VLSI DESIGN
1996, Vol. 4, No. 2, pp. 107–118
Reprints available directly from the publisher
Photocopying permitted by license only

An Integrated Hardware Array for Very High Speed Logic Simulation

E. SCOTT FEHR[a], STEPHEN A. SZYGENDA[b], and GRANVILLE E. OTT[a]

[a]*Applied Research Laboratories, The University of Texas at Austin, USA;* [b]*Electrical and Computer Engineering, The University of Texas at Austin, USA*

A hardware architecture is proposed which allows direct mapping of design simulation topology onto an acceleration platform. In order to clarify architectural principles, the simulation is confined to functional verification of unit delay, binary valued gate level logic designs. Under this approach, a rank ordered design description is executed on a massively parallel processor grid which implements an efficient and direct model of the design, similar to prototyping. Architectural innovation reduces logic complexity and execution time of boolean evaluation and fanout switching circuits, while large scale parallelism is integrated at die level to reduce cost and communication delays. The results of this research form the basis for a multiple order of magnitude improvement in reported state-of-the-art cost-performance merit for hardware gate level simulation accelerators.

1. INTRODUCTION

The increasing densities provided by VLSI technology have created a demand for CAD systems capable of quickly simulating and verifying large, complex designs. Previously, accelerator development efforts have evolved as either software algorithms implemented on general purpose computers or as special purpose hardware accelerators designed to optimize certain simulation algorithms [1–9]. This research establishes proof of concept for speed-up in a simulator architecture that matches digital designs rather than simulation style algorithms. A multiple order of magnitude cost-performance advantage over previous work is gained by the combination of architectural innovation and scalable technology-based integration. The architectural innovation reduces logic complexity and execution time of boolean evaluation and communication switching circuits, while large scale parallelism is integrated at die level to reduce cost and communication delays. The hardware architecture permits direct mapping of a rank ordered gate netlist onto the acceleration platform [10,11]. Assignment of a single netlist gate per processing element (PE), with fanout propagated by the interconnection network, simulates gate level design concurrency.

The key features of the architecture that provide the performance advantage are: 1) simple logic is used for the gate evaluation with a minimum number of transistors, providing a 100X reduction in cost; 2) the logic is streamlined using latches in push-pull fashion on both phases of the clock, allowing data processing to flow four times faster than a typical ALU for the same technology; and 3) a low cost multiple channel communication grid using transfer gates eliminates the primary bottleneck for gate level CAD, providing a 10X or more speed improvement.

In order to simplify design issues for purposes of architectural clarity, the simulation is confined to functional verification of unit delay, binary valued, synchronous, gate level logic designs. In order to match the hardware architecture, the simulation netlist is converted into equivalent four-input gates during preprocessing.

Related work has been done by Kravitz, et al [12], which identifies factors contributing to the performance of massively parallel simulators for large VLSI circuits. Although the subject of this work was switch level simulation, their analysis of the rank ordering and execution of the Boolean behavioral model created by COSMOS provides important results which are applicable to the execution model for our research. Kravitz's work was targeted to the Connection Machine [13], a general purpose parallel architecture with a dynamically scheduled message routing network not optimal for logic simulation. Their recognition that the communication pattern for this type of simulation "is static and could be supported with a much faster and simpler communication network" underscores the value of one of the principal architectural contributions of our work.

2. ARCHITECTURE OF THE ACCELERATOR

The accelerator platform provides concurrency by using scalable technology based integration to implement massive parallelism in a grid-connected array of PEs. Since the highest level of design granularity addressed in this research is the gate, placing a single netlist gate per PE is, in principle, an appropriate mapping to implement gate level design concurrency. In this architecture, the design execution state for an input stimulus at any point in simulation time is fully represented in concurrently executing parallel elements, so that the need for state-saving mechanisms is reduced. The resulting architecture executes an efficient and direct model of the design on the grid, providing benefits similar to prototyping.

The fundamental building block chip, designed for minimum cost and maximum speed, contains 256 PE cells arranged as two push-pull rows of 128 PEs each. This two row configuration is adequate to execute single stimulus patterns at full speed. Sequences of stimulus patterns can be executed in pipeline fashion by replicating the fundamental chip so that the number of rows is as great as the depth of the rank ordered netlist block to be executed. The active circuitry in the PE is designed for implementation in custom CMOS logic, which allows all fanin configuration switching and Boolean function evaluation for a logic rank to be done in one clock cycle. Fanout propagation between chips on a board is done at one clock per circuit rank, while wire delay between boards incurs an additional time step penalty.

The grid interconnect topology, shown in Figure 1, is designed to efficiently route arbitrary fanout patterns between successive ranks of a rank ordered design, including feedback paths. Fanin/fanout signals are routed over non-directional nearest neighbor interconnect wires. Each PE/gate output value is propagated between successive grid rows by a single wire. The lateral interconnection between adjacent PEs is composed of four bundled single-bit wires to correspond to the simulation netlist requirements of four fanins per gate, and therefore an average of four fanouts (although more are allowed by gating an out-

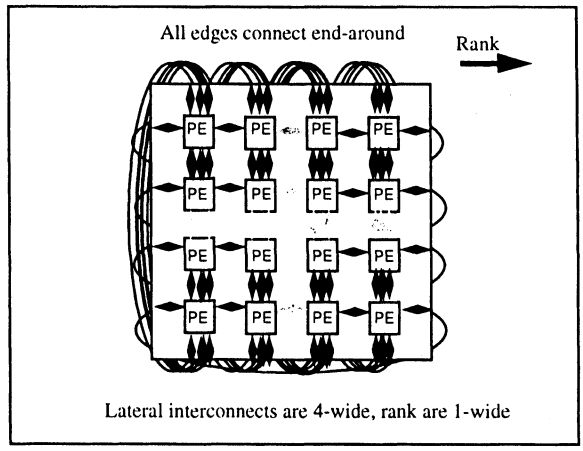

FIGURE 1 Grid interconnect topology

put along a single lateral channel to multiple inputs, or by expansion the of ranks). Wires at the grid edges wrap around to the opposite edge in both dimensions, forming a doubly connected mesh.

2.1. PE Cell

The PE cell logic block, Figure 2, includes the Boolean Evaluation Unit (BEU), I/O Crossbar, Node Descriptor Memory, and Output Latch. Each PE cell implements the communications switching function of the grid by allowing any one-to-one input-to-output pair mapping. This allows all routing permutations for each signal at a node. In addition, all one-to-many input-to-output mappings are possible, which enables fanout spreading.

I/O crossbar switch Figure 3 details the I/O crosspoint circuit which uses single MOS pass transistors to control the input to output signal mapping. Control of the crossbar requires three sets of four switches. The input function of the crossbar implements routing of nearest neighbor signals to the BEU. Along the lateral axis, the route can either pass through the node or switch onto the four BEU gate inputs. Along the rank axis, the crossbar will allow a signal to either pass through or switch onto the lateral axis; however, pass through should not occur during simulation, since the rank ordering algorithm will force a gate at each rank along a path, as explained in the discussion of netlist pre-processing and the example in section 3.2.

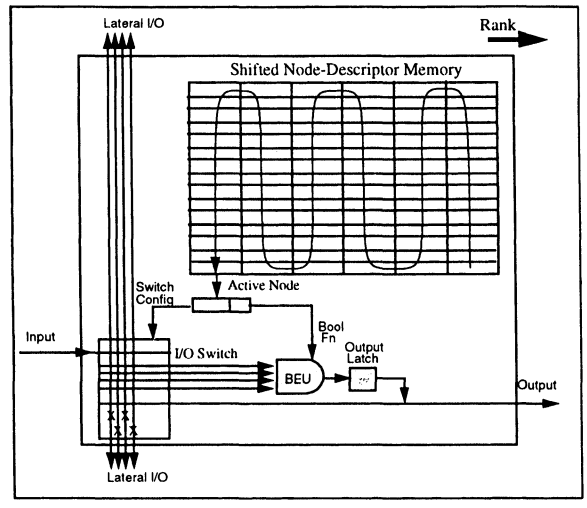

FIGURE 2 PE cell block diagram

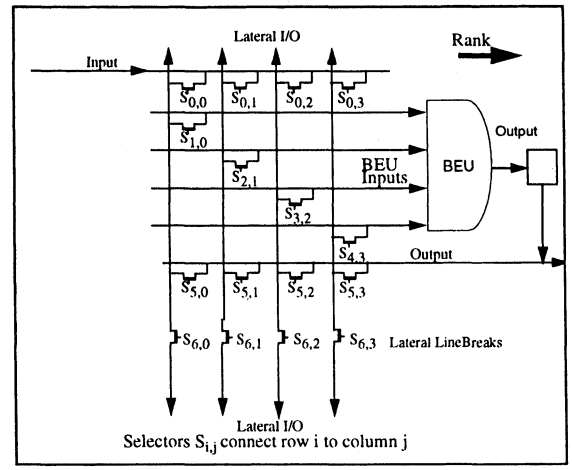

FIGURE 3 I/O crossbar switch

Boolean evaluation unit The BEU block of the PE, shown in Figure 3 and detailed in Figures 4 and 5, is implemented as a PLA which generates a preselected boolean function of four inputs. The node inputs are mapped to the BEU inputs by the crossbar switch. It is the responsibility of the netlist preprocessor to resolve potential conflicts in the assignment of BEU input lines to gate inputs. The output function of the crossbar implements routing of the BEU output latch value to any of the nearest neighbor interconnect wires, or back into the BEU input for the next evaluation cycle.

The basic operation of the BEU is to evaluate the designated function of the four selected signal values from the node inputs, and propagate a one bit result to the Output Latch. In its current design, the BEU is capable of performing sixteen boolean operations on four binary operands. Sixteen is considered an adequate number of boolean functions for commercial applications, although up to $(2^4)^4$ functions are pos-

sible for custom applications. Functions of fewer than four variables can be implemented by tying-off unspecified inputs such that the function is not altered. For example, the function XYZ has a 4-variable form XYZZ. This tie-off capability can be implemented by placing switches between the three adjacent pairings of BEU inputs which are configured by the preprocessor; i.e., W-to-X, X-to-Y, Y-to-Z. These three switches require three additional control bits in the node descriptor.

Node stacking memory Each PE has a local memory which contains the stack of netlist node descriptors assigned to that PE for evaluation. The stacking memory cell count is based on the predicted maximum number of ranks for commercial combinational nets. The number of netlist ranks required for largest anticipated commercial circuits is predicted to be less than 64. For an accelerator platform organized as two push-pull rows of processors, a net of 64 ranks can be simulated by two rows of processors, each capable of stacking 32 netlist nodes. Therefore, a stacking capacity of 32, or 2^5, nodes will be allocated for the accelerator PE memory. At execution time, each netlist node requires 23 bits derived as follows: gate function (4), rank I/O switches (4), BEU input switch (4), lateral I/O wire breaks (4), and tie-off (3). For an estimate of the stacking memory requirement, the 23 bit node descriptor is rounded up to 32 bits. Since each PE has the capacity to stack 32 descriptors, the resulting stacking memory requirement estimate is 2^{10} bits.

3. INTEGRATION OF ACCELERATOR DESIGN

Based on current and near-term commercial integration levels, an estimate of integration density and subsequently the total netlist node capacity for the mesh can be projected. First, a device count for the PE cell will be estimated. The cell includes two functional classes of logic: (1) active signal path logic, and (2) passive node stacking memory. The active logic block is fixed in size, while the stacking memory block is scalable in terms of the number of 23 bit node descriptors currently set at 32. Next, the number of PE cells that can be integrated onto a die is calculated based on commercial densities. Finally, the raw netlist node capacity of the mesh is calculated based on a projection of the number of chips which can be clustered per multi-chip module (MCM) or circuit board, and the number of boards per rack.

In order to compare the projected raw gate capacity to published capacities for other accelerators, an equivalence figure of merit is developed and then applied to the raw value to obtain an equivalent gate capacity value. In order to project from the number of devices per PE cell to the number of PE cells which can be integrated onto a die, an extrapolation is made from current and near term commercial integration levels. Current gate array technology provides a range of $2^{17-2^{20}}$ devices/die with higher densities eminent, so the 2^{20} figure is used.

Fabrication of the PE cell can be based on one of three design techniques for memory cells, listed in order of increasing density and denoting a MOS device by T: (1) gate array @ 20T/cell, (2) semi-custom @ 6T/cell, or (3) full custom @ 4T/cell. A fourth option, super custom or smart DRAM @ 1T/cell is a developing technology which is mentioned in passing, but not assumed for the PE cell. Since the PE is dominated by memory cells, the advantage of memory cell customizing is significant. Adopting custom design for the PE cell memory provides a factor of 4T/20T per cell advantage over generic gate array technology.

The active signal path logic for the PE cell includes the fanin/fanout configuration switch and horizontal link break switches, the BEU evaluation PLA, and the output latch. The fanin/fanout switch in Figure 3 utilizes PLA logic requiring 16T: 4 to switch the 4 lateral input/output wires onto the rank propagation wire, 4 to switch the lateral wires onto the BEU inputs, 4 to switch the BEU output back onto the lateral wires, and 4 to break the lateral paths. The BEU, Figures 4 and 5, requires about 250 or $\sim 2^8$T total for the evaluation and select modules: 4T for each of the 16 minterms, 4T for each of the 8 input

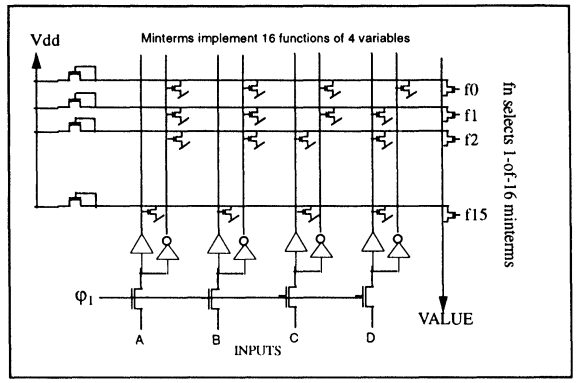

FIGURE 4 BEU function evaluation circuit

signal buffers, 4T for the input switches, 4T for the minterm pullups, and 16T for the minterm selector switch reaching a total of 120T for the evaluation circuit, and approximately the same number for the select circuit. The output latch, Figure 6, requires 6T for the buffers and phase gates. The passive node stacking memory is composed of 32 4T SRAM cells per node descriptor. For a 32 descriptor stack and the current-descriptor latch, $(32 \times 4 \times (32 + 1))T = \sim 2^{12}$, so that the complete PE cell requires $\sim (28 + 2^{12})T = \sim 2^{12}T$.

The total PE cell count per system can now be projected. Adopting 2^{20}T/die and 2^{12}T/PE cell allows $2^{20\text{-}12} = 2^8$ PE/die. Since there are currently 2^5 netlist node descriptors per PE cell memory block, the total number of netlist nodes/die = $2^8 \times 2^5 = 2^{13}$. Assuming 2^5 die/MCM-board and 2^4 MCM-boards/rack yields $(2^{13} \times 2^5 \times 2^4) = 2^{22}$ netlist nodes/rack. A two-rack chassis will allow a 2x capacity increase

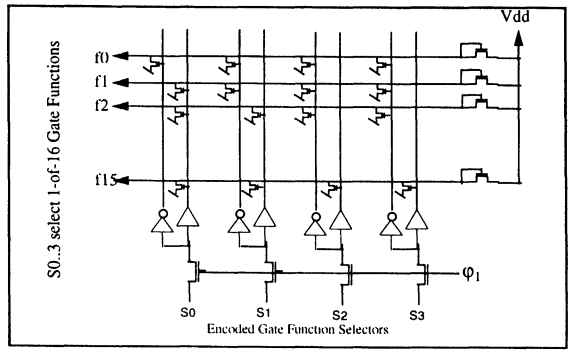

FIGURE 5 BEU function select circuit

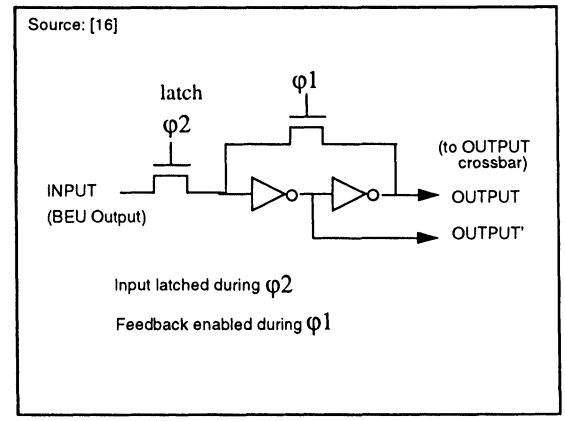

FIGURE 6 BEU output latch/register

and not increase maximum fanout length over a single rack, since the two racks allow a square backplane configuration. For the two-rack chassis, the total raw capacity of the accelerator is $(2 \times 2^{22}) = 2^{23}$, or 8 million, netlist nodes.

An equivalency factor is needed to compare the gate capacity of the proposed platform to other reported work. Two equivalency factors are used by Blank [14], percent of gates predicted to exhibit change of value, or active gates, and number of two-input gates per accelerator primitive element. Following Blank and other reported figures, the active/effective gate percentage will be adopted at 15% or $\sim 2^{-3}$. For the proposed accelerator, the primitive element is a four-input gate. All gates in a design do not require 4-input forms; i.e., any 2-input gates will be represented by a tied-off 4-input gate in the simulated net. Also, every 4-input gate can potentially represent three 2-input gates combining to create a 4-input function. This effect will be approximated by following the factor used by Blank, so a rough equivalency of 2.5 two-input gates per four-input gate is adopted, or roughly a more conservative value of ~2 two-input/four-input gates.

A third equivalency factor, redundancy, is required for this research to account for platform resources allotted to redundant gates inserted during rank ordering. A redundancy value of 20%, is assumed, which leaves a non-redundant residue of 83% original gate volume, so that the effect of redundancy will be set

to ~1. Summarizing, a final equivalence factor of $(2^{-3} \times 2^{1} \times 1) = 2^{-2}$ is obtained. Adjusting the raw netlist node chassis capacity of 2^{23} by the equivalence factor of 2^{-2}, an equivalent gate capacity of $2^{23} \times 2^{-2} = 2^{21} = 2$ million gates for a dual-rack chassis is derived.

3.1. Technology Scaling

Current technology minimum feature size is taken at 0.6μ for purposes of the integration level estimates in this research. The predicted effect of technology scaling on PE/die integration density is shown in Figure 7. As feature size scales from the current 0.6 to 0.3, it should be expected that PE/die integration density should scale from the current ~2^8 to 2^{10}. The potential for PE per die density increases as process advances move feature size toward the technology minimum of 0.25μ. The transit time, gate capacitance and drain-to-source current of every device in the system are assumed to scale proportionally; however, this linear scaling approximation may not hold below 0.5μ. A system's clock period is proportional to the τ of its smallest devices, and so becomes proportionally smaller

3.2. Cost-Performance Analysis

Cost-performance measurement of the simulation architecture is based on the trade-off between the amount of hardware, or cost, committed to execute the simulation versus the performance achieved. Performance of a compiled design simulation on the platform can be predicted as a function of two static characteristics of the design netlist: (1) dimensions relative to platform dimensions, and (2) netlist fanout density. Cost of a simulation platform is the amount of hardware and the associated dollar value required to achieve a desired (and available) performance level for a netlist size. If the size of the netlist exceeds the capacity of the platform after folding, then the platform is fully loaded with the partial netlist,

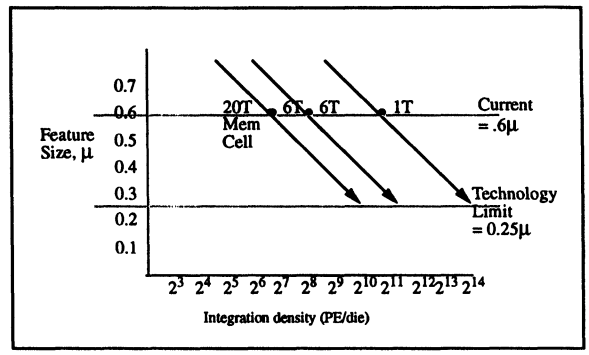

FIGURE 7 Effect of technology scaling on PE/die integration density.

and high speed DMA channels are used to swap the unloaded netlist to/from storage in order to sustain maximum execution rate. Netlist ranks in excess of platform dimensions can be folded over the platform such that multiple ranks can be stacked on a single platform row up to the node stacking capacity of the platform cell.

3.3. Netlist Pre-Processing

Netlist pre-processing, or compilation, performs four vital functions in preparing the raw design netlist for simulation on the accelerator: 1) rank order the resulting design netlist; 2) apply redundancy algorithms and fanout length delay penalties; 3) determine the switching patterns needed to propagate fanout; and 4) map the netlist onto the accelerator platform. Other efforts extending beyond the scope of this research are investigating performance optimizing techniques applied during compilation with hopeful results [15].

Rank ordering can be summarized as follows: If $l_k..l_l$ are design ranks, a linear string $S_j[k..l]$ of nodes identified and grouped during rank ordering can be mapped onto a sequence of l-k+1 platform rows at load time. During execution, members of S_j are evaluated in monotonic increasing order. Since string S_j is an ordering which is a subset of the design precedence graph generated during rank ordering, each l_j is dependent only on inputs generated at nodes $l_{i<j}$. Each node of the sequence of can be evaluated with-

out delay at its execution timepoint since the dependency requirements for fanins to l_j will have been satisfied.

Redundancy algorithms are used to fill out forward pass-through fanout paths, primary input paths and feed-back paths, and insert additional netlist ranks to relieve horizontal fanout saturation when it is detected. Pass-through fanout, primary input, and feedback paths must be represented by a gate at every rank in order to ensure that their values are explicitly represented at every simulation execution cycle. This potential requirement for redundant gates does not in principle effect execution speed, since synchronous logic nets do not evaluate gate inputs until the gate is a member of the currently executing netlist rank; however, insertion of redundant gates may widen the original netlist. Delay penalties based on fanout route lengths are discussed under System Timing, and include delays due to on-chip route length, chip-to-chip crossing, and board-to-board crossing.

Since the complete netlist execution state at a point in simulation time is represented in the values of the node outputs at the current rank, no auxiliary state representation mechanisms are needed. This reduces complexity and saves the cost of state saving logic.

An example of a rank ordered combinational net mapped onto a platform array is given in Figure 8 (a generic gate symbol is used to represent a PE in the array). Redundant gates, designated by (R), are required for primary inputs when entering beyond rank 0, and fanouts which pass through one or more ranks. Redundant rows/ranks are required to split gates with more than four inputs, or for a net width greater than the platform width. Primary inputs X11, X12, X13, and X14 all enter beyond rank 0 in the original net, which require redundant gates at each rank back to rank 0 to represent the signal state at each rank during simulation. The five inputs to gate I require a split to one 4 input gate and one 2 input gate, forcing the addition of rank 3. After the design ranks are loaded in order onto sequential platform rows, each rank is evaluated by concurrent execution of all processors in the corresponding row (primary inputs must be available prior to evaluation.)

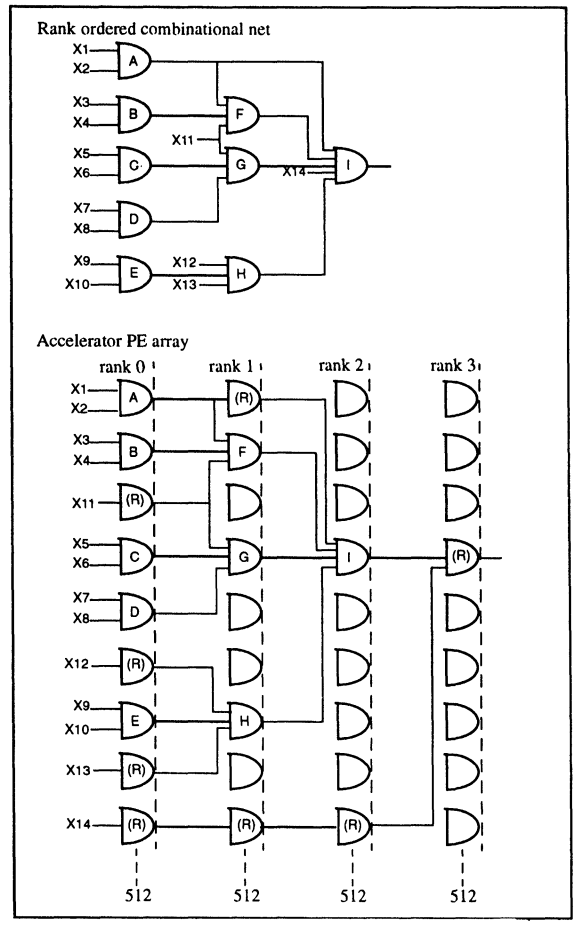

FIGURE 8 Rank ordered combinational design mapped onto accelerator.

4. SIMULATION SLOWDOWN FACTORS

The time required to execute a simulation on the platform is determined by the total number of simulation clock cycles required and the clock period. The number of clock cycles required to execute a simulation is determined by the number of clock cycles required to execute each rank in the compiled netlist, summed over all ranks. Since the accelerator executes at one clock cycle per rank, the number of clock cycles is equal to the number of ranks in the rank ordered net.

The period of the simulation clock cycle is constrained by logic delays to execute and propagate a simulation rank. The time required to execute a

netlist rank is determined by two types of delays: (1) gate delays through the PE active signal path–delays required to switch inputs, evaluate functions, latch outputs–a constant factor for each PE/simulation rank, and (2) fanout route delays which include wire delays for long fanout paths.

4.1. Interconnect Loading

The number of interconnect links consumed by all fanout connections for a net can be determined as follows. Between any pair of fanout connected nodes, the fanout distance is the path length, or number of single bit links that are consumed by the connection. That is, fanout distance on the grid can be determined as the sum of the number of horizontal links given by the column distance $|l-j|$ and the number of forward links given by the row distance $|k-i|$. For a net, the total number of links required for fanout connection is the sum of links for each fanout-connected pair in the net. The communications loading for each node is the number of fanouts for the node plus the routing paths of other fanouts which include the node.

Since there are four signal lines in each direction between node pairs, any lateral loading >4 produces collisions. In order to measure lateral crossover density, a figure of merit called the *collision degree* is defined relative to the lateral interconnect width of 4. A collision degree of zero means that no interconnect load is >4, so that all routing can be resolved by one circuit switching configuration between net ranks. A collision degree of one means that interconnect load is >4 but <=8, so that all collisions can be resolved by inserting one redundant rank between the original net ranks.

A figure of merit for the execution efficiency for a net can be derived from the sum of collision degrees for all rank pairs in the net. This follows directly from the circuit switching design of the grid, which allows propagation of fanouts between adjacent ranks in a single execution clock cycle. For a net having collision degree one, an additional clock cycle is required to propagate the redundant rank of interconnect load, and execution is slowed by a factor of two between the rank pair. Higher collision degrees are assessed similarly.

4.2. Netlist Redundancy Bounds

In order to predict bounds for time-space penalties due to netlist redundancy injection during preprocessing, a typical dense commercial circuit is characterized a 32-bit × 32-bit multiplier built up from 4-bit × 4-bit multiplier building blocks. For this circuit, partial results are summed in a Wallace tree to produce a 63-bit product, and Booth recoders and carry look-ahead generators are used to achieve high performance. This circuit is a typical logic design application for a 4-input sea-of-gates gate array requiring ~7500 gates, a non-trivial amount of logic.

The multiplier circuit requires 40–50 logic ranks (without pipelining). Three redundant logic ranks are required to distribute fan-out signals as follows: 1) one rank for the top of the logic tree for distributing 5 carries, 2) one rank for the 4-bit look-ahead, and 3) one rank for the 4-bit group look-ahead. The 3 redundant ranks represent 3/40, or < 10% redundancy. The maximum width of the multiplier circuit can be calculated as 16 × (4-in multiplexer) × 32 bits = 2^{11} gates. Feed-back paths could add 32 lines to this width, or (2^{11} + 32). A 32-die board configured as 2 push-pull rows with 28 PEs per die requires a system width of (2^8 PE/die) × (2^5 die/board) = 2^{13} PE/board, which is wider than the multiplier circuit. Therefore, no additional redundancy penalty is assessed based on width. As an example of upper bound worst case fanout crossing, logic design for a 32-bit barrel shifter could require up to 2x rows, or 100% redundancy.

Based on the above analysis, a liberal 20% figure will be adopted for typical redundancy to be used in calculating effective gate capacity for the accelerator platform.

4.3. Off-Chip Signal Slowdown

Simulation slow-down due to off-chip fanout propagation is calculated based on 1 ns/ft signal propagation speed as: 1) chip-to-chip fanout distance of <2 ft

is presumed, so that 5 ns, or 1 clock cycle (developed in section 4.4) slowdown is adopted; and 2) board-to-board fanout, including pad delays, is liberally penalized at 2 clock cycles slowdown.

Typical netlist fanout propagation slowdown, based on typical commercial circuits, is predicted based on 40-rank subnet depths, with 35 ranks executing in one clock cycle (all on-chip with complete fanout distribution), four ranks requiring two clocks (on-chip but delayed for dense fanout distribution), and one rank requiring two clocks (off-chip for wide ranks). This results in a predicted typical simulation slowdown of (4 × 1 clock) + (1 × 1 clock) = 5 clocks for a 40-rank net. Since the fastest possible simulation time for this net is 40 clocks, the simulation slowdown due to fanout propagation delay is 5/40, which is less than 20%.

4.4. System Timing

A cornerstone issue for cost-performance analysis, once accelerator functionality and architecture is defined, is "how fast can it go?" The answer can be derived in terms of the simulation clock period, which is a function of the system interconnect and logic topology delays derived from fabrication technology parameters. The required clock period can be bounded by calculating signal path delays through the PE cell active logic, and fanout propagation delays through the mesh interconnect.

Technology parameters for this research are based on currently reported fabrication process values applied to fundamental MOS relationships. In order to avoid current density problems and long signal delays along metal runs, metal width is currently being held at about 2μ with 0.05 Ω/sq, which will hold wire delays at $t_{wire} \approx 1$ ns/cm. A system's clock period is proportional to the τ of its smallest devices, and so becomes proportionally smaller as devices scale smaller. At a feature size of 0.6μ, a device value of $\tau \approx 0.02$ ns is predicted [14–16], and a system clock period of ≈ 2–4 ns. We will adopt the value $\tau \approx 0.03$ ns.

The BEU and latch design, Figures 4-6, require a total of six gate delays. One gate delay is required to generate the sixteen minterms as a function of the four inputs. During clock φ_1, the node inputs are gated into the EVALUATE block and the node descriptor function selectors are gated into the SELECT block, producing a BEU output value. Since the desired minterm is preselected by the SELECT block during φ_1, there is no additional delay on the BEU output. During φ_2, the BEU output value is gated into the OUTPUT latch. The latch requires two inverter delays plus two gate delays. The INPUT/OUTPUT switching block, Figure 3, requires no gate delays for signal propagation. All configuration switches are set within a single gate delay during φ_1. For a chain of PE cells forming a platform row, τ_{stage} is defined as the fundamental unit of on-chip fanout propagation, calculated at ~1.5 ns in [11]. For clock period measurement purposes, the switch-BEU-latch path delay can now be stated in terms of τ_{stage}, $\tau_{switch\text{-}BEU} \approx 2 \times \tau_{stage} \approx 3$ ns.

Fanout signal propagation delay between successive net ranks is composed of zero or more rank forward cell-to-cell wire delays and a single lateral cell-to-cell wire delay. Both delay directions may cross chip and/or board boundaries. All fanout propagation wire delays take place during φ_2. If wire delays are set at 1 ns/cm, then a φ_2 period of 3 ns would allow fanout propagation length of up to (3 ns × 1 cm/ns) = 3 cm. Based on the above calculations, the netlist pre-processor should observe the following rules for assigning redundant ranks to distribute delay periods based on fanout route lengths. A simulation slowdown of one clock cycle, and therefore redundant rank, will be assigned to the following fanout route configurations: every 16-cell segment of on-chip route length, every chip-to-chip crossing, and every board-to-board crossing.

In summary, allowing 2 ns for active path delay within the PE cell, and 3 ns for fanout propagation, the clock period could be set conservatively at 5 ns for the 0.6 μ technology. The SRAM cells have a structure equivalent to the BEU output latch, so that delay through the cell will be estimated at $\sim 2 \times \tau_{stage} \approx 3$ ns, or approximately equal to the PE signal delay. Allowing another 3 ns for fanout propagation, the clock period can be conservatively projected to be $\tau_{clock} = \tau_{PE} + \tau_{fanout} \approx 6$ ns. The push-pull circuit

uses two 5 ns clocks to read SRAM data, allowing up to 200 MHz simulation rate, which is in line with leading edge RISC chip + memory clocking frequencies.

5. COST-PERFORMANCE VALUE

For purposes of comparing performance predictions for this research to values reported for other acceleration work, a common performance figure of merit will be defined. Event-driven simulators measure performance in terms of *event* evaluations per unit time, where an event is caused by a change in the value of a gate input. Only gates for which events occur are subsequently evaluated, since gates without events cannot change value. Compiler-driven simulators, which include this research, evaluate all netlist gates without regard to events. In order to compare the performance of event-driven and compiler-driven simulators, we make an estimation of the percent of active gates (those with events) for typical circuits. The predicted size of this active subset of the netlist is then taken as a statistical measure of the number of gates which must be evaluated in order to propagate some input pattern through the net. The predicted time required by a simulator to evaluate these gates is then used to determine the predicted effective gate evaluation rate for the simulator, or *egevs* (effective gate evaluations per second, termed *active gate evaluations/second* in Blank [14].)

For software event-driven simulators, the best reported rates by Blank are on the order of 10^3 egevs. For previous hardware accelerators, the best reported event-driven rates are on the order of 6×10^7 egevs for the Zycad LE. The best reported compiler-driven rates are on the order of 9.6 x 10^8 egevs for the IBM EVE. Blank has estimated an average cost of \$0.10 per gate evaluation per second for reported hardware accelerator products and prototypes, in contrast to \$100 per gate evaluation per second for a \$200K Vax running at 2K gate evaluations per second.

For the accelerator architecture proposed in this research, the predicted performance is derived from the level of integration, the effective gate capacity of the system, the topology, and the clock rate. The fundamental building block die integrates 256 or 2^8 PEs arranged as 2 push-pull rows of 128 PEs each. Assuming 2^5 die/board, 2^4 boards/rack, and 2 racks, a row width of $(2^7 \times 2^5 \times 2^4 \times 2) = 2^{17}$ PEs is obtained. Using our raw-to-effective equivalence factor of 2^{-3} from section 3, $(2^{17} \times 2^{-3}) = 2^{14} = \sim 16 \times 10^3$ effective PEs per row. The previously derived clock rate of 200 MHz or 2×10^8 will now be used. Performance can now be predicted as $(16 \times 10^3$ equiv-gates$)(2 \times 10^8$ gate-eval/sec$) = \sim 3.2 \times 10^{12}$ egevs, which improves reported state-of-the-art compiler-driven performance (for the IBM-EVE) by three orders of magnitude. It should be noted that both the IBM EVE and this accelerator do not allow assignable gate delays.

The price of the accelerator system is estimated based on expected small quantity die costs together with the (nominal) cost of a general-purpose workstation host. For a SUN Sparcstation, upper bound on cost is \$10,000 or $\$10^4$. Small quantity die cost is expected to be ~\$500. The number of die, as previously predicted: 2^5 die/board $\times$ 2^4 boards/rack $\times$ 2 racks $= 2^{10}$ or 1024 die/system. Accelerator array cost can now be estimated at 1024 die/system $\times$ \$500 /die $= \sim$ \$500,000 or $\$5 \times 10^5$ (improvements in small volume production technology may reduce the cost 10X). Since the host cost is insignificant, we estimate the total accelerator system cost at $\$5 \times 10^5$ for a full size two card rack accelerator.

The resulting cost-performance figure of merit for this accelerator is conservatively predicted at $\$5 \times 10^5 / 10^{12}$ egevs $= \sim \$10^{-6} =$ \$0.000001 per egevs, or about 5 orders of magnitude better than the reported average industry accelerator cost-performance value of $\$10^{-1}$, as illustrated in Table I and Figure 9.

6. CONCLUSIONS

A hardware architecture is proposed which allows direct mapping of design simulation topology onto the acceleration platform. This approach is in contrast to other projects which have either tailored a simulation

TABLE I Accelerator cost-performance values, based on Blank [14].

Product	H/W-S/W, Algorithm	Max Gates	egevs	Average Cost-Perf	Est. Cost
Vax	S/W, Event	?	$\sim 2 \times 10^3$	$\$10^2$	$\$2\times10^5$
Daisy	H/W, Event	1×10^6	$\sim10^5$		$\sim\$10^4$
Tegas	H/W, Event	2.5×10^6	$\sim10^6$		$\sim\$2\times10^5$
Zycad LE	H/W, Event	1×10^6	$\sim 6 \times 10^7$	$\sim\$10^{-1}$	$\sim\$2\times10^6$
NEC HAL	H/W, Event	3×10^6	$\sim 3.6\times 10^8$		$\sim\$4\times10^7$
IBM EVE	HW, Comp	4×10^6	$\sim 9.6\times 10^8$		$\sim\$10^8$
Proposed	HW, Comp	1×10^6	$\sim 3.2\times10^{12}$	$\sim\$10^{-6}$	$\sim\$5\times10^5$

algorithm to match an existing machine, or reconfigured an existing machine to match a simulation algorithm. In order to isolate architectural principles for analysis, the simulation is confined to functional verification of unit-delay, binary-valued, synchronous gate level logic designs. Under this approach, a rank ordered design description is executed on a parallel processor grid which implements an efficient and direct model of the design, similar to prototyping. A multiple order of magnitude cost-performance advantage over previous work is gained by the combination of architectural innovation and scalable technology-based integration. The architectural innovation reduces logic complexity and execution time of boolean evaluation and communication switching circuits, while large scale parallelism is integrated at die level to reduce cost and communication delays. The results of this research form the basis for a multiple order of magnitude improvement in state-of-the-art cost-performance merit for hardware gate level simulation accelerators.

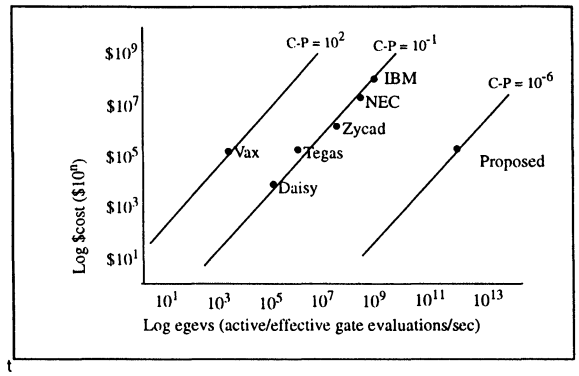

FIGURE 9 Cost-performance value

Advancements of this architecture under study at The University of Texas include mapping of event driven simulation to the hardware for efficient execution, and implementation of timing delays in the data paths without adding excessive hardware. Although not currently being pursued, the architecture could easily be extended to simulate four valued logic by widening all data paths to two bits per signal, increasing the area for data paths by a factor of two and requiring a modest increase in die size. Simulation of asynchronous logic designs is an interesting extension which will require further analysis of the net preprocessing algorithm.

References

[1] Denneau, M.M., "The yorktown simulation engine", 19th IEEE Design Automation Conference, 1982

[2] Kronstadt, E. and Pfister, G., "Software support for the yorktown simulation engine", 19th IEEE Design Automation Conference, 1982.

[3] Pfister, G.F., "The yorktown simulation engine: introduction", 19th IEEE Design Automation Conference, 1982

[4] Szygenda, S.A., "TEGAS2—Anatomy of a General Purpose Test Generation and Acceleration System for Digital Logic", Proc. ACM/IEEE Design Automation Workshop, June, 1972

[5] Szygenda, S.A. & Thompson, E.W., "Digital logic simulation in a time-based, table-driven environment. Part 1. Design verification", Computer, Mar 1975

[6] Szygenda, S.A., "Simulation of digital systems: where we are and where we may be headed", Computer-Aided Design of Digital electronic circuits and systems, North-Holland Publ. Co., 1979

[7] Barto, R.L., "Architecture for a hardware simulator", Proc. IEEE Conference on Circuits and Computers, 1980

[8] Wong, K., and Franklin, M.A., "Performance Analysis and Design of a Logic Simulation Machine", Proc. 14th Symposium on Computer Architecture, 1987.

[9] Zaidi, N.A. and Szygenda, S.A. "A Literature Survey of Hardware Accelerators", ECE Dept., Univ. of Texas at Austin, 1989

[10] Fehr, E.S. and Szygenda, S.A., "The process of digital logic simulation as a basis for an optimized hardware accelerator," International Workshop on CAD Accelerators, Univ. Oxford, UK, Sept. 1989
[11] Fehr, E.S. "An array-based hardware accelerator for digital logic simulation," Ph.D. dissertation, Univ. Texas, Austin, 1992
[12] Kravitz, S.A., Bryant, R.E., and Rutenbar, R.A., "Massively parallel switch-level simulation: a feasibility study", 26th ACM/IEEE Design Automation Conference, 1989
[13] Hillis, W.D., "The Connection Machine", MIT Press, 1985
[14] Blank, T., "A survey of hardware accelerators used in computer-aided design, IEEE Design & Test, Aug 1984
[15] Maurer, P.M., "Two new techniques for unit-delay compiled simulation", IEEE Transaction on CAD, v11, n9, Sept. 1992
[16] Mead, C.A.and Conway, L. "Introduction to VLSI systems", Addison-Wesley, 1980.

Authors' Biographies

E. Scott Fehr received the B.S. in Physics, M.S.E.E., and Ph.D in ECE degrees from The University of Texas at Austin, and was a student in the Ph.D. program in Computer Science at The University of California, Berkeley. He is currently a Research Engineer Associate in the Tactical Simulation Division of the Applied Research Laboratories, The University of Texas at Austin, conducting RD in software systems for C3I test instrumentation and rf mission planning. Areas of technical interest include CAD acceleration, simulation, parallel computer architecture, language and compiler design, and VLSI technology. Previously, he was an independent consultant for UNIX database systems, a contractor doing AIX systems programming and development at IBM, a Senior EngineerProject Manager for the TEGAS6 CAD behavioral language compiler development at Comsat General Integrated Systems, a Sr. Staff Architect for 286 and Z8 successor architectures at Intel and Zilog, a Logic Design Engineer for an APL hardware interpreter at Tektronix, and an Electrical Engineer with the 990 System architecture and operating system development at Texas Instruments.

Stephen A. Szygenda is a professor and Chairman of the Electrical and Computer Engineering Department at the University of Texas at Austin. He also holds Clint Murchison Sr. Chair of Free Enterprise. He is a pioneer in the areas of Simulation, CAD, Fault Tolerant Computing, and Software Engineering; with 30 years of experience, dating back to the early 1960s at Bell Telephone Laboratories. During that period he was involved in the development of the first generation CAD tools for the 1 ESS. After leaving Bell Labs. and joining the faculty of the University of Missouri, he began the development of the TEGAS system. This work was later carried on at SMU and the University of Texas, where he joined the faculty at 1972. In 1979, he left the university environment to form a company, CCSS, which was the first multiproduct simulation and test company, pruducing very large software and hardware systems. In 1980 this company was sold to COMSAT, where Dr. Szygenda remained as president until its subsequent sale to G.E., in 1983. At that time he founded the Rubicon Group; a unique high technology incubator. He rejoined the faculty of the University of Texas at Austin in 1986. He received his M.S. and Ph.D. degrees from Northwestern University, and has been active in numerous areas of the IEEE.

Granville E. Ott received his B. S., M. S., and Ph. D degrees in electrical engineering from The University of Texas at Austin, and is currently a Research Associate in the Tactical Simulation Division of the Applied Research Laboratories, The University of Texas at Austin working signal processing for EW, missile, and tactical systems and is principal investigator on a massive parallel processor project. Prior to returning to UT he was Senior Member of the Technical Staff at Texas Instruments where he was architect for TIAC 827A, and 870A seismic processors, the TI 860 and 980 control processors, and the TI99-4A home computer. He has 10 patents in computers and signal processing. Dr. Ott also was Professor of Electrical Engineering and Director Digital Power Research Group at the The University of Missouri, Columbia where he supervised 15 Ph.D dissertations in computer systems and simulation.