

## Chapter 1

# CHALLENGES FOR ARCHITECTURAL LEVEL POWER MODELING

Nam Sung Kim, Todd Austin, Trevor Mudge  
*The University of Michigan, Ann Arbor*  
{kimns,austin,tnm}@eecs.umich.edu

Dirk Grunwald  
*The University of Colorado, Boulder*  
grunwald@cs.colorado.edu

**Abstract** The power aware design of microprocessors is becoming increasingly important. Power aware design can best be achieved by considering the impact of architectural choices on power early in the design process. A natural solution is to build a power estimator into the cycle simulators that are used to gauge the effect of architectural choices on performance. Cycle simulators intentionally omit considerable implementation detail in order to be efficient. The challenge is to select the details that must be put back in if the simulator is required to also produce meaningful power figures. In this paper we propose how to augment a cycle simulator to produce these power figures.

**Keywords:** power estimation, cycle simulator, microarchitecture

## 1. Introduction

Power dissipation has become a significant constraint in modern microprocessor design. In many mobile and embedded environments power is already the leading design constraint. Although it may not be so apparent, it is almost as important in the design of general purpose high-performance computers [1]. It has become one of the primary design constraints along with performance, clock frequency, and die size. In addition to extra heat removal costs, high power consumption in em-

bedded processors also reduces the battery lifetime. Hence, a mobile computing system’s quality and reliability could be affected by its high power dissipation. In case of high performance microprocessors, high power dissipation leads to thermal issues like device degradation, higher packaging cost, and reduced chip lifetime.

The elevation of power to a “first-class” design constraint requires that power estimation be done at the same time as performance studies in the design flow. Performance analysis for a proposed design is usually accomplished during the design exploration phase with a cycle simulator. The natural solution would be to augment cycle simulators so that they can also provide power estimates. This has been done in the case of three recent simulators [2, 3, 4]. A cycle simulator models the behavior during each clock cycle of the processor. The goal of these simulators is to assess the impact on performance of cache and memory organizations, pipelining, multi-instruction issue, branch prediction, and other microarchitectural mechanism. The more sophisticated cycle simulators can boot an operating system and run significant parts of an application program within a several hour period. This enables the user to observe the performance under a variety of workloads for billions of cycles. Such long simulations are necessary to provide the confidence in the resulting performance figures. To simulate billions of cycles within a several hour period requires a high degree of efficiency in the cycle simulator. Efficiency is achieved by abstracting away the physical behavior of the microarchitecture — the very details that are required to obtain power figure. The challenge for architectural power modeling is to add enough detail back into the simulator so that power estimates are meaningful without unduly slowing the simulator.

The difficulty of this challenge is apparent from the results that one of us obtained by calibrating two of the recent power simulators mentioned above, the Cai-Lim simulator [2] and the Wattch simulator [3]. Figure 1.1 reproduces one of the figures from [5]. It shows power estimates obtained from two SPEC benchmarks, *mk88sim* and *lisp*. Three implementations of an 8-wide issue processor are examined: 1) an out-of-order machine with gated clocks; 2) an in-order machine; and 3) a half width machine, i.e., a 4-wide issue. Clearly the two models give significantly different results. The accuracy that we can expect is probably much less than the 10% claimed in [3]. This is not unacceptable for an early design tool. However, to be useful in the design process a successful model should indicate the trends accurately. The results in Figure 1.1 show that the two models also give conflicting views on trends. The Cai-Lim model indicates the in-order machine is the least power hungry, even allowing for a high level of imprecision in the results. The Wattch

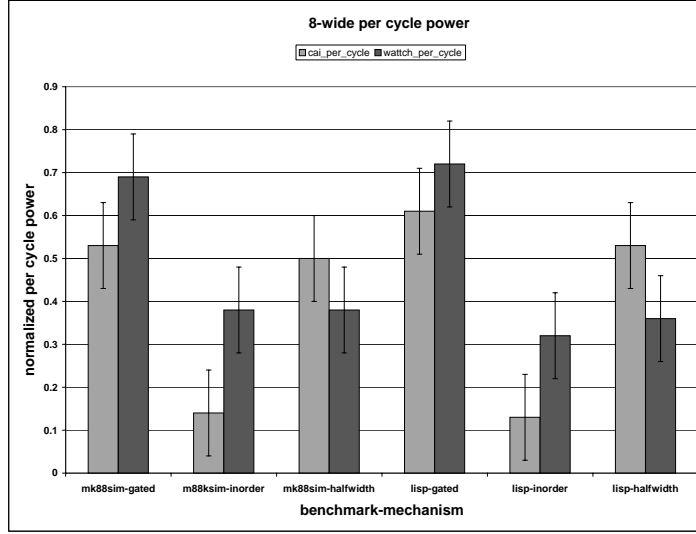


Figure 1.1. Normalized per cycle power estimation results for 8-wide microarchitectures. The error bars are based upon the 10% accuracy reported by Wattch (4).

model, on the other hand, does not show a clear preference. This paper will identify the causes of these inaccuracies, and propose how a cycle simulator can be augmented to correct for them.

The remainder of this paper is organized as follows. The next section reviews the power metrics required for power aware design. Section 3 discusses existing cycle simulators that measure power, and points out their weaknesses. Section 4 discusses the requirements needed by a cycle simulator to produce useful power figures. Section 5 considers the details of implementing such a power estimator. Section 6 conclude the paper and identifies some remaining open questions.

## 2. Power Metrics

There are three components that define the important contributions to power consumption in CMOS<sup>1</sup> technology:

$$P = ACV^2f + \tau AVI_{short} + VI_{leak} \quad (1.1)$$

<sup>1</sup>We focus on CMOS because it will likely remain the dominant technology for the next 5-7 years

The first component is perhaps the most familiar. It measures the dynamic power consumption caused by the charging and discharging of the capacitive load on the output of each gate. It is proportional to the frequency of the operation of the system,  $f$ , the activity of the gates in the system,  $A$  (some gates may not switch every clock), the total capacitance seen by the gate outputs,  $C$ , and the square of the supply voltage,  $V$ . The second term captures the power expended due to the short-circuit current,  $I_{short}$ , that momentarily,  $\tau$ , flows between the supply voltage and ground when the output of a CMOS logic gate switches. The third term measures the power lost due to leakage current that is present regardless of the state of the gate.

The first two terms can be lumped as activity based because they are directly related to the toggling frequency of the gates in the circuit. In contrast, the leakage term is unaffected by activity, because it is governed only by the number of gates and their threshold voltages. It is only affected by activity in the sense that leakage is reduced to zero when the gate is turned off. Unfortunately, this also results in any state in the circuit being lost. Thus to obtain power from the execution of a cycle simulator it is necessary to tie cycle behavior to activity at the gate level for the first two terms and to estimate the number of gates that the microarchitecture requires for the third term. The difficulties associated with cycle level power estimation arise directly from the difficulty of calculating these values with any level of accuracy.

If average power consumption were our main concern then the values for  $A$  and  $f$  could be calculated by sampling. Sampling would give us the time to perform more detailed power consumption calculations. This idea has been proposed in [6]. However, average power values can hide important details as can be seen in [7]. In this work the energy dissipation of individual ARM instructions is measured by running each instruction in a loop and measuring the average current drawn. The results show very little difference between the energy usage of different instructions, because the effect of averaging is to smooth out short lived effects like cache misses and pipeline stalls. In addition, there are two other power metrics that are important design constraints. The first is peak power. Typically, systems have an upper limit, which if exceeded will lead to some form of damage. The second is dynamic power. Sharp changes in power consumption can result in inductive effects that can result in circuit malfunction. The effect caused by “di/dt” noise. Equation 1.1 can be used to monitor peak power and estimate “di/dt” noise, if a running value of (1.1) is maintained. Sampling can miss these effects.

### 3. Previous Work

As we have noted, the abstractions necessary for efficiency make it difficult to base an accurate power estimation tool on a cycle simulator. The lack of detailed low-level physical design information such as interconnection capacitance, types of circuits for each microarchitectural block, clock trees, and I/O pads are among the principal sources of inaccuracy.

Prior work on microarchitectural level power estimation [2, 3, 4] has mainly relied on microarchitectural-activity-based calculations. A typical power dissipation figure for each block used in the target microprocessor is estimated. Then the activity of each block is recorded every cycle based on the behavior of the cycle simulator. Finally, the power dissipation of the microprocessor is estimated by combining the activity with the power figures.

In [3], Brooks et al. utilized detailed analytical power models for array structures and content addressable memories based on CACTI [8] to estimate the power consumption of memory-like microarchitectural blocks. In [2], Cai et al. introduced a power density model to estimate power dissipation of each microarchitectural block based on proprietary Intel design data. In [4], Vijaykrishnan et al. also considered bus transition-sensitivity by employing a register-transfer level power estimation technique. The power models for their blocks were implemented as look-up tables (LUTs) [9]. They also considered the power dissipation of the I/O pad and external memory bus.

In these simulators there is no accounting for changes in power due to data sensitivity. According to [10] and [11], the power consumption of a microarchitectural block is highly dependent on the input data characteristics applied to it. Figure 1.2 shows power dissipation simulation results for an 8-bit ALU and an 8-bit multiplier at 100MHz using the TSMC 0.25 $\mu$ m LEDA library, Synopsys Design Compiler, and PrimePower. These measurements agree with those in [12]. It can be seen that the variation in power dissipation is quite significant for different input patterns, suggesting we should consider data-sensitivity in the power estimation of microarchitectural blocks.

All but one of the earlier work also ignored power dissipation caused by accessing the external memory, which usually consumes a significant amount of power since the I/O pads of the microprocessor typically drive very large off-chip bonding wires. Furthermore, the earlier work has ignored power dissipation caused by the clock distribution network and global interconnect — the internal buses that interconnect the blocks. This is particularly tricky to do because it requires some notion of the

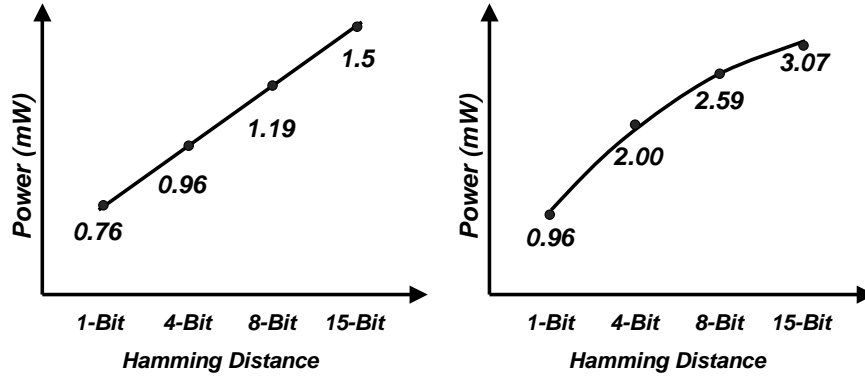


Figure 1.2. Power dissipation simulation for a 100 MHz 8-bit ALU and 8-bit multiplier. These microarchitectural blocks were designed using the TSMC  $0.25\mu m$  LEDA library, the *Synopsys Design Compiler*, and *PrimePower*. Power is plotted against the Hamming distance between consecutive inputs to the blocks

layout of processor. However, interconnect is becoming an ever more significant contributor to power consumption.

Clearly, there are a number of omissions in current cycle simulator-based power estimation tools that could account for their inaccuracy. In the next section we will propose a framework for augmenting cycle simulators so that the omissions can be included.

## 4. Augmenting a Cycle Simulator for Power Estimation

### 4.1 Details omitted from cycle simulators

As we have noted, cycle simulators derive their speed from abstracting out many of the physical details. We touched on some of the resulting omissions in the discussion of earlier power estimators. We will illustrate the extent of this abstraction by considering *SimpleScalar* [13], a cycle-based performance simulator that is widely used and forms the basis for the earlier power estimators. *SimpleScalar* simulates a specified architecture running a particular benchmark and returns performance in terms of total simulated clock cycles. In order to do this it only needs to trace address streams and some architectural activity. Examples of this activity include function unit usage, and the number of cache, TLB, and branch predictor accesses. These are combined with their respec-

tive access latencies to calculate the execution times of instructions. The effect of resource and data hazards is reflected in the access latencies. Thus cycle counts of program execution are calculated without having to model the detailed structure of pipelines. The simulator executes the instructions and stores their results in the simulated register file and main memory in the issue cycle of instructions interpreted. It simply advances the simulation clock after calculating the latencies that would have resulted from the execution of a real multi-issue pipeline.

There is no modeling of the movement of instructions and data between the various pipeline stages after the issue stage, or between other microarchitectural blocks. For instance, the fetch stage fetches instructions from the simulated main memory directly, not from a simulated L1 instruction cache. The cache is modeled by keeping track of the addresses of the instructions and noting when they are no longer included in a list of cache line addresses. When this occurs, a miss penalty is accounted for, and the cache line list is appropriately updated to reflect new cache entries and evictions.

To illustrate the effects of abstraction on memory buses, consider how SimpleScalar models a memory access over a memory bus. SimpleScalar checks the current memory access status and returns the access latency and the requested data blocks. The latency is determined from the number of the memory ports, the number of the requested data blocks, and whether or not the previous memory access cycle is complete. In real microprocessors, however, the memory access transaction occurs over several cycles, and the requested data blocks are transferred from/to the memory during the pertinent cycles according to the memory transaction type and the number of the requested data blocks as shown in Figure 1.3.

With only the latency information we have no idea about the details of the memory transfer cycles such as addresses and data values. This was also a shortcoming of earlier power estimators — they assumed that the transaction occurs in the currently accessed cycle regardless of the access latency, and the characteristics of the transaction cycle. To correct this, we need a mechanism for tracing data transactions on buses (internal as well as external) in a cycle accurate way. In order to provide this mechanism, it is necessary to augment the simulator to trace data bus streams. This can be accomplished with special routines at the interfaces of the microarchitectural blocks that capture the cycle accurate bus transaction cycles. In other words, we need to know what values are on the internal buses between the microarchitectural blocks and on the external I/O buses in each simulation cycle, in order to measure the switching power more accurately.

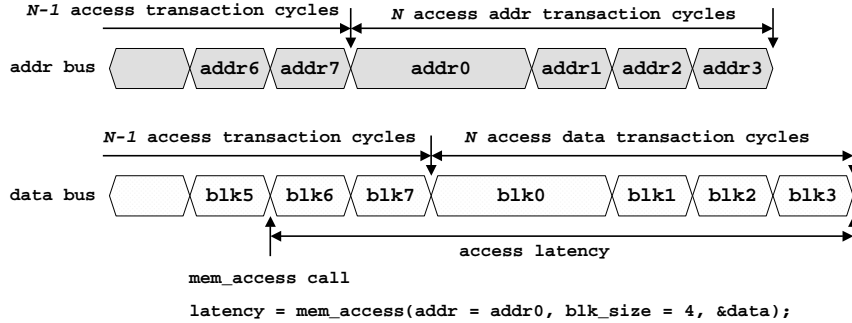


Figure 1.3. An example of memory access — real microprocessor vs. performance simulator. The waveform illustrates the timing of external addresses and data bus transaction of in real microprocessor. The function call is the abstraction

In summary, the principal abstraction in cycle simulators is to omit the modeling of data movements on the internal and external buses between microarchitectural blocks such as function units, caches, memories and pipeline stages.

## 4.2 Power Estimation Methodology

Figure 1.4 shows a proposed methodology for power estimation by augmenting a cycle simulator. First, target technology parameters need to be known or estimated — the *Berkeley Predictive Technology Model* is an example estimator. Other factors that are needed include the supply voltages, threshold voltage, and capacitance per area/length and sheet resistance values of the interconnecting material. In addition, the microarchitectural specification, target operating frequency, and circuit design style for each microarchitectural block should be determined as well.

Second, we need to construct power models for microarchitectural blocks based on the circuit or sub-system design styles. For example, the datapath of the microprocessor may be designed as full-custom, through standard cell synthesis, or by using a datapath compiler. The different cases require different circuit and power models. We can generate analytical power models [8, 14], empirical models [15, 16], or by employing other power macromodeling techniques [12, 17, 18, 19, 20, 21, 22, 23].



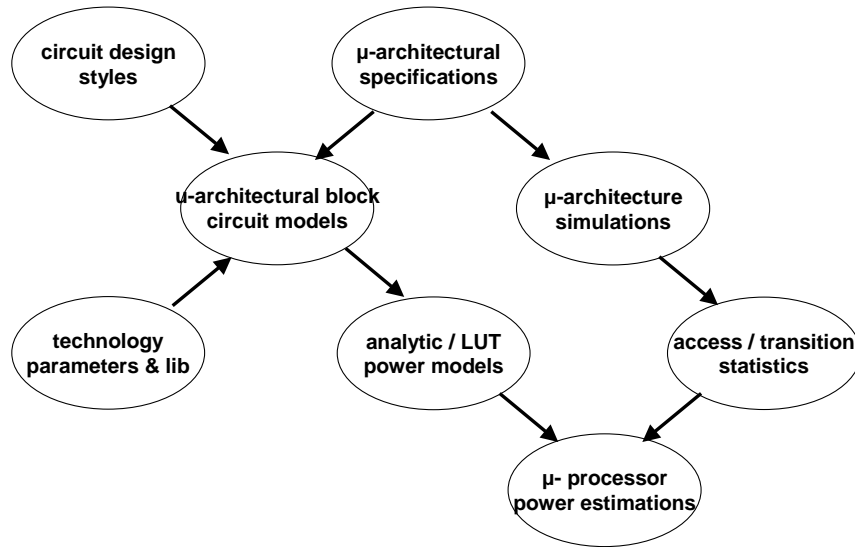


Figure 1.4. A microarchitectural power estimation methodology.

These models need only be developed once. Modeling techniques will be discussed further in Section 5.3.

Finally, the power estimator calculates the power dissipation of each microarchitectural block by obtaining the execution statistics from runs of the cycle simulator.

## 5. Implementing a cycle-accurate power estimator

The power dissipation of each microarchitectural block consists of three components:

- 1 The switching power of the load capacitance, which is directly proportional to the number of zero-to-one transitions of outputs per cycle. The number of transitions on a bus (zero-to-one and one-to-zero) can be summarized by recording the Hamming distance between successive bit patterns on that bus. The number of zero-to-one transitions is half this number on average.
- 2 The power dissipation of the microarchitectural block caused by the switching of its inputs, which is approximately proportional to

the number of transitions of inputs (zero-to-one and one-to-zero) per cycle. The power dissipation is actually quite specific to each type of microarchitectural block, and can be characterized in a LUT indexed by Hamming distance. The LUT could be replaced by a simple equation or a macromodel.

- 3 The leakage power from the reverse-biased diode current and sub-threshold leakage current that are approximately proportional to the area and the number of the transistors of the microarchitectural block. The last term is no longer negligible as supply voltages and threshold voltages decrease. We can implement a generic data structure for power estimation that can handle most of the microarchitectural block types by passing as arguments circuit parameters and LUTs indexed by the Hamming distances for the data on the input and output buses. This is combined with the switching activities from the runs of the simulator — the switching activities define the Hamming distances. As mentioned earlier, the LUT entries are calculated off-line. There are a number of well-known techniques for this, see [12, 17, 18, 19, 20, 21, 22, 23].

Thus, we need to implement an interface between the cycle simulator and the power estimator to collect the appropriate switching activities.

## 5.1 Implementation of the data structure and microarchitectural block models

Figure 1.5 shows a generic data structure for the architectural power estimators that will support flexible power models, and that will interface to SimpleScalar.

The circuit parameters in the data structure include a circuit design style — dynamic or static, a supply power voltage, an operating frequency, and complexity information such as the number of estimated transistors and area of the microarchitectural block.

The use of dynamic vs. static circuits affects the power dissipation characteristics of the block significantly. In current state-of-the-art microprocessors, multiple supply voltages are often fed to the chip. For example, 2.5V might be supplied to the internal core, and 3.3V to the I/O pads. Therefore, the data structure for each microarchitectural block should contain supply voltage information. In addition, some peripherals, which would typically be modeled as microarchitectural blocks may be integrated into the chip, and may operate at different frequencies as well. Thus, we support an independent operating frequency for each functional unit as well as a supply voltage. Furthermore, the complexity information, in particular the number of the transistors of the block

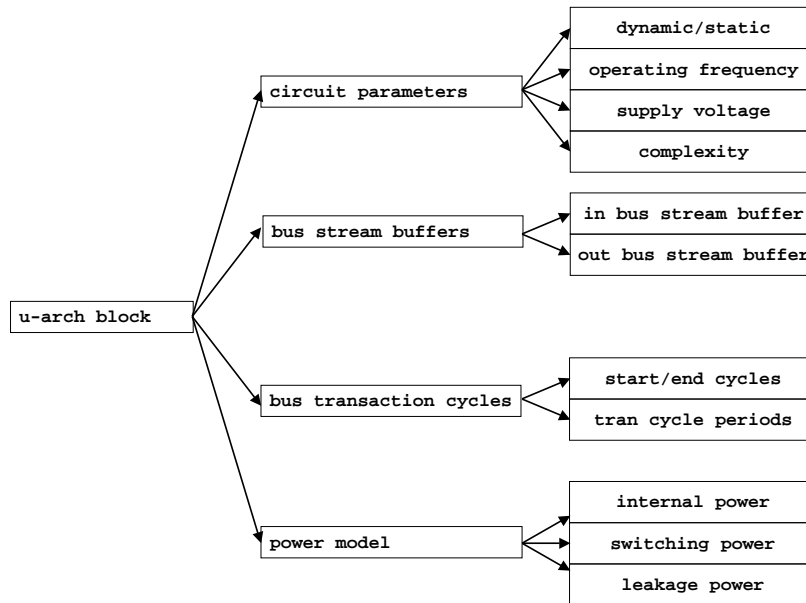


Figure 1.5. Generic data structure for a microarchitectural block.

determines leakage power, and is important for estimating the clock tree capacitance.

A microarchitectural block is modeled as a block with input and output buses (the internal buses between the blocks), see Figure 6. The input and output buses are implemented in the simulator as stream buffers in order to support Hamming distance calculations over complete bus transactions. In the case of blocks like ALUs, these buffers need only have one stage, but for memories and I/O buses they may need to be much deeper (recall Figure 1.4). These buffers are filled by the interface routines of the cycle simulator. In order to estimate the power of each block, we employ a method based on Hamming distance [12], as noted earlier. However, the information in the data structure does not limit us to Hamming distance methods. The availability of input and output statistics permits the use of other dynamic estimation techniques [22, 23].

In order to analyze Hamming distance or zero-to-one transition activity we have to keep the previous bus stream values. However, there

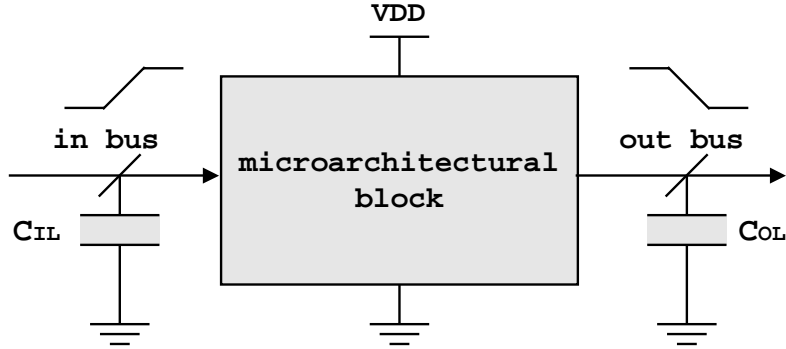


Figure 1.6. The microarchitectural block model

are different types of bus states to consider — unchanged, pull-up, pull-down, and high impedance. For instance, a directional bus usually keeps its value until the output driver of the microarchitectural block drives a different value on the bus, while a bi-directional bus that is in the high impedance state when it is not driven by any block has an unpredictable value due to leakage current. This value becomes predictable if keeper logic is used to prevent excessive current flow due to leakage current of the bus. Accounting for all these cases is not difficult, but it does require that we have to specify the bus type when we initialize the data structure shown in Figure 1.7.

Finally, microarchitectural blocks can be decomposed into more detailed blocks to increase the granularity and the accuracy of the power estimation. For instance, a cache can be viewed as one microarchitectural block, or split into decoders, tag arrays, data arrays, comparators, sense amplifiers, and output drivers, which we can model as a collection of separate blocks, assigning different power models to each block, see Figure 1.8.

## 5.2 Cycle simulator and power estimator interface

In order to track the bus streams for each block we need to implement an interface or API to collect the stream information for each microarchitectural block from the cycle simulator. The relative timing of the bus streams are dependent on the microarchitectural block and its cur-

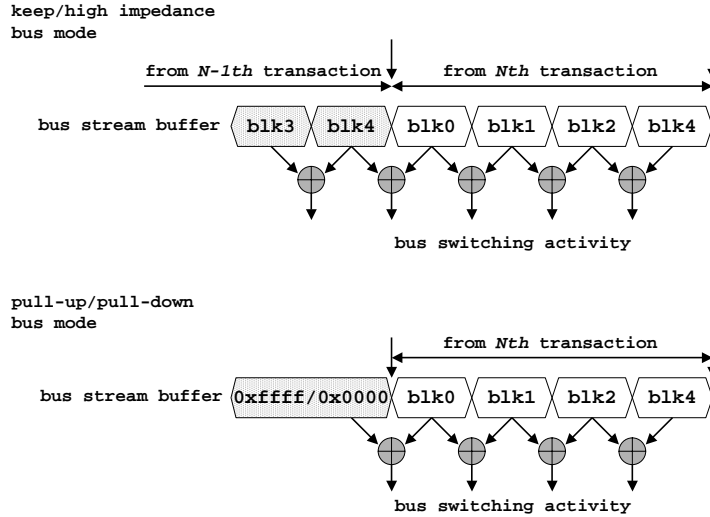


Figure 1.7. Bus modeling with data-activity

rent state. For example, reads and writes to a memory block create quite different bus switching activities, which are further modified by the occurrence of a cache miss.

Furthermore, as mentioned before, the cycle simulator often simulates the logical behavior of a microarchitectural block in a single cycle and then uses latency and hazard information to model the performance impact of the microarchitectural activities that may occur over several cycles. Figure 1.9 illustrates how stream information should be retained to correctly model the real data activity occurring on the bus.

### 5.3 Power Modeling Techniques

The estimation of the power dissipation is quite straightforward if we know the data activity of the microarchitectural block so that we can derive the effective capacitance of the block. In this section we will examine in more detail how we can derive the effective capacitance for several important microarchitectural blocks, the internal buses that connect them, and the clock tree that synchronizes them.

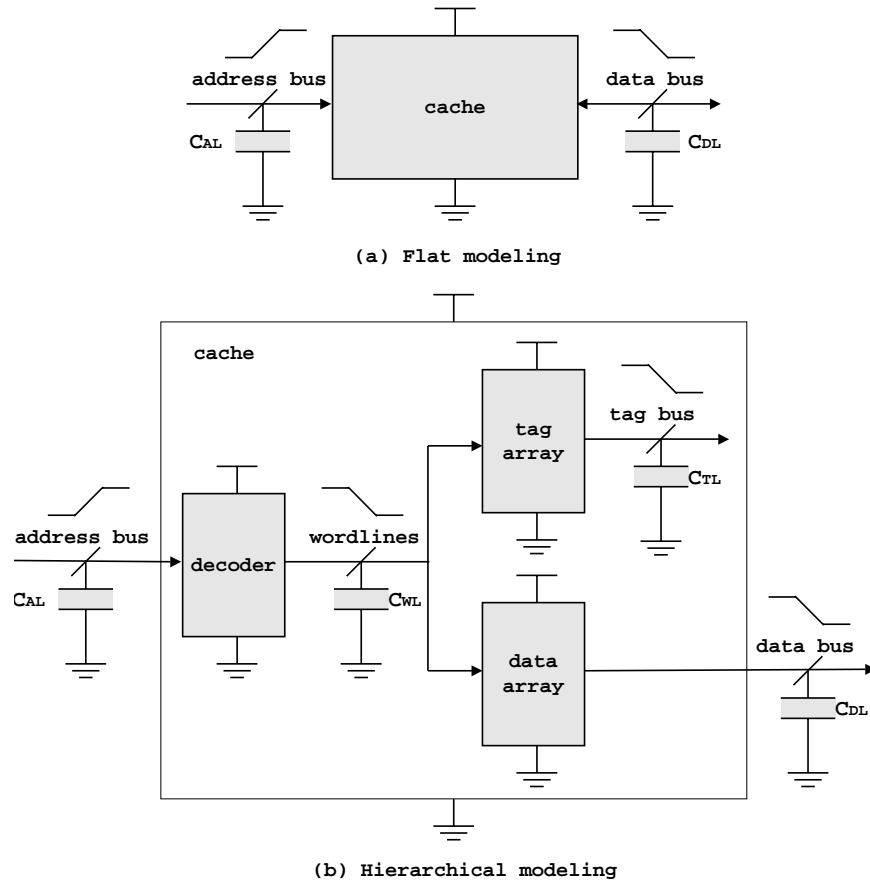


Figure 1.8. Hierarchical modeling of microarchitectural blocks.

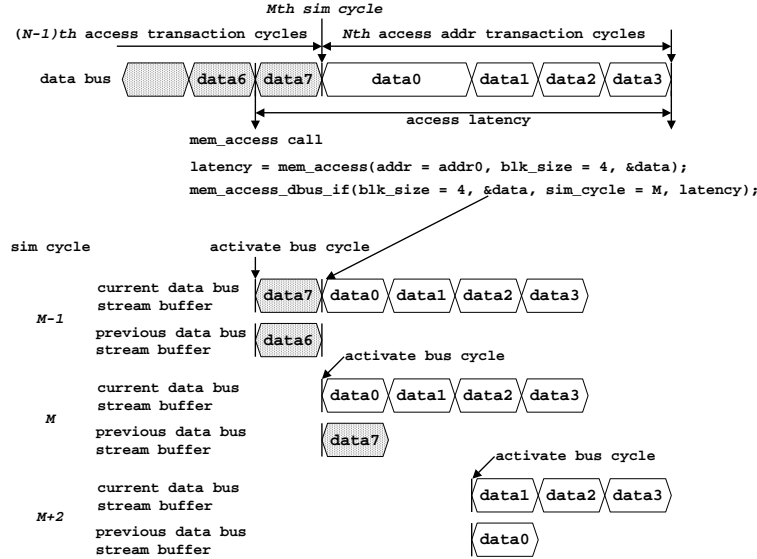


Figure 1.9. Interfaces between the simulator and the power estimation framework, and activation of bus transaction cycles.

**5.3.1 Memory models.** A microprocessor may contain many memory blocks including caches, TLBs, register files, reorder buffers, etc. Their fraction of the total area and their contribution to power consumption in a microprocessor can be as high as 40% [14]. Most memory structures consisting of five parts — an SRAM cell array, a row decoder/wordline driver, a column decoder, a sense amplifier, and a precharge circuit, see Figure 1.10.

The power consumption of most memory blocks can be estimated quite accurately by using a cache access time estimation model. Such models typically derive the access time from estimates of the capacitance and the resistance of the critical path during memory reads or writes. We can use these capacitances to obtain estimates of the power or energy consumption. Two examples are CACTI [8] and the analytical equations proposed in [14]. Both CACTI and the model in [14] compute the physical capacitance of each stage of the memory.

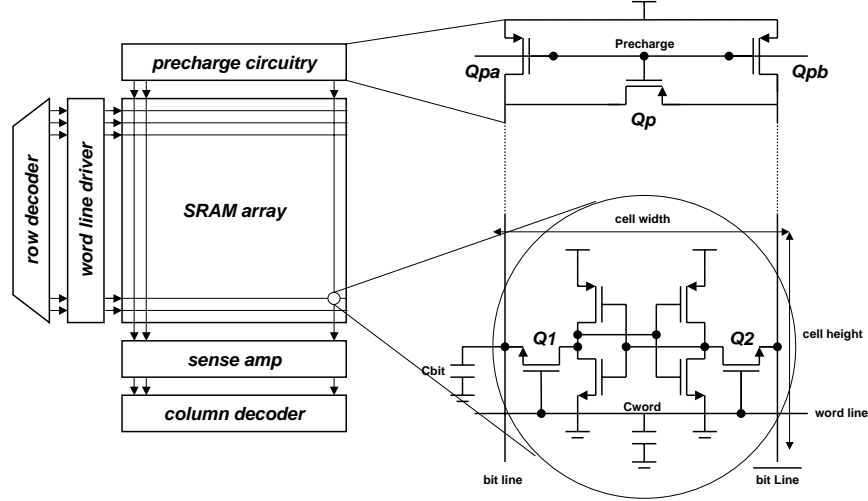


Figure 1.10. Memory structure.

We illustrate the method using CACTI. The total switching capacitance of a bit line  $C_{BitTot}$  represented in (1.2) consists of the metal line capacitance  $C_{bit}$  and the drain capacitance of transistors  $Q_1$ ,  $Q_p$ , and  $Q_{pa}$  in Figure 1.10.

$$C_{BitTot} = N_{rows} \cdot (C_{d,Q_1} + C_{bit}) + C_{d,Q_p} + C_{d,Q_{pa}} \quad (1.2)$$

Similarly, the total switching capacitance of a word line shown in (1.3) is composed of the metal line capacitance  $C_{word}$  and the gate capacitance of the pass gates  $Q_1$  and  $Q_2$ .

$$C_{WordTot} = N_{cols} \cdot (2 \cdot C_{g,Q_1} + C_{word}) \quad (1.3)$$

Therefore, the power dissipation of the SRAM array can be represented by (1.4). In the power dissipation of the bit line  $V_{BitSwing}$  should be used instead of  $V_{DD}$  because the voltage swing of the bit line is less than the full supply voltage.

$$P_{SRAMArray} = f \cdot (N_{cols} \cdot C_{BitTot} \cdot V_{BitSwing}^2 + C_{WordTot} \cdot V_{DD}^2) \quad (1.4)$$



In addition, there are two types of the bit lines used in the memory blocks. One is a single-ended bit line structure, and the other is a double-ended bit line structure. The bit lines dissipate most of the power in the memory structure. The power dissipation of the double-ended bit line memory structure is independent of the data activity of the memory cell because the operation of each bit line is complemented regardless of the contents of the SRAM cells.

In case of the single-ended bit line, its power dissipation of the bit lines depends on the values read from the cells. Thus we have to account for its data activity of the bit lines

**5.3.2 Datapath components.** A datapath typically contains ALUs, shifters, multipliers, register files, etc. Although most of datapath components also have very regular structure it is not a trivial problem to estimate the power dissipation because the effective switching capacitance is a complex function of the applied input sequences, and can vary non-linearly with the bit-width of the datapath.

Figure 1.11 represents the flow for the microarchitectural block power estimation. Either, we re-use small datapath component models, e.g., 4-bit, 8-bit, and 16-bit width of transistor- or gate-level existing designs or we generate combinational macro models for the datapath components using an HDL, a design library, and a synthesizer [20]. Second, we measure the average power dissipation for each possible Hamming distance by applying a large number of input sequences that have the same Hamming distance. Finally, we can extrapolate the results with regression-based techniques [17] to obtain the power model for 32-bit or 64-bit width datapath component.

This power modeling methodology has several advantages over other estimation techniques used in earlier power estimators. First, we can consider the effect of the technology parameters directly on the power dissipation because we measure the power consumption of the small block with the power simulator and the technology parameters.

Second, since the datapath components have a very regular structure we can reduce the power model construction time by using regression-based technique. The regularity makes it possible to apply regression-based estimation techniques without losing much accuracy.

Third, the Hamming distance based power estimation is simple to apply to the microarchitectural simulation if one traces the change of the input/output values presented to the blocks (the bus streams) during a simulation run.

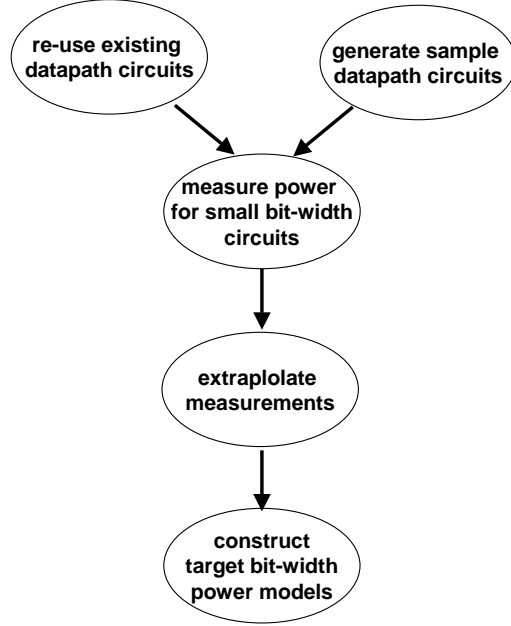


Figure 1.11. Power model construction flow for microarchitectural power estimation.

**5.3.3 Random logic and interconnections.** There is no exact way to predict, in the early design phase, the form of the random logic used for control. However, there are some empirical models based on Rent's rule [15, 16], which rely on parameters such as transistor count, area, the number of the pins, logic depth, etc. These model parameters can be determined based on similar existing designs.

Interconnection information cannot be determined at the microarchitectural design stage, but we can estimate its length by rough floorplaning. Its capacitance can be estimated by:

$$C_{Int} = L_{Int} \cdot W_{Int} \cdot C_{Area} + 2 \cdot (L_{Int} + W_{Int}) \cdot C_{Length} \quad (1.5)$$

In (1.5)  $C_{Int}$  is the interconnection capacitance of the internal input and output buses between the microarchitectural blocks,  $L_{Int}$  is the interconnection length,  $W_{Int}$  is the width of the interconnection line,  $C_{Area}$  is the capacitance per unit area, and  $C_{Length}$  is the capacitance per unit length of the interconnection layer. The estimated interconnect capaci-

tance for the interconnect between each major microarchitectural block is assigned to the data structure associated with each microarchitectural block. This is combined with the simulation statistics of the internal bus transition activity derived from the cycle simulation to estimate the power dissipation of the interconnect bus, see (1.6).

$$P_{Int} = HammingDist \cdot f \cdot C_{Int} \cdot V_{DD}^2 \quad (1.6)$$

**5.3.4 Clock distribution tree.** According to [24] the power dissipation of the clock distribution network can be responsible for 40% of the total power dissipation in high performance microprocessor design. As the chip size and the clock frequency increase the fraction of the power dissipation by the clock distribution network is becoming ever more significant.

There are several clock distribution styles. The most common are the H-tree and the balanced H-tree. The nodes of the clock tree includes all clocked transistors in the microprocessor core logic and memory, as well as the clock wiring and the clock driver. In addition, the clock network is distributed over the entire chip and therefore related to the overall chip area. The total load capacitance for the clock distribution tree contains three components: clock tree wiring, random logic and memory clocked nodes [15]:

$$C_{ClkTot} = C_{ClkWiring} + C_{ClkLogic} + C_{ClkMemory} \quad (1.7)$$

In (1.7)  $C_{ClkWiring}$  is a function of the interconnection capacitance of clock wire per length  $C_{Int/Length}$ , the total area  $A_{Tot}$  of the estimated chip die size, and the number of the levels in the clock distribution tree, which in turn is a function of the target clock skew and chip die size, see (1.8) and Figure 1.12:

$$C_{ClkWiring} = C_{Int/Length} \cdot A_{Tot} \cdot \sum_{i=1}^{N_{tree}} 2^{i-1} \cdot \frac{1}{2^{\lfloor i/2 \rfloor + 1}} \quad (1.8)$$

The effective switching capacitance of the clock distribution tree  $C_{ClkEff}$  includes the switching capacitance of the chain of the inverters forming the clock driver. This can be represented by:

$$C_{ClkEff} = C_{ClkTot} \cdot \left( \frac{1}{1 - 1/a_{clkdriver}} + 1 \right) \quad (1.9)$$

where  $a_{clkdriver}$  is the optimal stage ratio for the clock driver. From (1.7), (1.8), and (1.9) the total power dissipation of H-tree clock distribution network can be represented in terms of  $C_{ClkEff}$  as follows:

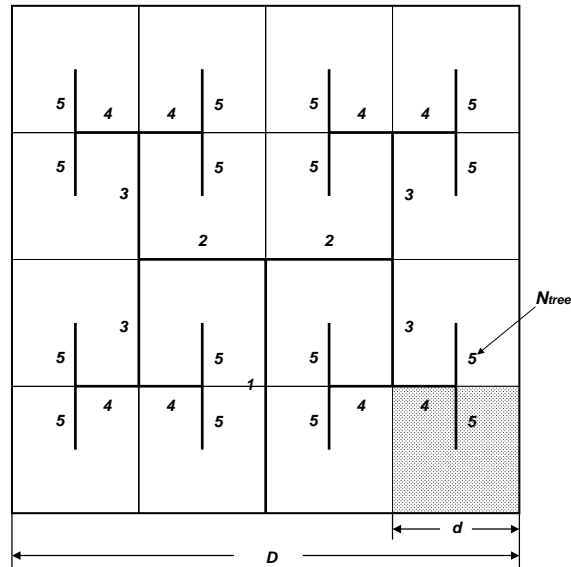


Figure 1.12. An example of H clock distribution network.

$$P_{Clk} = f_{Clk} \cdot C_{ClkEff} \cdot V_{DD}^2 \quad (1.10)$$

## 6. Conclusion and Future Work

Microarchitectural level tools are important for power estimation during the early design phase of microprocessor systems, particularly as we continue to explore new low power optimization techniques. In earlier work we found that recent power estimation tools are inaccurate enough that they may lead designers to make the wrong choices when deciding between several design trade-offs. In this paper we list the potential sources of error that could contribute to this inaccuracy. They are connected with the abstractions that are employed by cycle simulators. We propose a detailed methodology for reinstating some of the details lost in the abstraction.

Several important questions remain. The first is how much the power modeling slows simulation. We have implemented a preliminary version of this power estimation simulator on top of SimpleScalar and have ob-

served slow downs of about a 2x. The second question is how much accuracy is regained by our proposed methodology. This requires calibration against existing chips and remains for future work.

## Acknowledgments

This effort has been sponsored by the Defense Advanced Research Project Agency (DARPA) and Air Force Research Laboratory under contract No. F33615-00-C-1678. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

## References

- [1] Trevor Mudge, "Power: A First-Class Architectural Design Constraint," *IEEE Computer*, Vol 34. No. 4, April 2001.
- [2] G. Cai, and C. H. Lim, "Architectural Level Power/Performance Optimization and Dynamic Power Estimation," *Cool Chips Tutorial* in conjunction with MICRO 32, November 1999.
- [3] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proceedings of 27th International Symposium on Computer Architecture*, May 2000.
- [4] N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye, "Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower," *Proceedings of 27th International Symposium on Computer Architecture*, May 2000.
- [5] S. Ghiasi, and D. Grunwald, "A Comparison of Two Architectural Power Model," *Power Aware Computer Systems Workshop* in conjunction with ASPLOS-IX, November 2000.
- [6] T. M. Conte, M. A. Hirsh, and K. N. Menezes, "Reducing State Loss for Effective Trace Sampling of Superscalar Processors," *Proceedings of International Conference on Computer Design*, October 1996
- [7] A. Sinha, and A. P. Chandrakassan, "JouleTrack - A Web Based Tool for Software Energy Profiling," *Proceedings of 38th Design Automation Conference*, June 2001

- [8] S. Wilton, and N. Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches," Western Research Laboratory Technical Report 93/5, July 1994
- [9] H. Mehta, R. M. Owens, and M. J. Irwin, "Energy Characterization based on Clustering," Proceedings of 33rd Design Automation Conference, June 1996
- [10] P. E. Landman and J. M. Rabaey, "Activity-Sensitive Architectural Power Analysis," IEEE Transaction on CAD of Integrated Circuit and Systems, Vol. 15, No. 6, June 1996
- [11] P. E. Landman and J. M. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," IEEE Transaction on VLSI Systems, Vol. 3, No. 2, June 1995
- [12] G. Jochens, L. Kruse, E. Schmidt, and W. Nebel "A New Parameterizable Power Macro-Model for Datapath Components," Proceedings of Design Automation and Test in Europe Conference 1999
- [13] D. Burger, and T. M. Austin, "The SimpleScalar Tool Set, version 2.0," ACM SIGARCH Computer Architecture News, Vol. 25, No. 3, June 1997
- [14] M. B. Kamble, and K. Ghose, "Analytical Energy Dissipation Model for Low-Power Caches, Proceedings of International Symposium on Low Power Electronics and Design, August, 1997
- [15] B. Geuskens, and K. Rose, "Modeling Microprocessor Performance," Kluwer Academic Publishers, 1988
- [16] Bakoglu, "Circuits, Interconnections, and Packaging for VLSI," Addison Wesley, 1989
- [17] A. Bogliolo, L. Benini, and G. De Micheli, "Regression-based RTL Power Modeling," ACM Transaction on Design Automation of Electronic Systems, Vol. 5, No. 3, July 2000
- [18] L. Benini, A. Bogliolo, E. Macii, M. Poncino, and M. Surmei, "Regression-based RTL Power Models for Controllers," Proceedings of the 10th Great Lakes Symposium on VLSI, March 2000
- [19] S. Theoharis, G. Theodoridis, P. Merakos, and C. Goutis, "Accurate Data path models for fast RT-Level Power estimation," IEE Proceedings of Computers and Digital Techniques, Vol. 147, No. 4, July 2000
- [20] A. Bogliolo, R. Corgnati, and E. Macii, and M. Poncino, "Parameterized RTL Power Models for Combinational Soft Macros," Proceedings of International Conference on Computer-Aided Design, November 1999

- [21] M. M. Khellah, and M. I. Elmasry, "Effective Capacitance Macro-Modeling for Architectural-Level Power Estimation," Proceedings of the 8th Great Lakes Symposium on VLSI, 1998
- [22] C. Zhanping, K. Roy, and E. K. Chong, "Estimation of Power Dissipation Using a Novel Power Macromodeling technique," IEEE Transaction on CAD of Integrated Circuits and Systems, Vol. 19, No. 11, November 2000
- [23] S. Gupta, F. N. Najm, "Power Modeling for High-Level Power Estimation," IEEE Transaction on VLSI Systems, Vol. 8, No. 1, February, 2000
- [24] P. E. Gronowski, W. J. Bowhill, et al., "High Performance Micro-processor Design," IEEE Journal of Solid-State Circuits, Vol. 33, No. 5, May 1998