Stochastic Cycle Period Analysis in Timed Circuits *

Eric Mercer Chris J. Myers
Electrical Engineering Department
University of Utah
Salt Lake City, UT 84112

Abstract

This paper presents the stochastic cycle period as a performance metric for timed asynchronous circuits. The stochastic cycle period is a sum of weighted delays whose value represents the expected delay of a single cycle in the specification. Each weight denotes the amount of time a delay contributes to the cycle of the circuit when the circuit is in steady-state operation. This paper demonstrates how the stochastic cycle period is used to aggressively optimize timed circuits for average-case performance. It shows the use of the stochastic cycle period to restrict out low probability triggers, optimally order pins, and size transistors in gate implementations. All optimization efforts are focused to improve the average-case delay in the timed asynchronous circuit, at the possible expense of the worst-case delay. In addition, this paper shows the calculation of the stochastic cycle period and proposes a new method for calculating trigger probabilities in timed circuits. This new method is simulation based, does not require the timed state space of the specification, and can be applied to any arbitrary Petri net.

1 Introduction

Asynchronous design styles were engendered before many synchronous techniques, but were left to the wayside because of their perceived difficulty of implementation. Components in an asynchronous circuit operate as fast as they can and notify other components when they have completed their work. In this type of circuit, the traditional cost metric is redefined. In synchronous circuits, it is necessary to always optimize for the worst-case behavior. It is the worst-case that is going to set the clock frequency, regardless of how often that worst-case actually occurs. When designing an asynchronous circuit, the designer must optimize for the average-case, not the worst-case scenario.

*This research is supported by a grant from Intel Corporation and an NSF CAREER award MIP-9625014.

Pivotal to any design method is the ability to analyze and optimize circuits and circuit specifications throughout the design process. This work develops a performance analysis technique for *timed circuits* [1] which is integrated into the CAD tool ATACS. This technique, which is based on the *stochastic cycle period*, addresses the need of a performance metric for timed circuits.

The stochastic cycle period is a weighted sum of delays. The weight on a delay designates the relative likelihood that the delay contributes to the cycle of the circuit when the circuit is in its steady-state. The delay is derived from the actual gate implementation of the signals in the system and are trigger-dependent. A trigger-dependent delay is the time it takes for a transition at an input pin (the trigger) to cause a transition on the output pin. The value of the stochastic cycle period is the expected or average-case delay of the circuit because the weights on the delays are derived from the steady-state behavior.

Many methods for asynchronous performance analysis do exist and this work strives to build and improve upon those techniques. Specifically, the stochastic cycle period performance metric is rooted in [2] but differs in that it incorporates the use of bounded delays with distributions and allows choice constructs in the specification. When compared to [3, 4, 5], this performance metric avoids the pitfalls of Markovian analysis and reduces the number of statistical metrics to a manageable set. These methods either present a metric for every edge or state in the reachable system or they reduce the information down to a single value. The stochastic cycle period presents a greatly reduced set of metrics which denote the importance of trigger-dependent delays in the asynchronous design. Finally, the derivation of the stochastic cycle period uses simulation techniques that are similar to those in [6]. However, [6] can only present a single value that denotes the average time separation between one event to another. Although [6] does calculate the average-case delay of a cycle in the system, it does not present any information about the paths in the system that constitute

that delay. Not only does the stochastic cycle period present the average-case delay in the design, it shows the designer which transitions contribute to that delay, thereby revealing the critical average-path in the circuit.

This paper is organized as follows: Section 2 presents the system model, describes the stochastic cycle period, and briefly discusses how it is derived. Section 3 gives an example of the stochastic cycle period for a simple enhanced latch controller and discusses how the stochastic cycle period can be used to analyze and optimize circuit performance. Finally, Section 4 demonstrates runtimes for the stochastic cycle period on various designs and proposes areas of future work.

2 The Stochastic Cycle Period

The stochastic cycle period uses a *Timed Stochastic Petri Net* (TSPN) representation to model the timed circuit specification. Since delay in a timed circuit is a complex function of process variation and environment, delays for places in the *TSPN* are modeled as bounded stochastic distributions. In practice, the *TSPN* behaves much like a marked petri-net or *Signal Transition Graph* (*STG*) representation, only transitions cannot immediately fire when they become enabled in a marking. Rather, when a place is added to a marking, it undergoes a delay that is prescribed by its stochastic distribution. Once a place has completed its prescribed delay, the place becomes available to waiting transitions. If enough places are available to a waiting transition to enable it to fire, it fires instantly. A *trigger* is defined as the last place to become available to a transition causing it to fire [7]. The *TSPN* model forces interleaving semantics and only allows a single place to become available at a time to waiting transitions. Therefore, simultaneous switching is not considered in this model. This restricts the types of delays that a gate can undergo to trigger-dependent delays.

The stochastic cycle period ρ is defined as a weighted sum of delays, where each delay is derived from trigger dependencies in the circuit implementation. Formally, the stochastic cycle period is $\rho = \sum_{(u,v)\in T} w_{uv}\alpha_{uv}$, where T is the set of all possible signal transitions allowed in the system, α_{uv} is the delay caused when the gate implementing the transition v is triggered by transition u, and w_{uv} is a multiplier that determines the amount of time that α_{uv} contributes to the expected delay in a cycle of the circuit. Since weights are used for trigger transitions in the cycle, the circuit can be optimized to favor transitions which make significant offerings to the overall cycle period. Since each possible trigger in the cycle is weighted and considered, the metric provides a profile of the cycle period which reflects the notion of average-case performance.

The stochastic cycle period is computed using a *stochastic timing simulation*. The stochastic timing simulation is the natural extension to the timing simulation presented in [2] where the maximum values are replaced with stochastic information showing the contribution of each possible delay in the cycle period. With this substitution, the timing simulation for the *STG* in Figure 1 becomes:

$$\begin{aligned}
\tau(\mathrm{E}+,i) &= \rho i \\
\tau(\mathrm{D}+,i) &= \tau(\mathrm{E}+,i) + \alpha_{\mathrm{E+D+}} \\
\tau(\mathrm{A}+,i) &= w_{\mathrm{D+A+}}\left(\tau(\mathrm{D}+,i) + \alpha_{\mathrm{D+A+}}\right) + \\
&\quad\; w_{\mathrm{Lt\text{-}A+}}\left(\tau(\mathrm{Lt}\text{-},i) + \alpha_{\mathrm{Lt\text{-}A+}}\right) \\
&\vdots \\
\tau(\mathrm{E}+,i+1) &= w_{\mathrm{Rin+E+}}\left(\tau(\mathrm{Rin}+,i) + \alpha_{\mathrm{Rin+E+}}\right) + \\
&\quad\; w_{\mathrm{A\text{-}+E+}}\left(\tau(\mathrm{A}\text{-},i) + \alpha_{\mathrm{A\text{-}E+}}\right) \\
&= \rho\cdot(i+1),
\end{aligned}$$

where the function $\tau(v,i)$ is the time of the i^{th} transition of v and ρ is the cycle period. To solve for ρ in the timing simulation, the symbolic times for each of the i^{th} transitions are substituted into the function for $\tau(\mathrm{E}+,i+k)$. The function $\tau(\mathrm{E}+,i+k)$ is set equal to $\rho\cdot(i+k)$ and it is directly solved for ρ. The value k denotes the number of cycles used in the timing simulation before it converges to a steady-state solution. The original cycle period from [2] is an upper bound on the worst-case performance of the circuit, due to the role of the max function. The new stochastic cycle period shows the average-case performance of the circuit, due to the role of the weights. Accordingly, the effectiveness of optimizations both on and off the critical path of the circuit in improving average-case performance is duly reflected in the value, as well as the weights of the stochastic cycle period before and after the optimizations.

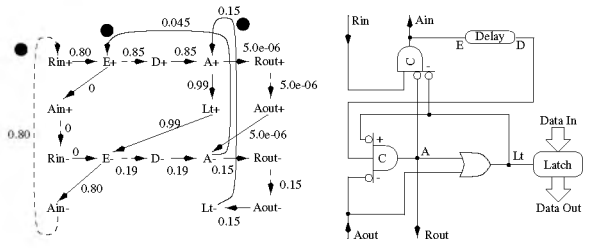

Figure 1: The *STG* and circuit implementation of an enhanced latch controller courtesy of [8].

The key to the stochastic timing simulation is the way in which the w_{uv} weight values are constructed during the simulation. On each cycle of the stochastic timing simulation, the weight values must be updated correctly to reflect the stochastic model of the *TSPN* representation. A *trigger probability*, $\Pr_{uv}$, is defined as the probability of a transition u triggering v. The trigger probabilities are used to update the weight values in the timing simulation. Therefore, the w_{uv} values are derived through simulation but seeded with trigger probabilities. The time of the i_{th} transition of a signal is found by taking each $\Pr_{uv}$ value for each trigger for the signal in question and adding it to w_{uv}. In [7], analytical, as well as simulation methods are presented to calculate trigger probabilities. However, each of the methods derive trigger probabilities by first finding steady-state and transition probabilities in the *timed Reachability Graph* (*RG*) of the *TSPN* system. Due to the size and nature of the *RG*, simulation time is costly. Moreover, while the analytical methods are fast, they can only be used with confidence in a narrow class of circuits. To overcome these limitations, this paper simulates directly the trigger probabilities using a stochastic simulation or a random walk. Note, the reachable state space is not stored in the simulation. The stochastic simulation can be applied to any class of TSPN (including arbitrary choice constructs), does not require the *RG* to exist, and does not require an input trace.

3 Example

Figure 1 shows the STG for the enhanced latch controller from [8] and the circuit synthesized by `ATACS`. The *STG* is translated to the *TSPN* model by causing the arc delays to be uniformly distributed across bounds that are set to be $\pm 20\%$ of the `SPICE`'ed delay values shown in [8]. Before generating the circuit, the stochastic cycle period for the enhanced latch controller is compared with an alternative latch controller design, such as the simple latch controller from [8]. This is done by using the expected delays of places in the *TSPN* as the α_{uv} delays in the stochastic cycle period. This simple analysis shows the enhanced latch controller to have an average-case performance that is 1.35 times faster than the simple latch controller. This number correlates well to the 1.47 times speedup shown in the `SPICE` results from [8]. The slight difference is attributed to the fact that results from the `SPICE` simulation are dependent on a single input trace with fixed times for things to happen, as well as the fact that `SPICE` cannot consider process and environment variations. It produces a fixed delay that is determined by the inputs and the model used the simulation.

The weights on the arcs of the *STG* in Figure 1 denote the w_{uv} values from the stochastic cycle period. The larger the weight, the greater the amount of delay the arc contributes to the average-case performance of the circuit. Therefore, if a trigger-dependent delay has the value α_{uv}, the amount of time that delay contributes to the average cycle of the circuit is $w_{uv} \cdot \alpha_{uv}$, where w_{uv} is the weight multiplier from the stochastic cycle period. Using these weights, further optimization to the circuit can be applied. According to the weights, the delay for the transition A+ is largely controlled by the trigger D+ and both D+ and A+ make significant contributions to the cycle period. Therefore, D+ should be near the output of the gate implementing A+ to optimize its performance. For the falling transition A-, D- controls the delay and thus D- should be near the output of the gate. The transition E+ is triggered by Rin+ and A-, but A- is not directly on the critical path, so Rin+ is moved near the output of the gate implementing E+. Accordingly, E- is triggered by Rin- and Lt+, but Rin- has negligible weight and is therefore not a contributor to the cycle period, so Lt+ should be moved near the output of the gate for E-. Finally, for the gate implementing Lt, the weights show Aout- to rarely contribute to the length of the cycle period, so A+ should be placed near the output of Lt to speedup the Lt+ transition.

A more aggressive optimization of this circuit involves tightening timing bounds to restrict out triggers in the actual implementation. The weights from the stochastic cycle period can be used to identify trigger signals that do not contribute to the cycle period. In this example, it is extremely unlikely that Aout+ triggers A-. If the bounded delay for Aout+ is tightened by 0.5%, the signal Aout+ is no longer needed in the implementation of A-. Similarly, a tightening of about 0.5% removes Rin- from the gate for E-. This shows how the stochastic cycle period and the trigger probabilities can be used to restrict out triggers that have low weights of occurring. Although some consider this type of optimization too aggressive, it is important to note that the delay assumptions in the beginning are very conservative. As the design matures, the timing assumptions are brought closer to the actual delay ranges. Moreover, at this point the circuit designer has a good understanding of the amount of slack found in the system. The stochastic cycle period is designed to better utilize this slack.

Another possible optimization is transistor sizing. For example, looking at the *STG*, transitions that make significant contributions to the delay of the circuit can be readily identified. With this information, it is possible to size the transistors in the gate implementations

to favor transitions that fall in the critical cycle. Consider the signal Lt. The high going phase of Lt is on the critical path with a weight value of 0.99, while the low going transition falls off the critical path with a weight value of 0.15. With this information, it is possible to skew the gate implementing Lt to favor the rising transition, since that is the critical edge. This can easily be accomplished by increasing the width of the transistors involved in the rising transition of Lt, with the transistor near the power rail having the largest width.

4 Results and Conclusions

Early results for the stochastic cycle period are promising. We have determined that we can do performance analysis on very large systems using stochastic simulation to compute trigger probabilities for the stochastic cycle period. To show this, we analyze a number of enhanced latch controllers connected in series on a 400 MHz Pentium II processor with 384 MB of memory. For a 4 stage enhanced latch, `ATACS` finds 2416 states in 222 seconds using 72 MB of memory. The simulation for the stochastic cycle period completes in 95 seconds using only 14 MB of memory. Note that the state graph does not need to be found for the simulation. In fact, `ATACS` is unable to find the state space for a 5 stage enhanced latch controller. However, the simulation can provide the stochastic cycle period in 156 seconds with only 14 MB of memory. This shows that the new simulation method for the stochastic cycle period is able to scale to large systems that could not otherwise be analyzed.

As another example, we analyze a simple asynchronous memory management unit (MMU) [9]. The MMU can receive a request to load or store data from a microprocessor. On a normal memory load or store, it appends the high order byte to the memory address from one of two internal registers (one for loads, another for stores). Two special addresses are used to load or store these internal registers. There are 6 possible cycles depending on choice in the environment. We assume that loads occur twice as often as stores. We also assume that each of the two internal registers are only accessed 1 percent of the time. The timing numbers are taken from [1]. This example illustrates our method applied to a specification containing choice constructs. The MMU has 1544 states and the analysis took 1048 seconds to complete. The results show that many of the falling requests in the control handshake do not affect the performance. They also show the relative importance of each of the 6 cycles.

This paper has presented the stochastic cycle period as a performance metric for timed systems. It has shown the stochastic cycle period to have the ability to optimize timed circuits for average-case performance by identifying delays in the circuit that significantly impact the cycle of a circuit when the circuit is in its steady-state behavior. From this, it is possible to restrict out triggers from gate implementations, optimally order pins, and size transistors to favor the average-case delay of the circuit. In addition, this paper has presented a new method of directly calculating trigger probabilities that does not require the presence of a *RG* for the *TSPN* model. The new method is based on stochastic simulation and is faster than previous simulation methods in [7], due to the fewer number of values that it must track.

Future work for this research includes the automation of transistor sizing and pin ordering, as well as, aiding the designer in identifying triggers to restrict from gate implementations. More importantly, future work includes methods for finding good bounded delays to use in the *TSPN* model and a method of sizing transistors to meet the specified delays.

References

[1] Chris J. Myers. *Computer-Aided Synthesis and Verification of Gate-Level Timed Circuits.* PhD thesis, Dept. of Elec. Eng., Stanford University, October 1995.

[2] Steven M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits.* PhD thesis, California Institute of Technology, 1991.

[3] Aiguo Xie and Peter A. Beerel. Symbolic techniques for performance analysis of timed systems based on average time separation of events. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 64–75. IEEE Computer Society Press, April 1997.

[4] Prabhakar Kudva and Venkatesh Akella. A technique for estimating power in self-timed asynchronous circuits. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 166–175, November 1994.

[5] P. Kudva, G. Gopalakrishnan, and E. Brunvand. Performance analysis and optimization for asynchronous circuits. In *Proc. International Conf. Computer Design (ICCD).* IEEE Computer Society Press, October 1994.

[6] Aiguo Xie, Sangyun Kim, and Peter A. Beerel. Bounding average time separations of events in stochastic timed petri nets with choice. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems.* IEEE Computer Society Press, April 1999.

[7] Eric G. Mercer. Stochastic cycle period anlaysis in timed circuits. Master's thesis, University of Utah, 1999.

[8] S. B. Furber and J. Liu. Dynamic logic in four-phase micropipelines. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems.* IEEE Computer Society Press, March 1996.

[9] C. J. Myers and A. J. Martin. The design of an asynchronous memory management unit. Technical Report CS-TR-93-30, California Institute of Technology, 1993.