

STOCHASTIC CYCLE PERIOD ANALYSIS IN TIMED CIRCUITS

Eric G. Mercer and Chris J. Myers

Electrical Engineering Department
University of Utah
Salt Lake City, UT 84112

ABSTRACT

This paper presents a technique to estimate the stochastic cycle period (SCP), a performance metric for timed asynchronous circuits. This technique uses timed stochastic Petri nets (TSPN) which support choice and arbitrary delay distributions. The SCP is the delay of the average path in a TSPN when represented as a sum of weighted place delays. A place delay is the expected value of its associated distribution and its weight denotes its importance in the average path of the TSPN. The approach analyzes finite execution traces of the TSPN to derive an expression for the weight values in the SCP. The weights can be analyzed with basic statistics to within an arbitrary error bound. This paper demonstrates the use of the SCP to aggressively optimize timed asynchronous circuits for improved average-case performance by reducing transistor counts, reordering input pins at gates, and skewing transistor sizes to favor important transitions. Each optimization effort is directed to improve the average-case delay in the circuit at the possible expense of the worst-case delay.

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 must be redefined. In synchronous circuits, one must 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.

Circuits designed for average-case using asynchronous techniques have a potential for higher performance. This potential is harnessed as early as 1955 in the design of an

asynchronous adder [1]. Through the use of dual rail encoding and completion detection, this simple ripple carry adder has an average-case performance near $\ln N$, where N is the number of bits in the addition. Traditional worst-case design techniques bound the performance at N . A more recent example of exploiting average-case performance is seen in [2]. This design of an asynchronous IA32 instruction decoder is aggressively optimized for the most common instruction lengths. The result is a three times improvement in throughput at half the latency, dissipating only half the power, and requiring about the same area as an existing synchronous IA32 decoder running at 400 MHz.

In order to exploit average-case performance, it is necessary to have an appropriate systematic method to analyze and optimize circuits and specifications throughout the design process. This paper develops the *stochastic cycle period* (SCP) as one such method. The SCP is a performance analysis metric for *timed circuits* [3] which is integrated into the CAD tool ATACS. The SCP analysis technique operates on *timed stochastic Petri nets* (TSPN), which are capable of modeling choice and arbitrary delay distributions. Given a timed asynchronous circuit modeled by a TSPN, the SCP is a sum of weighted place delays showing the average-path in the TSPN. Each place delay in the sum is the expected value of the delay distribution on that place, and its weight shows its importance in the average-delay of the TSPN. An expression for the weights is derived by analyzing finite unrollings of the TSPN. The expression can be evaluated using basic statistics. The value of the SCP using the place delays represents the average-delay of a cycle in the TSPN.

Many methods for asynchronous performance analysis do exist and this work strives to improve and extend these techniques. The SCP builds on the cycle period in [4] by incorporating bounded delays with arbitrary distributions and choice constructs in the specification. In [5], a technique is presented that calculates the average *time separations of events* (TSE) by analyzing finite unrollings of stochastic timed Petri nets. The analysis of the weights in the SCP uses the finite unrolling methods of [5]. However, the SCP uses the *max_diff* algorithm from [3] to derive the TSEs rather than a longest path analysis. The *max_diff*

This research is supported by a grant from Intel Corporation, NSF CAREER award MIP-9625014, and SRC contract 97-DJ-487.

algorithm removes the need to maximize over all longest paths found from each initially marked rule in the cut set of the TSPN and reduces the need to minimize the initially marked places in the cut set. Furthermore, the work in [5] only presents a single value that denotes the average time separation between two events. Although [5] 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. This approach augments the analysis in [5] to extract the constraining path from the TSPN. This path is used to derive the weight values in the SCP.

Markovian analysis has also been applied to the performance analysis of asynchronous circuits [6, 7]. SCP analysis avoids the computational complexity of Markovian analysis and reduces the number of statistical metrics to a manageable set. Moreover, the methods in [6, 7] either present a metric for every edge or state in the reachable system, or they reduce the information down to a single value. The SCP presents a greatly reduced set of metrics—on the order of the number of places in the TSPN—which denote the relative importance of different paths in the asynchronous circuit.

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

2. THE STOCHASTIC CYCLE PERIOD

The SCP uses a TSPN representation of timed circuits. 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 1-safe marked Petri net or *Signal Transition Graph* (STG), except that transitions cannot immediately fire when they become enabled in a marking. Rather, when a token enters a place, it undergoes a delay that is prescribed by its stochastic distribution. Once the token has completed its prescribed delay, the token becomes available to waiting transitions. If enough tokens are available to a waiting transition to enable it to fire, it fires instantaneously. A *trigger* is defined as the last place to become available to a transition causing it to fire [8]. The TSPN model forces interleaving semantics and only allows a single token to become available at a time.

The SCP, ρ , is defined as a sum of weighted delays given as $\rho = \sum_{(u,v) \in T} w_{uv} \alpha_{uv}$. T is the set of all possible transitions allowed in the TSPN. The place delay, α_{uv} , is the expected value of the delay for the distribution at the place between transitions u and v . This delay can be related to

the circuit by setting it to be the average measured delay between a transition of u and a transition of v on the gate implementing v when the gate is triggered by u (i.e., u is the last signal transition needed for v to transition). The w_{uv} value is a multiplier that determines the importance of α_{uv} in an average-cycle of the circuit. A w_{uv} value at or near 1 implies that the associated delay is often contributing to the average-delay of the circuit. A w_{uv} at or near 0 implies that the associated delay rarely, if ever, contributes to the average-delay of the circuit. This means that the transition happens in parallel with some other slower transition (or transitions).

Following notation in [5], the expression for the w_{uv} values develops as follows: let a *timed execution* (π) of a TSPN be an acyclic event graph or an unfolding of the TSPN with all choice resolved and places assigned delay values. An event $s^{(k)}$ is defined as the k^{th} instance of the transition s in the timed execution π . Formally $\pi = (T_\pi, F_\pi, \tau)$ where T_π is the set of all events, $F_\pi : T_\pi \times T_\pi$ is the flow relation, and $\tau : T_\pi \rightarrow \mathfrak{R}$ is a mapping function where $\tau(s^{(k)})$ returns the time of the k^{th} firing of s .

To generate a timed execution, it is first assumed that all initially marked places in the TSPN receive tokens at time zero. Each token is then assigned a random delay value sampled from its associated place distribution. The system clock is then iteratively advanced to the earliest token time that is to become available to waiting transitions. If when a token becomes available it enables a waiting transition to fire, then the transition is fired, tokens are moved to new places, and they are assigned random delays from their respective place distributions.

For a given timed execution pair $(s^{(k)}, t^{(k+\varepsilon)})$, their time separation is defined as $\gamma^{(k)}(s, t, \varepsilon) = \tau(t^{(k+\varepsilon)}) - \tau(s^{(k)})$. The *critical path function*, C , returns the sequence of events that determine the value of $\gamma^{(k)}(s, t, \varepsilon)$. Formally, C is defined as $C^{(k)}(t_1, t_n, \varepsilon) = (t_1^{(k)}, \dots, t_n^{(k+\varepsilon)})$ such that

$$\forall(1 \leq i < n), \forall(k \leq j \leq k + \varepsilon), \forall(0 \leq l \leq 1) :$$

$$(t_i^{(j)}, t_{i+1}^{(j+l)}) \in F_\pi \quad (1)$$

^

$$\begin{aligned} \gamma^{(j)}(t_i, t_n, \varepsilon + (k - j)) = \\ \gamma^{(j)}(t_i, t_{i+1}, l) + \\ \gamma^{(j+l)}(t_{i+1}, t_n, \varepsilon + (k - j - l)) \end{aligned} \quad (2)$$

This states that given any pair of adjacent events $(t_i^{(j)}, t_{i+1}^{(j+l)})$ in the critical path C , $(t_i^{(j)}, t_{i+1}^{(j+l)})$ must be in the flow relation as required by (1), and $(t_i^{(j)}, t_{i+1}^{(j+l)})$ must satisfy the *backtrace* requirement of (2). (2) states that the time separation of $t_i^{(j)}$ and the end of the sequence $t_n^{(k+\varepsilon)}$ must be equal

to the time separation of $(t_i^{(j)}, t_{i+1}^{(j+l)})$ plus the time separation of $(t_{i+1}^{(j+l)}, t_n^{(k+\varepsilon)})$. The l term accounts for adjacent events on a new cycle boundary, since it is possible to have t_i in cycle j and t_{i+1} in cycle $j+1$. Let $V : T_\pi \times T_\pi \times C \rightarrow \mathfrak{R}$ be the *value* function defined as

$$V(s^{(i)}, t^{(j)}, C) = \begin{cases} \frac{1}{\varepsilon} & \text{if } \exists s^{(i)}, t^{(j)} \in C \wedge \\ & (s^{(i)}, t^{(j)}) \in F_\pi \\ 0 & \text{otherwise.} \end{cases}$$

V returns $\frac{1}{\varepsilon}$ if the events $s^{(i)}$ and $t^{(j)}$ are found in the critical path C , and there is a $(s^{(i)}, t^{(j)})$ pair in the flow relation F_π .

Given a pair of transitions (u, v) , an event trace π , and a critical path $C^{(k)}(s, t, \varepsilon)$, the weight value $w_{uv}^{(k)}$ can now be defined as

$$w_{uv}^{(k)} = \sum_{k \leq i, j \leq k + \varepsilon} V(u^{(i)}, v^{(j)}, C^{(k)}(s, t, \varepsilon)),$$

which states that the weight of a given transition pair is equal to the sum of $\frac{1}{\varepsilon}$ for every given adjacent instance of the events $u^{(i)}$ and $v^{(j)}$ in the critical path $C^{(k)}(s, t, \varepsilon)$. Let Π represent the set of all possible timed executions of a given TSPN. If π is randomly sampled from Π , then $w_{uv}^{(k)}$ is a random variable with some mean and distribution. Thus, the sequence $\{w_{uv}^{(k)} : k = 1, 2, 3, \dots\}$ is a random process. Therefore, the average value of the weight sequence is the $\lim_{(n \rightarrow \infty)} \frac{1}{n} \sum_{k=1}^n w_{uv}^{(k)}$ which is defined as w_{uv} .

3. EXAMPLE

Figure 1 shows the *signal transition graph* (STG) for the enhanced latch controller from [9] and the circuit synthesized by ATACS. The STG is translated to the TSPN model by making the transition delays uniformly distributed across bounds that are set to be $\pm 20\%$ of the delay values from SPICE shown in [9]. Before generating the circuit, the stochastic cycle period for the enhanced latch controller is compared with an alternative latch controller design which

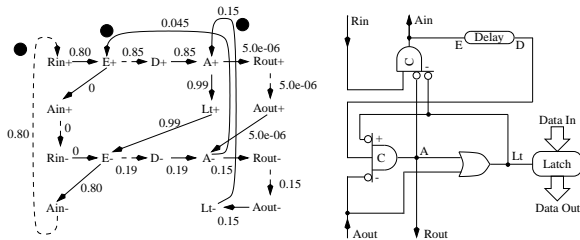


Figure 1: The STG and circuit implementation of an enhanced latch controller courtesy of [9].

is the simple latch controller from [9]. This is done by using the expected delays of places in the TSPN as the α_{uv} delays in the SCP. 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 [9]. 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 in the simulation.

The weights on the arcs of the STG in Figure 1 denote the w_{uv} values from the SCP. 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} , then 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 SCP. 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 of $A-$, $D-$ 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 it is thus, not a contributor to the cycle period. Therefore, $Lt+$ should be moved near the output of the gate for $E-$.

A more aggressive optimization of this circuit involves tightening timing bounds to restrict out triggers in the actual implementation. The weights from the SCP 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 SCP can be used to restrict out triggers that have low weights. Although this type of optimization may seem aggressive, it is important to note that the delay assumptions in the beginning are typically very conservative. As the design matures, the timing assumptions are brought closer to their actual delay ranges. Moreover, at this point the circuit designer has a better understanding of the amount of slack found in the system. The SCP is designed to better utilize this slack.

Another possible optimization is transistor sizing. For example, looking at the STG, transitions that make signifi-

cant 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, and 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 widths 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 SCP are promising. We have determined that we can do performance analysis on larger systems. To show this, we analyze a number of enhanced latch controllers connected in series on a 550 MHz Pentium III processor with 384 MB of memory. For a 4 stage enhanced latch, ATACS finds 2416 states in 222 seconds. The value of the SCP converges to 1176.6 ± 11.61 at a 95% confidence interval with a 1.0% relative error in 132 seconds. This correlates with the computed TSE, reported at the same confidence interval, to be 1197.3 ± 8.1 . In fact, ATACS is unable to find the state space for a 5 stage enhanced latch controller. However, the SCP is found in 150 seconds. This shows how the SCP scales to larger systems.

In [2], an asynchronous instruction length decoder is presented that makes extensive use of timed circuits. One of the principle timed circuits is the tag unit shown in Figure 2. Tag units are arranged in a matrix, and each column of the matrix is connected to a length decoder. When a column is the first byte of an instruction, its tag unit is alerted. The tag unit looks at the length of the instruction and then forwards the tag to the column of the first byte of the next instruction. This example illustrates our method applied to specifications containing choice constructs since the system must choose both an instruction length and an incoming tag. For this example, the SCP is determined in 35 seconds and reflects the frequency of instruction lengths as described by the specification.

This paper has presented the SCP as a performance metric for timed systems. It presented an expression for directly calculating the w_{iv} values in the SCP that does not require state space exploration. The new method is based on finite unrollings of the TSPN model and is faster than previous simulation methods in [8]. This paper has presented methods of optimizing circuits for average-case performance using the SCP. These methods are: pin ordering, transistor count reduction through trigger signal restriction, and unbalanced sizing to favor important transitions in the SCP.

Future work for this research includes the automation of

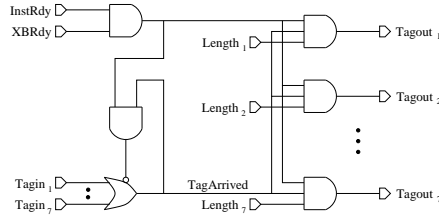


Figure 2: Intel RAPPID tag unit.

transistor sizing and pin ordering, as well as aiding the designer in identifying triggers to restrict from gate implementations. We also plan to develop methods for finding good bounded delays to use in the TSPN model and a method of sizing transistors to meet the specified delays.

Acknowledgements

We are indebted to Peter A. Beerel of the University of Southern California for helping us to better define the SCP, as well as his insight on its derivation. In addition, we would like to thank the following people of the University of Utah who contributed to this manuscript: Eric Peskin, Kip Killpack, and Robert Thacker.

5. REFERENCES

- [1] Bruce Gilchrist, J. H. Pomerene, and S. Y. Wong. Fast carry logic for digital computers. *IRE Transactions on Electronic Computers*, EC-4(4):133–136, December 1955.
- [2] Shai Rotem, Ken Stevens, Ran Ginosar, Peter Beerel, Chris Myers, Kenneth Yun, Rakefet Kol, Charles Dike, Marly Roncken, and Boris Agapie. RAPPID: An asynchronous instruction length decoder. In *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 60–70, April 1999.
- [3] Chris J. Myers. *Computer-aided synthesis and verification of gate-level timed circuits*. PhD thesis, Stanford University, October 1995.
- [4] Steven M. Burns. *Performance analysis and optimization of asynchronous circuits*. PhD thesis, California Institute of Technology, 1991.
- [5] Aiguo Xie, Sangyun Kim, and Peter A. Beerel. Bounding average time separations of events in stochastic timed Petri nets with choice. In *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 94–107, April 1999.
- [6] Aiguo Xie and Peter A. Beerel. Symbolic techniques for performance analysis of timed systems based on average time separation of events. In *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 64–75. IEEE Computer Society Press, April 1997.
- [7] P. Kudva, G. Gopalakrishnan, and E. Brunvand. Performance analysis and optimization for asynchronous circuits. In *Proceedings of International Conf. Computer Design (ICCD)*. IEEE Computer Society Press, October 1994.
- [8] Eric G. Mercer. Stochastic cycle period analysis in timed circuits. Master's thesis, University of Utah, 1999.
- [9] S. B. Furber and J. Liu. Dynamic logic in four-phase micropipelines. In *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, March 1996.