FAST_MET: A Fast and Accurate Tool for Multiple Event Transients

Adam Watkins*† and Spyros Tragoudas*
* Southern Illinois University Carbondale
Carbondale, IL 62901
† Los Alamos National Laboratories
Los Alamos, NM 87545
acwatkins88@lanl.gov, spyros@siu.edu

***Abstract*—The analysis of the soft error rate of a circuit has continued to be a difficult problem due to the calculation of the logical effect on a pulse generated by a radiation particle. The most common existing methods to determine logical effects use either exhaustive input pattern simulation or binary decision diagrams. The problem with both approaches is that simulation of the circuit can intractably time consuming. To solve this issue, a simulation tool is proposed which employs partitioning to reduce the simulation time and space overheads associated with the use of binary decision diagrams. In addition, the tool integrates an accurate electrical masking model. Compared to existing simulation tools, the proposed tool can simulate larger circuits faster.**

I. Introduction

As process technology continues to the scale down, the likelihood of a radiation induced error increases. This trend provides a need for accurate and efficient methods to calculate the soft error rate of a given circuit. Since it is expensive and time consuming to design and circuit and test at a later time, efficient tools that can accurately determine the error rate for a given netlist before fabrication can significantly reduce the design time. However, it has proven to be very difficult to characterize combinational circuits due to three pulse masking factors. The masking factors are as follows:

1) Logical Masking: The pulse is removed due to the boolean inputs on the gate during pulse generation or a controlling value on an off-input during pulse propagation.
2) Electrical Masking: The first order RLC effects within the gate cause the pulse to degrade and fully attenuate.
3) Temporal Masking: The pulse arrives at a flip-flop but does not meet the set-up and hold time parameters.

Considering the above factors, all current soft error simulators consist of modeling a particle at a specific energy which is used in an equation to model the error pulse shape [23]. Once the pulse is generated, it is propagated through the combinational logic to the output. It is then assumed that the output is connected to a flip-flop thus the set-up and hold times are considered. It has been shown in [10], [11] that concurrent estimation of all masking factors is required in order to ensure that the soft error rate is calculated accurately. However, due to limitations on simulation time and available memory, efficient and accurate consideration of all masking factors has proven to be a difficult and ongoing problem.

There have been many approaches to improve the estimation of the three factors. First the existing approaches to the estimation of logical masking will be discussed. The most common method to calculate the logical masking effects is the use of a Monte Carlo based simulation which consists of randomly applying patterns as in [19], [21], [13], [17], [16]. While this approach is very easy to implement, it suffers from accuracy and run time issues when the circuit has more than 30 inputs. In the case of large circuits, the simulator must resort to generating random input patterns which cover only a small subspace of the possible inputs. This implies that the error in logical masking estimation increases with the circuit size.

To enhance on Monte Carlo based simulation, the authors in [22] propose the use of probabilistic transfer matrices (PTM). This approach consists of representing the Boolean functions within a matrix in which Boolean operations can be applied. While being very accurate, the use of PTMs are non-ideal since they have high memory and simulation time costs that explode on small circuits. The authors in [7] propose an enhanced alternative to the PTM method which uses stochastic logic to calculate the signal probabilities for logical masking estimation. While their method is fast, it relies on the use of random input patterns that can limit the accuracy of the calculation.

Another common approach found in [2], [8] uses the correlation coefficient method (CCM) proposed in [4]. This method uses basic gate signal probability functions to determine the output. For example, the probability of an AND gate is given as $P(O) = P(A)P(B)$ where $P(O)$ is the output probability, $P(A)$ is the probability of input A being "1" and $P(B)$ is the probability of inputs B being "1". A drawback with using the basic probability functions is that they are unable to consider the correlations between signals. CCM improves on this by add a correlation factor to estimate the signal probability. While CCM does provide quick estimation of the logical masking effects with virtually no memory overhead, it can only estimate first order correlation. This implies that as the circuit gets larger, the accuracy of the method substantially decreases.

The last approach discussed in this section, which is of most concern in this paper, is the use of binary decision diagrams (BDDs). BDDs provide a very accurate way to determine the logical masking effects since they are a condensed canonical form of a Boolean function. The main problem with BDDs however, is that they scale exponentially with circuit size thus leading to the possibility of blow up on medium

to large circuits. Despite this problem, there have been many proposed simulators that the use BDDs [20], [10], [11]. The main advantage to the use of BDDs is that all input patterns can be considered concurrently thus allowing for exact calculation of logical masking probabilities.

To alleviate the inherent blow up problem, the authors in [20] propose the use of partitioning. In their work, they show that the use of partitioning allows for much faster calculation of the soft error with no threat of blow up at a cost of accuracy. The authors in [11] also investigate partitioning but do not give an in depth study on how partitioning effects the output error. In both approaches, the circuit is partitioned before simulation with each partition size being set to a predetermined value. This is problematic since the size may be much smaller or larger compared to the given time, processing power and memory. For this reason, this section proposes the use of an adaptive partitioning algorithm that scales the partition size based on the number of nodes. It is shown that the adaptive method allows for faster and more accurate estimation of the logical masking effect compared to the non adaptive approach.

In addition to the adaptive partitioning approach, the proposed simulator has the capability of considering multiple concurrent pulse strikes commonly referred to as multiple event transients (METs). The consideration of multiple strikes has become of greater concern due to the small feature size, leading to a lower critical charge for upset, and the closer placement of transistors [18]. These type of events can occur in two forms: single event multiple transient (SEMT) and multiple event multiple transient (MEMT). A SEMT is defined as the case where a single radiation particle upsets multiple transistors causing multiple transients pulses. A MEMT is defined as when multiple particles hit different circuit components simultaneously. While the probability of the SEMT vs the MEMT depends strongly on the radiation environment, methods developed for one type of error can easily be modified to consider the other.

There are a few existing methods that include the consideration of multiple event errors. The authors in [11] consider the correlation of METs however their method uses BDDs which tend to blow up. While they do investigate the use of partitioning, their method is only tested for small circuits using only a few concurrent pulses. An additional simulation tool used for MET analysis found in [5] uses probabilistic functions. This approach has shown to be fast and efficient but is not suited for the use of complicated electrical masking methods. To improve on the existing methods, the simulator FAST_MET (Fast and Accurate Simulation Tool for Multiple Event Transients) which uses BDDs to calculate the logical maksing effect in conjunction with the accurate pulse approximation tool proposed in Chapter **??**. FAST_SET improves on the method in [11] by allowing for the integration of accurate electrical masking models and through the extensive use and evaluation of partitioning to avoid BDD blow up.

II. Preliminaries

A crucial aspect in the estimation of the logical masking effects is accurate creation of the Boolean functions. To ensure full consideration of the logical masking effects, Boolean functions must be created to evaluate the circuit inputs for pulse generation and the off-inputs for pulse propagation. First the method to generate the functions for error pulse generation will be discussed.

When a high energy particle strikes a transistor, the magnitude and polarity of the pulse will depend on the configuration of the transistor. In the case of CMOS logic, a transient pulse is generated when a particle hits a blocking transistor. This will, in turn cause the transistor to temporarily conduct current allowing for the generation of the voltage pulse. In Fig. 1 a 2 input NAND gate is given to demonstrate this mechanism. In the NAND gate, a particle can hit a single PMOS or a NMOS transistor to cause an error. However, since the transistor must be blocking, the input pattern has a large effect on where the error will occur and the polarity of the pulse.

For example, in the 2 input NAND gate in Fig. 1, if both inputs have a value of "1", a strike on either of the PMOS transistors will likely cause a transient pulse. In the case of one input having a "0" value and the other a "1" value, a strike on the blocking NMOS transistor will cause an error. In the case of both inputs having a "1" value, the gate will only generate a transient pulse if both NMOS transistors are struck concurrently. Addition, the polarity of the pulse also requires consideration. If the output is has a "1" value, the pulse will start at a high value, go low and back to a high value. When the output has a "0" the pulse will start at a low value, go high and back to a low value. Based on this observation, the function for the pulse generation for the AND, NAND, OR and NOR are given in Table I for a N number of inputs with each input denoted as I_i.

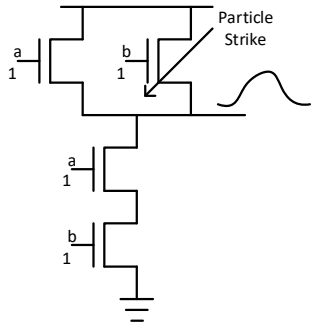

Fig. 1. A low-high-low transient pulse being generated by a strike on the PMOS.

TABLE I. Functions for Transient Pulse Generation

Gate	Polarity	Generation Function ($F(G)$)
AND	Rising	$F(G) = \neg(I_1 \wedge I_2 \wedge ...I_N)$
AND	Falling	$F(G) = I_1 \wedge I_2 \wedge ...I_N$
NAND	Rising	$F(G) = I_1 \wedge I_2 \wedge ...I_N$
NAND	Falling	$F(G) = \neg(I_1 \wedge I_2 \wedge ...I_N)$
OR	Rising	$F(G) = I_1 \wedge I_2 \wedge ...I_N$
OR	Falling	$F(G) = \neg(I_1 \wedge I_2 \wedge ...I_N)$
NOR	Rising	$F(G) = \neg(I_1 \wedge I_2 \wedge ...I_N)$
NOR	Falling	$F(G) = I_1 \wedge I_2 \wedge ...I_N$

For the consideration of the generation of METs, the above equations must be modified such that the joint probability

of the error occurring is considered. In the case of pulse generation, it is assumed that a single radiation particle will strike two or more gates causing transient pulses. As stated previously, in order for a pulse with a specific polarity to be generated, the output of the gate must have a specific value. Based on this, the probability of pulses being generated (P_{gen}) assuming strikes N gates with sufficient energy is given in equation 1 where $P(G_{i,p})$ is the probability that gate G_i is capable of generating a pulse of polarity p. If the pulse generated is rising, $G_{i,p}$ pertains to the probability that the output of $G_{i,p}$ is "0". If the pulse is falling, $G_{i,p}$ is the probability of the output being "1".

$$P_{gen} = \prod_{i=1}^{N} P(G_{i,p}) \tag{1}$$

A particle hitting two transistors simultaneously allows for multiple cases since the input may be such that the struck gate is not blocking. Based on this, the total number of possible ways a strike can produce a pulse (P_{num}) is given in equation 2 assuming a N number of struck gates and r as the number of gates that produce a transient pulse.

$$P_{num} = 2 * \sum_{r=1}^{N} \frac{N!}{(N-r)!} \tag{2}$$

Equation 2 is based on the fact that multiple pulses are not always generated when a particle strikes both transistors. As stated previously, the polarity and existence of the pulse depends on the input patterns. If one assumes that the particle hits a N number of gates, this creates a P_{num} number of cases where $P(C_i)$ represents the probability of case i calculated using equation 1. The total probability of the inputs having the values to generate a SET or MET P_{gen} is given below:

$$P_{gen} = \sum_{i=1}^{P_{num}} P(C_i) \tag{3}$$

In the case of a pulse being propagated to the input of gate, the pulse will be masked if one of the off-inputs has a controlling value. Fig. 2 gives a NAND gate with one of the inputs having a pulse and the other input having a controlling "0" value. In this case, the output will be a logical "1" value despite the pulse on the input.

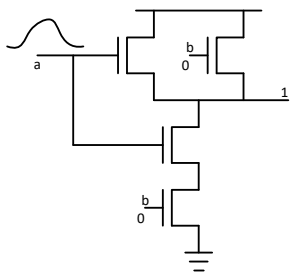

Fig. 2. A transient pulse being masking by a controlling value on the off-input.

To calculate the logical masking factor during pulse propagation, the functions of the off-inputs are calculated using the logical AND operation. Assume that $F(OI_i)$ is the Boolean function of the off-input OI_i representing the input patterns that allow a non-controlling value and there are a N number of inputs, the function representing the case where the pulse is propagated $F(M)$ is given below:

$$F(M) = F(OI_1) \wedge F(OI_2) \wedge ...F(OI_N) \tag{4}$$

In equation 4 only $N-1$ inputs are considered since the input which contains the transient pulse is not included in the calculation.

For the consideration of METs, two or more pulses may arrive at a gate. If the two pulse case is assumed, there are 3 possible ways that the pulses can be propagated: the pulse on a is propagated and b is masked, the pulse on b is propagated and a is masked and, lastly, the both pulses are propagated simultaneously providing an output pulse. Fig. 3 provides an illustration of the cases.

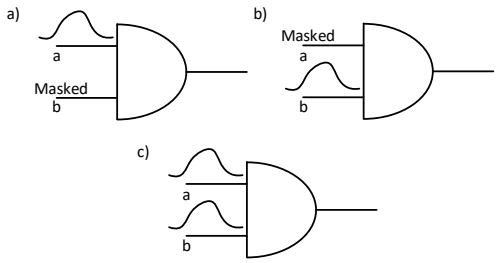

Fig. 3. (a) A transient pulse arriving on input a and masked on b. (b) A pulse arriving on input b and masked on a. (c) Two pulses arriving on the inputs simultaneously.

Table II gives the propagation functions $F(G)$ for each case with $F(a_p)$ and $F(b_p)$ being the functions representing the input values that allow for the pulse to be propagated to inputs a and b respectively. While this example is only for a two input gate, larger gates can be considered similarly.

TABLE II. FUNCTIONS FOR A MET ARRIVING ON THE INPUTS OF A NAND GATE

Case	Propagation Function ($F(G)$)
a	$F(G) = F(a_p) \wedge \neg F(b_p)$
b	$F(G) = \neg F(a_p) \wedge F(b_p)$
c	$F(G) = F(a_p) \wedge F(b_p)$

Based off equation 2, the total number of cases can be calculated as the number of combinations assuming $i = 1, 2, ...N$ simultaneous pulses. Assuming that P_{num} is the number of possible cases and r_i represents a i number of concurrent pulses, the equation for the total number of propagation cases is given below:

$$P_{num} = \sum_{i=1}^{N} \frac{N!}{(N-r_i)!} \tag{5}$$

Based on equation 5, the total probability of multiple convergent pulses at a gate G ($P(G_{tot})$) with the assumption that $P(G_i)$ is the probability of case i on gate G is given below:

$$P(G_{tot}) = \sum_{i}^{P_{num}} P(G_i) \tag{6}$$

When a MET is generated, it is represented as an event E_k with k being the event number. For each event, the generated transients are propagated through gates to the primary outputs. Once the pulse arrive at the gate output, the temporal masking probability can be considered. Based off the equation found in [15], let $P_{L,i}$ is the probability of the pulse being latch on gate i, W be the pulse width, t_{setutp} and t_{hold} being the setup and hold times respectively and T_{clk} be the clock period, the probability of a pulse being latch on an output flip-flop is calculated in the below equation:

$$P_{L,i} = \frac{W - (t_{setup} + t_{hold})}{T_{clk}} \tag{7}$$

Assuming that the pulses from event E_k propagate to a M number of primary outputs with each gate having a probability of $P_{L,i}$ calculated as in equation 7, the probability of error for event E_k is calculated as the following:

$$P(E_k) = \sum_{i=1}^{M} P_{L,i} \tag{8}$$

To evaluate the error probability for all events, the mean error susceptibility (MES), defined in [11] is used. The MES represents the average probability of error for all events combined. Assuming that there are a n_E number of events, n_d number of input probability distributions and that K is the number of injected events, the MES is found using the following equation:

$$MES = \sum_{k=1}^{K} \frac{P(E_k)}{n_E n_d} \tag{9}$$

Based off the MES, the soft error rate (SER) can be calculated as in below where R_{eff} represents the effective concentration of particle in the given area, P_{eff} is the probability of a particle hitting the sensitive region of a transistor and A as the area of the sensitive volumes.

$$SER = MES * R_{eff} * P_{eff} * A \tag{10}$$

III. Description of the FAST_MET Simulator

FAST_MET is a fast and accurate tool to estimate the soft error rate. The method aims to provide a good trade-off between simulation time and accurate SER calculation. To accurately determine the logical masking effects, BDDs are used. A BDD is a tree structure which can be used to represent a Boolean function. The first instance of the use of BDDs for function representation was proposed in [1] and has continued to be used regularly in circuit simulation and synthesis since. Fig. 4 gives the representation of a AND gate as BDD. Note that the structure consists of nodes which represent variables. For each node, there are two branches, one denoting a "true" value and the other a "false" value. The BDD graph also consists of multiple paths to two terminating nodes, one with a "true" value denoted by "1" and one with a "false" value denoted by "0".

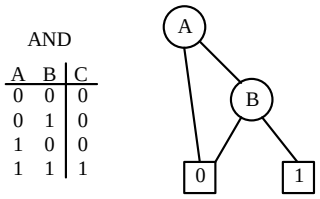

Fig. 4. The truth table and corresponding BDD for an AND gate.

The main advantage of a BDD is that they provide an efficient canonical form to represent a Boolean function. Specifically, this means that Boolean operations (such as AND, NAND, OR ... etc) can be easily and quickly applied to the graph. In the case of the FAST_MET simulator, BDD representation of the functions are used to both calculate the signal probability and the probability for pulse propagation.

In FAST_MET, the goal is to accurately and efficiently represent a sensitization function as a BDD. A sensitization function is defined as the Boolean function that allows for a generated transient pulse to propagate through a circuit. In other words, a sensitization is the function that represents all input patterns that sensitize the pulse to the output.

The FAST_MET simulator operates in a topological order. This implies that the primary inputs will be visited first. Once a primary input is visited, it is represented as a variable for in a BDD. Once all inputs are visited, the simulator starts on the gates within the circuit. When a gate is visited, the sensitization function representing the gate output is calculated. The "1" terminal on this BDD represents the Boolean function that makes the output a "1" value. To increase efficiency, the FAST_MET simulator generates a rising and falling pulse at each gate using the method proposed in Chapter **??** for a specific energy.

When a pulse is generated, the gate input functions are considered using the input sensitization functions. If the pulse is rising (starting from a low value then going high), the pulse is only generated when the output is low. To consider this, the BDD function is inverted such that a termination to "1" represents that case at which the pulse exists. Similarly, if the pulse is falling, the output must be a high value. Since the function already denotes a high value on the output, the sensitization function can be used directly. Additionally, as discussed in Section II, in the case of a gate having multiple inputs, the output function must be considered according to Table I. Using the table, the correct generation function can be found by applying the function for the gate specified. Fig. 5 provides an example of the creation of the generation functions for a gate.

To consider the MET case, the location and correlation

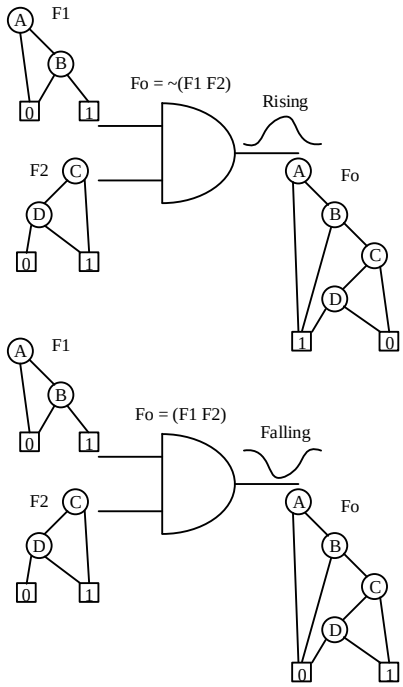

Fig. 5. An example of the BDD creation for a rising and falling pulse.

between the pulses must be considered. As discussed in [3], layout information is important in accurately modeling the location of the MET for a real design since gates that are near each other in a netlist may not necessarily be close in the layout. In this section, for the sake of simplicity, METs are generated using close proximity nodes from the netlist. For the sake of simple comparison and analysis, using the netlist is not problematic.

To determine the logical masking effects of a MET, the correlation between the inputs of both struck gates are found by using the logical AND operation between the generation functions derived as in Table I. The logic behind this operation is that all pulses must be logically sensitized concurrently. Since the functions may share common inputs, they may have dependencies that must be considered.

In the case of pulse propagation, the sensitization functions are represented as BDDs. As discussed in Section II, a pulse will propagate through a gate if all off-inputs have a non-controlling value. As can be observed in equation 4, the functions representing the case at which the off-input has a non-controlling value are logically AND'd together. The basis behind this idea is that all off-inputs must be non-controlling in order for the pulse to be propagated.

To consider pulse propagation in the FAST_MET simulator, the logic functions are first represented in a BDD. A logic function is the Boolean function which provides the variable states that allow a "1" and a "0" value. The sensitization function for the gate is then determined by using the logical AND operation on each off-input BDD. If the non-controlling value for the gate is a "0", such as in a OR gate, the BDDs are inverted so that the "1" terminal represents the case at which the pulse propagates. If the non-controlling value is a "1" the BDDs are left unchanged. The result of this operation is a single BDD in which the paths that lead to the "1" terminal represent all of the input patterns that allow the pulse to propagated while the paths that lead to the "0" terminal represent the patterns the will mask the pulse.

In the case of multiple pulses arriving at a gate simultaneously referred to a convergent pulses, the off-inputs are considered as discussed previously. As shown in Table II, a MET can result in multiple output cases which must be considered. The first case is where only one pulse will propagate which is calculated as previously discussed. All remaining cases include the circumstances where multiple pulses will arrive at the gate input simultaneously. In these cases, the logic function is determined by using the AND operation on the input function which have a pulse. The resulting function from this operation is then AND'd with the non-controlling off-input logic functions. The basis behind this idea is that the pulse sensitization functions give the cases where a pulse will arrive at the gate input. If two pulses arrive simultaneously, both pulses must be sensitized concurrently. The logical AND states that both functions must evaluate to true while all off-inputs must be non-controlling for the pulse to be propagated.

IV. FAST_MET Simulation Flow

The goal of the FAST_MET simulator is to calculate the SER of a circuit. To do this, the simulator must inject pulses in each gate. The main difficulty for this type of simulation is that for even relatively small circuits, it may take an intractable amount of simulation time and memory to consider all pulses. The FAST_MET simulator provides a good trade off compared to existing methods since it uses the pulse approximation method in Chapter **??** in conjunction with BDDs to accurately and quickly determine the SER.

The first step for the FAST_MET simulator is to partition the circuit. As shown in [20], [11], the use of BDD functions can lead to memory blow up. To alleviate this issue, the FAST_MET partitions the circuit into a user defined number of parts. It is shown in Section V that the use of partitioning can substantially reduce the simulation time at a cost of accuracy in the SER estimation. For this work, the circuits were partitioned using the Fiduccia and Mattheyses (FM) algorithm [6] which allows for a circuit to be partitioned into two parts in linear time. To achieve a k part partition, the FM algorithm can be applied recursively a k-1 number of times. This algorithm was chosen based off its linear time complexity and relatively easy integration into the FAST_MET tool. Through proper integration, one can also use other partitioning tools such as hMetis [9]. Partitioning the circuit improves the performance by reducing the size of the logic and pulse sensitization BDDs. For moderately large circuits or larger, partitioning must be used since the BDD functions are too large and complex for realistic simulation.

After partitioning, the FAST_MET simulator processes the primary inputs of the circuit. At each input node, the signal probability is set and the BDD variables are initialized such that each input is a variable. In this work, it is assumed that there are not pulses on the inputs, however, especially if the method is used for the consideration of sequential circuits such as in [12], pulses can be injected on the inputs. This is due to

the inputs being connected to the output of a flip-flop which drives the circuit.

After the inputs are processed, the internal nodes are visited in a topological order. At each node, at least one pulse is injected for a given charge. The particle effect is modeled as a current pulse using equation **??**. A current source is placed on the output of the gate to simulate the particle effect and the resulting pulse shape is determined using the method discussed in Chapter **??**. Multiple pulses can be injected concurrently in a gate, however, the simulation time and memory resources increase substantially as more pulses are injected.

Additionally, FAST_MET is able to simulate the effects of a MET. To accommodate this, each injected pulse or MET is given an event number k to create an event E_k. This is important since FAST_MET injects pulses at each gate concurrently to speed up the SER characterization. Thus, when an MET is generated, a pulse is created on two or more gates which all have the same event number.

Once the pulses are injected and the corresponding BDD function are determined, FAST_MET propagates all pulses found on the gate inputs. Each fanin gate is checked individually for pulses. If a pulse is found the propagation function is determined as in Section III. If the function does not evaluate to a false value, implying that there are no input patterns that allow the pulse to be propagated, the pulse is not considered further. However, if the function is not false, the output pulse shape is calculated. This operation continues for each pulse found on the inputs.

When a node is on the edge of a partition circuit, the effects of the previous partition must be considered. To do this, each node on the output edge of a partition is evaluated to determine their probability. This value is stored with the pulse which is propagated between the partitions. Since a partition may be separated from the primary inputs that drive it, virtual inputs are created which contain all pulses and their probabilities and the probabilities from the logic functions of the gate that was connected to input on the previous partition. Using these values, the probabilities are multiplied by the evaluated probability of the function to determine the overall probability. This is demonstrated in Fig. 6.

An issue that was found through numerous simulations using this method is that the functions will not evaluate to a false value even if the pulse does not propagate since a single partition does not consider the whole circuit. This, in effect, substantially reduces the number of pulses that are attenuated which can offset any improvement increases gained through partitioning. To solve this issue, it was found that setting a minimum probability threshold (P_{thr}) in which only pulses with a probability greater than this number are considered substantially saved simulation time with a slight increase in estimation error.

Once each individual pulse is propagated, a routine searches the inputs pulses and organizes them into a has table based on their event number. For each event E_k, the pulses are checked for overlap. If the pulses do overlap, they are treated as convergent pulses and are considered as previously discussed. Once issue that arises during this step is that there may be many possible cases in which the pulses overlap. For example, there may be 2, 3 or more concurrent overlapping

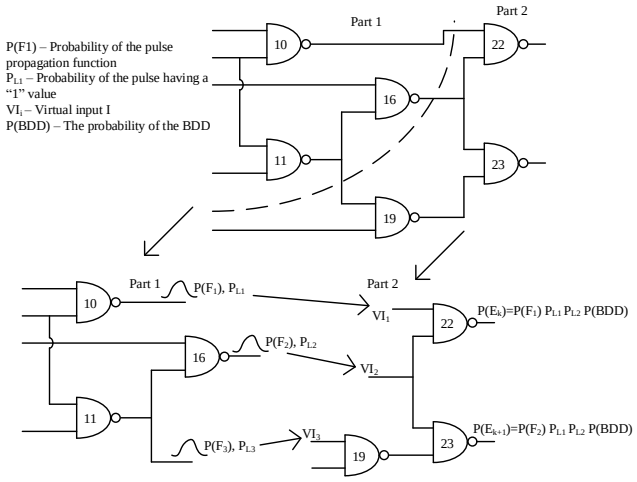

Fig. 6. An example on c17 showing how the pulses are propagated between partitions.

pulses. In this work, to reduce the simulation time, only two pulses are considered. While the electrical masking model can accurately consider the effects of 3 or more convergent pulses, it is assumed that the likelihood of the pulses propagating is low.

Lastly, when the simulator processes a circuit or partition output, the BDD functions for each pulse are solved to determine the probability. Using the probability the MES is determined as discussed in II. An overview of the whole simulation flow is given in Algorithm 1.

Algorithm 1 FAST_MET

```
Set Process Parameters
Parse Netlist
Organize Circuit Topologically
Parse Circuit Into k Parts
for Each Partition do
    Extract Partition
    Set Virtual Inputs
    for Each Gate do
        Generate Pulse
        Generate Sensitization Function
        Propagate Pulse
        Calculate Convergence
        if Primary Output then
            Calculate P_{L,i}
        else if Edge of Partition then
            Solve BDD Probability
        end if
    end for
end for
```

V. Results

The FAST_MET simulator was test on the ISCAS 85 combinational benchmark circuits. FAST_MET was implemented in C++ on a machine with a quad core i7 with superscaling and 16GB of RAM. The transistor look-up tables were characterized in HSPICE using the 32nm Predictive Technology

Algorithm 2 Generate Pulse

```
F = Calculate Generation Function
if F != false then
    Calculate Pulse Shape
    Add To Pulse List
end if
```

Algorithm 3 Propagate Pulse

```
F_sens = Sensitization Function
for Each Input Pulse i do
    F_sens = F_sens(i) ∧ F_sens
end for
if (F_sens != false) ∧ (Solve BDD(F_sens) > P_thr) then
    Calculate Pulse Shape
    Add To Pulse List
end if
```

Model (PTM) [22] where V_{dd} was set to 1.05V. The unit gate capacitance was set to a constant capacitance of 2 fF to simulate the loading effects of the gates and interconnects. To calculate the MES, a single pulse with an energy of 15 fC was applied to each gate using equation **??** with τ being set to $32x10^{-15}$. It was assumed that each output is connected to a flip-flop implemented in the same process library with a setup time (t_{setup}) of 22 ps and a hold time (t_{hold}) of -7 ps in accordance to [14]. For all simulations, the probability of MET was set to 10% and the error threshold for simulation P_{thr} was set to 0.025 when partitioning was used. When an MET occured, it was injected to a neighboring gate based on the netlist.

First, the accuracy of FAST_MET without partitioning is compared to Monte Carlo simulation on c17 and c880 to ensure that the probablistic functions provided here exactly compute the logical masking effect. To ensure that both simulators use the same pulses, the pulse injection routine in FAST_MET was moved to before the main simulation loop. Larger circuits were not tested since they lead to long simulations times and lead to the possibility of memory blow up for FAST_MET without the use of partitioning. As can be observed in the Table III, FAST_MET provides exact estimation of the output error with an over 5x speedup.

The FAST_MET simulator can provide an additional speedup through the use of partitioning. Table IV provides the MES and the simulation time for various ISCAS circuits while changing the partition size. According to the results, it shows that partitioning can provide an upward of 20x reduction in simulation time at a cost of accuracy compared to not using partitioning. Based off the results, it is shown that increasing the number of partitions follows the law of diminishing returns. For example in c880, the use of 2 partitions reduces the simulation time by 13x while the use of 4 partitions reduces the simulation time by 18x. While this is still a large decrease in simulation time, the error is increased by 4x. Additionally, it can be seen that in c17 the use of partitioning actually increases the simulation time. This is due to the time to simulate the circuit is less than the overhead incurred by the partitioning algorithm. Based on this result, it is suggested that the partition size is carefully selected to ensure that the error is minimized.

Algorithm 4 Calculate Convergence

```
Load Hash Table For Each Event
for Each Event do
    I = Check For Overlap
    if I == true then
        F = Calculate Convergence Function
        if (F != false) ∧ (Solve BDD(F) > P_thr) then
            Calculate Pulse Shape
            Add To Pulse List
        end if
    end if
end for
```

TABLE III. COMPARISION OF FAST_MET VS MONTE CARLO

Circuit	Simulator	MES	Run Time
c17	Monte Carlo	$3.92x10^{-4}$	0.429
c880	Monte Carlo	$4.15x10^{-4}$	9110.35
c17	FAST_MET	$3.92x10^{-4}$	0.0974
c880	FAST_MET	$4.15x10^{-4}$	1909.37

TABLE IV. FAST_MET PERFORMANCE VS NUMBER OF PARTITIONS

Circuit	Parts	MES	Run Time
c17	2	$3.83x10^{-4}$	0.1377
c880	2	$3.13x10^{-4}$	139.54
c880	4	$8.37x10^{-4}$	107.72
c1355	2	$1.96x10^{-4}$	459.00
c1355	4	$2.85x10^{-4}$	242.60
c1355	8	$3.67x10^{-4}$	167.961

VI. CONCLUSION

In this chapter, the FAST_MET simulator is proposed. First, probabilistic equations were provided which allow for accurate consideration of all propagation effects. These equations were then represented as BDDs and used to determine the logical masking effect. In the results it was shown that the tool provides a good trade-off between simulation time and accuracy due to the use of partitioning and an accurate masking model. Additionally, it is shown that the number of partitions reduces the simulation time up 18x compared to not using partitioning and 90x compared to Monte Carlo simulation.

REFERENCES

[1] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.

[2] Liang Chen, Mojtaba Ebrahimi, and Mehdi B. Tahoori. Cep: Correlated error propagation for hierarchical soft error analysis. *Journal of Electronic Testing*, 29(2):143–158, 2013.

[3] M. Ebrahimi, H. Asadi, R. Bishnoi, and M. B. Tahoori. Layout-based modeling and mitigation of multiple event transients. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):367–379, March 2016.

[4] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco. Estimate of signal probability in combinational logic networks. In *European Test Conference, 1989., Proceedings of the 1st*, pages 132–138, Apr 1989.

[5] M. Fazeli, S. N. Ahmadian, S. G. Miremadi, H. Asadi, and M. B. Tahoori. Soft error rate estimation of digital circuits in the presence of multiple event transients (mets). In *2011 Design, Automation Test in Europe*, pages 1–6, March 2011.

[6] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, DAC '82, pages 175–181, Piscataway, NJ, USA, 1982. IEEE Press.

[7] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi. A stochastic computational approach for accurate and efficient reliability evaluation. *IEEE Transactions on Computers*, 63(6):1336–1350, June 2014.

[8] Ji Li and Jeffrey Draper. Accelerating soft-error-rate (ser) estimation in the presence of single event transients. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 55:1–55:6, New York, NY, USA, 2016. ACM.

[9] Pekka Miettinen, Mikko Honkala, and Janne Roos. Using metis and hmetis algorithms in circuit partitioning.

[10] N. Miskov-Zivanov and D. Marculescu. Mars-c: modeling and reduction of soft errors in combinational circuits. In *2006 43rd ACM/IEEE Design Automation Conference*, pages 767–772, July 2006.

[11] N. Miskov-Zivanov and D. Marculescu. Multiple transient faults in combinational and sequential circuits: A systematic approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(10):1614–1627, Oct 2010.

[12] Natasa Miskov-Zivanov and Diana Marculescu. Mars-s: Modeling and reduction of soft errors in sequential circuits. In *Proceedings of the 8th International Symposium on Quality Electronic Design*, ISQED '07, pages 893–898, Washington, DC, USA, 2007. IEEE Computer Society.

[13] P. C. Murley and G. R. Srinivasan. Soft-error monte carlo modeling program, semm. *IBM J. Res. Dev.*, 40(1):109–118, January 1996.

[14] C. Nunes, P. F. Butzen, A. I. Reis, and R. P. Ribas. A methodology to evaluate the aging impact on flip-flops performance. In *Integrated Circuits and Systems Design (SBCCI), 2013 26th Symposium on*, pages 1–6, Sept 2013.

[15] M. Omana, G. Papasso, D. Rossi, and C. Metra. A model for transient fault propagation in combinatorial logic. In *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pages 111–115, July 2003.

[16] R. Rajaraman, J. S. Kim, N. Vijaykrishnan, Y. Xie, and M. J. Irwin. Seat-la: a soft error analysis tool for combinational logic. In *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*, pages 4 pp.–, Jan 2006.

[17] R. R. Rao, K. Chopra, D. T. Blaauw, and D. M. Sylvester. Computing the soft error rate of a combinational logic circuit using parameterized descriptors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3):468–479, March 2007.

[18] D. Rossi, M. Omana, F. Toma, and C. Metra. Multiple transient faults in logic: an issue for next generation ics? In *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, pages 352–360, Oct 2005.

[19] F. Wang and Y. Xie. Soft error rate analysis for combinational logic using an accurate electrical masking model. *IEEE Transactions on Dependable and Secure Computing*, 8(1):137–146, Jan 2011.

[20] Bin Zhang, Wei-Shen Wang, and M. Orshansky. Faser: fast analysis of soft error susceptibility for cell-based designs. In *7th International Symposium on Quality Electronic Design (ISQED'06)*, pages 6 pp.–760, March 2006.

[21] M. Zhang and N. R. Shanbhag. Soft-error-rate-analysis (sera) methodology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2140–2155, Oct 2006.

[22] Wei Zhao and Yu Cao. Predictive technology model for nano-cmos design exploration. *J. Emerg. Technol. Comput. Syst.*, 3(1), April 2007.

[23] J. F. Ziegler. Terrestrial cosmic rays. *IBM Journal of Research and Development*, 40(1):19–39, Jan 1996.