Statistical SRAM Analysis for Yield Enhancement

Paul Zuber, Miguel Miranda, Petr Dobrovolný, Koen van der Zanden, Jong-Hoon Jung‡
Digital Components, IMEC-Belgium
‡Design Technology Team, System LSI Division, Samsung Electronics Co.-Korea

***Abstract*—This paper presents an automated technique to perform SRAM wide statistical analysis in presence of process variability. The technique is implemented in a prototype tool and is demonstrated on several 45 and 32nm industry-grade SRAM vehicles. Selected case studies show how this approach successfully captures non-trivial statistical interactions between the cells and the periphery, which remain uncovered when only using statistical electrical simulations of the critical path or applying a digital corner approach. The presented tool provides the designer with valuable information on what performance metrics to expect, if manufactured. Since this feedback takes place in the design phase, a significant reduction in development time and cost can be achieved.**

I. Introduction

In this paper we present several case studies illustrating an approach for offering full memory statistical analysis capabilities addressing process variability effects and aiming at improving design productivity of embedded SRAM products.

We show how the most likely reasons for statistical failure can be anticipated at design time so as to correct weak design spots before tape-out, hence avoiding costly silicon spin iterations. The technique provides key help to memory and system designers to estimate parametric and functional yield loss due to statistical parametric spreads in threshold voltage and/or current gain of the devices. It is therefore of very high value to the design of embedded SRAM, which are considered to be the most process variation sensitive SoC components.

By placing a constraint either on a system level metric (like cycle time, power) or on stability metrics (as pass/fail checkpoints), we focus on the yield loss component caused by parametric deviations in the devices. Defect related yield is not addressed in this paper. Its treatment requires well known separate analysis techniques [1] and it can be considered orthogonal to parametric and functional yield analysis.

The strength of the approach lies in successfully capturing all (non-trivial) *memory-wide* statistical interactions between the SRAM cell and the periphery, not addressed when using statistical electrical simulations of the critical path alone.

The prototype tool requires three main input items. The first is a transistor level netlist description of a segment of the memory describing all circuitry involved from input to output. The second one is a set of parameters describing the internal architecture of the memory, including redundancy and error correction code infrastructure. The third one is information about the variability of the devices and interconnects used in the underlying technology. This information can be provided in either the form of statistical distributions of certain transistor parameters, scattered data obtained via statistical simulation of the device, or just plain DC current-voltage statistical relationships of fabricated devices obtained from silicon characterisation.

The analysis consists of two main phases. The first phase performs a statistical electrical simulation of the full SRAM critical path netlist. Such critical path includes not only bitline read/write circuits, but also the rest of the main SRAM circuit blocks (hereafter called islands). Examples are the row decoding, timing circuit and output stage buffers (a more formal definition of islands can be found in Section III). During this phase, statistically correlated parametric data and pass/fail information obtained from the critical path simulations is collected via inserted measurement and check point statements in the netlist.

Key in this strategy is the ability to complement the analysis of a nominal memory model under test with statistically sampled variants of the devices. An in-house developed enhanced Monte Carlo technique is used to significantly reduce the number of statistical simulations needed to achieve a particular level of confidence [2]. However any other existing technique for enhanced statistical sampling could be used as well [3], [4]. At this stage, sensitivity information of the impact of process variations resulting from the different SRAM islands in the targeted measurement point is collected.

During the second phase, the collected statistical data and sensitivity information is used to reconstruct the full memory parametric and pass/fail behaviour. This is done by using a mix of statistical and algorithmic data manipulation techniques. This phase is rather fast; ten megabyte memories can be analysed in a few seconds. This way, statistical information on the critical path percolates to the complete SRAM organisation level, resulting in a realistic prediction of the yield as perceived by the memory tester and/or equivalent BIST (built-in-self-testing) technique. In this paper, we focus on use cases and quantitative results for industrial inputs, and only resume the details of the method, left out for a follow-up publication.

II. Related Work

Emerging statistical techniques such as statistical static timing analysis (SSTA) or statistical circuit level simulation tools have been proposed to avoid conventional process corner analysis when handling random variability. Instruments to predict local process variations of circuits comprise Monte Carlo like runs around a nominal simulation for transistor level netlists or adding probabilistic awareness to e.g., timing analysis algorithms at the gate level. This adds a parametric yield dimension to the verification flow before tape-out, allowing the designer to take decisions and make improvements where/when (s)he still

978-3-9810801-6-2/DATE10 © 2010 EDAA

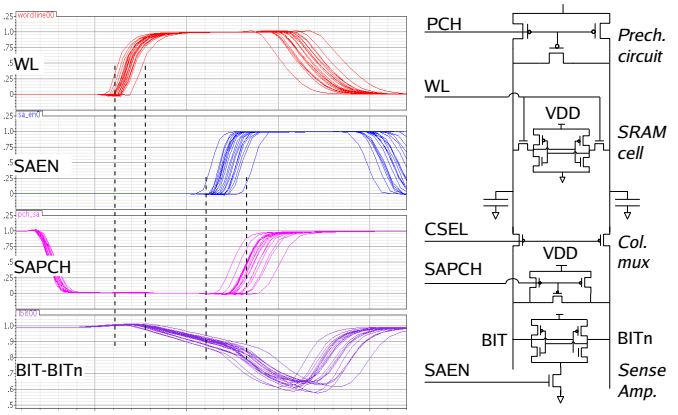

Fig. 1. Read voltage (sense amplifier input at activation time) variability originates from variations in the timing circuit (precharge and sense amplifier enable times), the bitcell drive strength, the wordline and driver, sense amplifier, and column multiplexers.

can. For memories, however, virtually no commercially available solution exists. Several issues render memories especially challenging:

Usually, the nominal simulation of a full memory is reduced to the critical path netlist, assuming all paths behave the same. This approach fails under local process variations where uncorrelated device-to-device variability makes every bitcell access operation to behave differently. Since a memory is as good as its worst path, the memory statistics for instance of the read voltage or access time are the distributions of the worst of all its cells. As a result, simulating the critical path netlist under variability does not model the full memory statistics correctly.

Works considering the bitcell alone without its periphery [5] manage to reduce the sample size and transistor counts effectively, but also entail incomplete analysis. We exemplify this in Figure 1, which shows how the different components influencing the read operation of the cell can affect its read margin. Indeed, variations affect the periphery circuit that times sense amplifier, row decoder, and precharge activation signals, the row decoder that activates a word line driver, and especially the cell's capability to discharge the bitline. As a consequence, we must simulate the entire equivalent circuit's operation under variability to obtain realistic results. Notice that the read margin is the difference between read voltage and sense amplifier input offset. As far as the remainder of the paper is concerned, we neglect the sense amplifier input offset, which did not exceed few mV under variability, and use the read voltage synonymously.

Attention must be paid to instance count and architectural correlations of bitcells and sense amplifiers and other memory parts. A worst case cell instance is not necessarily in the same path with the worst case sense amplifier or the worst case row driver logic so that a blind worst case combination would lead to over-pessimistic results again.

In order to get the bitcell statistics, a pure Monte Carlo based approach is unable to give results at all for large high yield realms. It simply fails to capture the tails of the real distribution for a reasonable amount of simulations. Moreover, the higher the instance count of a certain memory island circuit, the more tail exploration is required. Importance sampling variants to derive tail statistics of a single cell and to predict a memory failure probability have mushroomed [3], [6], [7], [8].

A generic analytical approach to predict the statistics of a system from the given Probability Density Function (PDF) of its sub-system components appeared in [9], and particularly for memories in [10]. However, both methods fail to accurately capture all the interactions between the different memory islands. The two typical phenomenons resulting from these extreme value problems are highly skewed and shifted PDFs. This is why simply using a statistically enhanced (importance sampling) engine on the critical path alone or even a more systematic Design of Experiments [11] approach to gather the mismatch statistics of the critical path netlist would fail as well.

In this work, we combine the considerations above and take them one step further. We report a method and its implementation in a prototype tool hereafter called Memory Variability Aware Modelling - or MemoryVAM in short - based on a technique that predicts the correct memory wide statistics of any parameter that can be measured in a SPICE/SPECTRE test bench, such as access time, power, stability checks such as read voltage, and so on. The method relies on a mix of sensitivity analysis of the different memory islands forming the critical path of the memory to process variations. Such sensitivity analysis is done at a larger granularity than the transistor level as commonly used so far for analog circuits [12], hence leading to a more efficient simulation needed for today's multi MBit memories and time to market requirements. At the same time we preserve the accuracy by taking all circuit interactions into account.

III. Statistical SRAM Analysis

In this section, we describe how the full memory statistics can be recovered by combining island statistics in a specific way. The only required inputs are the memory critical path netlist, process variability information, and a memory architecture description.

A. Sensitivity Analysis

We generate the sensitivity of critical path parameters to process variations in different islands. An island is formally defined as a set of transistors forming a memory component whose instance count (multiplicity) and connectivity differs to other islands. Our motivation is three-fold:

- Only a simulation of the entire critical path netlist allows to take any parameter measurement in the original SPICE testbench, especially those which cross island boundaries or are influenced by several islands. Assembling such parameters from smaller circuit measures (or equivalents) would be far less straight-forward, if not impossible.
- The different islands occur once along the critical path netlist but in a full memory, they occur with different

multiplicities. Sensitivities offer a way to account for real instance counts, as will be shown in Section III-D.
- Considering variability in subsets of all transistors, increases the effectiveness of any importance sampling technique [4], because the dimensionality of the problem decreases.

B. Technology input

In our experiments, we extracted populations of $V_{\rm th}$ and β pairs for different MOSFET types from a foundry provided statistical device compact model library. Comparison of our two-parameter model to the reference using 10,000 Monte Carlo runs shows excellent agreement of mean, spread and correlation of circuit performance metrics.

This is not true for those models assuming (equivalent) $V_{\rm th}$ as only source of random process variations such as random dopant fluctuations, line edge roughness or alike. These can cause some 20% difference in standard deviation of important metrics like gate leakage or delay. As shown for the delay of a gate, our $\Delta V_{\rm th}, \Delta\beta$ based model predicts well also the higher moments, and not only the mean. (see Figure 2).

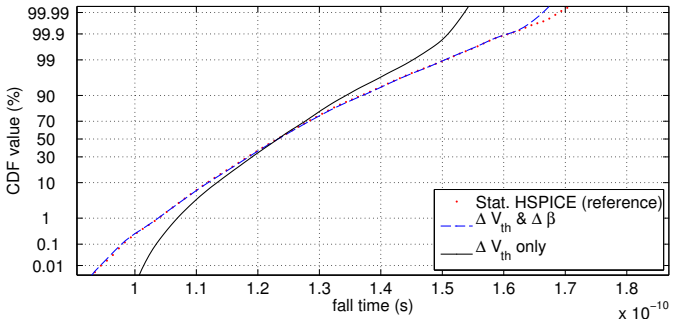

Fig. 2. Two- vs. one- vs. seven-parameter model

C. Variability Injection

Since SPICE itself does not support importance sampling, we introduce the concept of circuit Variability Injection. Variability Injection denotes the process of transforming a SPICE level netlist with any number and type of transistors into N variants of the same. These variants differ to the original netlist in such way, that variability injection sources are added to the MOSFET instances of an island. A voltage generator at the gate and a drain-current controlled current generator along source-drain are used to model $\Delta V_{\rm th}$ and $\Delta\beta$, respectively. The values of these injection sources differ among the variants and obey the underlying distribution. W and L are processed to comply with Pelgrom's rule [13].

D. Architecture Aware Scaling

After simulating the critical path sensitivity statistics for all islands, the final important stretch is deriving the memory statistics from the sensitivities. Since this happens under awareness of the connectivity of the islands, we refer to this step as Architecture Aware Scaling (AAS). It consists of the following consecutive steps.

- For every island $i = 1..I$, pick an M_i-sized sample from the sensitivity distribution ΔP_i of the critical path due to island i. The parameter P can be multi-valued and can contain any measure that was taken in the original SPICE testbench. ΔP is the variation of P shifted around nominal.
- Systematically list all M possible paths through a memory with all its island instances along them. There will be paths sharing sense amplifiers, others sharing row logic. All M bitcells are in unique paths, and the timing generator is shared by all paths. Any path instance in a real memory can thus be described by a particular combination of island indices: $p_j = (z_1, \ldots z_I), j = 1 \ldots M, z_i \in \{1 \ldots M_i\}$. This connects the ΔP_i statistics according to the memory topology, as described in the following.
- Re-assemble path parameters from sensitivities of islands that form the path. For every path, $P_j \approx P_0 + \Sigma_i \Delta P_i(z_i)$ where P_0 is the nominal point. This rule provided good accuracy ($\leq$1 % RMS error) on all examples used in this work. Optionally, a position dependent component $\Delta P_{\rm pos}(x, y)$ can be added to reflect systematic and random variations within a correlation length.
- Select the worst of the M paths and assign that value to the memory observation. Depending on the meaning of P this can be the max operator (e.g. late mode timing), the min operator (e.g. read voltage, early mode timing) or the average for power.
- Repeat the above steps n times with different random sequences to generate memory statistics.

IV. Results and Applications

MemoryVAM connects the techniques above in a tool implemented in MATLAB on top of SPICE. It can be configured with any number of islands and hierarchy depth, such as banks, local word lines, multiplexed output buffers and the like.

A. Comparison to other analysis methods

One attempt to derive the worst case behaviour of a memory might be a corner simulation of the critical path netlist. Another attempt might be to simulate the critical path netlist statistically. We show such results in Figure 3 for an industry provided SRAM block and compare it to MemoryVAM.

To stress the effect caused by local random variations we have separated the graph into local and global variations. It can be seen that even though the global variations (still) cause higher uncertainties, it is the local random variability that causes an additional shift of the cloud away from the nominal point.

It is interesting to note that the global variations of access time and read voltage are negatively correlated, while the local random variations of the memory are positively correlated. This effect is caused by a combination of maximum and minimum operators, on the two respective parameters.

Such observations cannot be made by just statistically simulating the critical path netlist, even if the approach is to simulate local variations around a global corner. Combined

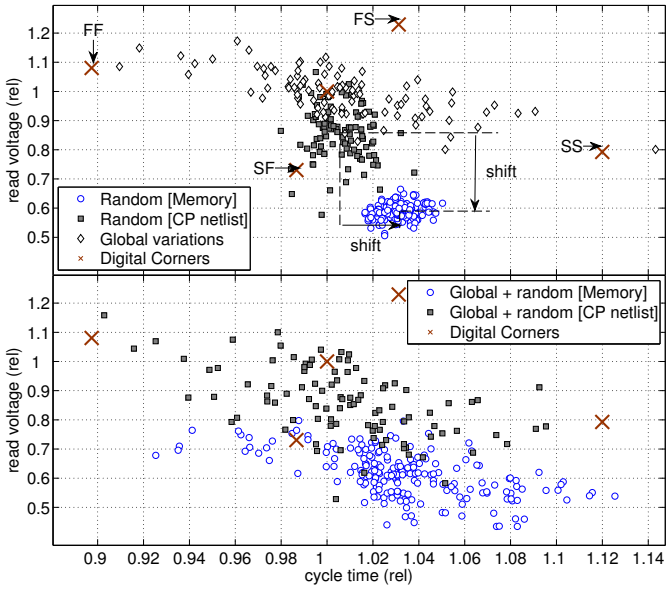

Fig. 3. Every single dot in the scatter plot represents one observation of a critical path netlist or of a full memory.

(global+local) corners would need on top of the global component also an equivalent local random margin to capture the architecture-dependent shift. Traditional digital corners are clearly inadequate for margining memory distributions at predefined percentiles.

B. Sensitivity Analysis

In Section III-A we proposed to analyse the sensitivity of critical path metrics to process variations in particular islands (blocks of transistors) for use as an input to architecture aware scaling, i.e. as intermediate results. As it turns out such intermediate results are per se already providing valuable information to the designer. Figure 4 shows such results for a 256 KBit memory block in 45nm. We selected the timing circuitry, the sense amplifier block and the bitcell islands, and analysed the effect of variations therein to the access time and read voltage. The user can easily see for example that for improving speed the best places to optimise are the periphery circuitry and the IO-blocks (sense amplifiers, precharge, R/W-logic, etc.), while

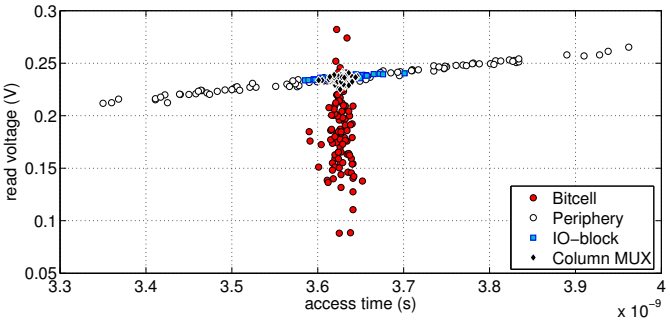

Fig. 4. Sensitivity analysis results for cycle time and read voltage due to variations in memory islands.

bitcells have little influence. The opposite is true for the read voltage. Similar graphs can be plotted for any other measure as well.

C. Stability and Pass/Fail checks

Apart from read margin, a series of other stability check metrics can be taken in a SPICE testbench and thus propagated through MemoryVAM. Read access examples are timing checks between precharge, senseamp enable, sensamp output, wordline, etc. or the dynamic noise margin in the cell. Write stability checks may include bitline write levels at write time, write margins and the like. We demonstrate the capability of our method on the latter two metrics.

We define here the write margin as the time between the cell has flipped and the earlier of the wordline or precharge closing times. The stability criterion demands that this time should be positive. At the same time the bitline write voltage must be small enough to be able to safely flip the cell. Figure 5 shows the results. As expected, both metrics shift to more critical values, though not dramatic in this case. This is a good example where the designer can reduce internal margins for higher speed and lower power while still staying in safe areas. Without loss of generality, we continue our work with the previously defined read voltage.

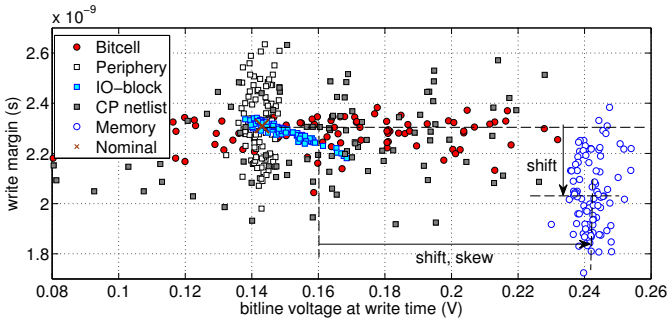

Fig. 5. Sensitivity analysis for two write metrics, and comparison between critical path netlist and shifted memory statistics.

D. Margin Control Knob

As shown in Figure 1, there is a strong influence of the sense amplifier activation time to the read voltage. It has become common in advanced memory design to implement test-time knobs to trade memory speed for robustness in terms of read voltage. This is done with a programmable delay line in the sense amplifier activation signal, as shown schematically in Figure 6. Note that the depicted circuitry is simplified as also other transitions such as the wordline closing slope are delayed. Of course, this artificial delay is subject to variability itself and must be accounted for and MemoryVAM can do so.

We have run MemoryVAM on one of our industry grade memories with and without activated margin control knob (MCK). The results can be seen in Figure 7. As expected, extra delay improves the read voltage. With the aggressive setting, the margin decreases. One can observe another less intuitive effect: while the nominal point of the read voltage

improves from approximately 240 to 310 mV, the median of the memory's read voltage improves from about 80 to only 105 mV. Apparently, the knob is less effective for increasing small read margins than for increasing high read margins. MemoryVAM enables an estimation of the efficiency of the knob on memory level (70 mV vs. 25 mV) before tape-out and thus to guide the knob design. Though less than predicted by the nominal SPICE simulation, there is still a noticeable effect of about 25 mV of this knob in the memory. Of course, this comes at the cost of extra delay, one nanosecond in this case.

If the IC is an aggressive design one wants to go for as high speed as possible. Only with a correct yield distribution as provided by our tool the designer knows for sure what minimum delay he can offer in the specification sheet assuming a given yield budget. The other way round, it is clear that if timing allows, the margin control knob (MCK) should be programmed to generate as much read margin as possible in order to minimise the chance of read voltage related failures during operation.

Figure 8 shows the memory statistics for a margin control knob with four positions (00 to 11) and a hard delay constraint of 5.5 ns cycle time. Memories that violate this constraint must be accelerated using a more aggressive setting. This is true for many observations under the safe (in terms of read voltage) setting MCK=11, some for the more aggressive setting MCK=10, and all remaining memories pass under MCK=01. The set of yielding memories in reality will exhibit a mix of differently programmed MCK signals. This set shows a median read voltage of about 100 mV (80 mV worst case).

A brute force statistical SPICE not including the right multiplicities and architecture correlations, or simply a worst case corner based approach, would have suggested MCK=01 for all memory observation, causing read voltages of about 90 mV down to 60 mV in the worst case. Assuming the designer would not accept such low read voltages in the fear of instable operation, she might have decided to–unnecessarily–further optimise timing at the cost of area and power. We demonstrate this effect in Figure 9 with the help of Cumulative Density Function (CDF) views of the cycle times for each knob setting and both methods. MemoryVAM predicts that about 30% of the memories will pass in the slowest setting. Less than 15% have to run in the second aggressive mode.

E. Power Analysis

For power one often finds the critical path netlist to already provide an 'internal' scaling from component to full memory level. This is usually done by monitoring the current within the components, multiplying this value by the instance count and summing up over the component types. Again the results are incorrect as soon as the assumption that all instances of an entity consume the same amount of power does not hold due to local random variations.

The sensitivity analysis of our method automatically produces the influences ΔP_i of the different islands i to the total memory power consumption. Different instance variants, for example of the IO-block, will cause memory power prediction values as if all IO-blocks were the same. In order to obtain the correct power consumption of a particular memory observation, we need to average all individual IO-block contri-

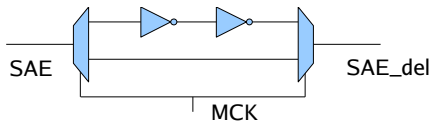

Fig. 6. Memory designers use programmable knobs (MCK) to advance or retard the sense amplifier activation time to trade timing and read voltage.

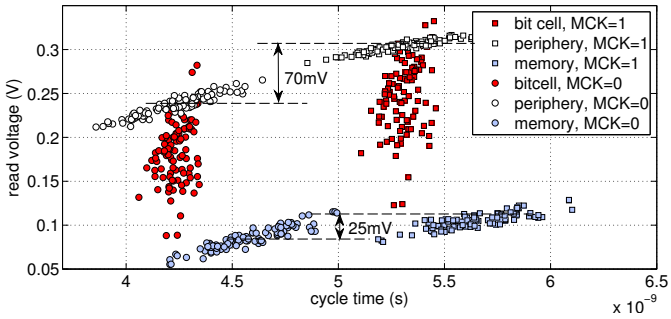

Fig. 7. Setting the margin control knob MCK to one causes later sense amplifier activation time and thus increased read voltage. This is reflected in a shift of the memory building block statistics and the full memory statistics.

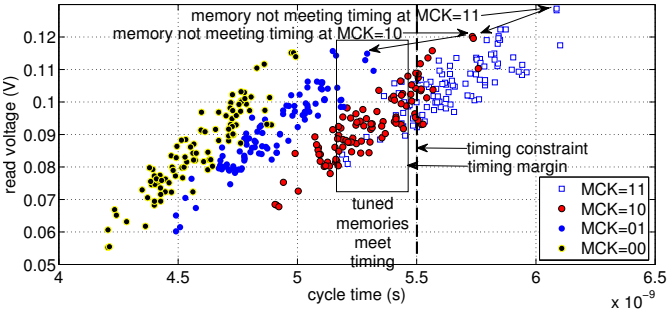

Fig. 8. Memories to the right of a constraint line do not meet timing. For those observations, MCK has to be decreased from 11 to 10 or even 01. The passing memories have still good average and worst case read margin.

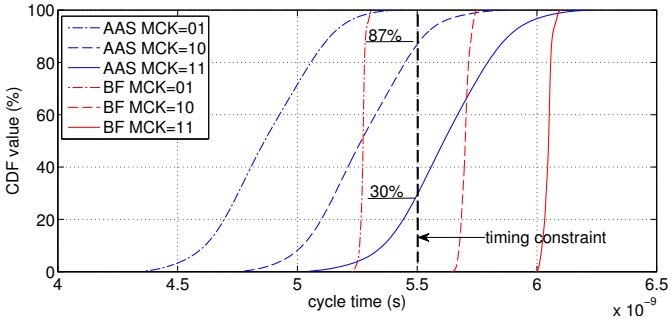

Fig. 9. CDF plots of the cycle time for different margin control knob settings computed by MemoryVAM and a pessimistic worst case method, which is little better than a corner approach. The latter falsely leads to the conclusion that the only safe setting for correct timing is so aggressive that the read voltages may get too low.

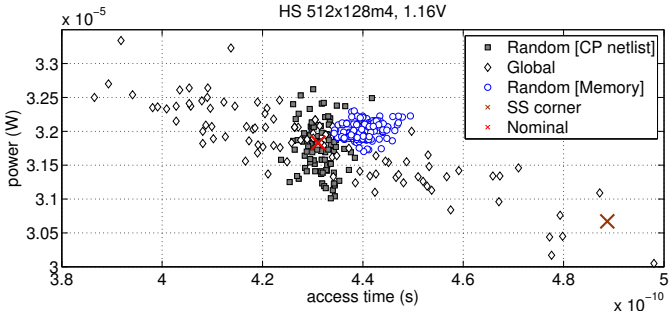

Fig. 10. The power consumption of a memory is predicted with the average operator over all path-based power values.

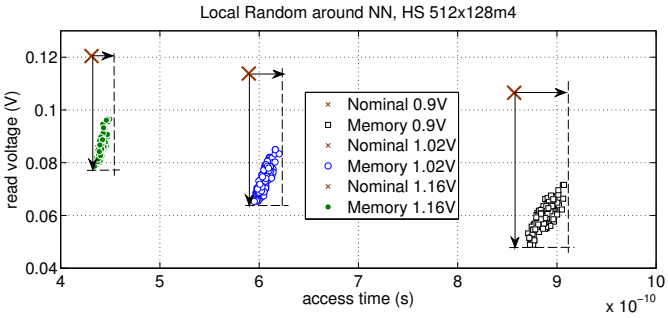

Fig. 11. Margins increase as voltage decreases.

butions. Figure 10 shows the total power of a write-read cycle for memory critical path netlist and of the full memory computed this way. As expected, the local random variability is decreased due to the averaging effect of the many instantiations of the components, mainly the bitcells. It may be worthwhile to note the preserved correlation between any two metrics from the testbench, which may be interesting for compound yield predictions, such as power and timing.

F. Critical Supply Voltage

Global Voltage Scaling (GVS) is a useful technique for a dynamic reduction of the memory voltage for minimum power under a timing constraint. A critical path replica can be used to report near-failure warnings of all paths to form a closed loop with the voltage regulator.

For GVS one is initially interested in predicting the global variability. It is not possible to adjust for local variability, which therefore must be respected as a margin. The amount of margin is essential. If this is too small, the risk of timing failures increases. If it is too high, the efforts of global voltages scaling may not pay off. Again, the nominal shift of timing due to the many parallel paths existing in the memory must be considered. Also the increased timing spread at lower voltages, as well as minimum tolerable read margin require a carefully selected lower bound for V_{DD}.

All such questions can be addressed with MemoryVAM. We have analysed the memory of Figure 10 with different voltages in order to find out the required margins for local random effects. Figure 11 shows the 99-percentiles of these margins, which increase with decreasing voltage.

G. Runtime considerations

Obviously simulating N variants of netlist variants for I (few) islands, costs a runtime overhead to nominal simulation. However, compared to a multiple corner approach this becomes relative. Above that, simulation of variants is predestined for compute farms. We required about 500 CPU minutes for deriving the sensitivities of 100 variants of the three islands in a SRAM using a single Pentium based desktop, but could proceed within an hour by using twelve servers in parallel. AAS run times are in the order of seconds to minutes.

V. Conclusions

In this paper we defined an automated approach to enable statistical timing, power and yield analysis for full SRAM arrays. Case studies based on industry-grade embedded SRAM in 45nm and 32nm technology nodes are used to illustrate the approach. We have shown quantitatively how our method can be used to avoid costly design iterations.

References

[1] R. Ott *et al.*, in *ASMC/SEMI: Proc. Advanced Semiconductor Manufacturing Conf.* ACM, 1999, pp. 87–91.

[2] P. Zuber, V. Matvejev, P. Roussel, P. Dobrovolný, and M. Miranda, "Using exponent monte carlo for quick statistical circuit simulation," in *Proc. of the Intl. Workshp. on Power And Timing Modeling, Optimization and Simulation*. Springer, 2009.

[3] R. Kanj, R. Joshi, and S. Nassif, "Mixture importance sampling and its application to the analysis of sram designs in the presence of rare failure events," in *DAC '06: Proceedings of the 43rd annual Design Automation Conference*. ACM, 2006, pp. 69–72.

[4] D. Hocevar, M. Lightner, and T. Trick, "A study of variance reduction techniques for estimating circuit yields," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 2, no. 3, pp. 180–192, July 1983.

[5] Y. Zhou, R. Kanj, K. Agarwal, Z. Li, R. Joshi, S. Nassif, and W. Shi, "The impact of beol lithography effects on the sram cell performance and yield," in *ISQED09: Proc. of the 2009 International Symposium on Quality Electronic Design*. ACM, 2009, pp. 607–612.

[6] H. Nho, S.-S. Yoon, S. Wong, and S.-O. Jung, "Numerical estimation of yield in sub-100-nm sram design using monte carlo simulation," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 55, no. 9, pp. 907–911, Sept. 2008.

[7] G. Chen *et al.*, "Yield-driven near-threshold sram design," in *Proc. Intl. Conf. Computer-Aided Design*, 2007, pp. 660–666.

[8] L. Dolecek, M. Qazi, D. Shah, and A. Chandrakasan, "Breaking the simulation barrier: Sram evaluation through norm minimization," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, Nov. 2008, pp. 322–329.

[9] M. Miranda *et al.*, "Variability aware modeling of socs: from device variations to manufactured system yield," in *Proc. Intl. Symp. Quality Electronic Design*. ACM, 2009, pp. 547–553.

[10] R. Aitken and S. Idgunji, "Worst-case design and margin for embedded sram," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, April 2007, pp. 1–6.

[11] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[12] T. McConaghy and G. Gielen, "Globally reliable variation-aware sizing of analog integrated circuits via response surfaces and structural homotopy," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 11, pp. 1627–1640, November 2009.

[13] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching properties of mos transistors," *Solid-State Circuits, IEEE Journal of*, vol. 24, no. 5, pp. 1433–1439, Oct 1989.