Efficient Stimulus Independent Timing Abstraction Model Based on a New Concept of Circuit Block Transparency

Martin Foltin
mxf@fc.hp.com
(970) 898 3755

Brian Foutz*

Sean Tyler
sct@fc.hp.com
(970) 898 7286

Hewlett Packard Corporation, 3404 E. Harmony Rd., MS-32
Fort Collins, CO 80528

ABSTRACT
We have developed a new timing abstraction model for digital circuit blocks that is stimulus independent, port based, supports designs with level triggered latches, and can be input into commercial STA (Static Timing Analysis) tools. The model is based on an extension of the concept of latch transparency to circuit block transparency introduced in this paper. It was implemented, tested and is being used in conjunction with transistor level STA for microprocessor designs with tens of millions of transistors. The STA simulation times are significantly shorter than with gray box timing models, which can decrease the overall chip timing verification time. The model can also be used in the intellectual property encapsulation domain.

Categories and Subject Descriptors
B.7.2 [**Integrated Circuits**]: Design Aids–*simulation, verification*

General Terms
Performance, Design, Verification.

Keywords
Timing analysis, timing model, VLSI design, circuit optimization

1. INTRODUCTION AND MOTIVATION
High performance VLSI designs, such as microprocessors or high performance chip-sets, rely on custom designed circuits to achieve maximum performance. To achieve sufficient accuracy, the timing analysis for such designs needs to be performed on the transistor level. Static Timing Analysis (STA) [1] tools are commonly employed to achieve complete path coverage in complex designs. However, a full chip transistor level STA for modern designs with tens of millions of transistors is prohibitively expensive.

Therefore, the design is usually divided into smaller blocks, with transistor level STA performed separately for each block. A timing abstraction model of the block is generated based on the STA results. This model is subsequently instantiated in a parent block. In the parent block, the STA tool will trace paths through timing abstraction models rather than through individual transistors. This process is repeated all the way up to the full chip level. This is the basis of hierarchical design methodologies [2]. Size and quality of timing abstractions determines timing accuracy and the overall turn-around time of the entire design.

The timing abstraction model must reproduce all characteristics of the block needed to time a circuit that contains this block. It should abstract away all unnecessary information from the block. Black box – like timing models assign parameters characterizing the block timing properties to block ports. These models are often used in designs with combinational logic and edge triggered latches (flip-flops). High performance designs, however, mostly employ level triggered latches to take advantage of time borrowing between latches to reach high operating frequencies. Black box models do not adequately model such designs because they do not support transparency, i.e., they cannot model input→output paths that pass level triggered latches.

Several timing models have been developed so far that can model transparent paths. The most relevant models are described in Section 2 below. The disadvantage of these models is that they are either not port based or they are input stimulus dependent. The models that are not port based retain some or all of the internal sequential nodes representing latches, domino logic, etc., from the modeled circuit. Timing analysis of circuits with such models is characterized by a large number of timing paths, long run times and large memory requirements, especially at the top hierarchy levels. The models that are input stimulus dependent are incompatible with hierarchical timing methodologies. In these methodologies, the timing of the design is optimized in an iterative manner. The timing abstraction model is generated based on the context files (i.e., the stimulus, slew rates and loads) rolled down from the previous STA iteration. Between the iterations, the blocks will change to fix timing violations. Hence, the context of the timing model instantiated for the current STA iteration may be different from the context used to create this model. This requires the model to be stimulus independent, i.e., to reproduce faithfully the timing of the block for any input signal arrival times.

Timing abstraction models find application also in the intellectual property encapsulation domain, where there is a need to accurately represent a block's timing behavior without exposing the block's internal implementation. Furthermore, the timing abstraction models may simplify design transfer between different

* Now with IBM Test Design Automation, Endicott, NY.

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

design teams. Stimulus independence is desired in these applications as well.

In this work we have developed a new timing abstraction model that is lightweight, port based, stimulus independent, supports level triggered latches and can be input into commercial STA tools. The model can be used on any level of hierarchy where it can be mixed with transistors and/or gates. The model was implemented, tested and is used in conjunction with transistor level STA for designs with millions of transistors. We will show that the model performance is significantly better and memory requirements are significantly smaller than for industry standard models.

2. PREVIOUS WORK

A straight-forward, flexible and accurate but not very efficient timing abstraction model for circuit blocks with level triggered latches is the gray box model [3]. This model abstracts away all combinational logic between the sequential elements and replaces it by delay arcs. All sequential elements (latches, domino logic, etc.) are retained and represented by model nodes. Timing checks performed on each model node are the same as on the corresponding sequential element in the modeled circuit. Timing analysis with gray box models will contain many internal paths that are unnecessary for verification of the timing behavior of the block in context of the upper level block. For designs with millions of transistors, the full chip timing analysis with gray boxes has long runtimes and high memory and storage demands. Advantages of the model are that it is clock frequency independent and the model paths are relatively easy to debug.

Striving to improve timing analysis efficiency, Segal et al. [4] developed a timing model that collapses together similar chains of latches in the modeled circuit. Two paths from a given input port are considered similar if the corresponding latches on the first and on the second path are controlled by the same clock signals. The delay arcs between the collapsed latches and from the last latch to output ports are adjusted so that i) the abstraction correctly models the total path delays, and ii) the setup check on each collapsed latch will equal to the most critical setup check from the two (or more) corresponding latches before they were collapsed. Additional latches are needed to model the clock→output paths from each original latch, but the authors conclude that the model will usually have fewer latches than the original design.

In a different approach, Venkatesh et al. [5] developed a simple abstraction model that represents all internal timing constraints of the block (the setup, hold and latch transparency requirements on all sequential nodes) in terms of a reduced set of requirements for clock edge separations and pulse widths. This was achieved by employing the symbolic formulation for signal arrival times, signal departure times and timing constraints on all sequential elements based on the general critical paths model [6]. Although this timing abstraction is stimulus dependent and falls more into the domain of optimal clocking, it is worth mentioning in context of this work because the underlying general critical paths model [6] provides numerical framework that may be used to derive the time constraints for signals at input ports in our new model.

Synopsys recently introduced an enhanced black box timing model that they named “transparent black” model [7]. For signals at input ports, this model measures the worst-case setup/hold time to any latch node through a chain of transparent latches. This is a significant enhancement to the conventional black box model that measures the worst-case setup/hold time only to the first latch node. Unfortunately, the model stops short of representing the input→output transparent paths in a way that would be independent on input stimulus. This prevents the model from being used in hierarchical design methodologies as described in the Introduction section above.

3. NEW TIMING ABSTRACTION MODEL

The new timing abstraction model presented in this work is a simplified circuit or a connectivity network that reproduces the most relevant timing characteristics of the modeled circuit block needed for timing analysis of this block in context of a larger circuit where the block is instantiated. The model is based on extension of the concept of latch transparency to circuit block transparency. Two different data path types are usually analyzed in STA – the longest paths and the shortest paths. In the discussion below we will describe timing abstraction model elements for the longest data paths that pass through sequential elements (latches, domino logic, etc.). Modeling of shortest data paths is simpler because they cannot pass transparently through the block. For shortest paths, the model gets simplified to a black box - like format. All purely combinational input→output paths are simply modeled by delay arcs (or delay nets) connecting the corresponding input and output ports.

3.1 Concept of Circuit Block Transparency

The level triggered latch is transparent when the clock signal connected to its clock pin is active (high or low, depending on the latch design). If the data signal arrives at the latch input before the clock signal becomes active, the signal does not pass to the latch output. In such case, the critical path will be triggered by the clock signal transition rather than by the data signal transition. This concept can be extended from latch to circuit block. Unlike a single latch, the block usually has several output ports. Hence, we need to define transparency separately for each output port. We classify the given output port as transparent if and only if the data signal arrives at this port no earlier than the latest clock signal that can get to this output port. Note that when considering the latest clock signal, this is not limited only to clock signals from last sequential elements in the fan-in of the given output port, but also includes clock signals from any preceding sequential elements as long as they pass transparently through the chain of latches (or domino logic, etc.) all the way to the given output port.

This port-based definition of transparency is more restrictive than considering transparency individually for each path. This is illustrated in Figure 1. The data signal from input **IN_1** arrives at latch **LAT_1** while the clock signal on this latch is active, so if we would consider just the path **IN_1→OUT**, this path would be transparent. However, the clock signal on **CK_2→OUT** path passing latches **LAT_2** and **LAT_3** arrives at output **OUT** later than the data signal on the **IN_1→OUT** path. Therefore, according to our definition of transparency, the block will not be transparent for the **IN_1→OUT** path. Blocking the **IN_1→OUT** path is entirely within the spirit of STA that strives to limit the analyzed paths only to the most critical (i.e., in this case the longest) paths. The designer is not interested in how the **IN_1→OUT** path propagates downstream from our block because

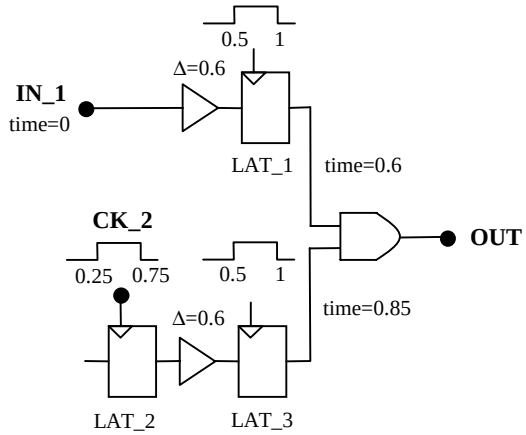

Figure 1. Example circuit with CK_2→OUT path that is more critical than IN_1→OUT data path

it merely trails the more critical path that reaches block output from the **CK_2→OUT** direction. Our definition of transparency therefore limits the number of unnecessary paths leaving the modeled block. Considering that such paths may branch several times in successive blocks, this definition may significantly reduce the number of paths to be traced in the design and therefore improves the STA efficiency and reduces designer's time to analyze STA results.

A timing abstraction model that is consistent with the block transparency definition from previous paragraphs can be designed as follows. Central to this design is that all transparent paths converging to a given output port **OUT** pass through a single latch attached to this port that serves as a transparency "gate" determining whether or not the given path will pass transparently to the output port. This assures the stimulus independence of our model. If, for example, for a different stimulus at input port **IN_1** the **IN_1→OUT** path from Figure 1 becomes longer than the **CK_2→OUT** path, than the signal on this path will arrive at the input of the model latch while the latch is open and the path will be traced to the output port.

The model latch will be controlled by signal from a derived clock node. The opening time of this latch can be easily obtained from results of STA analysis of the modeled block – simply take the output arrival time of the most critical clock→**OUT** path from the path report. Reference this time to a pin clock (i.e., a clock port of the block) that will be connected to the derived clock via delay arc (or delay net). Sections 3.2 and 3.3 will explain which pin clock to use to assure proper cycle advancement across the block boundaries. The model will be valid as long as the phase shift of clock waveforms between the model generation and the model instantiation is the same for all clock ports of the block. The clock frequencies and pulse widths on clock ports may not change. These are reasonable assumptions in most design environments, since the clock distribution tree and clock frequency usually do not change between individual iterations in the timing verification process. For the intellectual property encapsulation application, separate sets of model parameters may be generated for different clock frequencies.

Derivation of the latch closing time is slightly more complicated. Let's denote by **T(OUT)** the set of all input ports in the modeled circuit that may be connected for some input stimulus by a transparent path to the given output port **OUT.** Let's denote by **R(IN_*i*)** the latest time a signal can arrive at input port **IN_*i*∈T(OUT)** before a timing violation would occur on any sequential element that can be reached by any path from that input port **IN_*i***. This time can be obtained, for example, from the general critical paths model [6] where it is referred to as the required time. If **D(IN_*i*)** is the longest path delay between the input port **IN_*i*** and the output port **OUT**, then the closing time **TC** of the model latch attached to the output port **OUT** is obtained from formula

$$\mathbf{TC} = \max \forall\ \mathbf{IN\_}i \in \mathbf{T(OUT)},\ \mathbf{R(IN\_}i\mathbf{)} + \mathbf{D(IN\_}i\mathbf{)} \qquad (1)$$

This assures that all transparent paths can pass through this latch without timing violation.

To obtain the set of transparent paths **T(OUT)** and the transparent path delays **D(IN_*i*)**, we can, for example, perform a second STA run on the modeled block with stimulus on each input port **IN_*i*** set to the time **R(IN_*i*)**.

3.2 Timing Model of an Example Circuit

Figure 2 shows an example circuit block with two input ports, **IN_1** and **IN_2**, two output ports, **OUT_1** and **OUT_2**, and three transparent paths, **IN_1→OUT_1**, **IN_2→OUT_1** and **IN_2→OUT_2**. Figure 3 shows the timing abstraction model for this block. As discussed above, the timing model has one latch attached to each output port that serves as a transparency "gate" (see Output Latch in Figure 3). The timing model also has one latch attached to each input port that checks for timing violations on the most critical sequential element in the modeled circuit that can be reached from the given input port **IN** (see Input Latch in Figure 3). The closing time of this Input Latch is set to the time **R(IN)** discussed before. In general, the opening time **TO'** of the input model latch can be obtained from formula

$$\mathbf{TO'} = \min \forall\ \mathbf{OUT\_}j \in \mathbf{T'(IN)},\ \mathbf{V(OUT\_}j\mathbf{)} - \mathbf{D(OUT\_}j\mathbf{)}, \qquad (2)$$

where **T'(IN)** is the set of all output ports **OUT_*j*** in the modeled circuit that may be reached for some input stimulus by a transparent path from the input port **IN, V(OUT_*j*)** is the opening time of the model output latch attached to output port **OUT_*j*** and **D(OUT_*j*)** is the longest transparent path delay between the input port **IN** and the output port **OUT_*j***. Each Input Latch is connected by delay arcs (or delay nets) with delays **D(OUT_*j*)** to

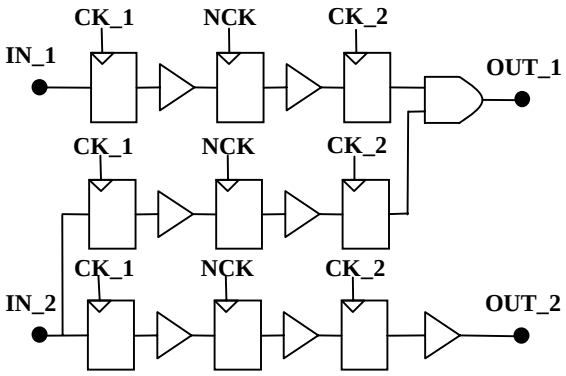

Figure 2. Example circuit with transparent paths IN_1→OUT_1, IN_2→OUT_1, and IN_2→OUT_2

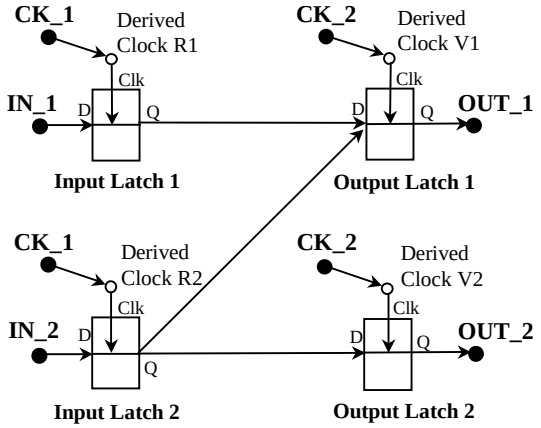

Figure 3. Timing abstraction model for the example circuit from Figure 2

Output Latches ***j*** attached to output ports **OUT_*j*∈T'(IN)**.

Let's consider STA of a parent block that contains, among other circuit elements, one or more timing abstraction models instantiated in place of children blocks. The timing path entering the timing abstraction model through input port **IN** will first pass through the Input Latch attached to this port. The timing slack on this latch will be exactly equal to the slack that would be observed in full transistor level STA of the modeled block on the most critical sequential element that can be reached from the input port **IN**. The name of this element will be encoded in the Input Latch name, so that if there is a negative slack on the Input Latch, the designer will know which sequential element in the modeled circuit is failing most and by how much. If there is no violation, the timing path will branch and propagate to Output Latch nodes ***j*** connected to output ports **OUT_*j*∈T'(IN)**. Depending on the actual signal arrival on Output Latch node (that is function of signal arrival at input port **IN**), the Output Latch will determine if the path should go transparently to the associated output port.

Note that the number of internal nodes in this timing abstraction model scales with the number of ports rather than with the number of paths. For most circuit blocks, there will be one internal node per each input port (the Input Latch) and one internal node per each output port (the Output Latch). The number of timing checks to be performed is limited to one check per each input port plus one check per each input port – output port pair that is connected by a transparent path for at least one input stimulus. Both the number of internal nodes, the number of timing checks and the number of paths are significantly smaller than in gray-box timing abstraction models. In spite of its light weight, the model provides all the most important timing information that designers may need – it flags the most critical timing violation for each input port and it models correctly the critical transparent input→output paths.

3.3 Cycle Advancement at Block Boundaries

If the sequential elements connected through combinational logic to output port **OUT** in the modeled circuit are controlled by pin clocks derived from different reference clocks or reference clock edges, we may need to attach more than one Output Latch to output port **OUT** in the timing model. This is to assure proper cycle count adjustment across the block boundaries as the signal passes from the "driver" Output Latch in the timing abstraction model to "receiver" latches in adjacent blocks downstream on the path. For similar reasons we may need to attach more than one Input Latch to input port **IN**.

In Section 3.1 we defined the set **T(OUT)** of all input ports that may be connected for some input stimulus by a transparent path to the given output port **OUT**. In this section we divide this set to subsets **T(OUT, CLK, CLK_edge)** according to the reference clock **CLK** and reference clock edge **CLK_edge** associated with the opening clock edge on the last sequential element along the transparent path **IN_*i*→OUT**, where **IN_*i*∈T(OUT)**. We will assign one Output Latch to each subset **T(OUT, CLK, CLK_edge)**. The opening time for each Output Latch is calculated similarly like in Section 3.1 except that now we consider only the most critical clock→**OUT** path that has the opening clock edge on the last sequential element derived from the reference clock **CLK** and reference clock edge **CLK_edge**. The Output Latch closing time is calculated from formula (1) where we replace **T(OUT)** by **T(OUT, CLK, CLK_edge)**.

To illustrate this, Figure 4 shows an example circuit that is modeled by the timing abstraction model shown in Figure 5. The modeled circuit has three transparent paths leading to output port **OUT**. The last sequential elements on these paths are open when the pin clocks **CK_3**, **CK_6** and **CK_9**, respectively, are high. The opening (rising) edges of **CK_3** and **CK_6** are derived from rising edge of reference clock **CLK**. The opening edge of **CK_9** is derived from falling edge of reference clock **CLK**. Therefore, the first two transparent paths will be in subset **T(OUT, CLK, R)**. The third path will be in subset **T(OUT, CLK, F)**. In the timing abstraction model, the first two transparent paths will pass through Output Latch controlled by a derived clock connected to pin clock **CK_3**. That is because the most critical clock→**OUT** path that determines opening time of the Output Latch for the first subset **T(OUT, CLK, R)** is the **CK_1→OUT**. The last latch on this path is controlled by pin clock **CK_3**. The third transparent path in the timing model will pass through a second Output Latch controlled by a derived clock connected to pin clock **CK_9**.

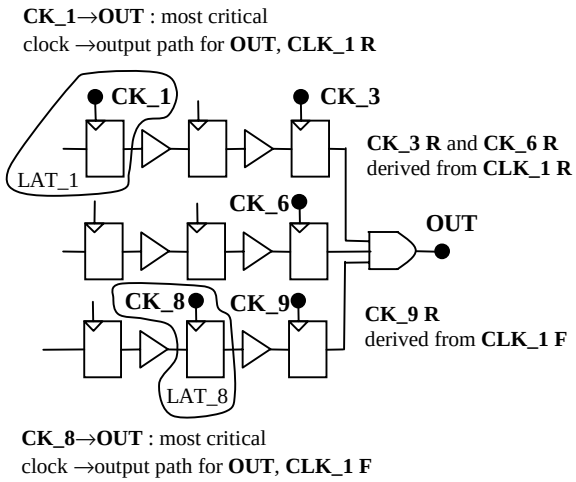

Figure 4. Example circuit with 3 transparent paths to output port OUT.

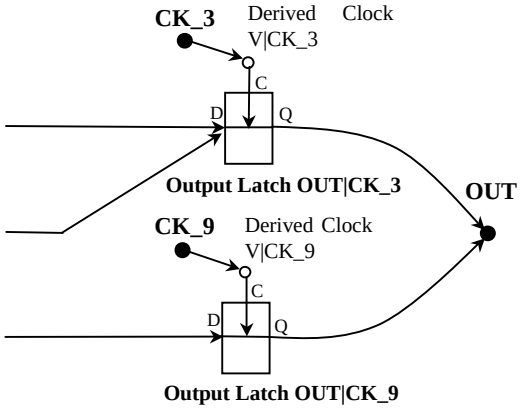

Figure 5. Output section of the timing abstraction model for the example circuit from Figure 4

Similarly, multiple Input Latches will be attached to each input port **IN** if the set **T'(IN)** defined in Section 3.2 needs to be divided to subsets according to the reference clock **CLK'** and reference clock edge **CLK_edge'** associated with the closing clock edge on the first sequential element on the **IN→OUT_*j*** transparent path.

3.4 Modeling Slope and Load Sensitivity

In this timing abstraction model, the model elements that are on block boundaries are the Input and Output Latches. The driver strength parameters of the Output Latch connected to output port **OUT** are derived from the driver strength of the last circuit element connected to that output port in the modeled circuit. Likewise, the slope (slew-rate) sensitivity parameters on Input Latch connected to input port **IN** are derived from the slope sensitivity of the first circuit element connected to that input port in the modeled circuit. In some circuit blocks it may happen that different paths from the set **T(OUT, CLK, CLK_edge)** have different load sensitivities. In the current model implementation, all paths from the set **T(OUT, CLK, CLK_edge)** that have driver strengths within certain interval (200 Ω) will pass through a single Output Latch. The driver strength of this Output Latch will be set to the minimum driver strength from all paths passing through this latch. Paths with driver strengths outside of this interval will pass through different Output Latch node(s). By tuning the driver strength interval knob we can trade accuracy for performance. Similarly, all paths from the set **T'(IN, CLK', CLK_edge')** that have slope sensitivities within certain interval will pass through a single Input Latch.

3.5 Expansion of Timing Paths

When analyzing STA results from an upper level block, the designer may wish to expand the path section passing through a particular timing abstraction to gain visibility to circuit elements that were abstracted away by the abstraction model. To facilitate timing path expansion, we create a database of relevant paths from the modeled circuit that can be plugged into the upper level path in place of path sections from the timing abstraction model. The database will contain the abstracted node names, stage delays and the clock information for the abstracted sequential elements. The Input Latch and Output Latch node names in the abstraction model are designed to serve as pointers to paths in this database. For each subset **T(OUT, CLK, CLK_edge)** there will be one path in the database that will correspond to path Type 4 from Figure 6 (i.e., the most critical clock→**OUT** path). The Output Latch node name must unambiguously identify this subset. This is achieved by including the output port name **OUT**, pin clock name **CK2** and the **CK2** opening edge in the Output Latch name (see Figure 6). Note that the **CK2** opening edge is derived from one and only one reference clock and edge (**CLK**, **CLK_edge**). Similarly, for each subset **T'(IN, CLK', CLK_edge')** there will be one path in the database that will correspond to path Type 1 from Figure 6 (i.e., the most critical **IN**→sequential element path). The Input Latch name is derived analogously to the Output Latch name (see Figure 6). The database will contain up to one transparent path of Type 3 (see Figure 6) for each pair of subsets **T'(IN, CLK', CLK_edge')** and **T(OUT, CLK, CLK_edge)**. The Input Latch and Output Latch names on path Type 3 will unambiguously identify this pair of subsets. The corresponding path from the database will be also used to expand path of Type 2, but a note will be printed for such path that it does not pass transparently through the block for the actual signal arrival time at input port **IN**.

The other reason for including the port and clock names as sub-strings in the Input and Output Latch names is to assure uniqueness for these latch names. For example, if the most critical sequential element **LAT_3** is in the fan-out of two different input ports **IN1** and **IN2**, the timing abstraction will contain two different Input Latch nodes **LAT_3|SL|IN1|CK1|f** and **LAT_3|SL|IN2|CK1|f**.

3.6 Timing Model Verification

For the purpose of model verification, the STA analysis will be performed on the instantiated timing model with the same stimulus that was used in the full transistor level STA run on the modeled block. All paths from the full transistor level STA analysis will be categorized according to **IN**→sequential element, **IN→OUT** and Clock→**OUT** types. Only the most critical **IN**→sequential element path will be considered for each combination **(IN, CLK', CLK_edge')**; only the most critical Clock→**OUT** path will be considered for each combination **(OUT, CLK, CLK_edge)**; and the longest **IN→OUT** transparent path will be considered for each combination **(IN,**

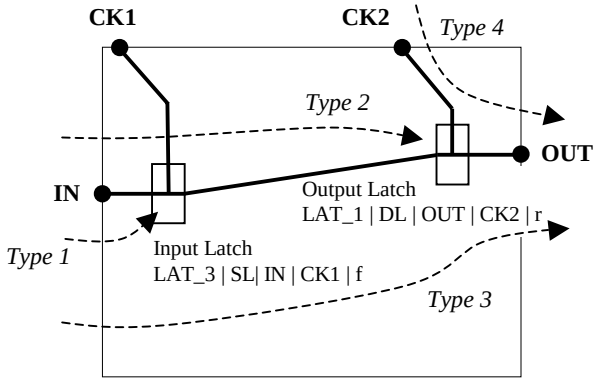

Figure 6. Timing path types passing through the timing abstraction model and the model node naming convention.

CLK',CLK_edge'), **(OUT, CLK, CLK_edge)**. Timing slacks on these paths will be compared with timing slacks on the corresponding paths from the STA of the timing abstraction model. The model will pass verification if the slack difference is smaller than a preset threshold (~ 5ps in our implementation).

4. RESULTS

The timing abstraction model developed in this work can be input into commercial STA tools. For example, in design environments that employ an STA engine supporting gray box - like timing abstractions, the new timing model can be represented by very lightweight gray box. In such case, the model Input Latches and Output Latches are represented by gray box nodes and model delay arcs are represented by gray box time arcs. This representation has far fewer internal nodes than a conventional gray box. For benchmarking, we choose a typical industrial block with about 2 million transistors that contains lower level blocks spanning two levels of hierarchy. The same STA engine is used both for STA with conventional gray boxes and with the new timing models represented by lightweight gray boxes. Because of the large number of lower level blocks with various types of logic (including domino and internal loops), the performance results presented in this section represent a good statistical average. In general, the model generation times will increase with the number of transparent paths in the model. The performance numbers are shown in Table 1 that lists the sums of generation times and runtimes for both representations for each hierarchy level as well as summed over all 3 levels. The performance advantage of the new model gets better as we ascend up the hierarchy level. This is due to the port-based nature of the model. For example, on the Level 3, the new abstraction model generation + runtime is 16-times shorter than for the conventional gray box. The memory requirements on this level are 8-times smaller and the model requires 23-times less disk storage space. Based on extrapolating these trends to the full chip hierarchy level, we anticipate even more performance gain on that level. When summed up over all 3 hierarchy levels, the total STA runtime that includes all model generation times and run times is 3–times shorter for the new timing models than for conventional gray boxes. Again, we anticipate that this factor will increase as we include performance numbers from the full chip level STA.

To test accuracy of the new abstraction model, we employed the verification algorithm from Section 3.6. Timing slacks on timing paths were compared for the full transistor level STA, the STA with the new timing model and the STA with the conventional gray box. We found that the new timing models yield results that agree with the full transistor level STA equally well or slightly better than the results from STA with conventional gray boxes.

5. CONCLUSIONS

We have developed a new timing abstraction model that is stimulus independent, port based, supports transparency and can be input into commercial STA (Static Timing Analysis) tools. Both the number of internal nodes, the number of timing checks and the number of paths in the model are significantly smaller than in the industry standard gray box timing abstraction models. The model size reduction is achieved by eliminating less critical paths, not by giving up accuracy. Performance studies revealed that STA for blocks containing the new timing models is several times faster than with conventional gray box models. The model limitations are that it is clock frequency dependent and for blocks with multiple ported clocks, the phase shift of clock signals between model generation and model instantiation must be the same for all ported clocks. The new timing abstraction model may find application also in the domains of intellectual property encapsulation and design transfer between different design teams.

6. REFERENCES

[1] Hitchcock, R.B., Sr., Smith, G.L., Cheng, D.D., Timing analysis of computer hardware, IBM J. Res. Develop. 26 (1982) 100-105

[2] McDonald, C., Indermaur, T., Buckley, M., Timing analysis for the PA-8000, Integrated System Design (isdmag.com), February 1997, Cover Story

[3] Avidan, J., Circuit analyzer of black, gray and transparent elements, U.S. Patent No. 6158022

[4] Segal, R.B., Extracting accurate and efficient timing models of latch-based designs, U.S. Patent No. 6023568

[5] Venkatesh, S.V., Palermo, R., Mortazavi, M., Sakallah, K.A., Timing abstraction of intellectual property blocks, IEEE 1997 Custom Integrated Circuits Conference, 99-102

[6] Burks, T.M., Sakallah, K.A., Mudge, T.N., Critical paths in circuits with level-sensitive latches, IEEE Trans. on VLSI systems, 3 (1995) 273-291

[7] PathMillPlus, Synopsys Products and Solutions Web page http://www.synopsys.com/products/analysis/pathmillplus_ds.html

Table 1. Performance comparison of the new timing abstraction models with the conventional gray box abstractions for a microprocessor block with 2 million transistors. Model generation + run times are shown in normalized time units.

Hierarchy Level	**Total**	**Level 3** (1 block)	**Level 2** (2 blocks)	**Level 1** (44 blocks)
New Abstraction Generation + Run Time (rel. u.)	12.90	1.00	4.00	7.90
Conventional Gray Box Generation + Run Time (rel. u.)	38.84	16.07	14.02	8.74
New Abstraction Maximum Memory (MB)	503	164	405	503
Conventional Gray Box Maximum Memory (MB)	1369	1369	814	468
New Abstraction File Size (MB)	65.1	4.4	11.2	49.5
Conventional Gray Box File Size (MB)	336.1	103.4	113.5	119.2