HAL
open science

Systematic Analysis of Energy and Delay Impact of Very Deep Submicron Process Variability Effects in Embedded SRAM Modules

Hua Wang, Miguel Miranda, Wim Dehaene, Francky Catthoor, Karen Maex

To cite this version:

Hua Wang, Miguel Miranda, Wim Dehaene, Francky Catthoor, Karen Maex. Systematic Analysis of Energy and Delay Impact of Very Deep Submicron Process Variability Effects in Embedded SRAM Modules. DATE'05, Mar 2005, Munich, Germany. pp.914-919. hal-00181235

HAL Id: hal-00181235

https://hal.science/hal-00181235

Submitted on 23 Oct 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Systematic Analysis of Energy and Delay Impact of Very Deep Submicron Process Variability Effects in Embedded SRAM Modules

Hua Wang*§, Miguel Miranda*, Wim Dehaene‡, Francky Catthoor*†, Karen Maex *†

* IMEC, Leuven, Belgium
wanghua,miranda,catthoor,maex@imec.be

‡ Dept. ESAT-MICAS, KUL, Leuven, Belgium
wim.dehaene@esat.kuleuven.ac.be

§ Also PhD student at the Katholieke Univ. Leuven, Belgium
† Also Professor at the Katholieke Univ. Leuven, Belgium

Abstract

Variability is becoming a serious problem in process technology for nanometer technology nodes. The increasing difficulty in controlling the uniformity of critical process parameters (e.g. doping levels) in the smaller devices, makes the electrical properties of such scaled devices much less predictable than in the past. In this paper, we study how these technology effects influence the energy and delay of a SRAM module. Despite the implications in the correct operation of the module, in practically all cases the affected memory implementations become also slower while consuming on average more energy than nominally. This is partly counter-intuitive and no existing literature describes this in a systematic generic way for SRAMs. In this paper, we identify and illustrate the different mechanisms behind this unexpected behavior and quantify the impact of these effects for on-chip SRAMs at the 65nm technology node.

1. Introduction

Technology process variations introduced during the fabrication of CMOS wafers such as mismatch in doping levels and/or lack of uniformity of the gate oxide cause transistor electrical characteristics to drift from their nominal values. For nanometer technologies (e.g., below 90nm) this is reflected in a stochastic component of variability that becomes much more prominent than any other systematic effects [1]. Due to the extremely small dimensions of the transistors in nanometer technologies, the same variation in the number of dopant atoms have a much higher impact in the electrical properties of the device than in earlier technology nodes. This makes digital circuits and systems more difficult to predict and control.

The analysis of the impact that this intra-die stochastic component of variability has on the performance and energy consumption of common IP blocks such as ALUs and other arithmetic components is relatively new [2, 3]. For SRAMs the focus has been mostly in functional yield and reliability issues [4], where the sensitivity of the SRAM circuits, especially the SRAM cell stability (e.g. signal to noise margin) and the design rules to compensate that has gained most attention [5, 6]. However, parametric yield which is related to meeting specific performance constraints for the memory library becomes even more critical than pure functional yield. Indeed embedded SRAMs are performance and energy critical components in modern SoCs, contributing to most of the delay and energy consumption of the system. Thus, understanding the performance and energy behavior of embedded SRAMs under the influence of process variation is key to achieving successful IP block and system design.

This paper investigates how the variability in the electrical properties of the transistors (e.g. Vt and β) makes the actual energy and delay per SRAM (read/write) operation to drift from the expected nominal value. Neighborhood transistors in the same die can make a simple circuit, such as an inverter, to become faster or slower or consume more or less energy hence leading to a "variability cloud" in the Energy/Delay (E/D) space with the nominal point centered in it. However, even when such effects are expected to manifest also in a similar way for more complex modules like SRAMs, this is not the case. As we show in this paper, these effects manifest at the SRAM level even more dramatic than at the inverter level, resulting also in variation clouds. However, in the case of SRAMs (and memories in general) all points become in practice slower and consume more energy in average than the memory not affected by variability (hereafter called the "nominal" memory).

Up to our knowledge, this is the first paper to analyze the different (global) interactions originating from intra-die stochastic variability effects inside and between the different blocks of an SRAM in a systematic way. We also quantify the impact that these effects have on both energy and delay at the 65nm technology node. Experimental results obtained from HSPICE simulations show that already for

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)
1530-1591/05 $ 20.00 IEEE

IEEE COMPUTER SOCIETY

small size SRAMs, process variability increases the delay and energy consumption up to a 40% above their nominal values. This values are expected to grow both with the memory size and further scaled technology nodes.

2. Stochastic model for variability

Process variation impacts many technology parameters. In this paper, our focus is on the effects of the front-end devices (e.g., transistors). The effect of variability in the interconnect is neglected here since for SRAMs its impact in the back-end processing is expected to be less dramatic than in the front-end. This is true for the long interconnect lines such as bit-lines, word-lines or decoder busses that are typically long enough so as to average variability effects [7].

Impact of variability at transistor level can be reflected in a drift in the drain current (I_D) from its nominal value. A first order approximation shows that, variations in the drain current are linearly related to variations in the current factor (β) and the threshold voltage (V_t) [8]. The current factor (β) depends on several physical parameters: $\beta = C_{ox}\mu W/L$. A first order analysis shows that the variabilities of these two electrical parameters for MOSFETs are proportional to the device dimensions [7]. Furthermore, they both follow a Gaussian distribution with the following standard deviation: $\sigma_{Vt} = \Delta_{Vt}/\sqrt{WL}$, $\sigma_\beta = \Delta_\beta/\sqrt{WL}$, where W, L are the width and length of the MOSFET device.

At the 65 nm, characterizations of that technology node towards (intra-die) variability impact leads to the following estimated values for $\Delta_{Vt} = 2.5mV\mu m$ and $\Delta_\beta/\beta = 0.7\%$ [9]. Given these estimates, one-sigma variation for the smallest representative transistors of that technology leads to about 10% drift from their nominal value in both V_t or β. These minimum sized transistors are very common in SRAMs in order to achieve maximum density, making these memories very vulnerable to the impact of process variability.

The transistor model used in our analysis is the BSIM4 model of Berkeley Predictive Technology Model [10] and the interconnect parameters are taken from the ITRS roadmap [11]. Both models are intended to reflect technology parameter specifications at the 65nm technology node. As variabilities are uncorrelated between transistors (stochastic effect dominant), every transistor in the netlist is injected with an independent V_t and β values generated according to a Gaussian distribution as described above.

In our experiments, we use HSPICE as transistor-level netlist simulation environment. The modeling of variation in V_t is straightforward in HSPICE since it is a parameter in the model card and an offset to that value can be explicitly specified at the netlist level. However, β can only be modeled in HSPICE via parameters of the device model. The oxide capacitance (C_{ox}) and mobility (μ) cannot be modified since they are correlated with other transistor parameters like gate capacitance. Therefore we have opted to reflect changes in β by using the ratio of W/L given the proportionality of β to that ratio while keeping the gate area ($W \times L$) constant. This ensures that the first order contribution of the extrinsic load driven by the transistor acting as driver does not become influenced by the variation in β.

3. SRAM architecture and model

The SRAM architecture we consider for analysis is the partitioned cell-array architecture [12] typically used in industry with the cell-matrix partitioned into several subarrays. In this architecture, the row decoder is split into two-stages, with a predecoder stage shared by all sub-arrays and a post-decoder locally placed in each sub-array. In our SRAM implementation, we use a set of three-input NAND gates for the predecoder and a set of three-input NOR gates for the postdecoder as illustrated in Figure 1.

The cell-array is made out of bitlines and wordlines with the cells implemented using a classical 6T SRAM cell structure. The memory cells and periphery circuits are dimensioned to perform stable operations at the 65 nm technology node, together with several timing interfaces present between the different blocks of the critical path of the SRAM. A very critical interface is the one devoted to activate the wordline once the decoding process is finished. That interfase is critical since the decoder is the first block in the critical path and variability has a large impact in its delay as shown in Section 5. That delay will be then propagated to the rest of the memory. To properly reflect the impact of this delay and specifically to the cell array, a self-timed style [13] is used when building up such interface. Such circuit style also enables memory to be read or written only when all the necessary blocks are ready so that the real variation in memory E/D can be reflected in the memory generated without affecting its functionality.

Clearly, the time consuming HSPICE simulations of the full SRAM are not directly practical for analysis of the process impact as many simulations are required to gather data with reasonable statistical significance. To speed up simulation time without sacrificing much accuracy, a lumped SRAM model is developed for this analysis. This model is composed out of a full row decoder plus a simplified cell-array where the number of wordlines is just a fraction of the ones present in the full memory and they are equally distributed along the height of the cell-array. The details of this model are left out since they are not the focus of this paper. Still, it is worth to mention that the results from HSPICE simulations indicate that this lumped model achieves more than 97% accuracy in average with a factor of 12 reduction in the simulation time compared to the full SRAM simulation.

IEEE COMPUTER SOCIETY

4. Methodology for capturing worst-case delay and energy under process variability

This section describes a methodology to define the critical path of the memory and specifically that of the decoder under process variability. The critical path of the SRAM starts from the row decoder and ends at the output buffer in the array. The delay of this path under process variations can be obtained in the netlist by connecting the row decoder critical path with the array and finding the delay of the slowest output out of all parallel outputs of the array.

4.1. Row decoder characterization

The task of finding the worst case decoder delay (hence the critical path) under process variability is not trivial as the decoder critical path depends not only on the selected row but also on the transition from the previous address. The reason for this is that the activated transition will change the (dis)charge paths of the parasitic capacitances of the decoder gates [14]. Hence it will also affect the actual measured delay. In fact, for a nominal decoder, a worst case transition exists but that one will not depend on the activated row. Under variability this is clearly not the case. The reason is that many parallel decoding paths exist in the combinational decoder. Hence, the slowest decoding path that determines the decoded row will have a different delay depending on the variability. Finding this critical path for the slowest decoded row implies also finding a worst case delay transition that excites that row.

A first analysis on the number of possible combinations leading to worst case response would require 2^{2N} simulations for a N-input decoder and that for each netlist where variability has been injected. Even for small decoders, this characterization is unfeasible. Therefore a methodology is required to find the critical path with the smallest possible number of transition vectors.

For that purpose, we have developed a heuristic that is based on decoupling the search for worst delay output for a given address from that one for the worst delay transition, irrespective of the output being selected. Based on this, an experimental methodology is developed as follows:

Step1. Find the initial vector that activates a set of worst case delay transitions for the decoder under process variability independently of the decoded row.

Step2. Identify the worst case address vector that excites the slowest decoded row under variability.

Step3. Find the set of worst case delay transitions for the full decoder by combining the initial vector to activate the worst case transitions (Step1) with the worst case decoded row address vector (Step2).

Step1: The goal of this step is to find the initial vector that activates a set of worst case delay transitions for

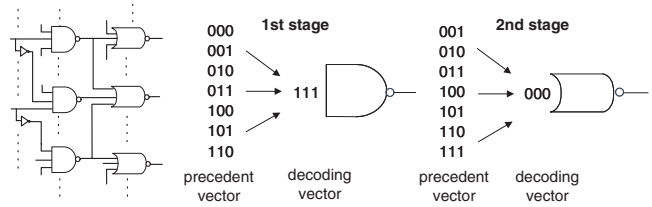

Figure 1. Decoder structure and possible set of worst case transitions for the (pre/post)decoder blocks

the decoder independent of the decoded row. A method has been developed for that by first finding the worst case transition set per building block of the decoder (e.g., address driver, predecoder and postdecoder) and then combining these transitions to find at the inputs of the decoder the worst delay transition for each decoded row. This analysis is possible since the transition only influences the intrinsic capacitance of the current block. The extrinsic load from the next block will not affect the relative ordering of each transition according to their associated delay.

For the two-stage decoder of the SRAM, the building blocks of the first stage is a set of three-input NAND gates and that of the second stage is a set of three-input NOR gates. According to the decoding feature of these gates (e.g., the input combination identifying one different output out of all possible ones in their Karnaugh map), the NAND and NOR gate will have only one decoding vector respectively (e.g., 111 for the NAND gate and 000 for the NOR gate). Hence, the set of transitions potentially leading to worst case delay are, for the NAND gate, the transitions of the type $XXX \rightarrow 111$, while for the NOR gate these are of the type $XXX \rightarrow 000$ (see Figure 1). To find out which of these transitions remain being the worst case under variability, a significant number of simulations (each under a random process variability scenario) has been performed. Figure 2 shows the results after simulations for the NAND gate. It is clearly visible that at the 65nm technology node, only the transition $000 \rightarrow 111$ remains worst case irrespective of the variability scenario for a NAND gate. While the critical transition for the NOR gate, obtained in similar way as that of the NAND, is the $011 \rightarrow 000$ one. Therefore, we can safely conclude that the influence of process variability in the internal (parasitic) loads of the netlist at 65nm is not yet dominant enough so as to change the impact of the ”nominal” worst case transition.

To find the initial vector that exercises a worst case transition for any decoded output, we just have to backtrack the gate-level worst case transitions all the way, from the postdecoder (NOR gate) through the predecoder (NAND gate), to the inputs of the address driver in the full decoder netlist.

Step2: The goal of this step is to identify the worst case address vector that excites the slowest decoded row. For this

IEEE COMPUTER SOCIETY

we could simply excite all decoder outputs one by one and detect the slowest one. However, due to the CPU time inefficiency of HSPICE that would also be expensive as the number of combinations grows as 2^n, being n the number of address decoder inputs. To avoid an expensive search, a technique based on exciting all decoder outputs at the same time and finding the slowest of them by using only one address vector has been developed.

This can be done by modifying the decoder input driver circuitry under test mode such that all input address bits and their complemented bits (see Figure 1) are supplied with the same "0 → 1" transition simultaneously irrespectively on whether a particular input should be complemented. This functionality of the address driver will violate the decoding principle by applying under test mode a 1...1 combination to the predecoder gates, hence activating all the decoder rows simultaneously. By monitoring the difference in delay of all the rows, the slowest one can be easily identified. Based on this, the worst delay vector can be obtained to activate that row under normal decoding operation. The input address transition will not play a role in the determination of this vector since the same transition (0 → 1) is generated in all outputs.

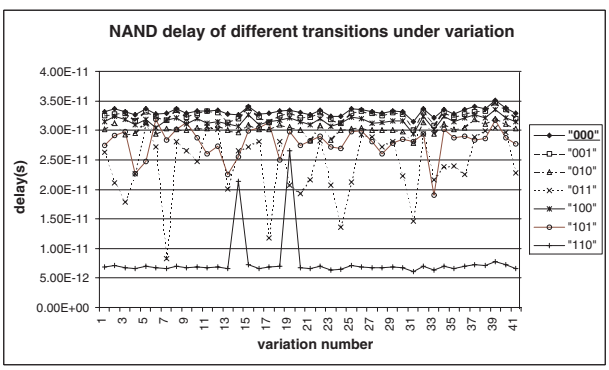

Figure 2. Impact of variability in delay for the set of potential worst case transitions for a NAND gate at the 65nm technology node.

Step3: The goal of this step is to find the set of worst case delay transitions for the full decoder. For this we simply combine the initial vector activating the worst case transitions obtained in Step1 with the worst case decoded row address vector obtained in Step2. This will lead to a set of two-address vectors where first vector is the initial vector exercising the worst case transition irrespectively of the decoded output, followed by the vector selecting the worst case output.

To capture the worst case energy of the decoder, the set of vectors we select is different although still based on the set found for delay characterisation. Since energy is also proportional to the switched capacitances involved in the decoding process, the worst case decoder transition for energy can be also obtained by applying two vectors. In this case, the first vector should be the same as the one found for delay in order to activate the worst-case load conditions of the gate's parasitic capacitances. The second vector is obtained by complementing the the bits of the vector identifying the slowest row. That should lead to an opposite switching direction in the nodes of the netlist hence maximizing the activity of these nodes.

4.2. Full memory characterization

Similar to the decoder, finding the delay of the array under process variability using a full search of all the cells, Sense Amplifiers (SA), and other related periphery is impractical. Hence, there is no better alternative than using a local search where decoder and cell-array interactions are neglected. However, in practice given enough Monte-Carlo simulations are made, the worst case delay range of both decoder and the cell-array when simulating both together is still captured.

For that purpose, the delay of the SRAM is obtained as follows:

Step A: Generate the nominal netlist of the target memory row decoder (see Section 3) and inject stochastic variability on the transistors of that netlist, hence generating a netlist for a given variability scenario. Then characterise the row decoder worst case delay and energy following the methodology described in Section 4.1. As a result the worst-case activated wordline both for energy and delay is also identified.

Step B: Generate the full memory lumped netlist (see Section 3) by connecting the critical rows with one of the wordlines of the lumped cell-array netlist.

Step C: Simulate the netlist generated in the previous step and measure the worst-case energy and delay of the write and read operations. The write operation worst case delay is measured from the moment the input address is given to the moment when the last cell in the accessed bitlines becomes stable. The read operation is performed right after the write operation and its associated worst case delay is measured from the time the address is given to the time when the slowest data bit is read out among the all the parallel bitlines.

To capture the SRAM energy/delay with enough accuracy, 200 simulations have been performed by each time generating a different process variability scenario. Given Monte-Carlo analysis, this should bring a confidence factor above the 99% since that one is inversely proportional to the square root of the number of experiments.

IEEE COMPUTER SOCIETY

5. Analysis of variability results

To evaluate the impact that process variability has at the SRAM level, a 1KB SRAM has been simulated following the methodology defined in Section 4. The circuits of this SRAM and especially the associated timing interfaces have been designed for nominal operation. Hence, no particular safety design margins have been taken when designing these circuits.

The experimental results shown in Figure 3 indicate that the decoder, under process variability tend to have larger delay than the nominal one. The large impact in decoder delay is due to the fact that the slowest path of the affected decoder will be slower than the nominal one for most of the cases. The energy increases also marginally and this is due to the existence of fanout paths initiated at the predecoder stage that reconverge at the input of the postdecoder gates. The delay of these paths are different depending on the actual variability scenario, hence leading to glitching activity at the input of the postdecoder gates. This is wasted energy that leads to a spread in decoder energy.

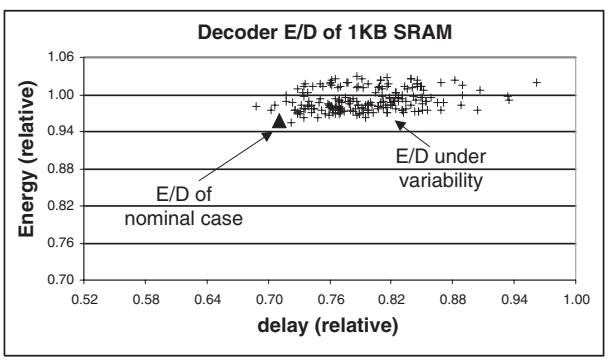

Figure 3. **Decoder energy and delay under process variation**

The delay/energy cloud of the memory for read and write operation are shown in Figure 4 and Figure 5 respectively. These Figures clearly indicate that the delay of both operations tend to become slower in most of the simulations. This increase in delay mostly comes from longer decoder delay but, also from the cell array. This is the case for small size SRAMs, since the decoder contribution in the overall delay is significant [15]. Also, as Figure 3 shows this is significantly affected by variability.

In the read operation, apart from the decoder, the cell array also tends to have larger delay because in most cases the slowest bitline (incl. the SA) among the ones being read will become slower than nominally. The increase in delay is related to a larger time required by the sense amplifiers to amplify the smaller-than-nominal voltage swing on this particular bitline to a full-rail voltage. This smaller swing is due to several effects, even a combination between them. For instance, the bitline discharge current can be reduced due to the difference in time when the pass transistors start activating the selected cell. This time can become later than nominally e.g., due to an increase in the V_t value of these transistors. Also, the inverters in the cell can have less driving capability due to the impact of variability in both V_t and β. Statistically, it is very much likely to have one of the bitlines reacting slower than the nominal case, hence the delay of the read operation is also in most cases larger than nominal.

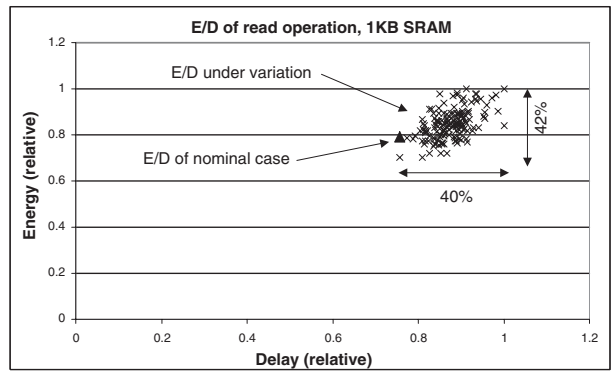

Figure 4. **SRAM energy/delay for read operation under process variation**

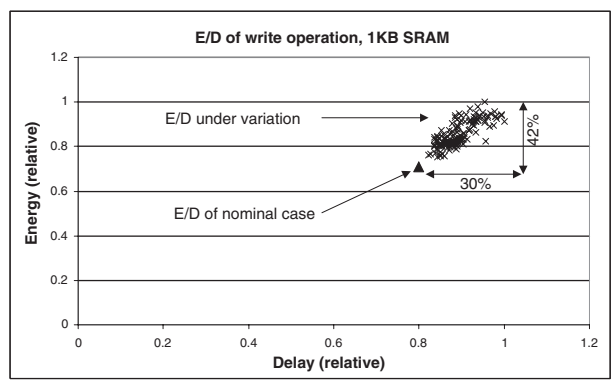

Figure 5. **SRAM energy/delay for write operation under process variation**

The energy variation of the read operation is dominated by the energy consumed on the bitlines, including the active ones (e.g., these being selected by the column muxes), the passive ones (e.g., these not being selected) and the SA. When bitlines have larger swing than nominal, they will also consume also more energy than in the nominal case due to the larger discharge current in the global bitline capacitance and this up to a level where the SA can amplify that swing up to a full rail. When the bitlines have less voltage swing, they will also consume less energy due to the smaller discharge in the associated bitline capacitances. In

this case, even when the SA needs more time, hence more energy to amplify this smaller swing to full rail, that one is not yet big enough (in their global contribution) so as to compensate the decrease in energy obtained in the bitlines due to the smaller discharge current. This balance is illustrated in Table 1 for both a nominal memory and one affected by variability showing less overall energy.

Table 1. Array read energy breakdown under variation where total energy is less than nominal

Relative energy	Active bitlines	Passive bitlines	SA	Total
nominal	0.163	0.6	0.237	1
variation	0.07	0.51	0.26	0.84

In the write operation, delay has the same trend as in the read operation for similar reasons. For instance, one end (e.g. n1 in Figure 6) of the cell being written is directly grounded by the write circuit in a very short time while in the other end (e.g. n2 in Figure 6) of the cell, the bit stored there has to be flipped by one of the inverters in that cell. That inverter may have under variability less driving capability than in the nominal case, which would result in a longer time for the cell to modify its state. Indeed, we would require all bitlines to react faster than the nominal bitline and that gain in delay to become larger than the excess in the decoder delay so as to have an overall memory with a faster reaction time than the nominal memory. That is in fact extremely unlikely to happen as simulation results confirm.

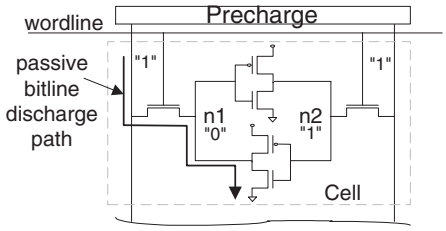

Figure 6. Simplified passive bitline structure in the Write (1) operation

The impact in energy in write operation is also similar to that in the read operation. Apart from the marginal energy increase in the decoder, the array consumes, in most cases, more energy due to the interaction between passive bitline and the active bitlines. Different from the array read delay which is mainly composed of bitline delay and SA delay, the array write delay is mainly determined by the bitline delay (the time the cell being written to become stable). When it becomes larger than nominal, the passive bitlines have more time to discharge through the pass transistors of the cells (see Figure 6). This will lead to a larger than nominal voltage swing on them which consequently increases their energy consumption. On the other hand, the active bitlines can also consume more energy due to larger swing on them and due to short circuit or leakage effects in the cells being activated. When the write operation becomes faster than the nominal case the energy associated becomes also smaller because of a similar mechanism.

6. Conclusion

SRAMs tend to be slower and consume more energy on average than their nominal implementation in most cases under variability. This is partly counter-intuitive so in this paper we have identified and illustrated the different mechanisms behind this unexpected behavior and quantify the impact of these effects for on-chip SRAMs at the 65nm technology node.

Acknowledgments

The authors gratefully acknowledge the fruitful discussion held with colleagues of IMEC's System Level Integration project as source of inspiration for the material presented here. In particular we thank to Antonis Papanikolaou and Jeroen Croon for their feedback.

References

[1] J. Bastos, et.al, Mismatch characterization of submicron MOS transistors *Analog Integrated Circuits and Signal Processing, vol. 12, no. 2, pp. 95-106, 1997*

[2] T.W.Chen, J.Gregg; A low cost individual-well adaptive body bias (IWABB) scheme for leakage power reduction and performance enhancement in the presence of intra-die variations *Prod. DATE 2004, pp240-245*

[3] C.Visweswariah, et.al, First-order Incremental Block-Based Statistical Timing Analysis *Prod. DAC2004,pp331 - 336*

[4] R. Heald; Managing variability in SRAM designs. *ISSCC04 uP Forum*.

[5] D.Burnett, et. al, Implications of fundamental threshold voltage variations for high-density SRAM and logic circuits *Proc. Symp. VLSI Tech, pp,15-16, June 1994*

[6] A.Bhavnagarwala, et.al, The impact of intrinsic device fluctuations on CMOS SRAM cell stability *IEEE Journal. of Solid-State Circuits, Vol.36,pp.658-665, April 2001*

[7] Pelgrom, et.al., Matching properties of MOS transistors. *IEEE Journal. on Solid-State Circuits*, 24(5):1433–1439, 1989.

[8] J.Croon, et.al, An easy-to-use mismatch model for the MOS transistor *IEEE Journal of Solid-State Circuits*, vol.37, pp.1056-1064, 2002.

[9] B.Tavel, et.al, Thin oxynitride solution for digital and mixed-signal 65nm CMOS platform, *IEDM 03 Technical Digest. pp:27.6.1 - 27.6.4*

[10] Y. Cao, et. al., New paradigm of predictive MOSFET and interconnect modeling for early circuit design. *Proc. of CICC*, June.

[11] International technology roadmap for semiconductors, 2004.

[12] B. Prince Semiconductor Memories: A handbook of design, manufacture, and application, second edition *John Wiley and Sons, 1991*

[13] Steve Furber; Asynchronous design for tolerance to timing variability. *ISSCC04 Advanced Solid State Circuits Forum*.

[14] J. Rabaey, et.al, Digital Integrated Circuits: a design perspective 2nd Edition *Pearson Education international, 2003*

[15] B.S. Amrutur, M.A. Horowitz; Speed and power scaling of SRAM's *IEEE Trans. Solid-State Circuits Vol.35, No.2, Feb,2000*