Latency Estimation of the Asynchronous Pipeline Using the Max-Plus Algebra

Jian Ruan, Zhiying Wang, Kui Dai, and Yong Li

School of Computer, National University of Defense Technology
Changsha, Hunan, P.R. China

Abstract. This paper presents a methodology to estimate the latency of the asynchronous pipeline without choices. We propose modeling an asynchronous pipeline with the timed event graph and characterizing its specification via the max-plus algebra. An evolution equation of the event firing epoch can be obtained, which is linear under the max-plus algebraic formalism. In terms of the above-mentioned equation, we can formulate the latency of the pipeline successfully. The case study shows that our method is simple, fast and effective.

Keywords: Asynchronous Pipeline, Max-Plus Algebra, Timed Event Graph, Evolution Equation, Latency Estimation.

1 Introduction

Performance analysis of the asynchronous circuit, especially the asynchronous pipeline, is attracting lots of research interest. In [1], Ramamoorthy et al. modeled asynchronous concurrent systems via the timed marked graph with deterministic firing delays, and then used Karp's theorem to determine the average throughput in polynomial time. In [2], Kudva et al. proposed adopting Generalized Timed Petri Nets (GTPNs) to model asynchronous circuits in which component delays are modeled using their mean values. As usual, the model is then converted into a continues time Markov chain (CTMC) and solved for its stable probability distribution. In [3,4], Xie and Beerel proposed a discrete time model that can handle arbitrarily distributed delays. The model is converted into a discrete time Markov chain (DTMC) and solved for its stationary distribution. To mitigate the state explosion problem, they developed a technique called state compression to speed up the symbolic computation of the stationary distribution of their DTMC models. Specifically, they reduced the state space of the DTMC to one of the feedback vertex sets of its state space. Nevertheless, even with these advances, the state explosion problem has limited Markov chain based approaches to small models.

In this paper, we apply the timed event graph to model the asynchronous pipeline. Using the max-plus algebra to describe the specification, we can derive a linear evolution equation of the event firing times under the max-plus algebraic formalism. Based on the periodicity of iterated evolution equation, the latency can be formulated effectively and estimated efficiently.

Y. Shi et al. (Eds.): ICCS 2007, Part IV, LNCS 4490, pp. 251–258, 2007.

The organization of this paper is as follows. Section 2 reviews the basics of the max-plus algebra. Section 3 describes the way to model asynchronous pipelines with the timed event graph. Section 4 illuminates the technique that how to estimate the latency of the asynchronous pipeline by its linear evolution equation in detail. Section 5 validates the conclusion by a 3-stage 4-phase bundled-data linear asynchronous pipeline and section 6 gives some conclusions.

2 Elementary Max-Plus Algebra

In this section we give an introduction to the max-plus algebra which will be used during the performance analysis of asynchronous pipelines. Most of the material presented is selected from [5,6], where a complete overview of the max-plus algebra can be found.

2.1 Basic Concepts

Definition 1. *The max-plus algebra* $\mathbb{R}_{\max}$ *is the set* $\mathbb{R}_\varepsilon \stackrel{def}{=} \mathbb{R} \cup \{-\infty\}$ *equipped with the two internal operations* $\oplus \stackrel{def}{=} \max$ *and* $\otimes \stackrel{def}{=} +$*, the neutral element for operation* $\oplus$ *is* $\varepsilon \stackrel{def}{=} -\infty$ *and the unit element for operation* $\otimes$ *is* $e \stackrel{def}{=} 0$.

$\mathbb{R}_{\max} = (\mathbb{R}_\varepsilon, \oplus, \otimes)$ is an idempotent commutative semi-ring, and the main difference with the conventional algebra $(\mathbb{R}, +, \times)$ is the fact that the operation $\oplus$ is idempotent, $a \oplus a = a$ and does not have an inverse.

Operations $\oplus$ and $\otimes$ can be extended to matrices with the classical construction,

1. $(A \oplus B)_{i,j} = A_{i,j} \oplus B_{i,j}$, where $A, B \in \mathbb{R}_\varepsilon^{m \times n}$;
2. $(A \otimes B)_{i,j} = \bigoplus_{k=1}^{p} (A_{i,k} \otimes B_{k,j})$, where $A \in \mathbb{R}_\varepsilon^{m \times p}$ and $B \in \mathbb{R}_\varepsilon^{p \times n}$;
3. $A^0 = I_n$, $A^k + 1 = A \otimes A^k = A^k \otimes A$, where $A \in \mathbb{R}_\varepsilon^{n \times n}$.[1]

2.2 Equation Solving

Solving linear equations in $(max, +)$ is different from the classical linear case. In general, it may or may not have a solution. Moreover, even if a solution exists, it may not be unique. However, there exists one class of linear equations for which we have a satisfactory theory.

Theorem 1. *Let* $A \in \mathbb{R}_\varepsilon^{n \times n}$, $B \in \mathbb{R}_\varepsilon^{n}$. *If matrix* A *is acyclic,* $A^* \stackrel{def}{=} \bigoplus_{i=0}^{\infty} A^i$ *converges and the vectorial linear equation of the form* $X = A \otimes X \oplus B$ *has a unique solution* $X = A^* \otimes B$.

Matrix $A \in \mathbb{R}_\varepsilon^{n \times n}$ is said to be acyclic iff $\exists (i_1, i_2, \ldots, i_{p-1}, i_p = i_1)$, s.t. $A_{i_1,i_2} \otimes A_{i_2,i_3} \otimes \ldots \otimes A_{i_{p-1},i_p} \neq \varepsilon$.

As for the recursion equation, the most attractive is not its solution but the periodicity.

[1] I_n is the $n \times n$ identity matrix with e on the main diagonal and ε elsewhere.

Theorem 2. *Let A be a $(max,+)$ irreducible matrix, there exists an unique real number λ, integer c and constant n_0, for all $n \geq n_0$, s.t. $A^{n+c} = \lambda^c \otimes A^n$.*

Matrix $A \in \mathbb{R}_{\varepsilon}^{n \times n}$ is said to be irreducible iff $\forall i,\ j,\ \exists (i,\ i_1,\ \ldots,\ i_p = j)$, s.t. $A_{i,i_1} \otimes A_{i_1,i_2} \otimes \ldots \otimes A_{i_{p-1},i_p} \neq \varepsilon$.

Corollary 1. *Let A be a $(max,+)$ irreducible matrix, the evolution equation $X(n+1) = A \otimes X(n)$ will end up in a periodic behavior, that is*

$$\exists n_0, c, \lambda, \quad \forall n \geq n_0, \quad X(n+c) = \lambda^c \otimes X(n).$$

3 Modeling the Asynchronous Pipeline

Event graph is widely used during the design, synthesis and verification of the asynchronous circuit[7] It describes the ordering of the events exactly, whereas the questions pertaining to when the events take place are not addressed. For questions related to the performance evaluation, it is necessary to introduce time.

In this section, we will expand on the way to model an asynchronous pipeline via timed event graph.

3.1 Timed Event Graph

Definition 2. *Petri net[8] is a bi-partite graph given by tuple $\mathcal{G} = (\mathcal{P}, \mathcal{Q}, \mathcal{E}, \mathcal{M}^0)$, where $\mathcal{P} = \{p_0, \cdots, p_m\}$ is the set of places, $\mathcal{Q} = \{q_0, \cdots, q_n\}$ is the set of transitions, $\mathcal{E} \subseteq (\mathcal{P} \times \mathcal{Q}) \cup (\mathcal{Q} \times \mathcal{P})$ is the set of arcs, $\mathcal{M}^0 : \mathcal{P} \to \{0, 1, 2, \cdots\}$ is the initial marking.*

We denote by $p^\bullet$ ($^\bullet p$) the set of downstream (upstream) transitions of place p, $q^\bullet$ ($^\bullet q$) the set of downstream (upstream) places of transition q.

Definition 3. *An event graph is a Petri net where each place has exactly one incoming transition and one outgoing transition.*

$$\forall p \in \mathcal{P}, |{}^\bullet p| = |p^\bullet| = 1$$

To analyze the latency of an asynchronous pipeline, we should consider the delay constraint. This can be done in two basic ways, by associating durations with either transition firings or with the sojourn of tokens in places[2].

1. Associating each place p with holding time σ_p. Holding time is the minimal time that a token must spend in a place before contributing to the enabling of the downstream transition, which can be used to represent the transportation or communication time. $\sigma_p(n)$ means the n-th holding time of place p.

[2] It is possible to convert a event graph with holding times into a event graph with firing times and vice-versa.

2. Associating each transition q with fireing time δ_q. Firing time is the time that elapses between the starting and the completion of the firing of the transition, which can be used to represent the process times in a functional block. $\delta_p\,(n)$ means the n-th firing time of transition p.

Now that the response times of the request and acknowledge between the adjacent pipeline stages are different, the former is much more appropriate.

3.2 Model of Asynchronous Pipeline

It is generally believed that an asynchronous pipeline can be viewed as a traditional "synchronous" data-path, consisting of latches and combinational circuits, which is clocked by some form of handshaking between neighbouring pipeline stages[7]. Taking a linear bundled-data asynchronous pipeline for example, whose structure is shown in Figure 1.

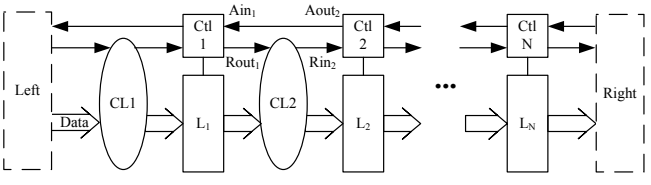

Fig. 1. Linear 4-phase bundled-data asynchronous pipeline

Since the latency of an asynchronous pipeline is only in reference to its structure, we may pass over the influence of the inputing. On the supposition that the environment deals with the request or acknowedge of the pipeline in no time, we may model the linear pipeline, along with its environment, as an autonomous timed event model, which is displayed in Figure 2.

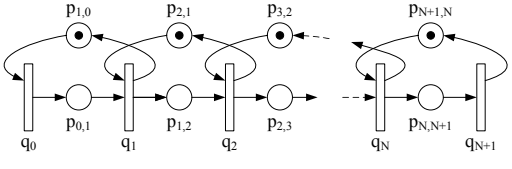

Fig. 2. Timed Event Graph Model

Transition q_0 stands for the left environment, its firing represents the inputing of new data. Transition q_{N+1} stands for the right environment, its firing represents that the computing result has been received. Moreover, transition

q_i $(0 < i \leq N)$ stands for the handshaking latch Hl_i of the pipeline[3], its firing represents that Hl_i is active.

Token in place $p_{i,i+1}$ represents that the input data is handled by stage $i+1$, holding time $\sigma_{i,i+1}$ amounts to the delay from new data on the input of stage $i+1$ to the production of the corresponding output (or to being received by the right environment), provided that the acknowledge signals are in place when data arrives. Token in place $p_{i,i-1}$ represents the reception of the acknowledgment produced by stage i, holding time $\sigma_{i,i-1}$ amounts to the delay from receiving an acknowledge from stage i to the corresponding acknowledge is produced by stage $i-1$ (or to the next request is produced by the left environment), provided that the request is in place when the acknowledge arrives[4].

4 Estimating the Latency of the Asynchronous Pipeline

4.1 Evolution Equation in the Max-Plus Algebra

Let us consider the autonomous timed event graph $\mathcal{G} = \left(\mathcal{P}, \mathcal{Q}, \mathcal{E}, \mathcal{M}^0, \sigma\right)$, we assume that,

1. $\mathcal{G}$ is *live* [5] and the initial marking $\mathcal{M}^0$ has at most one token per place,
2. For any place p, $(p \in \mathcal{P})$, the holding time σ_p is fixed[6].
3. For any token under the initial marking, it is available for the enabling of the downstream transition.

According to the firing rule of the timed event graph, transition i starts its n-th firing at time:

$$\begin{cases} X_i(1) = \max\left\{ \max\limits_{j \in {}^{\bullet\bullet}i} \left\{ X_j(1) + \sigma_{j,i} \middle| \mathcal{M}^0_{j,i} = 0 \right\}, 0 \right\} \\ X_i(n) = \max\limits_{j \in {}^{\bullet\bullet}i} \left\{ X_j\left(n - \mathcal{M}^0_{j,i}\right) + \sigma_{j,i} \right\} \end{cases}$$

Using the $(max, +)$ notation,

$$\begin{cases} X_i(1) = \bigoplus\limits_{j \in {}^{\bullet\bullet}i} \left\{ X_j(1) \otimes \sigma_{j,i} \middle| \mathcal{M}^0_{j,i} = 0 \right\} \oplus 0 \\ X_i(n) = \bigoplus\limits_{j \in {}^{\bullet\bullet}i} \left\{ X_j\left(n - \mathcal{M}^0_{j,i}\right) \otimes \sigma_{j,i} \right\} \end{cases}$$

This equation can be seen as a linear equation between the variables $X_i(n)$, with coefficients $\sigma_{j,i}$.

[3] Handshaking latch Hl_i is composed of conventional latch L_i and control circuit Ctl_i.
[4] The calculation of holding time differes from the circuit implementation styles.
[5] An event graph is live if and only if all circuits are marked under the initial marking.
[6] σ_p can be expressed as $\sigma_{i,j}$, if ${}^{\bullet}p = q_i$ and $p^{\bullet} = q_j$.

When written in vectorial form, it becomes

$$\begin{cases} X(1) = & A_0 \otimes X(1) \oplus \Gamma \\ X(n) = & A_0 \otimes X(n) \oplus A_1 \otimes X(n-1) \end{cases} \tag{1}$$

where $X(n)$ is the vector $\left[X_1(n), \cdots, X_N(n)\right]'$, Γ is a zero vector of size N and for $k \in \{0,1\}$, A_k is a $N \times N$ matrix defined by

$$\left(A_k\right)_{i,j} \stackrel{def}{=} \begin{cases} \sigma_{j,i} & \text{if } \mathcal{M}^0_{j,i} = k \\ \varepsilon & \text{otherwise.} \end{cases}$$

Since $\mathcal{G}$ is live, there is no empty circuit under its initial marking. As a consequence, matrix A_0 is acyclic. Applying Theorem 1, the unique solution of Equation (1) is:

$$\begin{cases} X(1) & = A_0^* \otimes \Gamma \\ X(n) & = A_0^* \otimes A_1 \otimes X(n-1) \stackrel{def}{=} \widetilde{A} \otimes X(n-1) \end{cases} \tag{2}$$

4.2 Evolution Equation of Asynchronous Pipeline

In terms of the timed event graph modeling of the linear asynchronous pipeline, we may get

$$A_0 = \begin{pmatrix} \varepsilon & \varepsilon & \dots & \varepsilon & \varepsilon \\ \sigma_{1,2} & \varepsilon & \dots & \varepsilon & \varepsilon \\ \varepsilon & \sigma_{2,3} & \dots & \varepsilon & \varepsilon \\ \vdots & & & & \\ \varepsilon & \varepsilon & \dots & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \dots & \sigma_{N-1,N} & \varepsilon \end{pmatrix}, \quad A_1 = \begin{pmatrix} \varepsilon & \sigma_{2,1} & \varepsilon & \dots & \varepsilon \\ \varepsilon & \varepsilon & \sigma_{3,2} & \dots & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \dots & \varepsilon \\ \vdots & & & & \\ \varepsilon & \varepsilon & \varepsilon & \dots & \sigma_{N,N-1} \\ \varepsilon & \varepsilon & \varepsilon & \dots & \varepsilon \end{pmatrix}$$

Using the rule of power operation,

$$\left(A_0^m\right)_{ij} = \begin{cases} \bigotimes_{k=j}^{i-1} \sigma_{k,k+1} & \text{if } i = j+m \\ \varepsilon & \text{otherwise.} \end{cases}$$

According the definition of A^* and $\widetilde{A}$, we can derive

$$\left(A_0^*\right)_{ij} = \begin{cases} \varepsilon & \text{if } i < j \\ e & \text{if } i = j \\ \sum_{k=j}^{i-1} \sigma_{k,k+1} & \text{if } i > j \end{cases}, \quad \widetilde{A}_{ij} = \begin{cases} \varepsilon & \text{if } i < j-1 \\ \sigma_{j,i} & \text{if } i = j-1 \\ \sum_{k=j-1}^{i-1} \sigma_{k,k+1} + \sigma_{j,j-1} & \text{if } i \geq j \end{cases}$$

Proposition 1. *With regard to the linear asynchronous pipeline, matrix* $\widetilde{A}$ *is irreducible.*

Proof. According to the expression of $\widetilde{A}$ shown above, we can discover that for all $i,\ j$,

1. if $i \geq j-1$, $\left(\widetilde{A}\right)_{ij} \neq \varepsilon$,
2. if $i < j-1$, there exists sequence $(i_1 = i,\ i_2 = i_1 + 1,\ \ldots,\ i_p = j)$, s.t. $\left(\widetilde{A}\right)_{i_1,i_2} \bigotimes \left(\widetilde{A}\right)_{i_2,i_3} \bigotimes \cdots \bigotimes \left(\widetilde{A}\right)_{i_{p-1},i_p} \neq \varepsilon$.

□

4.3 Latency of Asynchronous Pipeline

In this subsection, we will formulate the latency of a linear asynchronous pipeline by its evolution equation under the max-plus algebraic formalism.

Definition 4. *Latency is the delay between the input of a data item until the corresponding output data item is produced, i.e., the time difference between the firing of transition* q_0 *and that of transition* q_{N+1}.

$$T_{latency} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \left[X_0(i) - X_{N+1}(i) \right] \tag{3}$$

According to Proposition 1 and Corollary 1, we can formulate the latency

$$T_{latency} = \frac{1}{c} \sum_{i=n_0}^{n_0+c} \left[X_0(i) - X_{N+1}(i) \right] \tag{4}$$

5 Case Study

Now we apply our method to a 3-stage linear asynchronous pipeline composed of 4-phase bundled-data pipeline stages, illustrating that the analysis is quite feasible.

The linear pipeline's control circuit is implemented by the semi-decoupled latch controller[9]. SPICE analysis has been performed on the latch controller's layout for worst case conditions (Vdd=4.6V, slow-slow process corner, at 100^oC). The data processing function between latch stages is modeled with an delay elment, where $Cl_1 = 21.3nS$, $Cl_2 = 12.4nS$ and $Cl_3 = 27.2nS$. According to the SPICE analysis results and the matching delays of the processing logic, we may calculate the holding times and estimate the latency, shown in Table 1. The FIFO represents a pipeline with no data processing logics between latch stages.

From Table 1, we may find that the estimated results of $(max, +)$ method are very close to those of the SPICE simulation. On the one hand, it lies in the precise holding time from the SPICE analysis of the 4-phase bundled-data latch control circuit. On the other hand, it confirms the truth of our method, which is uncomplicated and in effect.

Table 1. Latency Estimation

	Holding Time (nS)								Result (nS)	
	$\sigma_{0,1}$	$\sigma_{1,2}$	$\sigma_{2,3}$	$\sigma_{3,4}$	$\sigma_{1,0}$	$\sigma_{2,1}$	$\sigma_{3,2}$	$\sigma_{4,3}$	$(max,+)$	SPICE
FIFO	4.1	4.1	4.1	0	0	4.7	4.7	4.7	16.4	14.1
Pipeline	24.4	16.5	31.3	0	0	4.7	4.7	4.7	103.5	94.7

6 Conclusion

The contribution of this paper is introducing a method for estimating the latency of an asynchronous pipeline, which is effective and efficient.

Nevertheless, the technique has significantly more room for improvement. More sophisticated stochastic holding time sequences can be used to replace the fixed time so as to characterize the performance precisely.

Acknowledgment

This work is supported by the Chinese National Natural Science Foundation under grant No.90407022.

References

1. Ramamoorthy, C.V., Ho, G.S.: Performance evaluation of asynchronous concurrent systems using Petri nets. IEEE Transitions on Software Engineering. **6(5)** (1980) 440–449
2. Kudva, P., Gopalakrishnan, G., Brunvand, E., Akella, V.: Performance analysis and optimization of asynchronous circuits. Proc. International Conf. Computer Design (ICCD). (1994) 221–225
3. Xie, A., Beerel, P.A.: Accelerating Markovian analysis of asynchronous systems using state compression. IEEE Transactions on Computer-Aided Design. **18(7)** (1999) 869–888
4. Xie, A., Beerel, P.A.: Symbolic techniques for performance analysis of asynchronous systems based on average time separation of events. Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC). IEEE Computer Society Press. (1997) 64–75
5. Baccelli, B., Cohen, G., Olsder, G.J., Quadrat, J.-P.: Synchronization and Linearity. Wiley (1992)
6. Altman, E., Gaujal, B., Hordijk,A.: Discrete-Event Control of Stochastic Networks: Multimodularity and Regularity. Springer (2003)
7. Sparsø, J., Furber, S.: Principles of Asynchronous Circuit Design: A Systems Perspective. Kluwer Academic Publishers (2001)
8. Murata, T.: Petri nets: properties, analysis and applications. Proceedings of the IEEE, **77(4)** (1989) 541-580
9. Furber, S.B., Day, P.: Four-phase micropipeline latch control circuits. IEEE Trans. Very Large Scale Integr. Syst. **4(2)** (1996) 247–253