ATM: A High Accuracy Extracted Timing Model for Hierarchical Timing Analysis

Kuan-Ming Lai[1], Tsung-Wei Huang[2], Pei-Yu Lee[3] and Tsung-Yi Ho[1]
r365460@gapp.nthu.edu.tw,twh760812@gmail.com,peiyu@maxeda.tech,tyho@cs.nthu.edu.tw
[1]CS Dept, NTHU, Taiwan; [2]ECE Dept, University of Utah, UT; [3]Maxeda Technology Inc.

ABSTRACT

As technology advances, the complexity and size of integrated circuits continue to grow. Hierarchical design flow is a mainstream solution to speed up timing closure. Static timing analysis is a pivotal step in the flow but it can be timing-consuming on large flat designs. To reduce the long runtime, we introduce ATM, a high-accuracy extracted timing model for hierarchical timing analysis. Interface logic model (ILM) and extracted timing model (ETM) are the two popular paradigms for generating timing macros. ILM is accurate but large in model size, and ETM is compact but less accurate. Recent research has applied graph compression techniques to ILM to reduce model size with simultaneous high accuracy. However, the generated models are still very large compared to ETM, and its efficiency of in-context usage may be limited. We base ATM on the ETM paradigm and address its accuracy limitation. Experimental results on TAU 2017 benchmarks show that ATM reduces the maximum absolute error of ETM from 131 ps to less than 1 ps. Compared to the ILM-based approach, our accuracy differs within 1 ps and the generated model can be up to 270× smaller.

1 INTRODUCTION

To reduce the long runtime of timing closure, hierarchical design flow is a popular solution for large designs. In hierarchical design, we divide a system into small manageable sub-modules and reuse each block to save time. Static timing analysis (STA) is a vital step in the flow to verify the timing behavior. Timing-driven optimizations iteratively call an STA engine to analyze and improve the timing of a design until all constraints are met. However, running STA on a large-scale flat design is a time-consuming process. Timing model can help designers to efficiently complete the timing closure by replacing large fixed sub-modules with small black boxes that represent the timing relationships between input and output ports. The STA engine can benefit from running on a small timing model instead of repetitively propagating timing on the flat fixed netlist. As a result, the model size and model accuracy are two primary concerns to generate timing models. When preserving more information in the timing model, we improve the accuracy but increase the model size. The challenge is to strike a balance between model size and accuracy.

ASPDAC '21, January 18–21, 2021, Tokyo, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431578

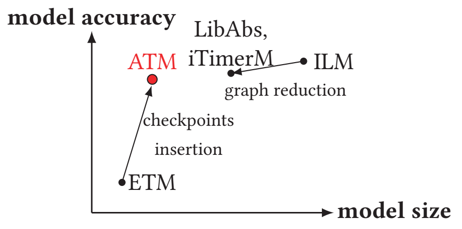


Figure 1: An overview of the start-of-the-art. ATM inserts a small number of checkpoint pins to improve accuracy.

Figure 1 draws the two popular timing model paradigms and highlights their pros and cons in terms of model size and model accuracy. The two common timing models, interface logic model (ILM) and extracted timing model (ETM), are proposed in [1]. ILM preserves the interface logic and discards the timing paths between two registers. Interface logic includes timing paths that start (1) from any primary inputs to any primary outputs/registers, (2) from any registers to any primary outputs. Timing information in interface logic is sensitive to the boundary timing information (e.g., input slew and output load). Since ILM contains all interface logic, it produces high accuracy but results in large model size. ETM abstracts a design into a single cell formatted in liberty. ETM condenses timing paths between two ports into a timing arc to establish their timing relationships. In the cell library, the timing arcs can be roughly classified into two types, combinational/sequential delay arcs and constraint arcs. Each timing arc stores timing values in two-dimensional lookup tables. For delay arcs, the slew and delay propagation are modeled as a function of input slew and output load. The constraint propagation in constraint arcs is a function of input slew and clock slew. Since ETM uses a timing arc to represent timing propagation between two ports, it is extremely compact. However, ETM compresses paths between ports into a timing arc and much information may be lost. This results in accuracy issues especially when paths pass through many output pins.

Previous studies, such as iTimerM [2] and LibAbs [3], observe that ILM is accurate but large in model size due to the preservation of all interface logic. They apply graph reduction techniques, including serial merging, parallel merging, tree merging, and biclique-start replacement to interface logic to reduce model size without affecting the accuracy. While these techniques typically achieve about 75% reduction in model size, it is still large compared to ETM (16−200× more). As a consequence, we introduce ATM, a high-accuracy extracted timing model for hierarchical timing analysis. We base ATM on the ETM paradigm and address its accuracy limitation. Our contributions are as follows:

- We identify the major cause of accuracy loss in the ETM paradigm and propose several techniques to address its deficiencies. Our techniques are generally applicable and can be implemented for other macro modeling frameworks.
- ETM only preserves timing arcs between input and output ports in the generated timing model, which can lead to accuracy problems. To improve the accuracy, we precisely identify a minimum set of checkpoint pins to include in ATM. A checkpoint pin can facilitate the compression of timing paths into a slew-capacitance function.
- Since timing information is stored in lookup tables, the timer obtains the timing values through interpolation. The indices of a table play a key role in deciding the accuracy and the model size. We develop an effective selection strategy of table indices that can prevent redundant table entries and insert essential indices for accurate interpolation.

We have evaluated ATM on a set of industrial designs released by TAU 2017 Timing Analysis Contest on Macro Modeling [4]. The experimental results show that ATM is nearly as accurate as the state-of-the-art model compressor, iTimerM [2], but we reduce the model size by 96× on average. Compared to the ETM engine in Synopsys PrimeTime [1], ATM reduces the maximum absolute error from 131 ps to less than 1 ps, using similar model sizes. We believe ATM stands out as a unique approach considering the tradeoff we explored between the model size and accuracy.

2 PROBLEM FORMULATION

We adhere to the problem formulation of TAU 2017 Timing Analysis Contest on Macro Modeling [4]. The contest provides a rigorous validation environment for us to conduct macro modeling research. Given a circuit, the goal is to extract the boundary timing relationships between input and output ports into a timing model. The output timing model is written in the format of a single cell in Liberty files. Moreover, ATM does not consider CPPR to generate timing models. The input circuit is described by the following files: (1) a gate-level netlist in a Verilog file (.v), (2) a Standard Parasitic Exchange Format (.spef) file to describe the parasitic RC network, (3) two standard cell libraries in Liberty files (.lib) for min (early) and max (late) mode, (4) a timing assertions file.

We evaluate ATM on benchmarks released from TAU 2017 Timing Analysis Contest on Macro Modeling [4]. The contest benchmarks provide operation conditions for each primary input port and output port; the range of primary input slew is from 5 ps to 250 ps and primary output load is from 5 fF to 250 fF. The evaluation is based on the model size, model accuracy, and model usage runtime. The model size is measured at the two liberty files size for early and late mode. Our model accuracy is evaluated on the output timing values of the generated model compared to the original flat design, using OpenTimer [5] as the golden reference to report boundary timing.

3 ALGORITHM

In this section, we highlight the accuracy limitations of ETM [1] and describe the algorithm of ATM.

3.1 Limitations of ETM

The accuracy problem of ETM is threefold.

3.1.1 Interpolation Error. ETM is formatted in liberty files where each timing saved in lookup tables. Timers use linear interpolation on lookup tables with table indices to calculate the value. However, there exists non-linear timing propagation. The interpolation error occurs as a result of using a linear-piecewise lookup table to approximate a nonlinear function. For example, according to [6], the transition propagation within a wire is a function of in_{slew} and out_{load}. It can be written as

$$out_{slew}(in_{slew}, out_{load}) = \sqrt{in_{slew}{}^2 + impulse(out_{load})^2} \quad (1)$$

where $impulse(load)$ is a quadratic function of $load$.

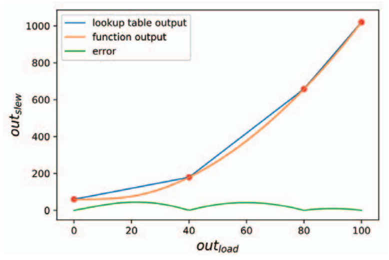


Figure 2: Interpolation error occurs in nonlinear function.

In Figure 2, Equation (1) is plotted in the orange curve with in_{slew} fixed. The equation is approximated to a lookup table consisting of four load indices, (0, 40, 80, and 100), shown in red circles, and the blue line shows the linear interpolation value. We further show the interpolation error in the green curve. There is a tradeoff between the number of indices and accuracy. Using more indices benefits the accuracy but results in larger table size. We propose a method to efficiently insert extra table indices to reduce the interpolation error. We describe the details in Section 3.7.4.

3.1.2 Multiple Output Load Variants. ETM abstracts paths between input and output pin into a delay arc with delay and transition tables associated. The two table indices of delay and transition tables are input transition time and output load. However, in some situations, we could not only use input transition time and output load to describe delay and slew propagation.

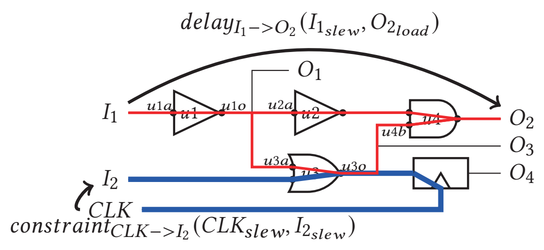

$delay_{I_1->O_2}(I_{1_{slew}}, O_{2_{load}})$
O_1
I_1
$u1a$
$u1$
$u1o$
$u2a$
$u2$
$u4$
$u4b$
O_2
O_3
O_4
I_2
$u3a$
$u3$
$u3o$
CLK
$constraint_{CLK->I_2}(CLK_{slew}, I_{2_{slew}})$

Figure 3: Two situations where ETM is hard to use two-dimensional lookup table to represent the delay, transition, and constraint between two ports.

For example, in Figure 3, ETM creates a delay arc from I_1 to O_2, and models the delay as a function of $I_{1_{slew}}$ and $O_{2_{load}}$. But in the original netlist the paths between I_1 and O_2 (highlighted in red) also connect to other output pins, O_1 and O_3. As a result, $O_{1_{load}}$ and $O_{3_{load}}$ affect the cell delay of $u1$ and $u3$, respectively.

3.1.3 Multiple Input Slew Variants. In timing analysis, slew values (slew for short) is propagated to calculate the delay along the path. Timer propagates the smallest/largest slew for each pin in min/max mode. For instance, in Figure 3, $u3o_{slew}$ is equal to $max(slew_{I_1->u3o}, slew_{I_2->u3o})$ in max mode.

ETM extracts a constraint arc from a clock pin to a input pin. A constraint table is a function of clock transition time and input transition time. For instance, ETM creates a constraint arc from CLK to I_2 in Figure 3 to replace the paths between CLK and I_2 (highlighted in blue). However, some pins along the paths are still sensitive to other input slews. For example, $u3o_{slew}$ is not only affected by $I_{2_{slew}}$ but also by $I_{1_{slew}}$. This causes errors to ETM by modeling the constraint only in function of $I_{2_{slew}}$ and CLK_{slew}.

3.2 Algorithm Flow

We first turn the input files into a timing graph. According to the user-defined operation condition, we propagate the minimum and maximum slews from the primary input. Next, we identify dirty pins and checkpoints. Based on the checkpoints, we partition the timing graph into several groups and extract timing arcs for each group. After extracting all the timing arcs, we select the table indices and then perform timing inference to obtain the values of each timing table entry.

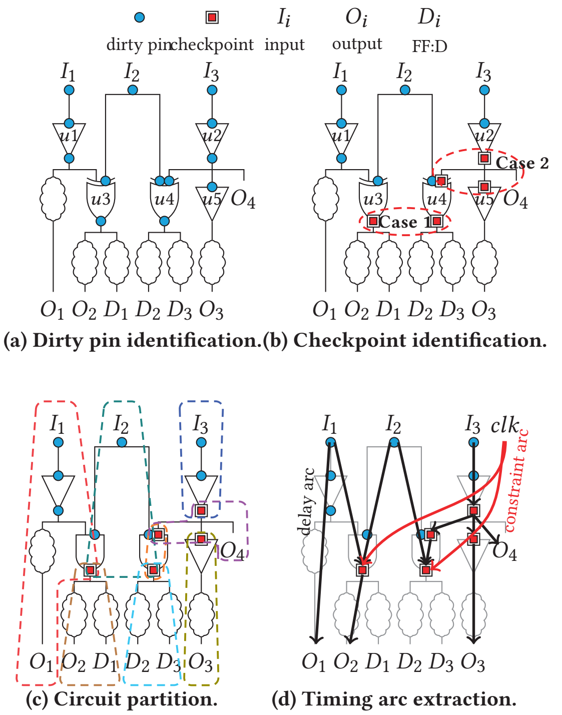
3.3 Minimum and Maximum Slew Propagation

In this step, we aim to obtain possible slew range for each pin under all possible operation conditions. We first set the minimum and maximum slews (5 ps and 250 ps) and loads (5 fF and 250 fF) to primary inputs and outputs, respectively. Then, we propagate the minimum and maximum slews in the topological order. The minimum and maximum values indicate the possible range of slew for each pin.

3.4 Dirty Pin and Checkpoint

For a pin, once the difference between minimum and maximum slews is less than a threshold, we set its slew as a constant. Due to the shielding effect, slew normally becomes constant after several propagation stages [2]. This represents that their slews are no longer influenced by the primary input slew and output load. For those pins of non-converged minimum and maximum slews, we consider them as *dirty pins*. We next define a *checkpoint*. If a path goes through a checkpoint (not start point and endpoint), then the timing information along the path is impacted by at least two input slews or two output loads. There are two cases to tell if a pin is a checkpoint.

- **case 1:** A dirty pin has at least two fanins both connected by primary inputs.
- **case 2:** A dirty pin is on a net connecting to primary outputs and the net has at least two fanouts.



(a) Dirty pin identification. (b) Checkpoint identification.

(c) Circuit partition. (d) Timing arc extraction.

Figure 4: We identify the checkpoint pins to reserve in ATM. Based on the checkpoint, we partition the interface logic into several groups and extract timing arcs for each group to represent the timing relationships in the interface logic.

Figure 4 (a) and (b) show examples of dirty pins and checkpoints identification. The output pin of $u3$ is a checkpoint because it has two fanins connected by primary inputs (I1 and I2), which belongs to case 1. For the case 2, primary output $O4$ entails other pins on the same net to be checkpoints.

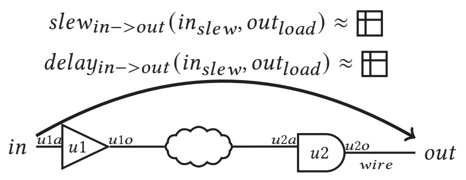
3.5 Circuit Partition

Modeling the timing propagation into a function of an input slew and an output load causes an accuracy issue to ETM. When the original paths include checkpoints, it is hard to estimate the timing propagation with one input slew and one output load. To handle this issue, we partition the interface logic into several groups. Each group has a *source* pin which is the lowest topological order within the group. The intuition of circuit partition is to ensure that paths starting from *source* to group members do not go through any checkpoints.

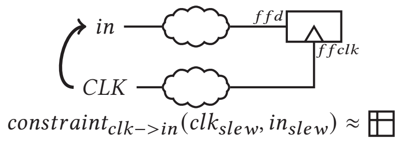
We build up a group for each primary input and checkpoint. For example, there are eight groups in Figure 4 (c) since there are three primary inputs and five checkpoints. To find out the group member, we begin with the depth-first search algorithm from each source pin and stop at either primary outputs, FF:D, or other checkpoints.

3.6 Timing Arc Extraction

After partitioning the circuit, we consider each group as an independent sub-circuit to build the extracted timing model. For each


$slew_{in->out}(in_{slew}, out_{load}) \approx$ ⊞
$delay_{in->out}(in_{slew}, out_{load}) \approx$ ⊞
in
u1a
u1
u1o
u2a
u2
u2o
wire
out

(a) Delay table and transition table are associated with a delay arc to approximate delay and slew propagation.



(b) A constraint table is associated with a constraint arc to approximate constraint.

Figure 5: ATM approximates slew, delay, and constraint propagation paths to 2-D lookup tables.

group, we create a constraint arc from clock to its *source* pin if there is any FF:D in its group member. In Figure 4 (d), although there are two FF:D in the group of the output pin of $u4$, ETM simply creates one constraint arc to represent the constraint from the same clock. This constraint arc represents the constraint from the first stage registers, D_2 and D_3, to the output pin of $u4$. The delay arcs are extracted from *source* to each checkpoint and each primary output in group member.

Lemma 3.1. *There are no multiple output load variants and multiple input slew variants in our extracted timing arcs.*

Proof. The timing arcs extracted by ATM do not cross group. Therefore, there are no checkpoints on the paths between the endpoints of the extracted arcs, and the timing along the paths is only impacted by the input slew and output load. □

Based on Lemma 3.1, ATM can accurately model each timing arc into a function with two variables. The timing information propagates properly from primary inputs through checkpoints to primary outputs. As a result, we solve the multiple output load variants and multiple input slew variants in ATM.

3.7 Lookup Table Index Selection

Herein, we introduce our lookup table index selection method. Timing propagating through a cell arc makes the propagation become a non-differentiable function, because the timing information of standard cells is stored in nonlinear delay lookup tables. We first pick up the non-differentiable points of the propagation function. While the spirit is similar to [2, 3], our algorithm is different in efficiently inserting extra indices to reduce the interpolation error from nonlinear functions. As mentioned in Section 3.1, even though we select non-differentiable points to be table indices, we may still lose accuracy.

3.7.1 Transition Table. The two indices of the transition table are the slew of the input pin and the load of the output pin. Figure 5 (a) shows an example about how we select these indices. In this example, we extract a delay arc from in to out. The slew propagation is equal to the slew output of the last wire. The equation can be written as:

$$slew_{in->out}(in_{slew}, out_{load}) = wire(u2o_{slew}, out_{load})$$

$u2o_{slew}$ can be replaced with the slew output of $u2$ cell.

$$u2o_{slew} = lookup(u2a_{slew}, u2o_{load} + out_{load})$$

where $lookup$ is a function interpolating the lookup table of cell $u2$. Due to the shielding effect, we could assume $u2a_{slew}$ is a constant and $lookup$ is a function only related to the output load. Based on Equation 1, the equation can be summarized as:

$$\begin{aligned} & slew_{in->out}(in_{slew}, out_{load}) \\ =& wire(lookup(u2o_{load} + out_{load}), out_{load}) \\ =& \sqrt{lookup(u2o_{load} + out_{load})^2 + impulse(out_{load})^2} \end{aligned} \tag{2}$$

$slew_{in->out}$ is a nonlinear piecewise function and we derive the non-differentiable points from load indices of $lookup$. We shift the load indices of $lookup$ by $u2o_{load}$ to get the load indices of $slew_{in->out}$:

$$\{index - u2o_{load} | index \in \text{load indices of } lookup\}$$

Slew indices are redundant since the equation is only related to out_{load}.

3.7.2 Delay Table. We decompose $delay_{in->out}$ into two parts, $delay_{in->u2a}$ and $delay_{u2a->out}$.

$$\begin{aligned} delay_{in->out}(in_{slew}, out_{load}) = delay_{in->u2a}(in_{slew}, u2a_{load}) + \\ delay_{u2a->out}(u2a_{slew}, out_{load}) \end{aligned} \tag{3}$$

Since $u2a_{load}$ and $u2a_{slew}$ are constants, the first part and second part are only sensitive to in_{slew} and out_{load}, respectively. For the first term, according to Elmore delay, wire delay is only related to the load but not slew. Thus, the wire delay in the first term is constant because the load of the internal pin is fixed and we only concern the cell arc. The input slew influences a few cell arcs because the minimum and maximum slews converge after several stages. We approximate $delay_{in->u2a}$ to the delay of a first cell arc plus a constant by considering the delay of cell arc as constant, except for the first one. The details are as follows:

$$\begin{aligned} delay_{in->u2a}(in1_{slew}) &\approx lookup(u1a_{slew}) + C \\ &= lookup(\sqrt{in_{slew}{}^2 + impulse(u1a_{load})^2}) + C \end{aligned} \tag{4}$$

where $lookup$ is a function interpolating the lookup table of cell $u1$. We derive the non-differentiable points from the slew indices of $lookup$:

$$\{\sqrt{index^2 - impulse(u1a_{load})^2} | index \in \text{slew indices of } lookup\}$$

For the second term, $delay_{u2a->out}$ is a summation of a cell delay and a wire delay. According to Elmore delay model, wire delay from $u2o$ to out is a linear function of out_{load}.

$$\begin{aligned} & delay_{u2a->out}(out_{load}) \\ =& lookup(u2o_{load} + out_{load}) + (a * out_{load} + b) \end{aligned} \tag{5}$$

where a and b are constants of $wire$, and $lookup$ is a function interpolating the lookup table of cell $u2$. $delay_{u2a->out}$ is a linear piecewise function since it is a linear combination of a linear piecewise function and a linear function. We derive the non-differentiable points by shifting the load indices of $lookup$ by $u2o_{load}$:

$$\{index - u2o_{load} | index \in \text{load indices of } lookup\}$$

3.7.3 Constraint Table. The two indices of the constraint table are the slew of the input pin and the slew of the clock pin. For different modes, different equations are used to obtain the constraint from clock pin to the primary input. For instance, in Figure 5 (b), $constraint_{clk->in}$ can be written as:

- **Hold:** $delay_{clk->ffclk} - delay_{in->ffd} + lookup_{hold}$
- **Setup:** $delay_{in->ffd} - delay_{clk->ffclk} + lookup_{setup}$

Constraint $lookup_{hold}$ and $lookup_{slew}$ are both a function of ffd_{slew} and $ffclk_{slew}$, and we consider them as constants due to the shielding effect. Since the internal load is fixed, $delay_{clk->ffclk}$ and $delay_{in->ffd}$ are a function of clk_{slew} and in_{slew} respectively. Their forms are the same as Equation 4. Therefore, we apply the same selection method to these two functions.

3.7.4 Extra Indices Insertion. In table indices selection, Equations 2 and 4 both show nonlinear propagation. For these nonlinear propagation, we introduce an extra indices insertion method to reduce the interpolation error. We observe that the function in each domain of Equations 2 and 4 are convex. Therefore, when we sample on the non-differentiable points, the interpolation error between two indices is a concave function. For each domain, inserting an extra index to the location of the maximum error can largely reduce the error. We use ternary search to find the maximum error for each interval, and recursively insert indices until the maximum error is less than a user-defined tolerance.

In the previous section, we assume that the input slew cannot influence the output slew, when selecting indices of the transition table. However, we should handle a special case when selecting the indices for transition table on a wire. In Figure 4 (d), when selecting indices of the transition table for a wire from the output of $u2$ to the input of $u5$, we notice that the output slew is both influenced by the input slew and output load. The slew propagation in Equation 1, is plotted in the green surface in Figure 6. To select slew and load indices, we begin with the boundary indices, (0, 0) and (250, 250). The bottom surface shows the interpolation error and red circles indicate the table indices we have selected. In the first iteration, we use 2D ternary search to insert an extra index to the location of the maximum error. After the first iteration, the region is divided into four subregions, and we recursively insert the table indices until the maximum is less than a user-defined tolerance.

4 EXPERIMENTAL RESULT

We implemented our algorithm in C++ language and ran on a machine of Intel Xeon 2.2 GHZ CPU and 96 GB memory. We evaluated ATM on a set of benchmarks released from TAU 2017 Timing Analysis Contest on Macro Modeling [4]. Table1 shows the statistics of the benchmarks. We compared ATM with iTimerM [2], the winner of TAU 2017 Timing Contest, which uses an ILM-based approach to construct a timing model. We also compared with the traditional ETM, where we use PrimeTime [1] to generate models. Comparing with industrial tools may not be fair due to the application scope, but it can be a good reference to highlight our quality. Given the range of input slew and output load, we randomly generated 200 sets of operation conditions to evaluate macro models.

Table 1: Statistics of benchmark.

Design	# PIs	# POs	# Pins	# Arcs
leon2_iccad	615	85	5179094	9111250
vga_lcd_iccad	85	99	768050	1359279
des_perf_ispd	234	140	371587	697145
mgc_matrix_mult_iccad	3202	1600	492568	948154
edit_dist_ispd	2562	12	416609	799167
fft_ispd	1026	1984	116139	226897

# PIs: # of primary inputs; **# POs:** # of primary outputs;

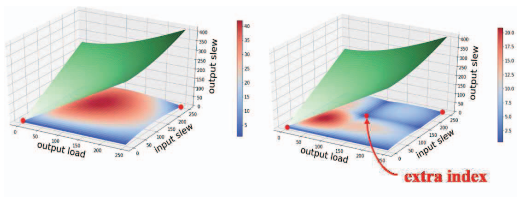


Figure 6: Extra indices to reduce the interpolation error.

4.1 Model Accuracy and Size

Table 2 shows the comparison of accuracy and model size. We measured the accuracy by comparing the difference of boundary timing between the original netlist and the generated timing model under various operation conditions. The boundary timing includes the required arrival time of primary inputs, the arrival time of primary outputs, and the transition time of primary outputs. We use PrimeTime to compare the accuracy of ETM by reporting the boundary timing of the original netlist and ETM. Due to the property of ILM, iTimerM is more accurate. Its maximum absolute error (MAE) is less than 1 ps for all designs. Although ETM produces compact models, it is less accurate. The top two highest MAE, 334 ps and 202 ps, are observed in the design edit_dist_ispd and mgc_matrix_mult_iccad. In contrast, ATM introduces internal pins and arcs to maintain accuracy. Our MAE and average error are all less than 1 ps, very close to iTimerM. While our model size is 1.2× bigger than ETM, it is 96× smaller than iTimerM. However, iTimerM considers CPPR and does not allow multiple arcs between two pins for positive and negative transitions. This may increase the model size. It is not fair to directly compare to iTimerM but it can be a good reference to highlight the accuracy quality of ATM.

4.2 Model Generation and Usage Runtime

All timing models are generated in 4 threads setting. In Table 3, we observe the ETM-based approach is significantly slower than the ILM-based approach in model generation. The similar trend could also be observed in [1]. The most time-consuming part of our approach is to perform timing inference to fill out the timing table. A large amount of paths condensed in a single extracted arc also

Table 2: Accuracy and model size comparison with iTimerM and ETM.

Design	MAE (ps)			Average Error (ps)			Model Size (MB)			Model Size Ratio		
	ATM	iTimerM	ETM	ATM	iTimerM	ETM	ATM	iTimerM	ETM	ATM	iTimerM	ETM
leon2_iccad	0.24	0.14	3.68	0.04	0.00	0.89	3.45	896.43	4.91	1.00	260.11	1.43
vga_lcd_iccad	0.16	0.13	3.70	0.04	0.00	0.27	0.42	112.87	0.66	1.00	270.67	1.57
des_perf_ispd	0.36	0.09	99.53	0.09	0.02	22.73	4.13	88.86	2.37	1.00	21.50	0.57
mgc_matrix_mult_iccad	0.45	0.37	202.71	0.13	0.01	3.55	22.21	352.47	15.33	1.00	15.87	0.69
edit_dist_ispd	0.36	0.20	334.47	0.08	0.02	68.32	18.96	155.24	12.65	1.00	8.19	0.67
fft_ispd	0.40	0.17	146.10	0.06	0.02	9.47	148.36	229.52	14.11	1.00	1.55	0.10
Average	0.33	0.18	131.70	0.07	0.01	17.54				1.00	96.32	0.84

MAE: Maximum absolute error; **Model Size**: size of min and max liberty files.

Table 3: Runtime comparison with iTimerM and ETM.

Design	Generation Runtime (s)			Usage Runtime (s)				Usage Runtime Ratio			
	ATM	iTimerM	ETM	Flat	ATM	iTimerM	ETM	ATM	Flat	iTimerM	ETM
leon2_iccad	1158.19	233.07	3129.04	87.76	<0.1	31.44	<0.1	1.00	1253.71	449.14	1.00
vga_lcd_iccad	66.10	20.04	359.30	9.77	<0.1	3.42	<0.1	1.00	488.50	171.00	1.00
des_perf_ispd	213.23	15.98	138.87	8.29	<0.1	2.54	<0.1	1.00	103.63	31.75	1.00
mgc_matrix_mult_iccad	557.73	36.30	9740.32	10.56	0.39	8.74	0.30	1.00	27.08	22.41	0.77
edit_dist_ispd	1040.72	25.65	4365.01	10.47	0.31	5.00	0.21	1.00	33.77	16.13	0.68
fft_ispd	178.73	22.20	220.57	3.18	4.69	7.19	0.25	1.00	0.68	1.53	0.05
Average Ratio								1.00	317.90	115.33	0.75

contributes to the model generation time. However, this process is usually done at once in the flow. We next compare the model performance measured by OpenTimer [5]. Table 3 shows the results. "Flat" indicates the runtime to complete the timing analysis on the original netlist. In general, our timing model outperforms iTimerM across all designs. On average, our model is 115× faster than iTimerM and 317× faster than the original design.

4.3 Tradeoffs between Model Accuracy and Size

In this experiment, we show tradeoffs between the model size and accuracy. We focus on the design fft_ispd which is very sensitive to the input slew and output load. Most pins and arcs are preserved both in our timing model and iTimerM for accuracy. In ATM, there are two tradeoffs. The first one is the slew convergence threshold to decide if a pin is a dirty pin. This threshold has a profound effect on the number of checkpoints. A low threshold means that pins are hard to be considered as converge, and we thus need to insert more checkpoints. As a result, a low threshold benefits accuracy but not model size. We experimented with different slew convergence threshold and the result is shown in Figure 7. The other tradeoff is the error tolerance when we insert extra indices for reducing interpolation error. A low error tolerance makes ATM insert more extra indices. Therefore, ATM can obtain more accuracy but result in large table size. As shown in Figure 7, when we allow high error tolerance, MAE increases, and model size decreases.

5 CONCLUSION

To speed up hierarchical timing analysis, we presented ATM, a high accuracy extracted timing model. ATM addresses the accuracy issues of ETM without increasing the unreasonable model size. Our key idea is to insert a small amount of checkpoints to improve accuracy. In our table indices selection method, we prevent redundant table entries and insert essential indices for accurate interpolation from non-linear timing propagation.

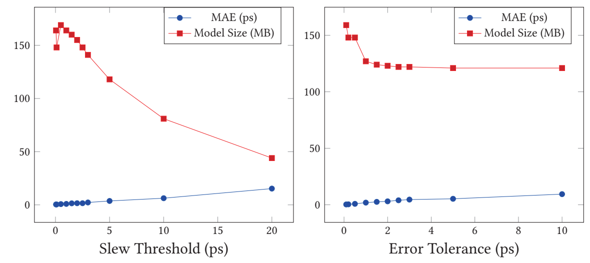


Figure 7: Tradeoff between model accuracy and model size on the slew convergence threshold and error tolerance.

REFERENCES

[1] Ajay J. Daga, Loa Mize, Subramanyam Sripada, Chris Wolff, and Qiuyang Wu. Automated timing model generation. DAC '02, page 146–151, 2002.
[2] Pei-Yu Lee, Iris Hui-Ru Jiang, and Ting-You Yang. iTimerM: Compact and Accurate Timing Macro Modeling for Efficient Hierarchical Timing Analysis. ISPD '17, page 83–89, 2017.
[3] Tin-Yin Lai, Tsung-Wei Huang, and Martin D. F. Wong. LibAbs: An Efficient and Accurate Timing Macro-Modeling Algorithm for Large Hierarchical Designs. In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, 2017.
[4] TAU 2017 Timing Contest on Macro Modeling, 2017. https://sites.google.com/site/taucontest2017/.
[5] Tsung-Wei Huang and Martin Wong. OpenTimer: A high-performance timing analysis tool. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 895–902, Nov 2015.
[6] C. V. Kashyap, C. J. Alpert, F. Liu, and A. Devgan. Closed-form expressions for extending step delay and slew metrics to ramp inputs for rc trees. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 509–516, 2004.