Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

APPLYING PARALLEL, DYNAMIC-RESOLUTION SIMULATIONS TO ACCELERATE VLSI POWER ESTIMATION

Dhananjai M. Rao

CSA Department
Miami University
Oxford, OH 45056, U.S.A.

Philip A. Wilsey

Dept. of ECECS
University of Cincinnati
Cincinnati, OH 45221–0030, U.S.A.

ABSTRACT

High resolution models of logic circuits need to be used in simulations to accurately track logic transitions or glitches, which contribute to the most dominant portion of VLSI power dissipated. Unfortunately, simulating large, high resolution models is a time consuming task. Although more abstract models that simulate faster can be used, they are less accurate as details of glitching activity are absent. This study proposes an alternatively approach that dynamically (i.e., during simulation) changes the resolution of a model to strike a better balance between accuracy and performance. Simulation-time resolution changes are performed using a novel methodology called Dynamic Component Substitution (DCS). This paper presents the issues involved in applying DCS to accelerate parallel power simulations of digital logic circuits. The experiments indicate that the proposed strategy can increase performance by 3x with negligible deviations in power estimates but consuming about 2x more memory.

1 INTRODUCTION

The steady increase in chip density and operating frequencies has made power consumption a major issue in Very Large Scale Integration (VLSI) devices and their design (Landman 1994). The increasing concern about power consumption have catalyzed the development of several Computer Aided Design (CAD) tools for *power estimation* (PE). A large number of the PE methods are either modifications to simulations or involve simulations in parts of the estimation process (Rao 2003). Simulation-based power estimation or power simulation is one of the earliest, most straightforward, and effective way to track dynamic power dissipation (Landman 1994). Dynamic power dissipation arises from numerous logic transitions ($0 \rightarrow 1$ or $1 \rightarrow 0$) called *glitches* that occur due to time varying sequences of inputs to a logic circuit. Glitches can contribute anywhere from 20% to 70% of the total power (Landman 1994). Consequently, the ability to accurately track glitches makes power simulation an attractive solution for PE and many PE CAD tools primarily focus on this aspect (Rao 2003). However, these simulations need to be conduced using high resolution or gate-level models to accurately track glitches. Unfortunately, such simulations are time consuming even when parallel simulation techniques are employed (Rao 2003). On the other hand, more abstract or behavioral models can be used for PE. The abstract models can be simulated much faster but lack fidelity as the glitching characteristics are lost (Rao 2003).

The aforementioned observations lead to the inference that an ideal scenario would be to simulate using low resolution configuration but using power dissipation from high resolution configurations. Accordingly, this study proposes to dynamically change the resolution of the models used in power simulations to strike a more optimal tradeoff between accuracy and performance. That is, the simulation strives to proceed using abstract sub-circuits but reporting power estimates collated earlier in the simulation from higher resolution configurations. However, if power dissipation for a specific input to a sub-circuit is not known then the sub-circuit is refined, power is estimated, and the sub-circuit abstracted for further simulation. Such a simulation in which the resolution of the model (or parts of the model) is changed during simulation is called Dynamic Resolution Simulation (DRS). Our research utilizes a novel methodology called Dynamic Component Substitution (DCS) (Rao 2003) in conjunction with component-based, multi-resolution models to achieve DRS — i.e., DRS is the concept while DCS is one specific implementation methodology. In other words, DCS is used to abstract and refine the model during simulation to strike a more optimal tradeoff between performance and accuracy of power simulations.

This paper presents the issues involved in applying DCS to accelerate parallel power simulations of digital logic circuits. A brief background on the core topics in this study is presented in Section 2. Section 4 presents some related research activities. An overview of DCS is presented in Section 3. The modeling and simulation environment used in this study is presented in Section 5. Section 6 describes the the proposed DCS-based strategy for PE in detail. The experiments conducted to evaluate the effectiveness of the proposed approach are discussed in Section 7. Section 8 concludes the paper summarizing the results from this research along with pointers to future work.

2 BACKGROUND

The topic of this paper spans the domains of modeling, parallel simulation, and VLSI PE. Accordingly, some brief background information on some of the pertinent topics is presented in this section. Readers are referred to the references for further details. Preliminaries of Time Warp (Rao 2003), the optimistic Parallel Discrete Event Simulation (PDES) synchronization strategy used in this study are presented in Section 2.1. A short review of VLSI PE is presented in Section 2.2.

2.1 Time Warp

This investigation utilizes parallel simulations that are synchronized using a strategy called Time Warp (Rao 2003). Time Warp in an optimistic synchronization mechanism. A Time Warp simulation is organized as a set of loosely coupled, asynchronous Logical Processes (LPs) that interact with each other by exchanging *virtual time* stamped events. Each LP processes its events asynchronously, as fast as it can in time-stamp order incrementing its Local Virtual Time (LVT), generating new events, and periodically saving its state. Due to the asynchronous nature of parallel computing, LPs may receive events in the past (time-stamp of the event is less than the LVT) called straggler events, that result in causal violations. On receiving a straggler event, the LP triggers a rollback mechanism in which the saved states are used to revert back to a time prior to the causal violation, cancel messages sent earlier using anti-messages, and reprocess messages in their correct time stamp order. A periodic garbage collection mechanism based on Global Virtual Time (GVT) is used to prune the state of the LPs. GVT is computed such that the simulation will never rollback below it (Rao 2003). The distributed simulation terminates when all the events in the system have been processed in their correct causal order.

2.2 VLSI Power Dissipation

Power dissipation of a VLSI chip depends on the type of transistor technology used for fabrication. Since the Complementary Metal Oxide Semiconductor (CMOS) technology is the most widely used (Rao 2003), this study uses it as the target technology for PE. The primary power dissipation of a CMOS transistor arises from static and dynamic power consumption (Li, Katkoori, and Mak 2004). Small amounts of leakage current, sub-threshold currents, and substrate injection currents give raise to negligibly small static power dissipation in CMOS transistors (Li, Katkoori, and Mak 2004; Rao 2003). On the other hand, dynamic power dissipation that primarily arises from charging and discharging parasitic capacitances in a CMOS circuit is the most dominant power component (Landman 1994; Li, Katkoori, and Mak 2004; Rao 2003). In other words, dynamic power arises when a CMOS switch transitions or glitches. Most PE CAD tools place heavy emphasis on dynamic power dissipation (Li, Katkoori, and Mak 2004; Rao 2003). Accordingly, the application emphasis of this study is to track transitions ($0 \rightarrow 1$ or $1 \rightarrow 0$) occurring in digital logic models. On detecting logic transitions, the models in this study use a suitable power dissipation factor (obtained from data sheets published by IC manufacturers) to compute power estimates. In practice, the power dissipation factor is set to a predetermined value based on the target CMOS fabrication technology and operating voltage (Rao 2003).

3 DYNAMIC COMPONENT SUBSTITUTION (DCS)

The dynamic-resolution simulations used in this study have been achieved using hierarchical component-based models (Rao 2003) and a novel, domain-independent methodology called Dynamic Component Substitution (DCS) (Rao 2003; Rao, Chernyakhovsky, and Wilsey 2000). DCS enables changes to the resolution of a model by replacing a module (a set of components with well defined interface and functionality) with an *equivalent* component or vice versa. Substituting a module with its equivalent component or vice versa is synonymous to abstracting or refining a model respectively. The equivalent component of a module must satisfy the following criteria: (i) *Interface equivalence*: it must have an interface that is identical to that of the module; and (ii) *Functional equivalence*: its exposed functionality must be the same (or within some acceptable error margin as defined by the modeler) as that of the original module. A more detailed discussion on equivalent components along with notions of functional equivalence are available in the literature (Rao 2003).

For example, consider a digital logic `FullAdder` circuit shown in Figure 1(a). The structural gate-level circuit can be substituted with more abstract, but functionally equivalent counterparts as shown in Figure 1(b) and Figure 1(c). For simulation, each logic gate is modeled as an independent component. The sub-circuits (like `Exclusive-Or`) are modeled as modules by appropriately interconnecting the components. The modules are tagged with their corresponding equivalent components in the model. The overall circuit is then modeled in a hierarchical fashion by suitably interconnecting modules (which can be viewed as an abstract component) and components. As indicated in Figure 1,

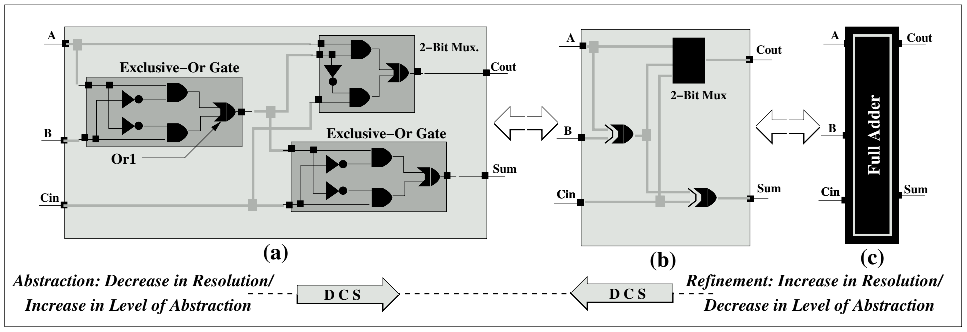

A
B
Cin
Exclusive–Or Gate
Or1
2–Bit Mux.
Cout
Exclusive–Or Gate
Sum
(a)
2–Bit Mux
(b)
Full Adder
(c)
Abstraction: Decrease in Resolution/ Increase in Level of Abstraction
D C S
D C S
Refinement: Increase in Resolution/ Decrease in Level of Abstraction

Figure 1: A Digital Logic FullAdder Circuit at Different Levels of Abstraction

during simulation, DCS is triggered to substitute a module with its equivalent component (or vice versa) to change the resolution of the model, thereby enabling dynamic resolution simulation. Note that, as illustrated in Figure 1, the top level interface (consisting of `A`, `B`, `Cin`, `Sum`, and `Cout`) to the `FullAdder` circuit does not change. DCS also involves appropriately mapping the states of the components participating in the transformation. The specifics regarding implementation of DCS in the simulation environment used in this investigation are discussed in Section 5.

The approaches used for triggering DCS are broadly categorized into *proactive* and *reactive* strategies. Approaches in which DCS transformations are scheduled to occur in the future (with respect to simulation time) are classified as proactive strategies. Proactive strategies are used when the scenarios for triggering DCS are already known during model development or can be identified during simulation. On the other hand, in reactive strategies, DCS transformations are scheduled at the current simulation time or in the past. Reactive strategies may require rolling back the simulation to an earlier time in addition to performing the desired transformation. Proactive strategies are easier to implement but identifying scenarios for proactive triggering can be complicated. On the other hand, triggering reactive DCS is much easier. However, it is more complex to implement and it typically incurs additional overheads during simulation (Rao 2003). Note that a combination of proactive and reactive strategies for abstraction or refinement may be employed in a single model. In this study proactive abstraction and reactive refinement have been utilized. A detailed discussion of the strategy for DCS is presented in Section 6.

4 RELATED RESEARCH

The topic of this paper involves two mature and active areas of research, namely Dynamic-Resolution Simulation (DRS) and VLSI PE. Unfortunately, a detailed literature survey of these broad areas is beyond the scope of this paper. Consequently, only a short survey of very closely related research activities in the aforementioned domains is presented in the following subsections. Readers are directed to the references and literature for further details.

4.1 Dynamic-Resolution Simulation (DRS)

The proposed strategy involves dynamic (i.e., during simulation) changes to the resolution of a model. A simulation in which the resolution of of a model changes dynamically is called a Dynamic Resolution Simulation (DRS). DRS maybe achieved via different techniques using multi-resolution, variable-resolution, or cross-resolution models (Rao 2003). In this study DRS has been achieved using a methodology called Dynamic Component Substitution (DCS). Lee and Fishwick discuss a taxonomy for abstraction of dynamic systems through structural simplifications, behavioral approximations, and data abstraction (Lee and Fishwick 1996). Their study focuses on selecting suitable configurations to represent a dynamic system in different operating conditions. They change the interface and functionality of the system during simulation. Alternatively, in DCS the overall functionality and interface of the model is preserved. Natrajan, Reynolds, and Srinivasan (1997) propose the use of Multiple Resolution Entities (MRE) to achieve DRS. In MRE all desired levels of resolution are active throughout the simulation. Contrarily, in DCS only one level of abstraction is active at any given point in simulation time with explicit concept of proactive and reactive resolution changes. A MRE typically has multiple interfaces at different levels of resolution. Conversely, in DCS only a single interface is used for the units undergoing transformations.

4.2 VLSI Power Simulation

Power simulation is a widely used approach for Power Estimation (PE). VLSI power simulations can be broadly classified based on the level of abstraction of the model used, in to the following categories: algorithmic or behavioral tools, Register Transfer Logic (RTL) or architectural level, and circuit or gate level tools. This study deals with circuit or gate level power estimators. Accordingly, only some of the earlier research in circuit-level power simulation is presented in this section. Readers are referred to the literature and references for more detailed discussions on VLSI power estimation (Rao 2003). Circuit-level refers to that level of abstraction for which the transistor or a logic gate is the most *atomic* element in the model. Many of the power estimation tools and techniques operate at this level of abstraction because accurate estimates can be generated. One of the earliest tools for power estimation at the device-level is SPICE (Nagel 1975) which provides one of the most accurate power estimates. Several power estimators have been developed by applying approximations to the models used by SPICE (Landman 1994). Event driven algorithms and relaxation techniques are used to reduce analysis times in SPICE (Deng 1994). Alternatively, Vanoostende et al. (1993) present the use of a hierarchy of simulators to improve performance. In a recent article, Chinosi et al. present a dynamic spatio-temporal circuit partitioning technique for identifying and selectively simulating sub-circuits for detailed simulation (Chinosi, Zafalon, and Guardiani 1999). Our research uses a combination of high resolution and low resolution model configurations to strike an optimal tradeoff between accuracy and performance. The changes in resolution are performed dynamically, during the course of power simulation an unique aspect that distinguishes the proposed strategy from all of the aforementioned methods.

```
xorGate(2, 1) : HardwareFactory.XorGate ”delay=15ns” {
  ComponentDefinitions {
    andGate(2, 1) : HardwareFactory.AndGate ”delay=5ns”;
    orGate(2, 1) : HardwareFactory.OrGate ”delay=5ns”;
    notGate(1, 1) : HardwareFactory.NotGate ”delay=5ns”;
  }
  ComponentInstantiations {
    notG1 : notGate; notG2 : notGate;
    andG1 : andGate; andG2 : andGate;
    orG1 : orGate;
  }
  Netlists {
    notG1(IN, 1) : xorGate(IN, 1); notG2(IN, 1) : xorGate(IN, 2);
    notG1(OUT, 1) : andG1(IN, 1); andG1(IN, 2) : xorGate(IN, 1);
    notG2(OUT, 1) : andG2(IN, 1); andG2(IN, 2) : xorGate(IN, 2);
    andG1(OUT, 1) : orG1(IN, 1); andG2(OUT, 1) : orG1(IN, 2);
    orG1(OUT, 1) : xorGate(OUT, 1);
  }
}
```

(a) XorGate Model

```
include “XorGate”
include “Mux2”

fullAdder(3, 2) : HardwareFactory.FullAdder ”delay=30ns” {
  ComponentDefinitions { }
  ComponentInstantiations {
    xorG1 : xorGate;
    xorG2 : xorGate;
    mux : mux2;
  }
  Netlists {
    xorG1(IN, 1) : fullAdder(IN, 1); xorG1(IN, 2) : fullAdder(IN, 2);
    mux(IN, 1) : fullAdder(IN, 1); mux(IN, 3) : fullAdder(IN, 3);
    xorG1(OUT, 1): mux(IN, 2) xorG2(IN, 2);
    mux(OUT, 1) : fullAdder(OUT, 1);
    xorG2(IN, 1) : fullAdder(IN, 3);
    xorG2(OUT, 1): fullAdder(OUT, 2);
  }
}
```

(b) FullAdder Model

Figure 2: Hierarchical Description for FullAdder Circuit Shown in Figure 1

5 MODELING & SIMULATION ENVIRONMENT

This section presents only a brief description of WESE, the web-enabled modeling and simulation environment used in this study. An overview of WESE's infrastructure for DCS is also presented. Readers are referred to the literature for detailed descriptions (Rao 2003; Rao and Wilsey 2000; Rao, Wilsey, and Carter 2001). WESE provides a component-based modeling language, a framework for developing a repository of components, and the infrastructure for parallel simulation. Model development in WESE involves two phases. First, a set of components involved in the model are developed using WESE's API and deployed as WESE factories. A WESE factory is a repository of components with additional capability for parallel simulation. In the next phase, models are developed by suitably interconnecting the components using a hierarchical modeling language called the System Specification Language (SSL). Figure 2 shows a snippet of the SSL specification for the `FullAdder` circuit illustrated in Figure 1.

As shown in Figure 2, the specification of a model or a SSL design file consists of a set of interconnected *modules* (such as: `xorGate` in Figure 2(a)). Each module consists of three main sections, namely: (i) the *component definition section* that contains the details of the components to be used to specify a module, including the Universal Resource Locator (URL) of a factory, name of the source object, and parameters; (ii) the *component instantiation section* that defines the various components constituting the module; and (iii) the *netlist section* that defines the inter-connectivity between the various instantiated components. Modules may be nested within each other in a hierarchical fashion by simply using a module as a component definition. In Figure 2(b), this technique is used to build the `fullAdder` module using two `xorGate` modules. As shown in the figure, SSL permits an equivalent component to be associated with each module. In WESE, DCS is performed by replacing the module with its equivalent component or vice versa.

A WESE server performs the task of collaborating with the distributed factories and coordinating the simulations. Once the simulation commences, the simulation sub-systems of each WESE factory (involved in the simulation) handle further processing. The *simulation sub-system* of a WESE factory has been developed using the WARPED simulation kernel. WARPED is an API for a general purpose discrete event simulation kernel with different implementations (Radhakrishnan et al. 1998). WESE utilizes the Time Warp (Radhakrishnan et al. 1998) based simulation kernel of WARPED.

WESE employs an event-driven approach to sequence the various phases involved in DCS. This approach makes the

DCS implementation immune to some of the idiosyncrasies of the underlying Time Warp synchronization mechanism. A component can trigger proactive as well as reactive DCS via appropriate Application Program Interface (API) method calls. Both types of DCS triggers result in scheduling a suitable *kernel* event except that in reactive calls WESE initially triggers an artificial rollback via WARPED API calls. In order to enable artificial rollbacks, the GVT in the simulation is simply throttled to lag by an user-specified time window. The modeler must specify a suitable GVT lag such that that the simulations never rollback below GVT. WESE also provides an API for mapping states of components during DCS. It is the responsibility of the modeler to utilize the API and suitably map states of components during DCS transformations. In WESE, all the components (including equivalent components for each module) at different levels of abstraction are created during simulation startup because WARPED currently does not support creation of LPs during simulation. Note that this purely an aspect of the implementation and is not a requirement for enabling DCS. The additional components are dormant and do not participate in the simulation until they are activated through a DCS transformation. However, in this study the presence of additional (but dormant) components has been exploited to optimize the implementation of the proposed strategy. Once again, this is purely an implementation optimization and is not a conceptual requirement or limitation of DCS-based PE. A more detailed description of WESE, WARPED, and DCS implementation are available in the literature (Radhakrishnan et al. 1998; Rao and Wilsey 2000; Rao, Wilsey, and Carter 2001).

6 DCS STRATEGY FOR POWER ESTIMATION

This study proposes an alternative, DRS-based approach for VLSI PE. The primary assumption underlying the proposed strategy is that abstract models can be simulated faster. This is usually the case for majority of digital logic circuits (with some exceptions — see Rao 2003). Abstract models simulate faster because a number of intermediate components and corresponding event-based interactions are eliminated. Reduction in events has a cascading effect of reducing intermediate events as the inputs propagate to the outputs. Accordingly, the proposed strategy aims to improve performance by simulating using abstract configurations of each module. However, the abstract configurations utilize power estimates collated using higher resolution configurations to ensure accuracy. The changes between higher and lower resolutions are performed as needed using DCS. More specifically, the strategy works as follows:

Step 1: **Initialization**: Initially the simulation starts off at the highest resolution so that glitches can be accurately tracked. The logic transitions for a given input is tracked and power dissipations are computed using a modeler-specified power factor. The power estimates and corresponding input vectors are stored in the state of the component. This value is used by the low resolution configuration to report accurate power dissipation.

As an optimization, in our implementation, each component *instance* reports its power dissipation to its immediate higher level equivalent component, if one has been specified in the model (see Figure 2(a)). For example, the `and`, `or`, and `not` gates constituting the `xorGate` module (see Figure 2(a)) report power dissipation to the `HardwareFactory.XorGate` component. As discussed in Section 5, equivalent components are created during simulation startup — our implementation leverages this fact and collates estimates directly in the equivalent component instances.

***Step 2*: Proactive Abstraction**: An equivalent component gathers power dissipation from each underlying component to determine overall power for the module. New power values are added only if a corresponding input vector entry does not exist in the state of an equivalent component. On the other hand, if an entry already exists in the state then the new estimate is ignored. If new entries are not added for several consecutive input vectors, the equivalent component triggers *proactive abstraction* for the given module. Proactive abstraction is performed by scheduling a DCS transformation to occur prior to starting the next input cycle. Note that different modules may undergo abstraction at different simulation times.

***Step 3*: Reactive Refinement**: When simulating using the equivalent component (low resolution), it uses estimates saved in its state to report power dissipations based on changes to steady state input vectors. However, the estimates for certain input combinations may not be present due to aforementioned proactive abstraction. In such a scenario, the equivalent component triggers *reactive refinement* to obtain the power estimate using high resolution configuration. Reactive refinement causes the simulation to rollback to the beginning of the input cycle, refine the module, and reprocess the inputs to collate power estimates. Note that different module instances undergo reactive refinement at different times in an asynchronous manner.

In order to enable reactive refinement, at least 2 states (except for the initial simulation cycle) from earlier simulation cycles must be available to rollback. Accordingly, the fossil collection rate is throttled through suitable API calls in WARPED. Only states (and events) that are more than 2 input cycles behind GVT are garbage collected. The delay in GVT is specified by the modeler based on the timing characteristics of the model. Noted that state saving is already performed by the Time Warp kernel to recover from rollbacks. The negative impact of delaying fossil collection is the increase in memory usage of the simulations.

The above process of proactive abstraction and reactive refinement is performed until the simulation is complete. As the simulation progresses, the frequency of DCS transformations decreases (see Figure 3(a)) because the number of new input vector combinations encountered by a module decreases. As the frequency of DCS transformations decrease, the simulation proceeds using low resolution configurations, thereby improving performance.

6.1 Source Of Error

The power dissipation of logic circuit is influenced by the glitching activity in the circuit. When sub-circuits are abstracted, they impact the glitching activity of other sub-circuits connected to them, which in turn impacts power dissipated. Depending on the model certain transformations increase glitches causing higher power estimates or vice versa. Consequently, in the proposed DCS-based approach introduces some error in the final power estimates. However, in practice (see Section 7), we find the positive and negative deviations tend to cancel out each other and the overall average errors are reasonably small.

7 EXPERIMENTS

The models used to evaluate the effectiveness of the proposed strategy have been developed using the modeling and simulation infrastructure of WESE. Note that WESE also provides the infrastructure for DCS. In accordance with the model development strategy (see Section 5), a set of fundamental logic components such as: `and`, `or`, `xor` gates have been developed using WESE's object oriented API (Rao 2003). In addition, more abstract components such as `multiplexer` (Rao 2003) and `fulladder` (Rao 2003) have also been developed. All of the common tasks involved in PE have been implemented in a common base class. The base class maintains the power tables for each component and tracks power dissipation. It also performs the task of triggering proactive abstraction or reactive refinement appropriately. Each logic component customizes the functionality of the base class just to provide the input to output transformation. Suitable timestamped events are used by the components to generate outputs. In addition to the logic components, testing components such as test vector generators and output displays have also been developed for experimentation. The logic components have been bundled into a WESE factory and deployed on the workstations used for parallel simulation.

Utilizing the components in the aforementioned WESE factory, several logic circuits have been developed using SSL. The SSL design also included appropriate higher level components for use during DCS. Some of the salient characteristics of the models are tabulated in Table 1. The column titled `Gates` and `Abstract` (in Table 1) indicates the number of basic gates and number of higher level components in a model. For example, the `32-bit Adder` (see Table 1) consists of 32 cascading full adder modules interconnected in a hierarchical fashion. Each full adder contains 2 exclusive-or gate modules which in turn contains basic gates. Each full adder and exclusive-or modules have an abstract equivalent component specified which gives raise to $(32+(32*2))$=96 abstract components as indicated in Table 1. The `32-bit ALU` and `32-bit Mul`tiplier models have been developed in a similar fashion. The multiplier model is composed in the form of a grid which gives raise to numerous rollbacks and a large number of glitches. The characteristics of the model make it a "stress test" for parallel simulation and the proposed strategy. The `pASIC` model (proposed by Navabi et al. (Navabi 1995)) is a very asymmetric model and is relatively flat with just 3 levels of hierarchy. It has very few components (such as 2x2 multipliers and 3-input `and` gates) to which DCS could be applied and can be viewed as a worst case scenario model.

The aforementioned DCS-based models were simulated using WESE on a varying number of workstations. The components were randomly partitioned onto the various workstations. Each workstation used in this study consisted of 2 Athlon MP processors (2.0 GHz comparable) with 1 GB of main memory running Linux. The workstations were networked using gigabit Ethernet. Note that immaterial of the number of workstations used, the models were exercised using the same set of unique input vectors. Consequently, the power dissipation and DCS behavior of the models are identical in all cases. The unique input vectors fall under the category of Uniform White Nose (UWN) model that is used by other researchers as well (Landman 1994). The power dissipation of the models with and without DCS is shown in Table 2. As shown in the table the deviation in power estimates is reasonably small. The 32-bit multiplier (`32-bit Mul`) shows more prominent deviation due to significant glitching effects inherently present in the model. DCS transformations influence glitching and therefore introduce deviations in the estimates. Our observations suggest that adding some additional model specific knowledge to the DCS triggering process could reduce the deviation. We are also exploring approaches for tracking possible deviations due to DCS and adjusting the PE using some post-simulation processing.

A trend of typical DCS transformations in the model is illustrated in Figure 3(a) using the DCS activity for the `32-bit Adder` model. The graph illustrates the number of active components at various simulation times. The active components decrease whenever abstractions occur in the model. On the other hand the number of components increases when refinement occurs. As indicated in the trend, the model initially undergoes a number of DCS transformations because new input vectors are encountered by different `FullAdder` modules in the model. Since power dissipation estimates for new vectors are not yet available, the modules undergo reactive refinement followed by proac-

Table 1: Characteristics of the Digital Logic Models

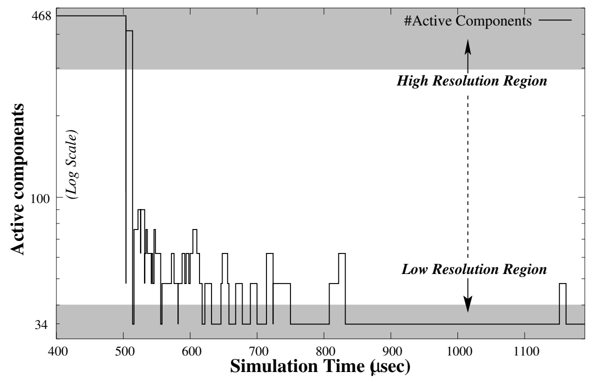
Model Name	Number of components				Number of Test Vectors
	Gates	Abstract	Others	Total	
32-bit Adder	482	96	2	580	20,000
32-bit ALU	642	128	2	772	20,000
32-bit Mul	16360	3096	2	19458	1,200
pASIC	825	25	2	852	1,000

tive abstraction. However, as the simulation progresses, the power estimates for various inputs are already available and fewer refinements are needed. Note that the `FullAdder` module constituting the `32-bit Adder` has only 3 inputs and therefore experiences only 8 unique input combinations. Consequently, although the primary inputs are unique, the `FullAdder` modules experience a repeating set of inputs after sufficient number of inputs have been processed and DCS frequency decreases. The peak memory consumption of the simulations, with and without DCS, are shown in Figure 3(b). The peak memory consumption of the simulations involving DCS are higher because GVT updating had to be delayed to enable reactive refinement as described in Section 6. The memory usage is higher due to the states and events that are maintained in memory to enable rollbacks to earlier simulation time. When additional processors (or workstations) are used for simulation, the overall memory consumption gets distributed across the workstations. Consequently, as shown in Figure 3(a), the peak memory usage tappers-off.

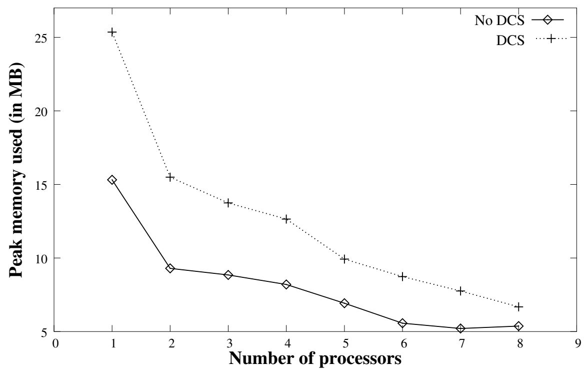
Table 2: Error in PE Due to DCS

Model	Power Dissipation (μW)		Error %
	No DCS	With DCS	
32-bit Adder	34199.66	34009.09	0.5%
32-bit ALU	45276.11	45320.0	0.096%
32-bit Mul	264935.04	255874.26	3.42%
pASIC	2173.2	2168.65	0.2%

The time taken for simulating the models without any DCS transformations is shown in Figure 4(a). Detailed statistics are not presented due to their large volume and readers are referred to the literature for further details (Rao 2003). The times are average of 10 simulation runs. The simulations using 1 processor, highlighted in Figure 4(a) were sequential simulations. As shown in the figure, in the case of the multiplier model, using 2 processors increases CPU time due to introduction of parallel simulation overheads such as state saving and rollbacks. However, as the number of processors are increased the overheads are superseded by the advantages of parallel simulations and the simulation time steadily decreases. In the case of the 32-bit ALU the model immediately begins to benefit from additional processors. However, the performance improvements tapper off because the communication overheads begin to dominate the computational overheads of the model. A similar trend is observed in the 32-bit Adder model except in 3 processor case where the random partitioning resulted in a configuration that gave raise to many rollbacks. In the case of pASIC model, the model had many loops in the model giving raise to numerous rollbacks. Consequently, as shown in Figure 4(a), the pASIC model does not gain from parallel simulation.

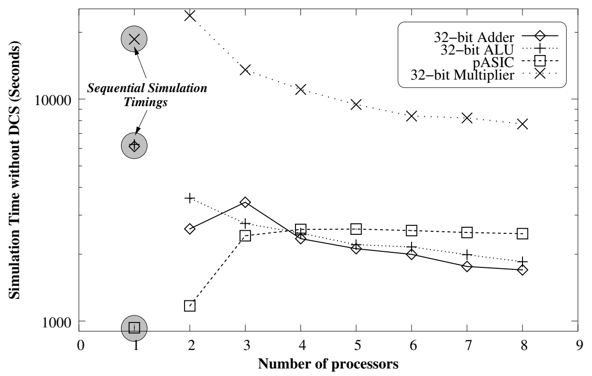


(a) DCS Trend

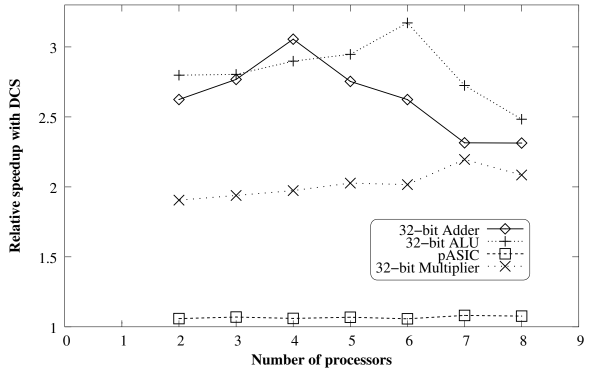


(b) Peak Memory Usage

Figure 3: Simulation Characteristics of 32-bit Adder



(a) Simulation Time without DCS



(b) Speedup Due to DCS

Figure 4: Simulation Time & Relative Speedup

The relative performance improvement accrued by employing the proposed DCS-based strategy is shown in Figure 4(b). The relative speedups (Patterson and Hennessy 2005) shown in Figure 4(b) have been computed by dividing the simulation time without DCS by the simulation time with DCS. As indicated by the graphs, all of the models benefit from using DCS. The gains in the Adder and ALU models are pronounced because the frequency of DCS transformations significantly reduced as the simulation progressed. On the other hand the multiplier model continued to exhibit a higher frequency of DCS transformations through the simulation. This is primarily because the number of input vectors used were fewer. The number of test vectors for the multiplier was kept small to ensure that the sequential simulations completed in reasonable time frames. In the case of pASIC model, the number of components undergoing DCS was small. Consequently, the gains from applying DCS is not very high. As indicated by the experiments the proposed strategy provides good performance improvements even in parallel simulations. However, it incurs additional memory overheads (about 2 times). Furthermore, the deviations in estimates is less than 3.5% for the models and lies within the acceptable error margins used in this field.

8 CONCLUSION

Power simulations are widely used to obtain power estimates for logic circuits and is an important aspect of VLSI design. The simulations need to be conducted using high resolution (or gate level) models in order to accurately track logic transitions (or glitches) that constitute significant part of VLSI power. Unfortunately, power simulations involving high resolution models is a time consuming task, even when parallel simulation techniques are employed. On the other hand, the simulations can be conducted using lower resolution (or more abstract) models. However, low resolution models do not yield accurate estimates as glitching effects are lost. This paper presented an alternative approach which aimed to leverage the accuracy of high resolution models and the performance of low resolution (or abstract) models. More specifically, the strategy aggressively strives to simulate using abstract model configurations but utilize estimates obtained using high resolution models for a given input transition. If the power estimate for a given input transition is not known, the strategy uses the high resolution configuration to obtain accurate estimate. Once the estimate is known the model is reverted back to the abstract configuration to simulate rapidly.

Changes to the resolution of a model was performed dynamically (i.e., during simulation) to adapt to the needs of PE. Such a simulation, in which the resolution of the model is changed dynamically is called Dynamic Resolution Simulation (DRS). In this study, DRS was achieved using a novel methodology called Dynamic Component Substitution (DCS). DCS has been used to pro-actively abstract the model to improve performance of PE. However, if the PE for a specific input transition is not known, the model is reactively refined and simulated to obtain accurate power estimates. Reactive refinement involves rolling-back the model and reprocessing the inputs. The paper presented observations and inferences drawn from experiments conducted to evaluate the effectiveness of the proposed DCS-based approach. The experiments indicate that proposed strategy provides notable performance improvements (1 to 3 times faster) by trading off some (about 0.5% to 3.5%) accuracy and incurring additional (about 2 times) memory overheads. The loss in accuracy occurs due to difference in glitching activity as part of the models are asynchronously abstracted and refined. Note that the loss is accuracy is well within acceptable margins when compared other PE techniques that typically incur about 10% to 15% errors. Research is underway to track and further minimize the deviations in the estimates. The memory overheads in the simulations arise from the need to save state to enable reactive refinement. However, state saving is not a significant overhead because it is already performed as part of the Time Warp synchronization mechanism used in this study. The inferences drawn from this study indicate that parallel, DRS provide an unique avenue to accelerate VLSI power simulations as well as other simulation-based PE techniques.

ACKNOWLEDGMENTS

Support for this work was provided in part by the Ohio Board of Regents.

REFERENCES

Chinosi, M., R. Zafalon, and C. Guardiani. 1999, June. Parallel mixed-level power simulation based on spatio-temporal circuit partitioning. In *Proceedings of the 36th ACM/IEEE conference on Design Automation Conference (DAC'99)*, 562–567.

Deng, A. 1994, April. Power analysis for cmos/bicmos circuits. In *1994 International Workshop on Low Power Design*, 3–8.

Landman, P. E. 1994. *Low-power architectural design methodologies*. Ph. D. thesis, University of California at Berkely.

Lee, K. S., and P. A. Fishwick. 1996, December. Dynamic model abstraction. In *Proceedings of the 1996 Winter Simulation Conference*, 764–771.

Li, H., S. Katkoori, and W.-K. Mak. 2004. Power minimization algorithms for lut-based fpga technology mapping. *ACM Trans. Des. Autom. Electron. Syst.* 9 (1): 33–51.

Nagel, L. W. 1975. Spice2: A computer program to simulate semiconductor circuits. Technical report, University of California at Berkeley. Technical Report ERL-M520.

Natrajan, A., P. F. Reynolds, and S. Srinivasan. 1997, June. MRE: A flexible approach to multi-resolution modeling. In *Proceedings of the 11th Workshop on Parallel and Distributed Simulation (PADS'97)*, 156–163.

Navabi, Z. 1995. Exemplar synthesizable pasic controller model. (available online).

Patterson, D. A., and J. L. Hennessy. 2005. *Computer organization and design: The hardware / software interface*. 3 ed. San Francisco, CA 94111: Elsevier Inc.

Radhakrishnan, R., D. E. Martin, M. Chetlur, D. M. Rao, and P. A. Wilsey. 1998, December. An Object-Oriented Time Warp Simulation Kernel. In *Proceedings of the International Symposium on Computing in Object-Oriented Parallel Environments (ISCOPE'98)*, ed. D. Caromel, R. R. Oldehoeft, and M. Tholburn, Volume LNCS 1505, 13–23. Springer-Verlag.

Rao, D. M. 2003. *Study of dynamic component substitution*. Ph. D. thesis, University of Cincinnati.

Rao, D. M., V. Chernyakhovsky, and P. A. Wilsey. 2000, January. WESE: A Web-based Environment for Systems Engineering. In *2000 International Conference On Web-Based Modelling & Simulation (WebSim'2000)*. Society for Computer Simulation.

Rao, D. M., and P. A. Wilsey. 2000, December. Dynamic component substitution in web-based simulation. In *Proceedings of the 2000 Winter Simulation Conference (WSC'2000)*. Society for Computer Simulation.

Rao, D. M., P. A. Wilsey, and H. W. Carter. 2001, April. Optimizing costs of web-based modeling and simulation. In *Proceedings of the First International Workshop on Internet Computing and E-Commerce (ICEC'01)*. IPDPS.

Vanoostende, P., P. Six, J. Vandewalle, and H. J. DeMan. 1993, January. Estimation of typical power of synchronous cmos circuits using a hierarchy of simulators. *IEEE journal on Solid-State Circuits* 28 (1): 26–39.

AUTHOR BIOGRAPHIES

DHANANJAI M. RAO is currently a visiting faculty member in the Computer Science & Systems Analysis (CSA) Department, Miami University, Oxford, Ohio, USA. He received his Ph.D. and Masters in Computer Science and Engineering from University of Cincinnati in 2003 and 2000 respectively. His research interests include parallel simulation, distributed computing, and web-based simulation. He research also includes applying simulations to various domains, such as: digital logic simulations, conventional and active networks, ATM networks, mobile ad hoc networks, and Epidemiology. His web address is `<http://www.users.muohio.edu/raodm>`.

PHILIP A. WILSEY is an Professor in the Department of Electrical & Computer Engineering and Computer Science at the University of Cincinnati. He received PhD and MS degrees in Computer Science from the University of Louisiana at Lafayette and a BS degree in Mathematics from Illinois State University. His current research interests are parallel and distributed processing, parallel discrete event driven simulation, computer aided design, formal methods and design verification, and computer architecture. He is a senior member of the IEEE and is a member of the IEEE Computer Society and the ACM. His web address is `<http://www.ececs.uc.edu/~paw>`.