Virtual Prototyper (ViPro): An Early Design Space Exploration and Optimization Tool for SRAM Designers

Satyanand Nalam[1], Mudit Bhargava[2], Ken Mai[2], and Benton H. Calhoun[1]
University of Virginia[1], Carnegie Mellon University[2]
svn2u@virginia.edu

ABSTRACT
SRAM design in scaled technologies requires knowledge of phenomena at the process, circuit, and architecture level. Decisions made at various levels of the design hierarchy affect the global figures of merit (FoMs) of an SRAM, such as, performance, power, area, and yield. However, the lack of a quick mechanism to understand the impact of changes at various levels of the hierarchy on global FoMs makes an accurate assessment of SRAM design innovations difficult. Thus, we introduce Virtual Prototyper (ViPro), a tool that helps SRAM designers explore the large design space by rapidly generating optimized virtual prototypes of complete SRAM macros. It does so by allowing designers to describe the SRAM components with varying levels of detail and by incorporating them into a hierarchical model that captures circuit and architectural features of the SRAM to optimize a complete prototype. It generates base-case prototypes that provide starting points for design space exploration, and assesses the impact of process, circuit, and architectural changes on the overall SRAM macro design.

Categories and Subject Descriptors
B.3.1 [**Memory Structures**]: Semiconductor Memories – *Static memory (SRAM)*

General Terms
Design

Keywords
SRAM, Virtual prototype, optimization, design space exploration, iterative design

1. INTRODUCTION
While process scaling has enabled ever-larger embedded memories, scaling trends such as process variability, device leakage, soft error susceptibility, and interconnect delay make memory design increasingly difficult. In the face of such scaling effects, the best way to design SRAMs that are optimal in terms of global figures of merit (FoMs), which we define as energy, performance, area, and yield, at the 32nm process technology node and below, largely remains an open question. Researchers have proposed a number of techniques at the technology and circuit level to deal with problems such as variation and leakage [1][2][3][4], but they tend to address only certain individual components of the memory. A change in any one of the key memory circuits will alter the optimal circuit topologies, array partitioning, and architecture for the entire memory. This makes it difficult to evaluate a particular circuit technique without assessing the global benefits and overheads.

Though a back-of-the-envelope estimation of overheads and impact on SRAM global FoMs early in the design is possible, it largely depends on the assumptions made about the SRAM architecture and component circuits, some of which may not yet be designed. These assumptions can vary from designer to designer and lead to vastly different estimates about the impact of a particular circuit technique on SRAM global FoMs. Alternatively, the designer could create complete SRAM prototypes to evaluate each new technique, which impractically increases design time and reduces productivity. This makes it difficult for designers to make an accurate comparison of available design options and to choose one that results in an optimal design, especially in deep nanoscale processes. Thus, there is a need for a methodology through which designers can generate and evaluate prototypes at every step of the SRAM design process that account for process and circuit level issues in terms of global FoMs.

To address this problem, we present a Virtual Prototyping tool (ViPro), which enables early design space exploration by creating virtual prototypes of a complete SRAM macro, even when many design details are missing (hence "virtual"). As the design process proceeds, the prototypes become more accurate and complete. Thus, ViPro helps the designer do what he would want to do anyway (e.g. design space exploration), but much more efficiently, making it *design automation* in the truest sense. ViPro has four key features that make it a valuable tool for SRAM designers. First, it generates a base-case SRAM prototype - a general purpose, robust SRAM with reasonable performance. Second, since the model allows the components to be described using varying levels of detail, designers can define and work on a complete prototype quite early in the design cycle. Third, it can quickly re-optimize the design if a circuit component or the process models change. Finally, it performs its own process characterization, and thus can be used with any process with defined SPICE/Spectre device models.

The novelty of the tool is that it generates a virtual prototype of the SRAM with as much information as is available at every stage of the design process, and gives the best possible estimate of the global FoMs and trade-offs between them. It is not a stand-alone tool or a compiler that designs an SRAM. Rather, the designer is actively involved, and uses his expertise to make

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC'10, June 13-18, 2010, Anaheim, California, USA
Copyright 2010 ACM 978-1-4503-0002-5 /10/06…$10.00.

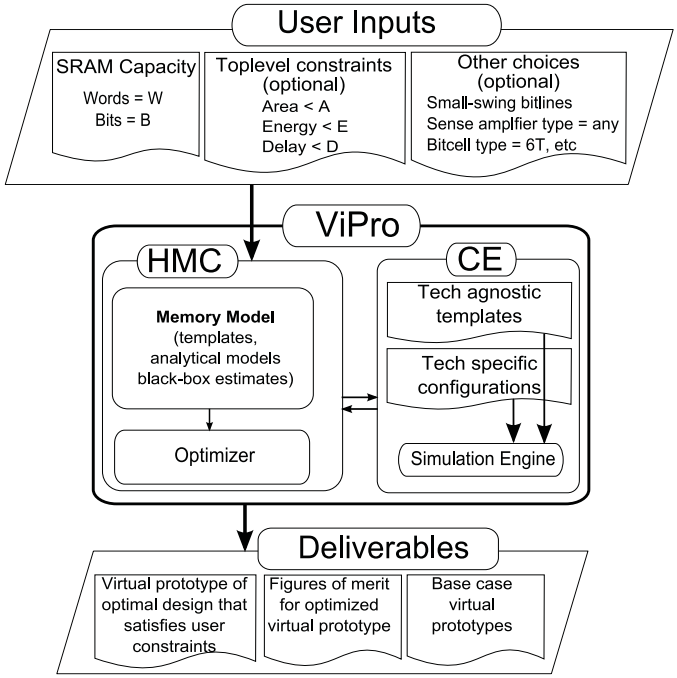

Fig 1: Top-level structure of Virtual Prototyper (ViPro). The characterization engine (CE) and the hierarchical meta-compiler (HMC) implement the two main features of ViPro - technology-agnosticism and a flexible hierarchical memory model.

design decisions based on the output of ViPro, which ultimately leads to an optimal design after several iterations.

2. RELATED WORK

There are a few memory design and FoM characterization tools available, but they do not support integrated process-circuit-system co-design like ViPro. At one end of the spectrum are architecture-level modeling tools like CACTI [5], used by computer architects to obtain quick estimates of SRAM access time, power, and area. CACTI 6.0 [6] facilitates high level design space exploration by using an optimization cost function that accounts for a user-weighted combination of delay, leakage, dynamic power, cycle time and area. Our tool also supports architectural exploration, but it differs from CACTI in two key ways. First, CACTI makes fixed assumptions regarding the circuits comprising the SRAM, so it optimizes at the architecture level only. ViPro allows designers to generate circuit information (via simulation) specific to any given technology or to add/alter the underlying circuits. Thus, it supports circuit-architecture co-design, which leads to better overall designs. Second, CACTI supports a limited set of process technologies and assumes ITRS [7] parameters for all its calculations. These assumptions may not be accurate, especially for advanced process nodes. ViPro uses a technology-agnostic simulation environment [8] to characterize its circuit components in any process using SPICE before generating the virtual prototypes, so it uses accurate technology-specific circuit parameters for any process.

At the other end of the spectrum are transistor-level optimizers (e.g., [9][10]) that are good at choosing optimal device characteristics (e.g. W/L, V_T, V_{DD} etc.) for a given circuit topology, but are not helpful in choosing an optimal circuit topology or micro-architecture. In addition, since they are designed to thoroughly explore the design space under device and environmental variations, they are not suitable for quick, early design space exploration.

Finally, memory compilers (e.g., [11][12]) help generate memories based on user-defined parameters, but do not provide trade-off information or facilitate design space exploration and optimization. They are more a deliverable *from* memory design teams rather than a tool *for* memory design teams. ViPro fills the gap between these tools by providing an optimization and design space exploration tool for SRAM that supports circuit-system co-design.

An analogous design tool for the design and optimization of high speed links was previously presented in [13], which couples circuit level parameters with global FoMs and demonstrates the value of doing so for exploring a broad design space. The tool relies heavily on analytical expressions that are specific to high speed links. ViPro targets another complicated mixed signal design problem, SRAM, with a more generalized approach. While it can support analytical modeling for speeding up the optimization process, it also supports a fully simulation based approach or a mixed modeling/simulation framework (e.g. by supporting variable levels of detail in describing the SRAM circuit components).

The rest of the paper is organized as follows. In section 3, we discuss the implementation of the tool in detail and describe the SRAM design methodology using it. In section 4, we present an example usage and demonstrate how it helps understand the impact of process, circuit, and architecture level decisions on SRAM FoMs. Finally, we conclude in section 5.

3. OVERVIEW OF VIPRO

Fig 1 shows the structure of ViPro, which comprises two main blocks: a characterization engine (CE), and a hierarchical meta-compiler (HMC, 'meta' to distinguish it from a true compiler that produces complete final designs). The HMC implements an editable and flexible hierarchical model of the memory that allows a designer to define components of the memory with varying levels of detail and accuracy. The CE provides a technology-agnostic framework for generating data from simulation of those components, so that ViPro can operate in any process technology. Since the CE produces the building block components for the HMC, we begin by describing the CE.

3.1 CHARACTERIZATION ENGINE (CE)

In [8], the authors propose a technology-agnostic simulation environment (TASE) that abstracts out the process dependencies from the simulation set-up using simulation templates. It then combines these templates with process-related data to produce the simulation-ready netlist and simulates it using Spectre. We incorporate TASE into the CE for two purposes. First, we use the CE to characterize a technology or process through device-level simulations (e.g. I-V curves, FO4 delay, etc.). Second, the CE includes a user-expandable library of templates of SRAM components that can be characterized in terms of global (e.g. energy and delay) and component-specific FoMs (e.g. noise margins for a bitcell and offset for a sense amplifier). These

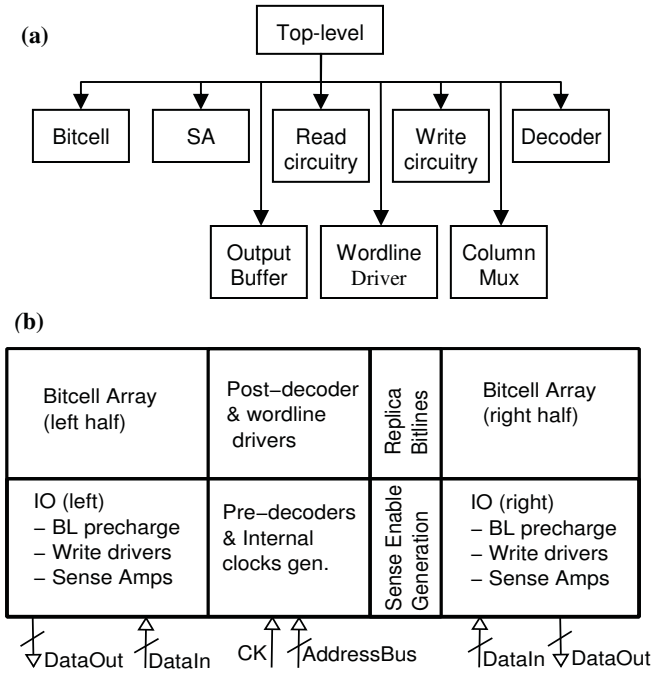

Fig 2: Example of (a) Memory hierarchy and (b) single-macro architecture.

templates also use Monte Carlo analysis to capture statistical data required for yield estimation. As new circuits are added to the template library over time, the CE incorporates a larger range of options that the HMC can then use for optimization. The technology-agnostic nature of TASE allows the CE to work with any technology that has a SPICE model file, which in turn makes ViPro usable for any technology.

3.2 HIERARCHICAL META-COMPILER

The HMC has two functions within ViPro. First, it contains and manages a hierarchical model of the memory (HMM) that acts as a prototype of the system. The HMM is implemented using object-oriented programming in MATLAB. Fig 2 depicts an example memory hierarchy and architecture. For ease of demonstration, we show only two levels of hierarchy that support low-capacity, single macro memories.

The second function of the HMC is to calculate the global FoMs for each component and to determine an optimal SRAM configuration subject to constraints on the FoMs. The HMC calculates the total energy by simply adding the energy of the component circuits and the delay by adding the delays of the components on the critical path. For the demonstration in this paper, a brute force optimization method is used with the array configuration (number of rows and columns per block) being the variables and either energy or delay being a constraint or optimization objective. In general, the number of variables and constraints can be larger, with the resulting higher complexity optimization problem handled by more powerful convex optimization tools such as MOSEK [14]. In particular, the SRAM optimization problem will only be truly complete if area and yield are also considered along with energy and delay as the FoMs. However, for the purpose of demonstrating the SRAM

(a)

```
classdef SRAM_Component
  properties
        W; % transistor min. width
        L; % transistor min. length
        VDD; % nominal operating voltage
        NR; % Number of rows per block
        ...
  end % end properties

  methods
        function obj = E()
        % use black-box estimates or CE
characterized blocks for energy
        end
        ...
  end % end methods
```

(b)

```
% SA is a sub-class of the parent class
SRAM_Component
classdef SRAM_SA < SRAM_Component
  properties
        offset;
        % NR, by inheritance
        <other attributes local to SA>
  end % end properties

  methods
        function obj = E()
        % overloaded function to calculate
energy for SA component
        end
        ...
  end % end methods
```

Fig 3: (a) SRAM parent class and (b) example component class showing circuit attributes and FoM estimation.

design methodology using virtual prototyping, considering a sub-set of the metric suffices.

Each component in the SRAM hierarchy is defined as a class, and each class has properties defined for parameters (device sizes, voltages, etc.) and FoMs associated with the component. The methods (functions) in the class that determine the values for an instance's FoMs support varying levels of detail. For example, the method that computes delay can simply assign it a constant value, equivalent to treating the component as a black box with estimated constant behavior. This may be the best option for a new circuit block about which little details are known. The method can alternatively use an analytical expression or macro-model to compute the FoMs from the component's parameters. For instance, the energy of a wordline (WL) driver is estimated using CV^2 calculations, with the value of C determined by the bitcell pass-gate and WL wire capacitance. The delay of a component is calculated using logical effort or using an equivalent simplified R-C circuit. Finally, the components can be specified by transistor level netlists from which the CE can directly obtain FoM data across different values of the input parameters, leading to complete tradeoff curves.

A key feature of the HMM is the use of class inheritance that allows properties common to a branch of the hierarchy to filter

down to the leaf node of that branch. Fig 3 shows the parent class and the SA class that is inherited from the parent class. By defining common parameters whose optimum value is affected by multiple blocks as properties of the parent class, the HMM can capture interactions between different blocks when optimizing the overall design. For example, the offset of the sense amplifier (SA) and the strength of the bitcell's access transistor both influence the optimum number of bitcells on a single bitline (i.e. number of rows (NR) in the memory array). Thus, the HMM defines a property "NR" in the parent SRAM class that filters down to both the bitcell and the SA classes (see Fig 3). The local component methods capture how "NR" depends on both the offset in the SA block and the drive strength of the bitcell block. These local definitions allow top-level optimization to coordinate local dependencies. For instance, the SA can adjust its offset (e.g. by changing its device sizes) or the bitcell can adjust its drive strength (e.g. by strengthening its access transistor).

3.3 TOOL FLOW

This section describes the interaction of a user with ViPro. The designer provides the following inputs in a configuration file similar to CACTI.

- Process technology
- Top-level memory specifications – Capacity, word-size, supply voltage, operating temperature, etc.
- Constraints on metrics like energy, delay and area (optional – may be local or global)
- Component specifications (optional) – black-box estimates, analytical models or CE simulation templates

Fig 4 shows the steps involved in using ViPro for generating virtual prototypes.

The first step (once per technology) is to characterize the process technology through device-level simulations. In the second step, the CE characterizes components from the library (once per technology). Any components unavailable from the CE must be defined using black-box estimates, analytical models, or new templates added to the CE library. As more components are added to the library, the accuracy and scope of the virtual prototypes improves. In the third step, using existing components and built-in analytical models, ViPro can generate an optimized base-case prototype that provides a convenient starting point for a designer interested in creating a more optimized custom design. He can explore different circuit options (e.g. assist techniques, alternative bitcells, etc.) to further optimize his design. By changing the specification of one or more components and running the tool iteratively, the designer can steer the design effort towards an optimal design. Alternatively, the designer can exploit the tool's technology-agnostic nature to compare prototypes for different process or device options. This kind of comparison is especially useful for many "fab-less" companies that have to choose between different processes for their design. Thus, for example, when porting an existing design to a new technology, the designer can quickly see how the optimum configuration of the design changes. This technology agnostic feature also allows for process-circuit co-design, since the optimal circuit and architecture selections will change in response to process alterations.

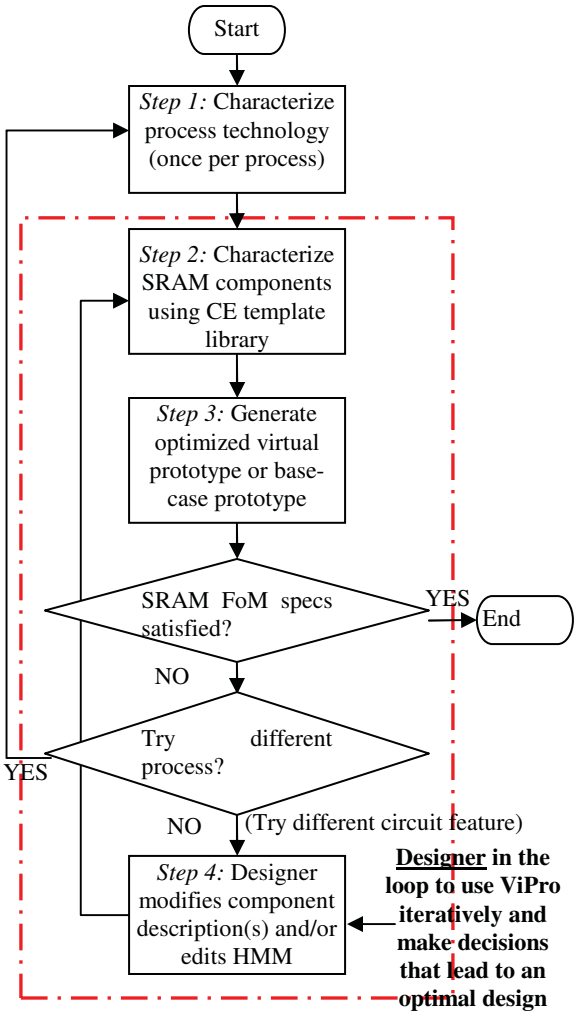

Fig 4: Methodology of using virtual prototypes for SRAM design

A key insight here is that the designer is an integral part of the tool flow. As the designer runs the tool multiple times and compares several virtual prototypes, his understanding of the trade-offs involved in the design increases. Thus, he finalizes more and more components, which improves the accuracy of the prototype. Ultimately, as the design nears completion, all the components in the memory are specified in terms of circuit netlists from the template library of the CE, and our tool becomes closer to an actual SRAM compiler.

4. USING VIPRO FOR SRAM DESIGN

We now demonstrate how ViPro can be used for SRAM design. First, we generate a base-case prototype and show how ViPro helps track changes in process models. We then use it to re-optimize the design after making changes to the underlying circuits and architecture. Finally, we show how it helps in porting designs to other technologies.

4.1 BASE-CASE GENERATION

We begin the SRAM design in a 45 nm bulk CMOS process by using ViPro to generate an optimal base-case prototype using

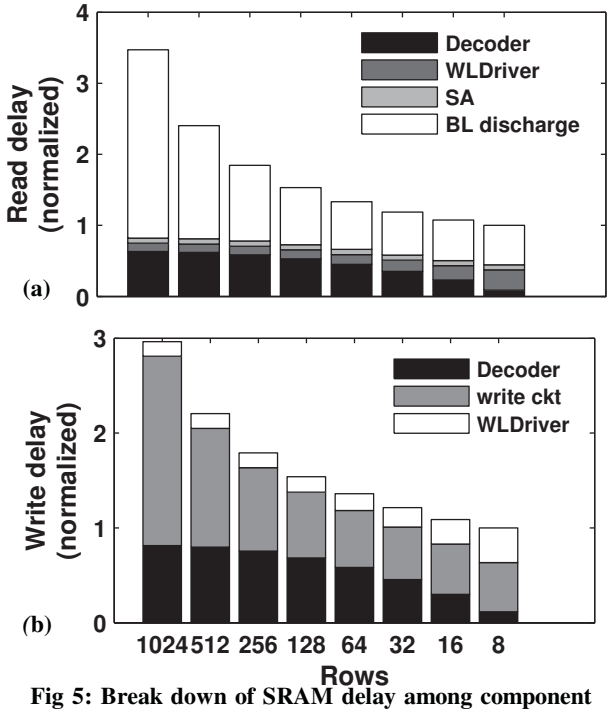

Fig 5: Break down of SRAM delay among component circuits for (a) read and (b) write, with capacity = 16 kb, word-size = 16 bits, technology = PTM 45nm.

built-in components. The base-case prototype provides the following information.

- Design parameters for optimal design
- FoM information for each component and the top-level
- Optimal trade-off curves

In the current implementation, the design parameter estimated by the tool is the array configuration (rows, columns) for the optimal design, with all other design parameters (e.g. device sizes) fixed, but ViPro expands to support additional metrics and variables. The FoMs considered are energy and delay, and the tool generates the optimal trade-off curve between these metrics. Fig 5 shows the delay FoM information for various array configurations, for read and write. It also produces similar break down information for energy. The energy is dominated by the bitline for both read and write.

Fig 6 shows the optimal E-D curve for the base-case prototype generated using 45 nm PTMs [16]. Depending on the top-level requirements of energy or delay, we can choose a design point from this curve, which provides the design parameters (rows, columns in this example) for a compiler to generate a full design. When designing SRAMs in cutting-edge technologies, designers have to deal with process models that are in constant flux. We can use ViPro to deal with this problem by using the CE to re-characterize the process for the new device models, and re-generate the base-case prototype. To demonstrate this, we increase the threshold voltage (V_T) of the transistors specified in the PTM model and re-optimize the base-case prototype for the changed model files. Now, since the new transistors are slower, fewer bit cells can be placed on a bitline to get the same performance as before. Though an experienced designer could easily draw this conclusion qualitatively, it would be much

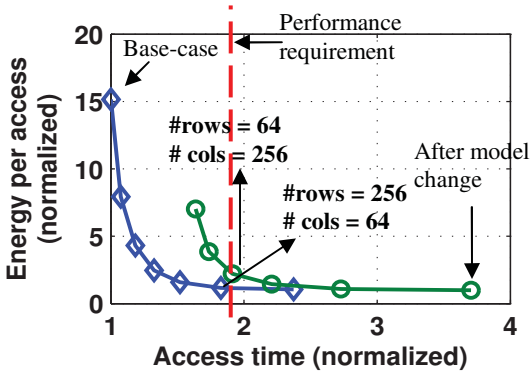

Fig 6: Optimal E-D curve for generated base-case prototype using 45nm PTMs and re-generated base-case E-D curve after process model change. Memory capacity = 16kb, word-size = 16 bits.

harder to decide the actual number of optimum cells per bitline, since the higher V_T transistors impact the FoMs of all the SRAM components.

Fig 6 shows the optimal E-D curve for the changed models. We observe that the same performance requirement (e.g. 1.9 units) leads to a different optimal design, with a different array configuration (e.g. 256x64 for the original models and 64x256 for the higher V_T model). Thus, ViPro provides quantitative knowledge about the impact of a process model change on the optimal SRAM design.

4.2 DESIGN RE-OPTIMIZATION

With the base-case prototype as a starting point, we can now explore various circuit options (e.g. different SA or bitcell) or architectural choices (e.g. different word-size), and observe how the optimal SRAM design changes with these decisions.

First, suppose we want to incorporate a new SA into our SRAM design. The SA is still being designed but is targeted to have 24% lower delay at the cost of 15% higher offset, when compared to the SA in the base-case. We simply plug in these estimates for the SA component in the HMM model of the SRAM (by defining them in the configuration file), as we do not have a completed design of the SA yet. Now, the higher SA offset requires a larger bitline discharge. Thus, we replace the bitcell in the HMM with a larger bitcell that provides higher read current. Since the bitcell is already designed, we replace the bitcell schematic used in the base-case with the larger one (by picking the bitcell from the CE library). Thus, we specify different components of the SRAM with different level of detail. After making these changes to the SRAM, we run ViPro to re-optimize the modified design.

Fig 7 shows the tool output for the base-case and the re-optimized design. We see that if the performance requirement is more than 1.6 units, the re-optimized design has higher energy than the base-case for the same delay, or conversely has higher delay than the base-case for the same energy, making the base-case more optimal. On the other hand, below 1.6 units of delay, the new design is more optimal. Thus, ViPro helps us decide whether to choose the new design or not, even when the actual SA design was not complete. Note that we have only considered energy and delay as the metrics of interest in this example. The

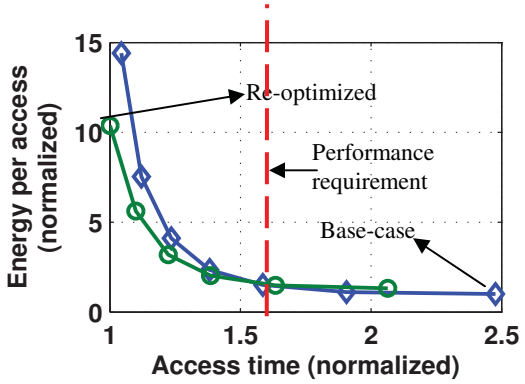

Fig 7: Optimal E-D curve for SRAM prototype in PTM 45nm after changes in the bitcell and SA circuits. Memory capacity = 16kb, word-size = 16 bits.

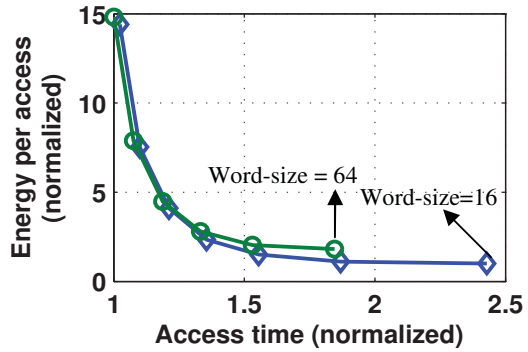

Fig 8: Optimal E-D curves in 45nm PTM for 16 kb SRAM prototypes using word-size of 16b and 64b

optimization result and consequently, the design decision would change if area is considered since the larger bitcell would increase the SRAM area.

We can also use ViPro to look at the impact of changes at the architecture level on the SRAM FoMs. For example, suppose we had to choose between a word-size of 16 and 64 for our design. We simply change the word-size input to generate the optimal E-D curves for both designs, as shown in Fig 8. We observe that one word-size may be slightly more optimal than the other depending on the energy and delay requirements.

Thus, ViPro provides information about how circuit and architectural level choices affect the global FoMs and helps the designer make critical design decisions right from the beginning of the design, without requiring complete or detailed description of the component circuits.

4.3 CROSS-TECHNOLOGY USAGE

When porting an SRAM design to a new technology, it is likely that the optimal design changes even with no change to the design requirements. ViPro helps re-optimize the design by generating a base-case prototype in the new technology after re-characterizing the components and the technology through the CE (e.g. by defining device sizes relative to the minimum width and length), similar to the re-optimization that can be done when process models change in an existing technology. Fig 9 shows the base-case tool output for 45 nm and 65 nm PTM

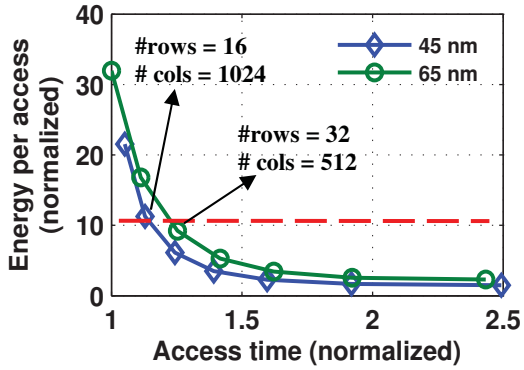

Fig 9: Optimal base-case E-D curves for 65nm and 45nm PTMs generated by ViPro. Memory capacity = 16 kb, word-size = 16 bits.

technologies. We see that for the same energy budget of 10 units, different array configurations are optimal for different technologies (e.g. 16x1024 for 45 nm, and 32x512 for 65 nm).

5. CONCLUSIONS

In this paper, we have presented ViPro, an SRAM design tool. It aids SRAM designers by quickly generating virtual prototypes and top-level SRAM trade-off data at every step of the design process. We have described a design methodology using ViPro and demonstrated how it can be used to first, provide a starting point design, and second, to re-optimize the design to reflect changes at the process, circuit, and architectural levels, while not requiring completed description of the component circuits.

6. REFERENCES

[1] A. Bhavnagarwala, et al., “Fluctuation...,” IEDM, 2005.

[2] T. Suzuki, et al., “A Stable SRAM Cell Design...,” Symp. VLSI Circuits, 2006.

[3] M. Bhargava, et al., "Low-overhead...," CICC, 2009.

[4] Y. Kim, et al., "New SRAM...," Design and Test of Nano Devices, Circuits and Systems, 2008.

[5] P. Shivakumar and N. P. Jouppi, “Cacti 3.0...,” Tech. Rep. Western Research Lab (WRL) Research Report, 2001/2.

[6] N. Muralimanohar, et al., “Optimizing NUCA...,” *MICRO*, 2007.

[7] International Technology Roadmap for Semiconductors. http://www.itrs.net/, 2006 Update.

[8] S.Nalam, et al., "A Technology-Agnostic Simulation Environment...," ICCD, 2009.

[9] X. Bai, et al., "Uncertainty-aware circuit...," DAC, 2002.

[10] V. Sundararajan, et al., "Fast and...," TCAD of ICS, 2002.

[11] K. Chakraborty, et al., "A physical...," TVLSI, 2001.

[12] A. Chandna, “GaAs MESFET...,” Ph.D. dissertation, Univ. Michigan, 1995.

[13] R. Sredojevic and V. Stojanovic., “Optimization-based Framework...,” ICCAD, 2008.

[14] “The MOSEK Optimization Software”, http://www.mosek.com/

[15] B.S. Amrutur, B.S and M.A. Horowitz., "Fast low-power...," JSSC, 2001.

[16] W. Zhao and Y. Cao., "New generation of predictive...," ISQED, 2006.