Response Time Properties of Asynchronous Linear Pipelines.

Robert Berks and Jo Ebergen*
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
{rtberks,jebergen}@maveric.uwaterloo.ca

October 7, 1997

Abstract

Extending previous work [4], we give bounds on the worst and average-case response times of asynchronous linear pipelines. The delays of the cells are bounded by a general variable delay model that allows for both forward and reverse latencies. Each cell may have a delay distribution different from and independent from other cells. We derive simple closed-form formulae that bound the worst and average-case response time. Our results are simple, provable, require no simulation, and allow high-level reasoning about performance of pipelines.

1 Introduction

We present some properties on the response time of linear pipelines that are an extension of ideas presented in [4] to make the results both simpler and more easily applicable, yet far more general.

We consider linear pipelines where each of the component cells may have a variable delay, and each component may have different delay characteristics from other components. Pipelines of asynchronous cells can have these features, and it is interesting to see the timing properties of such an architecture. Unlike synchronous pipelines, we cannot derive throughput information from the clock frequency, so we require a different kind of analysis.

Consider a linear array of $L + 1$ cells as indicated in Figure 1 (where $L = 3$). We are only

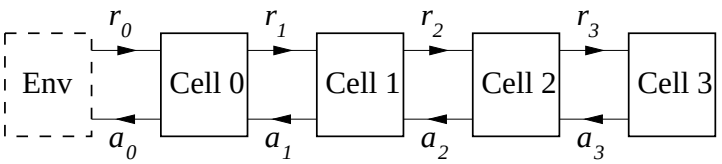

Figure 1: A linear array of cells

*Currently on leave at Sun Microsystems Laboratories, Mountain View, USA.

interested in the communication behaviour at the interface of each cell as an ordering of events. Each interface between two cells has the same communication behaviour, viz., a repetition of a request r followed by an acknowledgment a. A request followed by an acknowledgment is called a handshake. The end cell handshakes at its only interface, the other cells perform handshakes at both interfaces such that in each cycle one handshake occurs at either side.

The requests and acknowledgments are considered as instantaneous events, and delays can be incurred in the cells or the environment of the pipeline. For each cell we assume that the delays between the receipt of the *last* input causing an output and the production of that output is always bounded from above and below. Unlike in [4], each cell may have a set of bounds different from other cells, and we also distinguish between whether a signal is transmitted through a cell in a "forward" or "reverse" direction. The lower bounds for the forward and reverse latency of cell i are given by $\delta_{i,f}$ and $\delta_{i,r}$ respectively. Similarly, the upper bounds for the forward and reverse latency for cell i are given by $\Delta_{i,f}$ and $\Delta_{i,r}$ respectively. We also assume a lower bound (δ_e) and upper bound (Δ_e) for the environment delay. Surprisingly, despite this great increase in the generality of the delay model over our previous work, the results are even simpler than before.

The delays of the cells do not have to be fixed, but may vary between a fixed upper and lower bound. Delays may depend on the data received, or they may vary over different instances of the same cell, or they may vary over time.

Response time (RT) is defined as the time between a request from the environment (r_0) and the successive acknowledgment (a_0). We give a closed-form formula for the upper bound of response-time RT under any distribution of delays. We also give both lower bounds and upper bounds for the average response time of pipelines, denoted by AR, under any distribution of delays.

Although our results give bounds, not specific values, our approach has the following advantages.

- We need not give specific implementations of cells or compute specific distributions of delays. Different implementations must only satisfy the upper bounds and lower bounds for the delays of the basic cells.
- The results can be proved analytically, without any simulation.
- We can express the results as closed-form formulae. This allows us to reason at a higher level about an architecture. We can also experiment rapidly with different parameters values.

As an example of our formulae, we show that

- $RT \leq \max(\{\Delta_{k,r} + \Delta_{k-1,f} + \sum_{h=0}^{k-1}(\Delta_{h,r} - \delta_{h,r}) - \delta_e \mid 1 \leq k \leq L\} \cup \{\Delta_{0,r}\})$
- $mc - \Delta_e \leq AR \leq MC - \delta_e$

The parameters are defined in Table 1 and will be further explained in the following sections. The upper bound for the response time RT is strongly related to the reverse latencies in the summation term and the pipeline length L. Surprisingly, there is only one term in this formula relating to forward latencies. Furthermore, the bounds for the average response AR time depend on certain pipeline bottlenecks, as defined by mc and MC, and is independent of L.

Our architectural model is restricted to pipelines. Nevertheless this offers a rich area of designs. There are many pipeline designs for various computations all where data is flowing in one or both

directions, see [1, 5, 6, 10, 15, 17] for example. If we know what the values are for each of the parameters for certain implementations, we can quickly determine what the upper bounds for the worst-case and average response times are. Our delay model is also appropriate when we use data dependent completion signals, as in [11, 14, 19] for example.

Symbol	Usage
$\delta_{h,r}$	Lower bound on "reverse" latency of cell h
$\Delta_{h,r}$	Upper bound on "reverse" latency of cell h
$\delta_{h,f}$	Lower bound on "forward" latency of cell h
$\Delta_{h,f}$	Upper bound on "forward" latency of cell h
δ_e	Lower bound on the environment delay
Δ_e	Upper bound on the environment delay
L	Number of non-end cells and index of end cell
RT	Worst-case response time
AR	Average response time
mc	Minimum cycle time between adjacent cells $mc = \max\{\Delta_e + \delta_{0,r}, \delta_{0,f} + \delta_{1,r}, \delta_{1,f} + \delta_{2,r}, \ldots, \delta_{L-1,f} + \delta_{L,r}\}$
MC	Maximum cycle time between adjacent cells $MC = \max\{\delta_e + \Delta_{0,r}, \Delta_{0,f} + \Delta_{1,r}, \Delta_{1,f} + \Delta_{2,r}, \ldots, \Delta_{L-1,f} + \Delta_{L,r}\}$

Table 1: Summary of symbols used.

2 Related Work.

The delay assumptions and dependency graphs (also called process graphs) in this paper are inspired by the model used in [8, 9]. In [8, 9] algorithms are described to obtain exact bounds for the delay between two given occurrences of events in a given dependency graph under a variable delay model. A much larger class of networks and communication behaviours is allowed than we use here. In contrast, give formulae for a small set of regular process graphs. Our formulae have parameters, like the depth of the network and the upper and lower bounds for certain delays.

More detailed performance analysis and optimization techniques, all the way down to transistor sizing, are given in [3]. Techniques for analyzing the throughput and latency of micropipelines and rings are proposed in [18], where formulae are derived for the throughput as a function of the forward and backward latencies of the stages and the number of tokens in the ring. However, a fixed-delay model is used. See also [16], where these techniques have been applied. A fixed delay model is also used in [13], where an algorithm is presented to compute the average cycle time of arbitrary Signal Graphs. In [7] formulae for upper bounds of the utilization of pipelines are presented under various assumptions for the delay distributions for the latencies of the cells. In [15] yet another model is used for analyzing the response time of implementations. The model is based on the notion of sequence functions and is more restrictive than a variable delay model.

The algorithms in [20] compute average time separation of events using an iterative numerical procedure to find the steady state. For each set of values for the parameters, the algorithm can compute the average. The algorithm is applicable to a wide range of process graphs. A simulation approach is also found in Kearney and Bergmann [12].

3 Communication Behaviours

Consider the cell in Figure 2 from a linear array. At interface i, the communication behaviour for

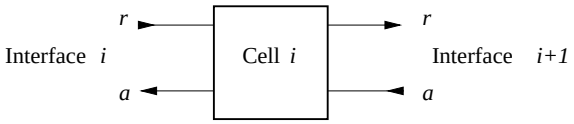

Figure 2: One cell

the handshakes can be given by $*[r_i?; a_i!]$. That is, we take an input request, $r_i?$, and then we output an acknowledgement $a_i!$. (? denotes input, while ! denotes output.) This behaviour then repeats (indicated by $*[..]$). At interface $i+1$ the communication behaviour is given by $*[r_{i+1}!; a_{i+1}?]$.

Cell i synchronizes the handshakes at interface i and $i+1$. We restrict ourselves to behaviours for cells where for every handshake on interface i there is a handshake on interface $i+1$. More precisely, the communication behaviours we look at are given by

$$(r_i?;\ *[\,(a_i!;\ r_i?) \| (r_{i+1}!;\ a_{i+1}?)\,])$$

In other words, every occurrence of the segment $r_{i+1}!;\ a_{i+1}?$ takes place in parallel with the handshake $a_i!;\ r_i?$.

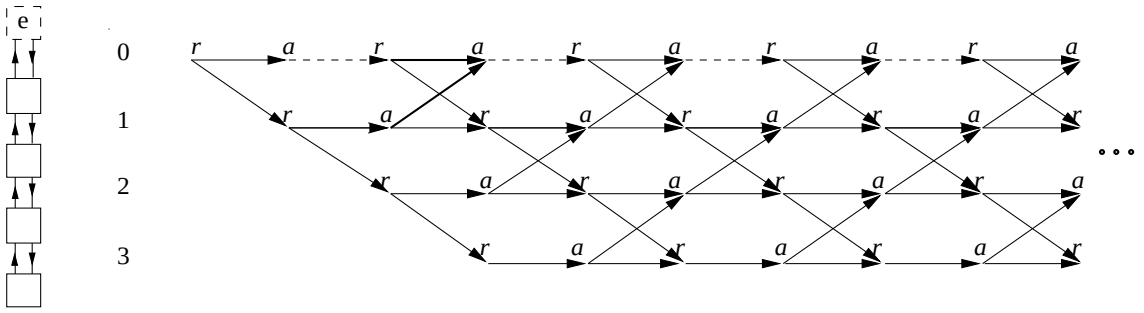

Figure 3: Unfolded communication behaviour for a linear array of four cells.

If we consider a complete network of 3 non-end cells, one end cell, and the environment of the pipeline, then we can depict its behaviour in a process graph (Figure 3), where all edges represent direct precedences brought about by some component in the network. The dashed arrows represent direct precedences caused by the environment of the pipeline. Let us denote the k-th occurrence of event a_i by (a_i, k), where $k \geq 0$. Such a process graph gives a precedence relation for occurrences of events. A precedence between occurrences of events e_0 and e_1 is denoted by $e_0 \rightarrow e_1$. For example, we have the following precedences highlighted in Figure 3.

$$(r_0, 1) \rightarrow (a_0, 1) \quad \text{and} \quad (a_1, 0) \rightarrow (a_0, 1)$$

These precedences indicate that occurrence 1 of event a_0 can only happen after occurrence 1 of event r_0 and occurrence 0 of event a_1 has happened.

4 The Delay Model

We need to add a delay model to our behavioural model. Let c be a node in a (unfolded) process graph with c an output of cell h. $T(c)$ is the time assigned to node c and

$$T_{pred}(c) = \max\{T(b) \mid b \text{ is a direct predecessor of c}\}$$

We will stipulate that the delay is always measured from the time $T_{pred}(c)$, when the *last* directly preceding event occurs, to the actual occurrence of the event c. Our delay model prescribes that any timing assignment T of the process graph must satisfy the following constraints.

$$T_{pred}(c) + \delta_{h,f} \leq T(c) \leq T_{pred}(c) + \Delta_{h,f}$$

if node c is a request, and

$$T_{pred}(c) + \delta_{h,r} \leq T(c) \leq T_{pred}(c) + \Delta_{h,r}$$

if node c is an acknowledgment.

The constraints prescribe that "forward" delays through cell h have lower bound $\delta_{h,f}$ and upper bound $\Delta_{h,f}$, that is

$$\delta_{h,f} \leq \text{cell } h \text{ forward delays} \leq \Delta_{h,f}$$

Similarly, "reverse" delays for cell h are bounded by $\delta_{h,r}$ and $\Delta_{h,r}$.

$$\left.\begin{array}{l} r_h \rightarrow r_{h+1} \\ a_{h+1} \rightarrow r_{h+1} \end{array}\right\} \text{ forward latencies for cell } h.$$

$$\left.\begin{array}{l} r_h \rightarrow a_h \\ a_{h+1} \rightarrow a_h \end{array}\right\} \text{reverse latencies for cell } h.$$

The end cell is associated with a reverse latency only and the environment is associated with a forward latency. We label the lower bound and upper bound on the environment delays δ_e and Δ_e respectively.

For worst-case response time we will minimize the environment delay when we maximize the pipeline delay, and vice-versa for best-case response time. Observe the definitions of Maximum and Minimum Cycle time given in Table 1.

4.1 A Micropipeline

Suppose we have a micropipeline control-circuit as in Figure 4. Micropipelines were described in [17] and are used by the asynchronous versions of the ARM processor[6]. In our model Figure 4 corresponds to a pipeline with three non-end cells and one end-cell. Hence, we would model it as a linear pipeline with parameter $L = 3$. Micropipeline stages have one handshake on the right to every handshake on the left.

We assume that the only delays in the circuit are the buffer delays. That is, we assume that the C-elements and wiring have zero-delay. This causes no loss of generality, because we can always model the delays of the C-elements with the delays of the buffers.

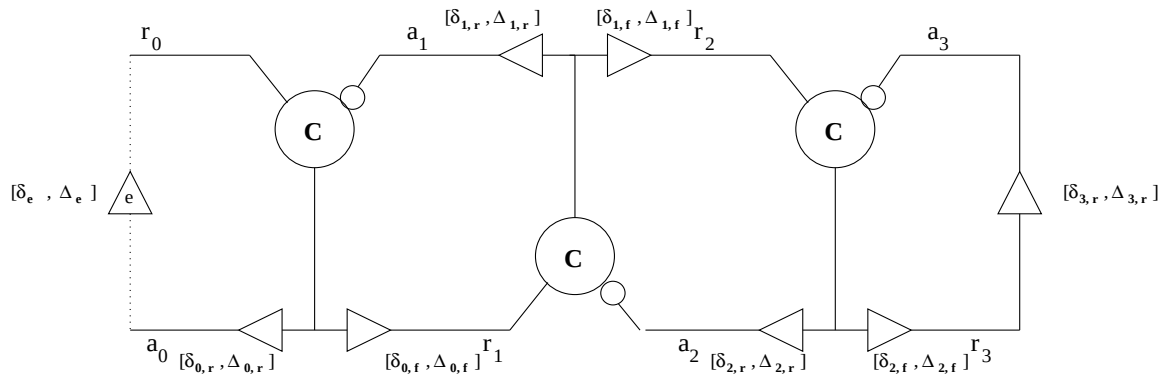

Figure 4: A micropipeline with $L = 3$. The environment delay is labelled with an e.

4.2 Critical Paths

When a dependency graph and a timing assignment is given, we define the useful notion of *critical paths*. We define an edge $b \to c$ to be a *critical edge* if either

(i) $c = r_0$, and $\delta_e \leq T(c) - T(b) \leq \Delta_e$ where b must be a_0.

(ii) $c = r_{h+1}$, and $\delta_{h,f} \leq T(c) - T(b) \leq \Delta_{h,f}$ for some b.

(iii) $c = a_h$, and $\delta_{h,r} \leq T(c) - T(b) \leq \Delta_{h,r}$ for some b.

So we have three cases: environment delays, forward delays and reverse delays. The lower bound δ_x ($x \in \{e, (h,f), (h,r)\}$ depending on the type of edge) is automatic by our statement that edge delays must be at least δ_x. The upper bound may not always be true, because other dependencies may make event c occur later. A directed path is a *critical path*, if all its edges are critical edges.

We can view the critical edges as pieces of elastic which can "stretch" between lengths δ_x and Δ_x. If we stretch an edge beyond Δ_x it breaks. On a critical path, we know that no bit of elastic on that path is broken, since, by the choice of the edges on the path, none of the bits of elastic is stretched beyond its upper bound, Δ_x.

5 Worst-Case Response Time

Theorem 1 *The worst-case response time RT for any pipeline (as defined in Section 3 operating under a delay model of Section 4) is bounded from above as*

$$RT \quad \leq \quad \max(\{\Delta_{k,r} + \Delta_{k-1,f} + \sum_{h=0}^{k-1}(\Delta_{h,r} - \delta_{h,r}) - \delta_e \mid 1 \leq k \leq L\} \cup \{\Delta_{0,r}\})$$

□ *(Proof omitted)* [2]

Notice that the summation term depends only on the reverse latencies. As the length of the pipeline gets larger the summation term can get larger, though ultimately the worst case depends on the cycle time between cells k and $k-1$. Consequently, the bound becomes smaller if we put the

adjacent cells with the slowest cycles (i.e. $\Delta_{k,r} + \Delta_{k-1,f}$) near the entrance of the pipeline, as this is where the summation term is minimized.

The bound above may not be tight and cannot be simplified further without further assumptions about the variables. In many implementations the variation among cell delays is only in the forward latencies. The reverse latencies may be fixed, i.e., $\Delta_{h,r} = \delta_{h,r}$ for $0 \leq h \leq L$. With this assumption the summation term disappears and the result is independent of the length of the pipeline.

$$\begin{aligned} RT &\leq \max(\{\Delta_{k,r} + \Delta_{k-1,f} + \sum_{h=0}^{k-1}(0) - \delta_e \mid 1 \leq k \leq L\} \cup \{\Delta_{0,r}\}) \\ &= \max(\{\Delta_{k,r} + \Delta_{k-1,f} - \delta_e \mid 1 \leq k \leq L\} \cup \{\Delta_{0,r}\}) \\ &= \max(\{\Delta_{k,r} + \Delta_{k-1,f} \mid 1 \leq k \leq L\} \cup \{\Delta_{0,r} + \delta_e\}) - \delta_e \\ &= MC - \delta_e \end{aligned}$$

In other words, in a linear pipeline with no variation in reverse latencies per cell, the worst-case response time is at most the maximum cycle time between adjacent cells.

6 Average-Case Response Time

In a previous paper[4] we described amortized response time as an upper bound on the average response time. That is, over many request/acknowledgment cycles, we also calculate the worst possible delay caused by the pipeline, and divide this by the number of cycles. Here we also find a lower bound on the average response time by a similar technique. Averaged over many request/acknowledgment cycles we calculate the *minimum* possible delay caused by the pipeline.

Theorem 2 *The average response time AR for any pipeline (as defined in Section 3 operating under a delay model of Section 4), is bounded as follows*

$$mc - \Delta_e \leq \quad AR \quad \leq MC - \delta_e$$

Proof. (Upper bound.) From acknowledgment (a_0, j_0) there is a critical path to some request (r_0, i_0) with $i_0 \leq j_0$ and on this critical response path there is no other event at level 0. Let this path be of length $2k + 1$ with $k \geq 0$. There may be many paths between (r_0, i_0) and (a_0, j_0) of length $2k+1$. We label each edge on each path with maximum delays and then we choose a critical path P. This path P is the longest path (in terms of delay) between (r_0, i_0) and (a_0, j_0) excluding paths through the environment. We call the delay on this critical path of length $2k + 1$, D_k.

Between (r_0, i_0) and (a_0, j_0) there are k environment edges (i.e, $a_0 \rightarrow r_0$) and $k + 1$ "response" edges (i.e, $r_0 \rightarrow a_0$). Environment delays are at least δ_e. In k handshake cycles, the total time for the environment delays is at least $k\delta_e$. To obtain the upper bound on the average response time we subtract the environment delay over k handshake cycles and divide by the number of responses.

$$\begin{aligned} AR &\leq \max\{D_k - k\delta_e)/(k+1) \mid k \geq 0\} \\ &= \max\{D_k + \delta_e)/(k+1) \mid k \geq 0\} - \delta_e \end{aligned}$$

We look in more detail at the first term $\max\{D_k + \delta_e)/(k+1) \mid k \geq 0\}$.

Suppose there is either an edge $r_i \rightarrow r_{i+1}$ or an edge $a_{i+1} \rightarrow r_{i+1}$ in our path, P. r_{i+1} is the output request signal of cell i. The path must pass back through cell i at some point which means

there must be an acknowledgement a_{i+1} forthcoming from cell $i+1$. So, for each edge on the path that ends in r_{i+1} there must be a corresponding edge that ends in a_{i+1}. For each pair of edges we have a maximum delay of $\Delta_{i,f} + \Delta_{i+1,r} \leq MC$. There are k edges ending in a request in the path and so we have k pairs with total delay less than or equal to $k \cdot MC$. We also have one "loose" reverse edge at the end of the path. So the total delay on the path is bounded as $D_k \leq k \cdot MC + \Delta_{0,r}$.

$$\begin{aligned} \max\{D_k + \delta_e)/(k+1) \mid k \geq 0\} &\leq \max\{k \cdot MC + \Delta_{0,r} + \delta_e)/(k+1) \mid k \geq 0\} \\ &\leq \max\{k \cdot MC + MC)/(k+1) \mid k \geq 0\} \\ &= MC \end{aligned}$$

But we also note that there exist paths P that almost exclusively consist of handshake cycles with delay MC, except for a begin part and end part. As $k \to \infty$, we have for delay D_k on these paths

$$\lim_{k\to\infty} (D_k + \delta_e)/(k+1) = \lim_{k\to\infty} (k \cdot MC + \Delta_{0,r} + \delta_e)/(k+1) = MC$$

From the above two equations we find that $\max\{D_k + \delta_e)/(k+1) \mid k \geq 0\} = MC$. Consequently

$$AR \leq \max\{D_k + \delta_e)/(k+1) \mid k \geq 0\} - \delta_e = MC - \delta_e$$

We can use a similar explanation to prove the lower bound. Let the same paths exist as before except the delay on each non-environment edge is minimized. Let the critical path of length $2k+1$ have delay d_k. Since we are trying to find a lower bound, we have to maximize the environment delay when we subtract it. We obtain

$$\begin{aligned} AR &\geq \max\{d_k - k\Delta_e)/(k+1) \mid k \geq 0\} \\ &= \max\{d_k + \Delta_e)/(k+1) \mid k \geq 0\} - \Delta_e \end{aligned}$$

The same steps as above show that $\max\{d_k + \Delta_e)/(k+1) \mid k \geq 0\} = mc$ and the result follows. □

The above theorem states that the average response time of a pipeline is bounded by two bottlenecks. One bottleneck is described by the minimum cycle time mc and the other by the maximum cycle time MC. These bottlenecks need not occur at the same place in the pipeline. If one has the freedom to rearrange stages in the pipeline one might be tempted to split up stages that have a large cycle time between them so as to minimize MC. Two remarks are in order, however. Firstly, if the reverse latency of stages are identical, then there is no benefit in rearranging stages. Secondly, our formulae are only dealing with bounds. As a consequence, these formulae say what happens to the bounds of the average-case response time under any rearrangement of stages, but they do not say what happens with the actual average-case response time.

7 An Example Micropipeline

We refer back to the example given in section 4.1 where we have a pipeline with $L = 3$. We instantiate the delay elements as in Table 2. After some calculations we find that the minimum cycle time $mc = \Delta_e + \delta_{0,r} = 4+1 = 5$ and the maximum cycle time $MC = \Delta_{1,f} + \Delta_{2,r} = 5+3 = 8$.

Suppose we are interested in average response time. Since $mc - \Delta_e \leq AR \leq MC - \delta_e$ we have $1 \leq AR \leq 6$. If we are interested in worst-case response time, we need to do (a little) more effort

to instantiate our formulae. Skipping some steps we have,

$$\begin{aligned} RT &\leq \max(\{\Delta_{k,r} + \Delta_{k-1,f} + \sum_{h=0}^{k-1}(\Delta_{h,r} - \delta_{h,r}) - \delta_e \mid 1 \leq k \leq L\} \cup \{\Delta_{0,r}\}) \\ &= \max(\{3,6,7\} \cup \{1\}) \\ &= 7 \end{aligned}$$

h	$\delta_{h,f}$	$\Delta_{h,f}$	$\delta_{h,r}$	$\Delta_{h,r}$
e	2	4	$--$	$--$
0	2	4	1	1
1	1	5	1	1
2	1	3	1	3
3	$--$	$--$	2	4

Table 2: Delays for a pipeline of length $L = 3$.

8 Conclusions

For linear pipelines with a simple handshake behaviour, we have given formulae for bounds on the average and worst-case response time. Although the formulae are simple, one must always be aware that we are dealing with bounds and that the formulae do not express what happens with the actual worst and average-case response times under a specific delay distribution.

The bounds on the average-case response time are dependent only on two bottlenecks in the pipeline and not on the pipeline length. One bottleneck constrains the maximal average throughput, and the other bottleneck constrains the minimal average throughput. Both bottlenecks depend on a cycle of forward and reverse latencies between two stages. This observation led to the conclusion that we can minimize the bounds by minimizing the cycle times mc and MC between cells.

The upper bound for the worst-case response time depends mainly on the reverse latencies and the length of the pipeline. The formula contained only one term with a forward latency. This formula led to the conclusion that we can minimize the bound for the worst-case response by putting a stage with a big delay closer to the front of the pipeline.

References

[1] K. v. Berkel and M. Rem. VLSI programming of asynchronous circuits for low power. In G. Birtwistle and A. Davis, editors, *Asynchronous Digital Circuit Design*, Workshops in Computing, pages 152–210. Springer-Verlag, 1995.

[2] R. Berks and J. Ebergen. Response time properties of linear pipelines with varying cell delays. http://maveric0.uwaterloo.ca/publication.html, 97.

[3] S. M. Burns and A. J. Martin. Performance analysis and optimization of asynchronous circuits. In C. H. Séquin, editor, *Advanced Research in VLSI: Proceedings of the 1991 UC Santa Cruz Conference*, pages 71–86. MIT Press, 1991.

[4] J. Ebergen and R. Berks. Response time properties of some asynchronous circuits. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, Apr. 1997.

[5] J. Ebergen and R. Hoogerwoord. A derivation of a serial-parallel multiplier. *Science of Computer Programming*, 15:201–215, 1990.

[6] S. Furber. Computing without clocks: Micropipelining the ARM processor. In G. Birtwistle and A. Davis, editors, *Asynchronous Digital Circuit Design*, Workshops in Computing, pages 211–262. Springer-Verlag, 1995.

[7] M. R. Greenstreet and K. Steiglitz. Bubbles can make self-timed pipelines fast. *Journal of VLSI Signal Processing*, 2(3):139–148, Nov. 1990.

[8] H. Hulgaard, S. Burns, T. Amon, and G. Borriello. An algorithm for exact bounds on the time separation of events in concurrent systems. Technical Report TR-94-02-02, University of Washington, Department of Computer Science and Engineering, Feb. 1994.

[9] H. Hulgaard and S. M. Burns. Bounded delay timing analysis of a class of CSP programs with choice. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 2–11, Nov. 1994.

[10] M. B. Josephs, P. G. Lucassen, J. T. Udding, and T. Verhoeff. Formal design of an asynchronous DSP counterflow pipeline: A case study in handshake algebra. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 206–215, Nov. 1994.

[11] D. Kearney and N. W. Bergmann. Bundled data asynchronous multipliers with data dependant computation times. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, Apr. 1997.

[12] D. Kearney and N. W. Bermann. Performance evaluation of asynchronous logic pipelines with data dependant processing delays. In *Asynchronous Design Methodologies*, pages 4–13. IEEE Computer Society Press, May 1995.

[13] C. D. Nielsen and M. Kishinevski. Performance analysis based on timing simulation. In *Proc. ACM/IEEE Design Automation Conference*, pages 70–76, June 1994.

[14] S. M. Nowick, K. Y. Yun, and P. A. Beerel. Speculative completion for the design of high-performance asynchronous dynamic adders. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, Apr. 1997.

[15] M. Rem. Trace theory and systolic computations. In J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, editors, *PARLE: Parallel Architectures and Languages Europe, Vol. I*, volume 258 of *Lecture Notes in Computer Science*, pages 14–33. Springer-Verlag, 1987.

[16] J. Sparsø and J. Staunstrup. Design and performance analysis of delay insensitive multi-ring structures. In *Proc. Hawaii International Conf. System Sciences*, volume I, pages 349–358. IEEE Computer Society Press, Jan. 1993.

[17] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.

[18] T. E. Williams. Analyzing and improving the latency and throughput performance of self-timed pipelines and rings. In *Proc. International Symposium on Circuits and Systems*, May 1992.

[19] T. E. Williams and M. A. Horowitz. A zero-overhead self-timed 160ns 54b CMOS divider. *IEEE Journal of Solid-State Circuits*, 26(11):1651–1661, Nov. 1991.

[20] A. Xie and P. 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*. IEEE Computer Society Press, Apr. 1997.