$ per corner with regular F_{max} at the LP corner

Tab. 8.3 lists the maximum clock frequencies as well as their ratio normalized to regular F_{max} per corner. In comparison to applying a strong forward body bias to the memory domains, the leakage penalty incurred with ALUH is neglectable and is compensated with the savings in memory domains. In this regard, an individual ALUL domain is also sensible, as it helps to keep the leakage penalty low. While the leakage reduction reaches a minimum at the `fir` filter application, it is still a solid leakage reduction.

In sum, these findings prove that proper fine-grained partitioning into body bias domains using measures of timing criticalities and activation leads to the desired, further improvements over coarse-grained body biasing.

8.3 Discussion

As the original MuCCRA architectures were built for use with SRAM context memories, the inferred memory devices in this implementation are a lot faster than the originally anticipated memory. While this enhances the positive results, the general trend would be still the same even with SRAM devices. Up till now however, there are no detailed studies on the behavior of SRAMs in modern FDSOI under strong bias. Thus, an evaluation with inferred memories had to be chosen.

The general insights that can be derived from these results section can be split into two types of body biasing:

1. Reverse body biasing minimizing leakage to reach the lowest power consumptions possible (ultra-low-power)

2. High-Performance applications scaling clock frequency using forward body biasing to maximize energy efficiency

In regard to the first category, large parts of the chip or the entire chip are using strong reverse body bias, as such modes usually do not require high performance. In comparison to dynamic power, such modes often exhibit relatively large static power consumption. This static power consumption can be efficiently reduced by employing reverse body biasing. As at such low clock frequencies the timing criticality of all components is virtually non-existent, the entire part or chip can be put into a single body bias domain. If, however, the chip is geared towards ultra-low-power and performance, it could

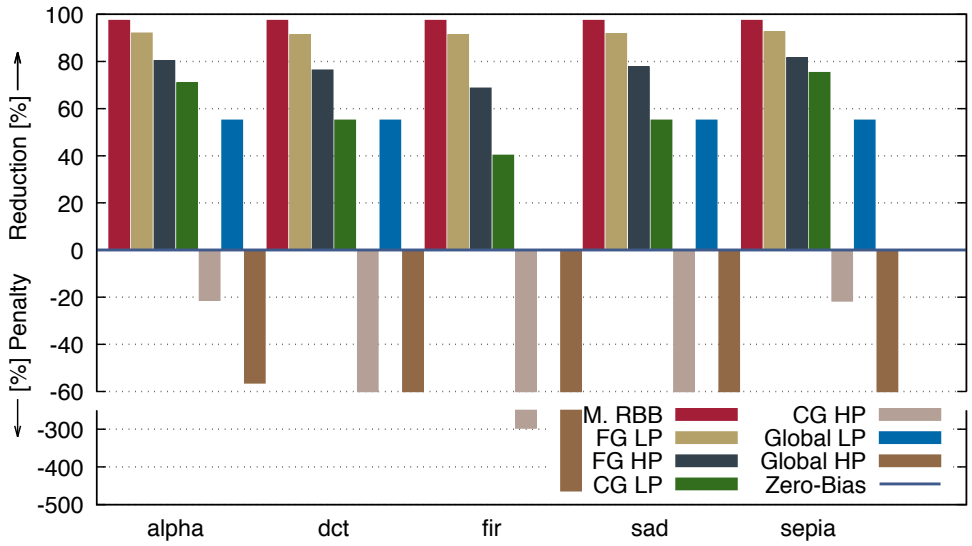


Figure 8.37: Comparative overview of fine-grained versus coarse-grained versus core-grained body biasing schemes applied to five benchmark applications at $V_{DD} = 0.55V$

tremendously benefit from the extremely high-frequency boost of forward body bias at low supply voltages, thus killing two birds with one stone: attaining ultra-low-power as well as high performance considering the supply voltage. For this purpose, low supply voltages are combined with the high-performance corner discussed in the following paragraph.

When a design should be geared toward high performance at maximum attainable energy efficiency, it could possibly benefit from having as many body bias domains as there are parts with different timing criticality and activation. Only when a design is partitioned along these lines, the severe penalties of forward body biasing used for frequency scaling can be mitigated. Furthermore, a direct relation between the granularity and the leakage penalty or reduction can be drawn.

Consider Fig. 8.37, which is a overall comparison of all considered body biasing schemes from core- to fine-grained body biasing evaluated per application using the test chip described in chapter 7. Each application block starts with maximum reverse body biasing at a frequency of $F_{max,RBB}$, followed by the LP corner at F_{max} and finally followed by the HP corner at $F_{max,FBB}$. Each of these corners is evaluated for fine-grained (first), coarse-grained (second) and global body biasing (last). Thus, it is clearly visible that fine-grained schemes get very close to the theoretical optimum of maximum reverse body biasing, however at more than five times the clock frequency. Furthermore, this comparison also shows that fine-grained body biasing is superior to all other approaches proposed in this thesis.

These findings are especially important with smaller technology nodes where leakage will be considerably exacerbated. As FDSOI is reported to be scalable down to at least 10nm [29], it will be vital to be able to contain leakage during forward body biasing. This aspect also has to be seen in the light of the available alternatives. Consider the only actual contender: supply voltage scaling. In order to justify all the additional hardware, e.g. voltage level shifters, performant voltage regulators etc., considerably sized domains are required. This increased granularity also exacerbates the incurred penalty, leakage and dynamic power wise. In stark contrast, body biasing needs nothing more than the actual body bias creation with power straps supplying the tap cells. This allows to pinpoint portions of a chip that need an extra performance boost while reducing power consumption at the same time in parts that need less performance than available at a given supply voltage. Also, compared to power gating, body biasing offers an interesting alternative. By combining clock gating with strong reverse body biasing, power consumption of the affected parts is minimized while neither power switches nor retention cells are needed as the state of the circuit is retained.

In this light, body biasing can also be used as a last resort artificial pipeline balancer, adjusting all parts to a given clock cycle. Although of course, this is a very desirable feature as despite best design efforts, totally balanced designs are hard to come by. Furthermore, by using body biasing in its most desirable application, i.e., the above-mentioned frequency scaling, it of course effects pipelining as well. If circuits can operate faster, additional pipeline stages may be omitted altogether, which is a low-power design method itself. These advantages combined with the ability to actually lower supply voltage while staying at the same clock frequency should disperse any doubts in regard to body biasing as a low-power technique.

9 Conclusion and Outlook

The FDSOI construction of processes as manufactured by STMicroelectronics, Renesas and Globalfoundries significantly extended body biasing capabilities, but the state of the art exhibited a big gap to exploit body biasing in FDSOI fully. The conducted research can be broken down to one simple golden rule, by which the approaches proposed in this thesis function: *The more focussed, either spatially or temporally, the better the effect of body biasing*. This points towards the central contribution of this thesis, to explore methods to break up granularity limitations of previous approaches and evaluate them in a detailed manner, allowing projections for complex, future computing architectures consisting of great numbers of processing elements.

In contrast to DVFS, the implementation of body biasing requires far smaller overheads and thus allows highly effective fine-grained solutions. While DVFS would also benefit considerably from more fine-grained approaches, it is simply not feasible at present. Without the need for level-shifters, voltage generators that are capable of driving huge dynamic loads and the accompanying infrastructure, options for more advanced application schemes arise. In FDSOI with body biasing, even small groups of standard cells can be pinpointed and adjusted to minimize power consumption and to maximize energy efficiency. This, however, introduces great complexity in regard to which standard cells should be biased together, by how much and when, also in the light of the still very existent constraints and the small, but in sum possibly expensive overheads of aggressive partitioning.

To tackle this challenge, this thesis proposed two principal partitioning approaches: Guided pre-partitioning with subsequent leakage trade-off based partitioning into body bias domains, and, fully automatized, standard cell netlist based domain candidate exploration. To demonstrate the effectiveness of the proposed methods, a Dynamically Reconfigurable Processor architecture called MuCCRA4 has been used. DRPs with their arrays of PEs provided a good trade-off between generality, pointing towards present and future many-core processors, as well as the simplicity that allowed this in-depth evaluation regarding body bias applications and which made test-chip manufacturing feasible. This DRP architecture has then been partitioned into varying granularities using several approaches. Following the trade-off aspects of body biasing, activation and timing were determined to be suitable guidelines to determine body bias domains. Where the first approach employing guided, pre-partitioning by hand, domain candidate exploration successfully replaced this step with a more powerful automatized approach. This is done by algorithmically grouping standard cells into domain candidates which were then also merged with a subset based merging algorithm.

In contrast to DVFS, however, body biasing introduces another level of complexity as it is not a single supply voltage that needs to be governed, but the bias voltages of multiple domains which may be interdependent. This phenomenon was described as cross-domain resource sharing, where body bias assignments could not be simply determined by biasing the domain with the seemingly lowest leakage current to yield optimal results in regard to energy efficiency. Furthermore, changing the timing of one domain also affects the timing of dependent domains. These effects have been successfully accommodated using a metric called body biasing impact. This metric was defined to determine iteratively in each step the most leakage and timing efficient domain for a discrete change in body bias, resulting in an energy efficient body bias domain partitioning with leakage minimal body bias assignments. With different instructions being used, leveraging different application's timing slack allowed for further optimization of body bias application.

Simulative results showed that significant gains in energy efficiency over the state of the art could be attained by applying body bias in a more fine-grained manner. Together with the implementation of a fine-grained body biasing enabled DRP design, the validity of the proposed approaches was then also shown using in-silicon measurements of the leakage current affected by the partitioning, assignment and application dependent schemes. The in-silicon results clearly showed leakage reductions very close to and beyond SPICE based evaluations of fine-grained body biasing. While in SPICE simulations, maximum leakage reductions converged at around 80% versus naive body biasing, in-silicon evaluations showed results crossing the 90% mark. These values certainly demand caution but are very well within the capabilities FDSOI technologies offer and demonstrated before. In the case of this thesis, however, leakage increases in forward body bias direction or clock frequency degradation in the reverse body bias direction have been minimized to a fraction of the penalties incurred in previous technology demonstrations.

There are, however, also many aspects that could not be sufficiently addressed in this thesis and which, among others, point to remaining research that needs to be conducted to fully exploit FDSOI and its body biasing capabilities. Body biasing in general, but especially the aspects of dynamic body biasing proved to require far more research in architectural, software, EDA, technological and electrical respects. First of all, body biasing can be exploited architecturally in a multitude of ways. One intriguing architectural technique is artificial pipeline balancing. With body biasing, the timing of pipeline stages or of combinatorial circuits inside a pipeline stage can be easily adjusted using body biasing if considered during design. This allows designing pipelined design more freely, with balancing, or even dynamic balancing after manufacturing. Furthermore, body biasing also could be used to incorporate more actual specialization while compensating the burden on static power consumption via reverse body biasing when certain specialized circuits are not required. Additionally, on the software side, many optimizations can be geared towards scheduling of operations by timing slack or other methods considering the need to switch body biases for optimal operation. Moreover, in regard to design automation, current EDA tools are not even close to incorporate body biasing in a more fine-grained

fashion than existing DVFS like approaches. Also in regard to sign-off and reliability analysis, variability or even only temperature variation, these effects have not been not considered sufficiently.
But then, of course, while all this applies to body biasing with temporal granularities like DVFS, i.e. where switching overheads can be neglected, it is the case all the more for actual dynamic body biasing. Regarding dynamic body biasing, there is also still a lot of unused potential on the technological and electrical engineering side, apart from the actual scaling challenges. Transistor architecture still has much potential for optimization, such as reducing body capacitance or optimizing body effect utilization for improved dynamic body biasing capabilities.

All these challenges in mind, body biasing is a powerful technique. While body biasing is virtually independent of technology scaling as a technique, and even though it can be applied to most transistor architectures, FDSOI depends significantly on it. Its exploitation is thus imperative for the success of FDSOI technology which does not only have fierce competition but also has a couple of aces up its sleeve. The first one is body biasing which has been discussed at great lengths in this thesis, but also economics, as cost per transistor is no longer decreasing. Together with all the benefits of being planar technology, FDSOI has the right attributes to become a new commodity technology for an extensive range of applications. From automotive and aerospace applications with known planar reliability and radiation profiles, highly energy efficient high-performant consumer hardware to ultra-low-power, but flexibly capable sensor node applications. This broad application range and the body bias enabled capabilities to broaden its power and performance range will make sure that FDSOI will stay around for a long time.

Bibliography

[1] R. Merrit, "Arm cto: power surge could create 'dark silicon'," *EETimes*, oct. 2009.

[2] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. IEEE, 2011, pp. 365–376.

[3] M. Wolf, "Ultralow power and the new era of not-so-vlsi," *IEEE Design & Test*, vol. 33, no. 4, pp. 109–113, 2016.

[4] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[5] A. Schmidt, "Analog circuit design in pd-soi cmos technology for high temperatures up to 400° c using reverse body biasing (rbb)," Ph.D. dissertation, Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften» Elektrotechnik und Informationstechnik, 2014.

[6] D. Jacquet, F. Hasbani, P. Flatresse, R. Wilson, F. Arnaud, G. Cesana, T. Di Gilio, C. Lecocq, T. Roy, A. Chhabra *et al.*, "A 3 ghz dual core processor arm cortex tm-a9 in 28 nm utbb fd-soi cmos with ultra-wide voltage range and energy efficiency optimization," *Solid-State Circuits, IEEE Journal of*, vol. 49, no. 4, pp. 812–826, 2014.

[7] T. Ohtou, N. Sugii, and T. Hiramoto, "Impact of parameter variations and random dopant fluctuations on short-channel fully depleted soi mosfets with extremely thin box," *Electron Device Letters, IEEE*, vol. 28, no. 8, pp. 740–742, Aug 2007.

[8] J.-P. Colinge, *Silicon-on-Insulator Technology: Materials to VLSI: Materials to Vlsi*. Springer Science & Business Media, 2004.

[9] H.-K. Lim and J. G. Fossum, "Threshold voltage of thin-film silicon-on-insulator (soi) mosfet's," *Electron Devices, IEEE Transactions on*, vol. 30, no. 10, pp. 1244–1251, 1983.

[10] T. Ishigaki, N. Sugii, R. Tsuchiya, S. Kimura, and Y. Morita, *Ultralow-power LSI Technology with Silicon on Thin Buried Oxide (SOTB) CMOSFET.* INTECH Open Access Publisher, 2010.

[11] T. Takahashi, T. Matsuki, T. Shinada, Y. Inoue, and K. Uchida, "Comparison of self-heating effect (she) in short-channel bulk and ultra-thin box soi mosfets: impacts

of doped well, ambient temperature, and soi/box thicknesses on she," in *Electron Devices Meeting (IEDM), 2013 IEEE International*. IEEE, 2013, pp. 7–4.

[12] L. Wang, A. R. Brown, M. Nedjalkov, C. Alexander, B. Cheng, C. Millar, and A. Asenov, "Impact of self-heating on the statistical variability in bulk and soi finfets," *Electron Devices, IEEE Transactions on*, vol. 62, no. 7, pp. 2106–2112, 2015.

[13] T. Kuroda and T. Sakurai, "Body biasing," in *Leakage in Nanometer CMOS Technologies*. Springer, 2006, pp. 105–140.

[14] H. Okuhara, K. Kitamori, Y. Fujita, K. Usami, and H. Amano, "An optimal power supply and body bias voltage for a ultra low power micro-controller with silicon on thin box mosfet," in *Low Power Electronics and Design (ISLPED), 2015 IEEE/ACM International Symposium on*. IEEE, 2015, pp. 207–212.

[15] T. Sakurai and A. R. Newton, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *Solid-State Circuits, IEEE Journal of*, vol. 25, no. 2, pp. 584–594, 1990.

[16] S. H. Dhong, J.-T. Tzeng, K. M. Babaji, R. Krishnan, L.-C. Lu, and T.-P. Guo, "Planar compatible fdsoi design architecture," May 14 2013, uS Patent 8,443,306.

[17] H. Amano, "A survey on dynamically reconfigurable processors," *IEICE transactions on Communications*, vol. 89, no. 12, pp. 3179–3187, 2006.

[18] M. Motomura, "A dynamically reconfigurable processor architecture," in *Microprocessor Forum, 2002*, 2002.

[19] T. Oppold, T. Schweizer, J. Oliveira Filho, S. Eisenhardt, and W. Rosenstiel, "Crc–concepts and evaluation of processor-like reconfigurable architectures (crc–konzepte und bewertung prozessorartig rekonfigurierbarer architekturen)," *it–Information Technology (vormals it+ ti)*, vol. 49, no. 3, pp. 157–164, 2007.

[20] R. Tsuchiya, M. Horiuchi, S. Kimura, M. Yamaoka, T. Kawahara, S. Maegawa, T. Ipposhi, Y. Ohji, and H. Matsuoka, "Silicon on thin box: A new paradigm of the cmosfet for low-power high-performance application featuring wide-range back-bias control," in *Electron Devices Meeting, 2004. IEDM Technical Digest. IEEE International*. IEEE, 2004, pp. 631–634.

[21] N. Sugii, R. Tsuchiya, T. Ishigaki, Y. Morita, H. Yoshimoto, K. Torii, and S. Kimura, "Comprehensive study on vth variability in silicon on thin box (sotb) cmos with small random-dopant fluctuation: finding a way to further reduce variation," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*. IEEE, 2008, pp. 1–4.

[22] N. Sugii, R. Tsuchiya, T. Ishigaki, Y. Morita, H. Yoshimoto, and S. I. Kimura, "Local variability and scalability in silicon-on-thin-box (sotb) cmos with small random-dopant fluctuation," *Electron Devices, IEEE Transactions on*, vol. 57, no. 4, pp. 835–845, 2010.

[23] T. Ishigaki, R. Tsuchiya, Y. Morita, N. Sugii, and S. I. Kimura, "Effects of device structure and back biasing on hci and nbti in silicon-on-thin-box (sotb) cmosfet," *Electron Devices, IEEE Transactions on*, vol. 58, no. 4, pp. 1197–1204, 2011.

[24] N. Planes, O. Weber, V. Barral, S. Haendler, D. Noblet, D. Croain, M. Bocat, P.-O. Sassoulas, X. Federspiel, A. Cros *et al.*, "28nm fdsoi technology platform for high-speed low-voltage digital applications," in *VLSI Technology (VLSIT), 2012 Symposium on*. IEEE, 2012, pp. 133–134.

[25] P. Magarshack, P. Flatresse, and G. Cesana, "Utbb fd-soi: A process/design symbiosis for breakthrough energy-efficiency," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 952–957.

[26] X. Federspiel, D. Angot, M. Rafik, F. Cacho, A. Bajolet, N. Planes, D. Roy, M. Haond, and F. Arnaud, "28nm node bulk vs fdsoi reliability comparison," in *2012 IEEE International Reliability Physics Symposium (IRPS)*, 2012.

[27] Globalfoundries, "Globalfoundries launches industry first 22nm fd-soi technology platform," July 2015, available online: http://globalfoundries.com/newsroom/press-releases/2015/07/13/globalfoundries-launches-industry-s-first-22nm-fd-soi-technology-platform, accessed February 17th, 2016.

[28] O. Weber, E. Josse, F. Andrieu, A. Cros, E. Richard, P. Perreau, E. Baylac, N. Degors, C. Gallon, E. Perrin *et al.*, "14nm fdsoi technology for high speed and energy efficient applications," in *VLSI Technology (VLSI-Technology): Digest of Technical Papers, 2014 Symposium on*. IEEE, 2014, pp. 1–2.

[29] Q. Liu, B. DeSalvo, P. Morin, N. Loubet, S. Pilorget, F. Chafik, S. Maitrejean, E. Augendre, D. Chanemougame, S. Guillaumet *et al.*, "Fdsoi cmos devices featuring dual strained channel and thin box extendable to the 10nm node," in *2014 IEEE International Electron Devices Meeting*. IEEE, 2014, pp. 9–1.

[30] S. Natarajan, M. Agostinelli, S. Akbar, M. Bost, A. Bowonder, V. Chikarmane, S. Chouksey, A. Dasgupta, K. Fischer, Q. Fu *et al.*, "A 14nm logic technology featuring 2 nd-generation finfet, air-gapped interconnects, self-aligned double patterning and a 0.0588 μm 2 sram cell size," in *Electron Devices Meeting (IEDM), 2014 IEEE International*. IEEE, 2014, pp. 3–7.

[31] S.-Y. Wu, C. Y. Lin, M. Chiang, J. Liaw, J. Cheng, S. Yang, M. Liang, T. Miyashita, C. Tsai, B. Hsu *et al.*, "A 16nm finfet cmos technology for mobile soc and computing applications," in *Electron Devices Meeting (IEDM), 2013 IEEE International*. IEEE, 2013, pp. 9–1.

[32] E. IQ, "Fully gate-all-around silicon nanowire cmos devices," 2014, available Online: http://www.electroiq.com/content/eiq-2/en/articles/sst/print/volume-51/issue-5/features/nanotechnology/fully-gate-all-around-silicon-nanowire-cmos-devices.html Accessed: January 2014.

[33] K. T. Lee, W. Kang, E.-A. Chung, G. Kim, H. Shim, H. Lee, H. Kim, M. Choe, N.-I. Lee, A. Patel *et al.*, "Technology scaling on high-k & metal-gate finfet bti reliability," in *Reliability Physics Symposium (IRPS), 2013 IEEE International*. IEEE, 2013, pp. 2D–1.

[34] A. Paul, A. Bryant, T. Hook, C. Yeh, V. Kamineni, J. Johnson, N. Tripathi, T. Yamashita, G. Tsutsui, V. Basker *et al.*, "Comprehensive study of effective current variability and mosfet parameter correlations in 14nm multi-fin soi finfets," in *Electron Devices Meeting (IEDM), 2013 IEEE International*. IEEE, 2013, pp. 13–5.

[35] T. Kobayashi and T. Sakurai, "Self-adjusting threshold-voltage scheme (sats) for low-voltage high-speed operation," in *Custom Integrated Circuits Conference, 1994., Proceedings of the IEEE 1994*. IEEE, 1994, pp. 271–274.

[36] T. Kuroda and T. Sakurai, "Threshold-volgage control schemes through substrate-bias for low-power high-speed cmos lsi design," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13, no. 2-3, pp. 191–201, 1996.

[37] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *Solid-State Circuits, IEEE Journal of*, vol. 37, no. 11, pp. 1396–1402, 2002.

[38] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*. ACM, 2002, pp. 721–725.

[39] C. Neau and K. Roy, "Optimal body bias selection for leakage improvement and process compensation over different technology generations," in *Proceedings of the 2003 international symposium on Low power electronics and design*. ACM, 2003, pp. 116–121.

[40] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Mitigating parameter variation with dynamic fine-grain body biasing," in *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*. IEEE, 2007, pp. 27–42.

[41] A. Sathanur, A. Pullini, L. Benini, G. De Micheli, and E. Macii, "Physically clustered forward body biasing for variability compensation in nanometer cmos design," in *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2009, pp. 154–159.

[42] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *Proceedings of the 39th annual IEEE/ACM international symposium on microarchitecture*. IEEE Computer Society, 2006, pp. 347–358.

[43] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*. IEEE, 2007, pp. 38–43.

[44] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*. IEEE, 2008, pp. 123–134.

[45] A. W. Yin, L. Guang, E. Nigussie, P. Liljeberg, J. Isoaho, and H. Tenhunen, "Architectural exploration of per-core dvfs for energy-constrained on-chip networks," in *Digital System Design, Architectures, Methods and Tools, 2009. DSD'09. 12th Euromicro Conference on*. IEEE, 2009, pp. 141–146.

[46] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom *et al.*, "A 48-core ia-32 message-passing processor with dvfs in 45nm cmos," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*. IEEE, 2010, pp. 108–109.

[47] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: adaptive dvfs and thread packing under power caps," in *Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*. ACM, 2011, pp. 175–185.

[48] T. Kolpe, A. Zhai, and S. S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 2011, pp. 1–6.

[49] L. Benini, G. De Micheli, E. Macii, M. Poncino, and R. Scarsi, "Symbolic synthesis of clock-gating logic for power optimization of synchronous controllers," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 4, no. 4, pp. 351–375, 1999.

[50] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 47, no. 3, pp. 415–420, 2000.

[51] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on rtl clock-gating," in *Design Automation Conference, 2003. Proceedings*. IEEE, 2003, pp. 622–627.

[52] F. Emnett and M. Biegel, "Power reduction through rtl clock gating," *SNUG, San Jose*, 2000.

[53] H. Li, S. Bhunia, Y. Chen, T. Vijaykumar, and K. Roy, "Deterministic clock gating for microprocessor power reduction," in *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*. IEEE, 2003, pp. 113–122.

[54] K. Usami and N. Ohkubo, "A design approach for fine-grained run-time power gating using locally extracted sleep signals," in *Computer Design, 2006. ICCD 2006. International Conference on*. IEEE, 2007, pp. 155–161.

[55] L. Leinweber and S. Bhunia, "Fine-grained supply gating through hypergraph partitioning and shannon decomposition for active power reduction," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 373–378.

[56] A. Sathanur, A. Pullini, L. Benini, A. Macii, E. Macii, and M. Poncino, "A scalable algorithmic framework for row-based power-gating," in *Design, Automation and Test in Europe, 2008. DATE'08*. IEEE, 2008, pp. 379–384.

[57] Y. Saito, T. Shirai, T. Nakamura, T. Nishimura, Y. Hasegawa, S. Tsutsumi, T. Kashima, M. Nakata, S. Takeda, K. Usami *et al.*, "Leakage power reduction for coarse grained dynamically reconfigurable processor arrays with fine grained power gating technique," in *FPT 2008. International Conference on Field-Programmable Technology*. IEEE, 2008, pp. 329–332.

[58] S.-Y. Chen, R.-B. Lin, H.-H. Tung, and K.-W. Lin, "Power gating design for standard-cell-like structured asics," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2010, pp. 514–519.

[59] N. S. Kim, D. Blaauw, and T. Mudge, "Leakage power optimization techniques for ultra deep sub-micron multi-level caches," in *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society, 2003, p. 627.

[60] P. Gupta, A. B. Kahng, P. Sharma, and D. Sylvester, "Selective gate-length biasing for cost-effective runtime leakage control," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 327–330.

[61] ——, "Gate-length biasing for runtime-leakage control," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 8, pp. 1475–1485, 2006.

[62] A. Calimera, E. Macii, M. Poncino, and R. Bahar, "Temperature-insensitive synthesis using multi-vt libraries," in *Proceedings of the 18th ACM Great Lakes symposium on VLSI*. ACM, 2008, pp. 5–10.

[63] C. Yeh, Y.-S. Kang, S.-J. Shieh, and J.-S. Wang, "Layout techniques supporting the use of dual supply voltages for cell-based designs," in *Design Automation Conference, 1999. Proceedings. 36th*. IEEE, 1999, pp. 62–67.

[64] R. Puri, L. Stok, J. Cohn, D. Kung, D. Pan, D. Sylvester, A. Srivastava, and S. Kulkarni, "Pushing asic performance in a power envelope," in *Proceedings of the 40th annual Design Automation Conference*. ACM, 2003, pp. 788–793.

[65] F. Li, Y. Lin, and L. He, "Vdd programmability to reduce fpga interconnect power," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*. IEEE Computer Society, 2004, pp. 760–765.

[66] Y. Akgul, D. Puschini, S. Lesecq, E. Beigne, P. Benoit, and L. Torres, "Methodology for power mode selection in fd-soi circuits with dvfs and dynamic body biasing," in *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2013 23rd International Workshop on*. IEEE, 2013, pp. 199–206.

[67] Y. Akgul, D. Puschini, S. Lesecq, E. Beigné, I. Miro-Panades, P. Benoit, and L. Torres, "Power management through dvfs and dynamic body biasing in fd-soi circuits," in *Proceedings of the 51st Annual Design Automation Conference*. ACM, 2014, pp. 1–6.

[68] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 3s, p. 111, 2014.

[69] R. Jevtic, H.-P. Le, M. Blagojevic, S. Bailey, K. Asanovic, E. Alon, and B. Nikolic, "Per-core dvfs with switched-capacitor converters for energy efficiency in manycore processors," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 23, no. 4, pp. 723–730, 2015.

[70] F. Conti, D. Rossi, A. Pullini, I. Loi, and L. Benini, "Energy-efficient vision on the pulp platform for ultra-low power parallel computing," in *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*. IEEE, 2014, pp. 1–6.

[71] D. Rossi, A. Pullini, I. Loi, M. Gautschi, K. Gurkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beignè, F. Clermidy, F. Abouzeid, P. Flatresse, and L. Benini, "193 mops/mw @ 162 mops, 0.32v to 1.15v voltage range multi-core accelerator for energy efficient parallel and sequential digital processing," in *Low-Power and High-Speed Chips (COOL CHIPS XIX), 2016 IEEE Symposium in*. IEEE, 2016.

[72] M. Hioki, T. Sekigawa, T. Nakagawa, H. Koike, Y. Matsumoto, T. Kawanami, and T. Tsutsumi, "Fully-functional fpga prototype with fine-grain programmable body biasing," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2013, pp. 73–80.

[73] M. Hioki, C. Ma, T. Kawanami, Y. Ogasahara, T. Nakagawa, T. Sekigawa, T. Tsutsumi, and H. Koike, "Sotb implementation of a field programmable gate array with fine-grained vt programmability," *Journal of Low Power Electronics and Applications*, vol. 4, no. 3, pp. 188–200, 2014.

[74] T. Toi, N. Nakamura, Y. Kato, T. Awashima, and K. Wakabayashi, "High-level synthesis challenges for mapping a complete program on a dynamically reconfigurable processor," *IPSJ Transactions on System LSI Design Methodology*, vol. 3, pp. 91–104, 2010.

[75] M. Suzuki, Y. Hasegawa, Y. Yamada, N. Kaneko, K. Deguchi, H. Amano, K. Anjo, M. Motomura, K. Wakabayashi, T. Toi *et al.*, "Stream applications on the dynamically reconfigurable processor," in *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*. IEEE, 2004, pp. 137–144.

[76] N. Suzuki, S. Kurotaki, M. Suzuki, N. Kaneko, Y. Yamada, K. Deguchi, Y. Hasegawa, H. Amano, K. Anjo, M. Motomura *et al.*, "Implementing and evaluating stream applications on the dynamically reconfigurable processor," in *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*. IEEE, 2004, pp. 328–329.

[77] Y. Hasegawa, S. Abe, H. Matsutani, H. Amano, K. Anjo, and T. Awashima, "An adaptive cryptographic accelerator for ipsec on dynamically reconfigurable processor," in *Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, 2005.* IEEE, 2005, pp. 163–170.

[78] K. Deguchi, S. Abe, M. Suzuki, K. Anjo, T. Awashima, and H. Amano, "Implementing core tasks of jpeg2000 encoder on the dynamically reconfigurable processor." in *ARCS Workshops*. Citeseer, 2005, pp. 12–18.

[79] T. Toi, N. Nakamura, Y. Kato, T. Awashima, K. Wakabayashi, and L. Jing, "High-level synthesis challenges and solutions for a dynamically reconfigurable processor," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. ACM, 2006, pp. 702–708.

[80] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "Adres: An architecture with tightly coupled vliw processor and coarse-grained reconfigurable matrix," in *International Conference on Field Programmable Logic and Applications*. Springer, 2003, pp. 61–70.

[81] W.-J. Lee, S.-O. Woo, K.-T. Kwon, S.-J. Son, K.-J. Min, G.-J. Jang, C.-H. Lee, S.-Y. Jung, C.-M. Park, and S.-H. Lee, "A scalable gpu architecture based on dynamically reconfigurable embedded processor," *High Performance Graphics*, pp. 5–7, 2011.

[82] B. Mei, S. Vernalde, D. Verkest, and R. Lauwereins, "A tightly coupled vliw/reconfigurable matrix and its modulo scheduling technique," in *New Algorithms, Architectures and Applications for Reconfigurable Computing*. Springer, 2005, pp. 15–28.

[83] G. Lee, K. Choi, and N. D. Dutt, "Mapping multi-domain applications onto coarse-grained reconfigurable architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, pp. 637–650, 2011.

[84] W.-J. Lee, Y. Shin, J. Lee, J.-W. Kim, J.-H. Nah, S. Jung, S. Lee, H.-S. Park, and T.-D. Han, "Sgrt: A mobile gpu architecture for real-time ray tracing," in *Proceedings of the 5th high-performance graphics conference*. ACM, 2013, pp. 109–119.

[85] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt, "Pact xppa self-reconfigurable data processing architecture," *the Journal of Supercomputing*, vol. 26, no. 2, pp. 167–184, 2003.

[86] J. M. Cardoso and M. Weinhardt, "Xpp-vc: Ac compiler with temporal partitioning for the pact-xpp architecture," in *International Conference on Field Programmable Logic and Applications*. Springer, 2002, pp. 864–874.

[87] S. Eisenhardt, T. Oppold, T. Schweizer, and W. Rosenstiel, "Optimizing partial reconfiguration of multi-context architectures," in *2008 International Conference on Reconfigurable Computing and FPGAs*. IEEE, 2008, pp. 67–72.

[88] T. Schweizer, T. Oppold, J. Oliveira Filho, S. Eisenhardt, K. Blocher, and W. Rosenstiel, "Exploiting slack time in dynamically reconfigurable processor architectures," in *Field-Programmable Technology, 2007. ICFPT 2007. International Conference on*. IEEE, 2007, pp. 381–384.

[89] M. Jo, V. P. Arava, H. Yang, and K. Choi, "Implementation of floating-point operations for 3d graphics on a coarse-grained reconfigurable architecture," in *2007 IEEE International SOC Conference*. IEEE, 2007, pp. 127–130.

[90] C. Brunelli, F. Garzia, D. Rossi, and J. Nurmi, "A coarse-grain reconfigurable architecture for multimedia applications supporting subword and floating-point calculations," *Journal of Systems Architecture*, vol. 56, no. 1, pp. 38–47, 2010.

[91] S. M. Jafri, T. N. Gia, S. Dytckov, M. Daneshtalab, A. Hemani, J. Plosila, and H. Tenhunen, "Neurocgra: A cgra with support for neural networks," in *High Performance Computing & Simulation (HPCS), 2014 International Conference on*. IEEE, 2014, pp. 506–511.

[92] H. Amano, Y. Hasegawa, S. Tsutsumi, T. Nakamura, T. Nishimura, V. Tanbunheng, A. Parimala, T. Sano, and M. Kato, "Muccra chips: Configurable dynamically-reconfigurable processors," in *Solid-State Circuits Conference, 2007. ASSCC'07. IEEE Asian*. IEEE, 2007, pp. 384–387.

[93] S. Saito, Y. Kohama, Y. Sugimori, Y. Hasegawa, H. Matsutani, T. Sano, K. Kasuga, Y. Yoshida, K. Niitsu, N. Miura *et al.*, "Muccra-cube: A 3d dynamically reconfigurable processor with inductive-coupling link," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*. IEEE, 2009, pp. 6–11.

[94] Y. Saito, T. Sano, M. Kato, V. Tunbunheng, Y. Yasuda, M. Kimura, and H. Amano, "Muccra-3: a low power dynamically reconfigurable processor array," in *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*. IEEE Press, 2010, pp. 377–378.

[95] T. Nishimura, K. Hirai, Y. Saito, T. Nakamura, Y. Hasegawa, S. Tsutsusmi, V. Tunbunheng, and H. Amano, "Power reduction techniques for dynamically reconfigurable processor arrays," in *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*. IEEE, 2008, pp. 305–310.

[96] T. Katagiri and H. Amano, "動的再構成プロセッサ muccra-4 の実装 (implementation of a dynamically reconfigurable processor muccra-4)," 研究報告システム *LSI* 設計技術 *(SLDM)*, vol. 2014, no. 22, pp. 1–6, 2014.

[97] ——, "A high speed design and implementation of dynamically reconfigurable processor using 28nm soi technology," in *Conference on Field Programmable Logic and Applications*. IEEE, 2014.

[98] J. M. Kühn, H. Amano, T. Katagiri, and W. Rosenstiel, "Leakage reduction using coarse-grained static body biasing in a dynamically reconfigurable processor," in *Fifth International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, 2014, https://dl.dropboxusercontent.com/u/3355605/isheart2014.pdf.

[99] J. M. Kühn, D. Peterson, H. Amano, O. Bringmann, and W. Rosenstiel, "Spatial and temporal granularity limits of body biasing in utbb-fdsoi," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015.

[100] J. M. Kühn, H. Amano, and W. Rosenstiel, "Using body biasing for energy efficient frequency scaling in a dynamically reconfigurable processor," in *Online Proceedings of The 19th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, 2015.

[101] J. M. Kühn, H. Amano, O. Bringmann, and W. Rosenstiel, "Fine-grained body biasing for frequency scaling in advanced soi processes," in *Low-Power and High-Speed Chips (COOL CHIPS XVIII), 2015 IEEE Symposium in*. IEEE, 2015.

[102] ——, "Leveraging fdsoi through body bias domain partitioning and bias search," in *Proceedings of the 53rd Design Automation Conference (DAC), Austin TX*, 2016.

[103] EDAUtils, "Verilog netlist parser with java and tcl api," 2016, available online: http://www.edautils.com/VlogNetlistParser.html, accessed May 29th, 2016.

[104] K. Kitamori, H. Su, and H. Amano, "Power optimization of a micro-controller with silicon on thin buried oxide," in *The 18th Workshop on Synthesis And System Integration of Mixed Information technologies*, 2013, pp. 68–731.

[105] J. M. Kühn, A. B. Ahmed, H. Okuhara, H. Amano, O. Bringmann, and W. Rosenstiel, "Muccra4-bb: A fine-grained body biasing capable drp," in *2016 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS XIX)*. IEEE, 2016, pp. 1–3.

[106] Synopsys Inc., "Primetime - static timing analysis," 2016, available online at: http://www.synopsys.com/Tools/Implementation/SignOff/PrimeTime/Pages/default.aspx, accessed June 14th 2016.

[107] ——, "Accelerate design innovation with design compiler," 2016, available online: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/default.aspx, accessed: June 23rd, 2016.