AN EFFICIENT TIMING MODEL FOR HARDWARE IMPLEMENTATION OF MULTIRATE DATAFLOW GRAPHS

Nitin Chandrachoodan, Shuvra S. Bhattacharyya [†] *and K. J. Ray Liu*

Department of Electrical and Computer Engineering,
University of Maryland, College Park, MD 20742
(nitin,ssb,kjrliu@eng.umd.edu)

ABSTRACT

We consider the problem of representing timing information associated with functions in a dataflow graph used to represent a signal processing system in the context of high-level hardware (architectural) synthesis. This information is used for synthesis of appropriate architectures for implementing the graph. Conventional models for timing suffer from shortcomings that make it difficult to represent timing information in a hierarchical manner, especially for multirate signal processing systems.

We identify some of these shortcomings, and provide an alternate model that does not have these problems. We show that with some reasonable assumptions on the way hardware implementations of multirate systems operate, we can derive general hierarchical descriptions of multirate systems similarly to single rate systems. Several analytical results such as the computation of the iteration period bound, that previously applied only to single rate systems can also easily be extended to multirate systems under the new assumptions.

We have applied our model to several multirate signal processing applications, and obtained favorable results. We present results of the timing information computed for several multirate DSP applications that show how the new treatment can streamline the problem of performance analysis and synthesis of such systems.

1. INTRODUCTION

High-Level Synthesis (HLS) refers to the task of deriving an efficient architecture and implementation of a system based on an abstract description of its functionality. In HLS for digital signal processing (DSP) applications, the algorithm is often represented in the synchronous dataflow (SDF) model [1] as an SDF graph whose vertices represent functions and edges represent communication or dependencies. This model uses *consumption* and *production* parameters on edges to model multiple sample rates in an application. To map such a dataflow graph onto an architecture (either hardware or software) efficiently, we need to annotate the application specification and architecture with information about the execution times of vertices, and the area utilization and power consumption of processing resources. The timing information is used to generate a set of constraints related to the system that the actual implementation must satisfy.

The conventional model for describing timing in this context is derived from the method used in combinational logic analysis. Here each vertex is assigned a single value (called the "propagation delay") representing the maximum delay among all its input-output pairs.

A major disadvantage of this approach is that it does not allow an efficient hierarchical description of the system timing when the system contains delay elements (iterative systems) and a hardware implementation is desired. Delay elements roughly correspond to registers in a hardware implementation, but are more flexible in that they do not impose the restriction that all the delay elements are activated at the same instant of time [2, 3, 4]. This kind of variable phase clocking has been recognized as a useful feature even in sequential logic synthesis [5, 6].

In multirate systems under the SDF model, the most common interpretation of *execution time* is as follows: each vertex is assumed to be enabled when sufficient dataflow tokens have enqueued on its inputs. Once enabled, it can *fire* at any time, consuming a number of tokens from each input edge equal to the consumption parameter on that edge, and producing a number of tokens on each output edge equal to the production parameter on that edge. The execution time of the vertex is the time between the (instantaneous) consumption and production events.

This model has been used in the context of SDF to derive several useful results regarding consistency, liveness, and throughput of graphs modeling DSP systems. However the treatment is quite different from that for single-rate graphs, and many analytical results for single rate systems cannot be extended to multirate systems.

To the best of our knowledge, there does not appear to be any other timing model that addresses the hierarchical timing issues for dataflow based DSP system design. Conventional models cannot easily be used to represent systems that are either hierarchical or contain multirate elements. Models such as the processor timing data used in [7] capture the effects of real system parameters and latency for single rate systems, but they do not provide ways to take advantage of skewed clock phases or multirate graphs directly. Multirate systems are usually handled by some technique such as deriving the homogeneous equivalent expanded graph (which can lead to an exponential increase in the graph size), while hierarchical systems need to be completely flattened and expanded in the context of the overall graph.

In this paper, we propose an extension of the hierarchical timing pair (HTP) model [8] that overcomes these difficulties. For multirate systems, the new model allows a treatment very similar to that for normal single rate systems, while still allowing important features of the multirate execution to be represented. The

[*] This research was supported in part by the US National Science Foundation Grant #9734275 and NSF NYI Award MIP9457397

[†] Also with the University of Maryland Institute for Advanced Computer Studies.

0-7803-7041-4/01/$10.00 ©2001 IEEE

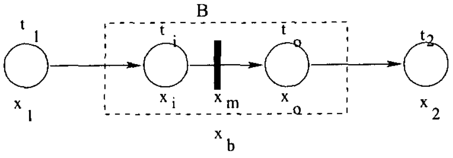


Figure 1: Timing of complex blocks.

model also allows several analytical results for single rate systems to be applied to multirate systems. As an example we derive an expression for the iteration period bound of a multirate graph.

We have used our model to compute timing parameters of a number of multirate graphs used in signal processing applications. The results show that the new model can result in compact representations of fairly large systems that can then be used as hierarchical subsystems of larger graphs.

In the next section, we present an overview of the HTP model for single rate systems. Section 3 then considers multirate systems, and extends the timing pair model to handle multirate systems. Section 4 presents results of applying the model to several examples from signal processing, and finally we present our conclusions and some interesting directions for further work.

2. THE HIERARCHICAL TIMING PAIR MODEL FOR SINGLE RATE SYSTEMS

The Hierarchical Timing Pair model is a new model for representing timing information in dataflow graphs that is described in [8]. We present a brief overview of the main features of the model in order to understand how they can be extended to multirate systems. Further details on the application of the model to both single and multirate systems is found in [8].

The HTP model is developed around the concept of the "constraint time" of a path in a dataflow graph. Consider a path in a dataflow graph such as the path from x_1 to x_2 in the graph of Fig. 1. The first step is to recognize that the timing information associated with each vertex in the graph is used primarily for the purpose of establishing constraints on the earliest time that the vertex can execute (when its inputs are ready).

To provide timing information for a complex block, we should be able to emulate the timing characteristics that this block would imply between its input and output. For Fig. 1, if we were to write the constraints in terms of the internal blocks x_i and x_o, we would obtain

$$x_i - x_1 \geq t_1; x_o - x_i \geq t_i - 1 \times T; x_2 - x_o \geq t_o.$$

We would now like to compute certain information such that if we were to combine the complex block B under the single start time x_b, we would still be able to write down equations that would provide the same constraints to the environment outside the block B. We see that this is achieved by the following constraints:

$$x_b - x_1 \geq t_1; x_2 - x_b \geq t_i + t_o - 1 \times T.$$

In other words, if we assume the execution time of the block B is given by the expression $t_i + t_o - 1 \times T$, we can put down constraints that exactly simulate the effect of the complex block B.

Consider a path from input $v_i = v_1$ of a graph G to output $v_o = v_k$ given by $p : v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$, with edges $e_i : v_i \rightarrow v_{i+1}$ for $i = 1 \cdots k - 1$. Let t_i be the execution time of v_i, and let d_j be the number of delays on edge e_j. We define the *constraint time* of this path as $t_c(p) = \sum_{i=1}^{k} t_i - T \times \sum_{j=1}^{k-1} d_j$.

We use the term c_p to refer to the sum $\sum_{i=1}^{k} t_i$, and m_p to refer to the sum $\sum_{j=1}^{k-1} d_j$. The ordered pair (m_p, c_p) is referred to as a *timing pair*.

When there are multiple paths between the pair of vertices with different numbers of delay elements, it is possible that for different values of the system iteration period T, different paths have the maximum constraint time and are responsible for the actual operational constraint. To handle this, we can store all possible timing pairs in a list and use this list to compute the actual constraint time for a given value of T. Note that the list will not be too large, since we can remove redundant elements using the simple rules which are discussed in [8]. In [8], we also detail an efficient algorithm for computing the timing pair lists required to completely specify the timing information associated with a graph.

3. MULTIRATE SYSTEMS

In this section, we consider some problems that arise in the treatment of multirate systems. We examine some examples to see how these difficulties can be overcome, and motivate new restrictions that make mathematical analysis more tractable. Using these restrictions, which frequently hold in practical hardware implementations of DSP applications, the timing pair model can be extended to multirate systems also.

The conventional interpretation of SDF execution semantics has been based on token counts on edges. A vertex is enabled when each of its input edges has accumulated a number of tokens greater than or equal to the consumption parameter on that edge. At any time after it is enabled, the vertex may fire, producing a number of tokens on each output edge equal to the production parameter on that edge. In the following discussion, we use c to refer to the consumption parameter on an edge, and p to refer to the production parameter. The edge in question will be understood from context.

This interpretation, though very useful in obtaining a strict mathematical analysis of the consistency and throughput of such multirate systems and for design of software implementations, has some unsatisfactory features with regard to dedicated hardware implementations. One is the fact that it results in tokens being produced in bursts of p at a time on output edges and similarly consumed in bursts of c at a time. This is not the consumption pattern in a synchronous hardware implementation of a DSP application, where tokens refer to data samples on edges, and as such, will usually appear periodically at the sample rate specified for that edge.

Another important problem is with regard to the criterion used for firing vertices. Consider the example of the 3 : 5 rate changer shown in Fig. 2. According to the SDF interpretation, this vertex can only fire after 5 tokens are queued on its input, and will then instantaneously produce 3 tokens on its output. However, a real rate changer need not actually wait for 5 tokens before producing its first output. .

Figure 2 illustrates these issues. This is the timing pattern of a 3 : 5 fractional rate conversion assuming a 7-tap filter is used for interpolation. It is clear from the filter length and interpolation rate that the first output in each iteration (an iteration ends when the system returns to its original state) depends on the first 2 inputs only, the second depends on inputs 2 to 4, and the third depends on

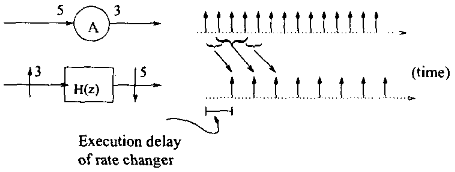


Figure 2: 3 : 5 sample rate changer.

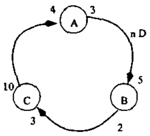


Figure 3: Deadlock in an SDF system: if $n < 10$ the graph deadlocks.

inputs 4 and 5. Therefore, the delay pattern shown in the figure is valid as long as there is sufficient time for the filters to act on their corresponding inputs. In other words, it is not necessary to wait for 5 inputs to be consumed before starting to produce the outputs.

The interpretation we use for execution of SDF graphs is therefore as follows: each node receives its inputs in a periodic stream, and can start computing its outputs some time after the first input becomes available (this time would depend on internal features such as the number of taps in the filter in the above example). The outputs are also generated in a periodic stream at the appropriate rate required for consistency of the system.

An important effect of this alternate interpretation is that it changes the criteria for deadlock in a graph. Under normal SDF semantics, the graph in Fig. 3 would be deadlocked if the edge AB has less than 10 delays on it. On the other hand, 6 delays are sufficient on edge BC, while 16 delays are required on edge CA in order to prevent deadlock. Under the new interpretation, as long as each cycle in the graph contains at least one token, deadlock is broken and the system can execute. This is the same condition that applies to single rate graphs.

It is important to understand that this interpretation of multirate SDF execution is useful because in the context of synchronous hardware implementations, multirate DSP applications rarely require the conventional interpretation in terms of token consumption. Typical multirate blocks in DSP applications are decimators and interpolators, multirate filters (rate changers), block coders and decoders etc. A notable feature of these applications is that few of them actually require a consumption of c tokens before starting to produce p tokens. Even for block coders and serial-to-parallel converters, the data are still periodic, or can be assumed to be so without much loss of performance. This assumption is also useful as it can simplify buffering requirements.

3.1. The HTP model for Multirate systems

We now specify how the HTP model can be applied to the analysis of multirate systems in the SDF formulation with the above execution and timing model. For simplicity and clarity, we assume that the unit to be modeled is a Single-Input Single-Output (SISO) system and that the propagation delay through sub-units is constant. However, our model can easily be extended to handle the Multiple-Input Multiple Output (MIMO) case.

Given a multirate system represented as an SDF graph, we follow the usual technique [1] to compute the repetitions vector for the graph. The *balance equation* on each edge $e : u \rightarrow v$ in the graph is given by $p_e \times q_u = c_e \times q_v$, where p_e is the production parameter on e, c_e is the consumption parameter, and q_u and q_v are the repetition counts for the source and sink actors of the edge. Let T denote the overall iteration period of the graph. This is the time required for each actor to execute the minimum number of times required to return the system to its starting state (the repetition count of the actor). Therefore, the sample period on edge e is given by $T_e = \frac{T}{q_u \cdot p_e} = \frac{T}{q_v \cdot c_e}$.

Now extending the analogy of the single rate case, we define the constraint time on a path as

$$t_c(p) = \sum_{i=1}^{k} t_i - \sum_{j=1}^{k-1} (d_j \times T_j)$$

where T_j is the sample period on edge j. By noting that the effect of a delay on any edge (in both the single rate and multirate cases) is to give an offset of $-T_e$ to the constraint time of any path through that edge, we can see that this gives the correct set of constraints. Also, the values of the starting times for the different vertices that are obtained as a solution to the set of constraints will give a valid schedule for the multirate system.

It is possible to view the constraint times in terms of "normalized delays". Here the delays on each edge are normalized to a value of $d_n(e) = \frac{d_e}{q_u \cdot p_e} = \frac{d_e}{q_v \cdot c_e}$. In terms of the normalized delays, the expression for constraint time becomes the same as that for the single rate case.

For single rate graphs, the minimum iteration period that can be attained by the system is known as the iteration period bound and is known to be equal to the maximum cycle-mean (MCM) [9]. So far, no such tight bound is known for multirate SDF graphs, though some good approximations have been found [10]. Under our proposed model, it is easy to determine an exact bound similar to the one for single rate graphs. By considering the cumulative constraints around a loop for the single rate case, we can easily obtain the iteration period bound $T_{min} = \max_{c \in C} \frac{\sum_c t_u}{\sum_c d_e}$, where C is the set of directed cycles in the graph. Similarly, for the multirate case, we can obtain the result

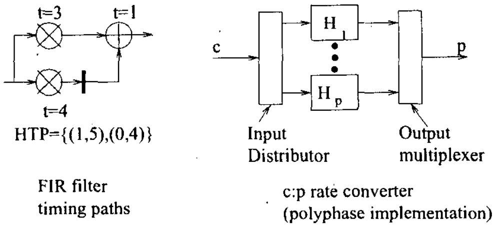
$$T_{min} = \max_{c \in C} \frac{\sum_c t_u}{\sum_c d_n(e)},$$

where T_{min} is the iteration period of the overall system as discussed above. In addition, the start times for each operation are directly obtained as a solution to the constraint system that is set up using the timing information.

One factor here is that, unlike the single rate case, the number of timing pairs in the list for a path is not bounded by the number of delay elements. However, in practice this number is usually quite small.

4. EXAMPLES AND RESULTS

We have applied the HTP model to the SDF graphs representing typical multirate signal processing applications. The examples we have taken are from the Ptolemy system [11] (CD-DAT, DAT-CD conversion, 2 channel non-uniform filter bank) and from [12, p.256] (QMF bank).



FIR filter timing paths

c:p rate converter (polyphase implementation)

Figure 4: Multirate FIR filter structure.

Benchmark	Timing pairs
QMF bank (input to y_3)	$\{(7,15),(3,14),(1,13),(0,12)\}$
CD-DAT (160:147)	$\{(93/32,20),(0,16)\}$
DAT-CD (147:160)	$\{(15/7,15),(0,12)\}$
2 ch. Non.Unif. FB	$\{(5/2,10),(1,9),(0,8)\}$

Table 1: Timing pairs for multirate systems.

The basic unit in several of these examples is the multirate FIR filter that is capable of performing rate conversion as described in section 3. Due to the polyphase implementation of this filter, computing it's timing parameters requires care, as different filter components receive different numbers of inputs. This problem is explained in greater detail in the technical report [8]. For the purpose of the examples we are considering, we assume for the sake of the other multirate examples that any rate conversion is performed using a MR FIR filter that has the timing parameters $\{(1,5),(0,4)\}$.

For the examples we considered, the rate conversions result in several I-O paths with different numbers of delays at different rates. The resulting timing pairs that are obtained for these systems are summarized in Table 1.

To understand these results in context, note that each example contains roughly 5-10 vertices, and the same order of edges in its natural hierarchical representation. If we flatten a design containing 10 such multirate systems, the resulting graph has over 50 vertices. If this is expanded, the result is a very large graph, since the sample rates here result in a large expanded equivalent graph. Since most network algorithms including shortest paths have complexity $O(|V||E|)$ or higher where $|V|$ is the number of vertices and $|E|$ the number of edges, the saving obtained by the new model can be substantial, since it avoids the need both for hierarchical flattening and for deriving the homogeneous equivalent expanded graph.

A general observation we can make about the timing model is that systems that have delay elements in the feed-forward section, such as FIR filters and filters with both forward and backward delays, tend to have more timing pairs than systems where the delay elements are restricted to a relatively small amount of feedback. This is because feedback delay elements must necessarily exist in a loop that has a total negative constraint time, which means they will not contribute towards a dominant constraint time in the forward direction.

5. CONCLUSIONS AND FUTURE DIRECTIONS

We have presented a multirate extension of the Hierarchical Timing Pair model for use in analysis and synthesis of hardware realizations of multirate dataflow graphs. The HTP model is able to efficiently store information about multirate graphs, and allows exact computation of important system parameters such as the iteration period bound easily. We have shown that the HTP model overcomes many limitations of conventional timing models, while making certain frequently applicable assumptions on the execution patterns of multirate systems.

We have considered several typical multirate DSP applications and computed timing pairs for these models. The results demonstrate the power of our approach. In particular, the results show that the model can be used to obtain large reductions in the amount of information about the circuit that we need to store in order to use its timing information in the context of a larger system.

The model as it exists now requires the ability to choose the start times of operations (variable phase clocking). We are currently examining ways of extending the model to more general circuits, that include some fixed phase registers along with other nodes in which the delay can be adjusted.

6. REFERENCES

[1] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, Sep 1987.

[2] M. Potkonjak and M. Srivastava, "Behavioral optimization using the manipulation of timing constraints," *IEEE Trans. on Computer Aided Design*, vol. 17, no. 10, pp. 936–947, Oct 1998.

[3] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASIC's," *IEEE Trans. on Computer Aided Design*,v ol. 8, no. 6, pp. 661–679, Jun 1989.

[4] S. M. H. de Groot, S. H. Gerez, and O. E. Herrmann, "Range-chart-guided iterative data-flow graph scheduling," *IEEE Trans. on Circuits and Systems - I*, vol. 39, no. 5, pp. 351–364, May 1992.

[5] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. on Computers*, vol. 39, no. 7, pp. 945–951, Jul 1990.

[6] H. V.Jag adish and T.Kailath, "Obtaining schedules for digital systems," *IEEE Trans. on Signal Processing*, vol. 39, no. 10, pp. 2296–2316, Oct 1991.

[7] D. W. Trainor, R. F. Woods, and J. V. McCanny, "Architectural synthesis of a digital signal processing algorithm using iris," *Journal of VLSI Signal Processing*, vol. 16, no. 1, pp. 41–56, 1997.

[8] N. Chandrachoodan, S. S. Bhattacharyya, and K. J. R. Liu, "The hierarchical timing pair model for synchronous dataflow graphs," Tech. Rep. UMIACS-TR-2000-75, Nov 2000, *http://dspserv.eng.umd.edu/pub/dspcad/papers/*.

[9] R. Reiter, "Scheduling parallel computations," *Journal of the ACM*, vol. 15, no. 4, pp. 590–599, Oct 1968.

[10] R. Schoenen, V. Zivojnovic, and H. Meyr, "An upper bound of the throughput of multirate multiprocessor schedules," in *ICASSP 97*.IEE E, 1997.

[11] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: A framework for simulating and prototyping heterogeneous systems," *Int. Jour. Computer Simulation*, vol. 4, pp. 155–182, Apr 1994.

[12] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall Signal Processing Series, 1993.