Automating the Design of Register Files in VLSI Chips: A Prototyping Tool

Kevin Linger
University of Virginia

ABSTRACT

Register files can be difficult to design for low power due to their emphasis on speed. Register files present many opportunities for decreasing power in systems that have slack in performance budgets. A parameterized model was created for performing energy vs. delay analyses of register files. The model allows a designer to substitute in different write and read periphery circuits for comparison, such as in the case of a static and dynamic read periphery. This model was integrated with the tool ViPro, which automates the process of collecting data, and can be an effective way to design register files.

1. INTRODUCTION

Management of energy consumption in register files (RF) is difficult because they are performance-critical components of CPU cores with a large number of design variables. RFs can account for 30% of a CPU's power consumption [1], and better methods of managing energy and performance tradeoffs are necessary for improving energy-efficiency. This can be done by creating a parameterized model of RFs that account for local changes at a global level.

Virtual Prototyper (ViPro) is a tool that assists in automating the design space of SRAM arrays by splitting up the SRAM system into modular components [2]. RF bitcells are similar to SRAM bitcells, which makes them an ideal target for analysis with ViPro. RFs are typically low in capacity, which lessens the importance of area that is typically seen in SRAM arrays. The read and write operations are mostly isolated from each other, which allows for more freedom when designing periphery circuits. An efficient way is needed to design different periphery circuits and compare against them.

A RF model was created and integrated with ViPro to assist in managing the energy/delay tradeoffs made when changing design knobs. It proves to be effective in quickly generating E/D curves for different periphery circuits.

2. REGISTER FILE OVERVIEW

RFs can be characterized as low-capacity SRAMs with multiple read and write ports. Reads and writes can happen simultaneously, because each port has its own access transistor in the bitcell. Each port has an input address that is decoded to select a bitcell to be read or written to, and each write port has an input word to be written. The overall structure can be seen in Figure 1.

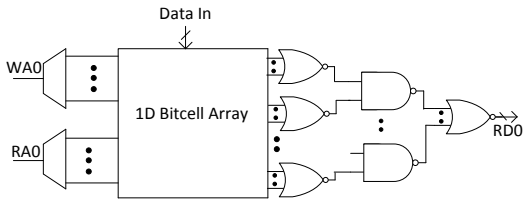

Figure 1. RF organization with one read and write port.

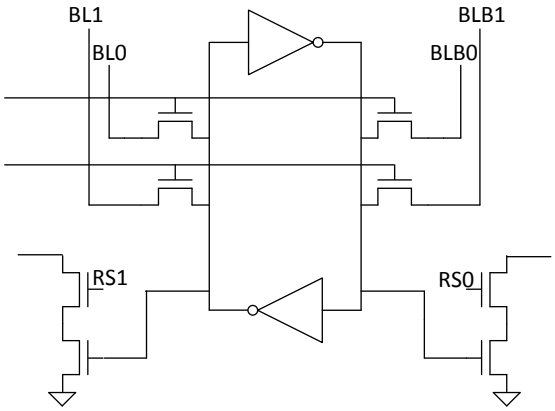

Figure 2. Bitcell of a RF with 2 read and 2 write ports

An 8T bitcell style is used to isolate the differential write from the single-ended read (Figure 2) [3]. Each write port adds two pass gate transistors to the bitcell, while each read adds a pull-down transistor.

2.1 Array Organization

Because RFs are normally small, they can be arranged as a one-dimensional array with one word per row and no banking. There is one row decoder for every port. With large RFs, it might be advantageous to bank the register file to reduce write bitline capacitance [1].

2.2 Read

A read operation in an RF is a wide fan-in multiplexor (Figure 1). Bitcells are grouped together in one dynamic NOR gate to form local bitlines. These local bitlines are then grouped together in a second dynamic gate to create the output. Dynamic circuits are used to achieve fast read latency. The pull-down network consists of a select transistor activated by a row decoder and an input driven by the value in the bitcell. Another option for multiplexing is to use transmission gates [4] at the expense of performance. This improves energy when the

same value at a particular node is read twice, because the bitlines are no longer being precharged every cycle.

2.3 Write

A write operates in a similar way as a standard SRAM array. A pair of differential bitlines is connected to the pass gates of every bitcell. The input data drives the bitlines while a row decoder is used to turn on the pass gates of the target bitcell. The values in the bitcell will then be written after a delay that is dependent on the number of read and write ports, because of the extra capacitance added at each node. The write bitlines do not need to be precharged because there is no longer a need for a differential read. Data gating the input can be used at multiple granularities to make the bitline charging more power efficient by splitting the bitline into localized groups [1].

3. MODELING REGISTER FILES

It may be enticing to optimize the read and write periphery separately because of the organization of RFs. However, completely separating the two is challenging because changes in the peripheries may overlap in the bitcell. For example, if the number of read ports is increased, the write bitcell needs to account for the extra gate capacitance in the bitcell.

A modular RF model was created with ViPro for calculating energy and delay for register files by breaking them into isolated and parameterized components. These components are not directly connected together, but account for other parts of the system, such as in the example above. The RF components consist of a flip flop, decoder, write periphery connected to a bitcell array, and read periphery. Models for decoders and the flip flop were already integrated into ViPro.

3.1 Read Periphery Model

The model for a dynamic read is shown in Figure 3. Two dynamic NORs are buffered together with a NAND gate that feeds into another dynamic NOR. The widths of the two dynamic gates are split evenly, which makes it roughly equivalent to a square root function. When they cannot be split evenly, the nearest powers of two that are used which gives one of them a larger fan-in. The pull-down transistors were upsized to be 10 times the effective keeper strength [3]. This pull-down strength impacts both the performance of the read and the write, as the bitcell is directly connected to the gate of one of the transistors. Load circuits are added to account for the leakage of the other gates.

The model for static reads is similar to the dynamic (Figure 4). Two stages of transmission gates are used with an inverter stage between [4]. The multipliers are calculated in the same way as the dynamic, with a slight variation because there is no NAND gate that groups two local bitlines together. The energy consumption is taken as the average of both a changing output and a repeat output to take advantage of static gates saving energy when the same value is read twice.

3.2 Write Periphery Model

The write periphery includes both the bitcell array and the data input drivers (Figure 5). The array consists of a test bitcell along with a load bitcell that adds capacitance to the bitlines and accounts for leakage in the bitcells. The write driver is a tri-state inverter, and is connected to all bitcells. The delay is taken as the sum of the bitline charge delay and the bitcell node flip delay.

3.3 Bitcell Model

The model for the bitcell is shown in Figure 6. For every write port, two pass gate transistors are added. For every read port, a capacitor is added that is equivalent to the gate capacitance for the technology used multiplied by the effective strength of the access transistor. For dynamic read systems, this is the width of the pull-down transistor used in the dynamic NOR gate. For static reads, it is the sum of the widths of the inverter used in the read periphery. This capacitance is split evenly between the nodes of the bitcell as the read ports are split. When there are an odd number of read ports, one side is given more capacitance. This capacitor encapsulates how changing the number of read ports or the style of read periphery can have an impact on the write delay and allows for the read periphery to be optimized apart from the write.

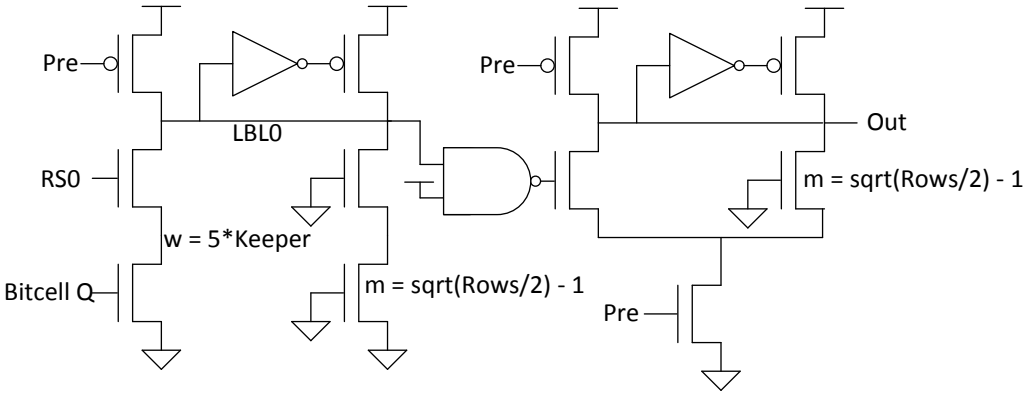

Figure 3. Model for dynamic read periphery

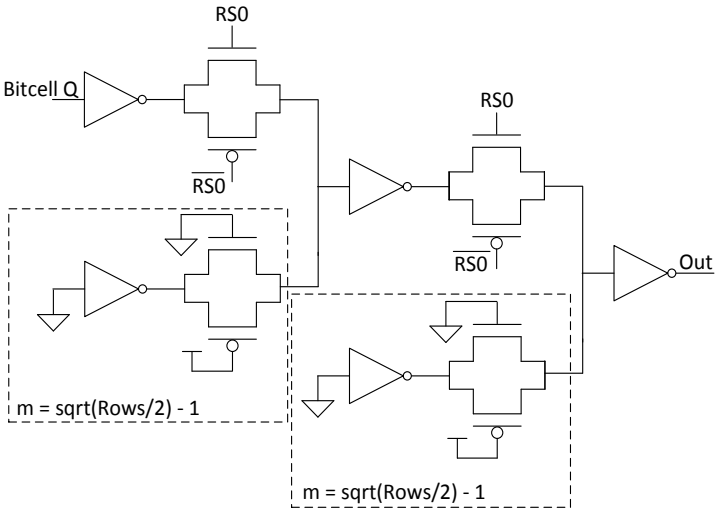

Figure 4. Model for static read periphery

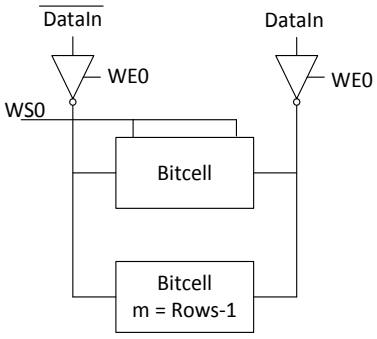

Figure 5. Model for write periphery

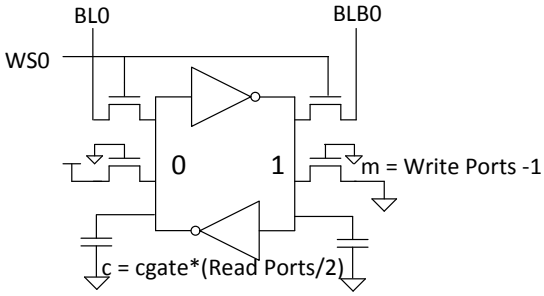

Figure 6. Bitcell model

3.4 Delay Calculation

The results from the various components can be coalesced together to get a total delay and energy for the system.

The delay for a read is:

$$Read\ Delay = DFF + \max(Precharge, Decode) + Read\ Periphery$$

The precharge and decode happen on a different edge of the clock than the actual periphery, thus the maximum between the two sets the frequency.

The delay for a write is:

$$Write\ Delay = DFF + \max(Bitline\ Charge, Decode) + Bitcell\ Flip$$

The bitlines can be charged while the address decoding happens because all bitcells in the array are connected to the bitline. The decode stage is typically not a part of the critical path [3].

The total delay is then the maximum of the read and write delays. One way to implement a register file is to have the write happen on one edge of the clock and the read on the other. The read delay typically dominates and would set the clock frequency in the case [5].

3.5 Energy Calculation

The energy is the sum of the energy of all components. Each component in the model only considers one bit in each word, and one port. These are accounted for in the final calculation.

$$\begin{aligned} Energy\ = DFF * \big((ws * WP) + (\log(NR) * RP)\big) \\ + Read\ Periphery * RP * ws \\ + Write\ Periphery * WP * ws \\ + Decode * (RP + WP) \end{aligned}$$

Where *WP* and *RP* are the number of write ports, and *ws* is the word size. There is a flip flop for each input address (1 per port), and for each data input (1 per write port).

4. OPTIMIZATIONS

The strict speed requirement of RFs can be relaxed in low-performance systems to allow for improved area or power efficiency. Two ways of doing this are reducing the number of physical ports in the system and localizing the input data during a write.

4.1 Reducing Physical Ports

A RF can be multi-ported without having a physical port for every logical port. Adding a port to a RF adds more input drivers, increases bitcell capacitance, and has a quadratic impact on area [5]. These effects can be mitigated by using one physical port for two logical ports. The read delay typically dominates the write delay [5], meaning two writes can be done for every read with a penalty in clock frequency. This allows two write ports to be mimicked with one, which puts two less access transistors in the bitcell and decreases the number of leakage paths. The area is decreased by having less global bitlines, less decoders, and simpler wiring in the bitcell. The delay with double-pumped writing is more complex than simply doubling the delay with two ports because of

the decrease in bitcell capacitance. The model used in ViPro accounts for this and can give a more accurate write delay.

4.2 Data Gating Write Bitlines

Write data drivers lose energy-efficiency by charging the large global bitlines which are connected to every bitcell, particularly in large RFs. This write data can be gated to split the global bitline into local bitlines [1]. The input address can then be pre-decoded to determine subgroup of cells is going to be written to, which decreases the capacitance the input needs to drive. This extra buffer stage can add delay to the system, but may make the global bitline charge faster. ViPro can assist in this analysis by making it easy to change the write periphery and examine how it changes the whole system.

5. RESULTS

ViPro makes analysis of register files more efficient by simulating parameterized models. This section demonstrates a few examples of experiments that can be run, and how two different read styles can be analyzed simultaneously.

ViPro was used to examine the E/D curve of the read and write operation for static and dynamic read periphery with varying inputs. Figure 6 shows the E/D of both reads and writes at voltages varying from 0.8 to 1.2 volts. The curve reveals the difference between energy consumed in a static read vs. a dynamic read, as static reads show to consume 75% less energy, with a small delay tradeoff. The write operation with static read periphery is also faster and consumes less energy than with dynamic, because the dynamic read periphery adds more capacitance to the bitcell. ViPro makes it easy to sweep any parameter to generate these curves with different periphery circuits.

A RF designer might want to know how many read ports can be afforded under a given power or speed budget. For read periphery, there is no impact on delay and the energy can simply be calculated by multiplying the energy by the

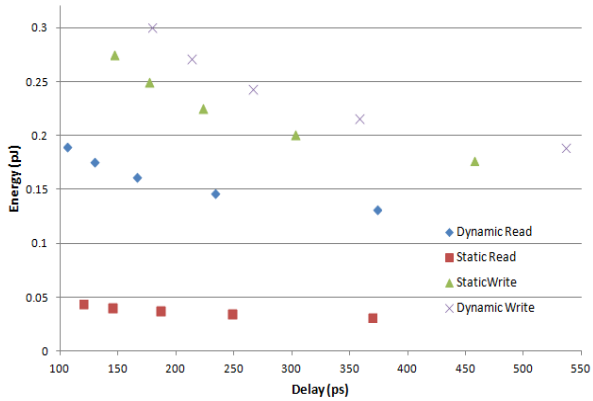

Figure 7. E/D curve comparing static and dynamic read styles. Shows both the read and write operations.

number of read ports. It is more interesting for the write operation, where a more detailed calculation is needed. The results of this analysis are shown in Table 1.

Read Ports	Write Energy (pJ)	Bitline Charge Delay (ps)	Bitcell Flip Delay (ps)	Write Delay (ps)
1	.442	75	94	169
2	.454	75	153	228
3	.492	75	195	270
4	.505	75	252	327

Table 1. E/D of write operation as read ports are increased

This data shows that increasing the number of read ports has a very small impact on energy, but a drastic impact on the time it takes for the bitcell node to be flipped. This is due to the pull-up transistor in the bitcell not being strong enough to flip one side of the bitcell quickly from a 0 to a 1. ViPro can then be used to see how upsizing the bitcell and making the pull-up stronger can change the write delay. The results are shown in Table 2.

Cell Size Factor	Write Energy (pJ)	Bitline Charge Delay (ps)	Bitcell Flip Delay (ps)	Write Delay (ps)
2	.612	87	200	287
4	.792	103	147	250
6	.960	119	110	229
8	1.126	135	91	226

Table 2. E/D of write operation as the bitcell is upsized. 4 read ports are used.

This data shows that as the bitcell is initially upsized, performance is gained in the flip delay, but lost in the bitline charging. Overall, the gains made in the bitcell flip delay outweigh the losses on the bitline charging delay. This comes at a large energy cost.

5. CONCLUSION

Register files are ideal targets for automation due to their low capacity and the separation of the read and write operations. A model was created to account for how changes in read periphery can impact the write periphery. ViPro makes it easy to plug in new periphery circuits and sweep parameters and measure the E/D curves. Future work should be done to explore energy-efficient write drivers through localized data gating.

5. REFERENCES

[1] E. Donkoh, T. Siong Ong, Y. N. Too, P. Chiang, "Register file write data gating..." *Low power electronics and design*, pp. 149-154, 2012

[2] S. Nalam, M. Bhargava, K. Mai, B. H. Calhoun, "Virtual Prototyper…" DAC, pp. 139-143.

[3] R. Krishnamurthy, et al., "A 130-nm 6-GHz 256 × 32 bit leakage-tolerant register file," *Solid-State Circuits*, pp.624-632, 2002

[4] S. Okumara, et al., "Which is the best dual-port SRAM…" ICICDT, pp. 55-58, 2008

[5] R. K. Montoye, et al., "A 4R2W register file…" ISSCC, pp. 256-258, 2011.