The Hierarchical Timing Pair Model for Multirate DSP Applications

Nitin Chandrachoodan, Shuvra S. Bhattacharyya, and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—The problem of representing timing information associated with functions in a dataflow graph is considered. This information is used for constraint analysis during behavioral 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 for sequential and multirate systems. Some of these shortcomings are identified, and an alternate timing model that does not have these problems for hardware implementations is provided.**

We introduce the concept of *timing pairs* to model delay elements in sequential and multirate circuits and show how this allows us to derive hierarchical timing information for complex circuits. The resulting compact representation of the timing information can be used to streamline system performance analysis. In addition, several analytical results that previously applied only to single rate systems can now be extended to multirate systems.

We present an algorithm to compute the timing parameters and have used this to compute timing parameters for a number of benchmark circuits. The results obtained on several ISCAS benchmark circuits as well as several multirate dataflow graphs corresponding to useful signal processing applications are presented. These results show that the new representation model can result in large reductions in the amount of information required to represent timing for hierarchical systems.

***Index Terms*—Hierarchical dataflow graphs, high-level design, iteration period, multirate DSP, timing analysis.**

I. Introduction

BEHAVIORAL synthesis refers to the task of constructing an architecture and binding and scheduling an algorithm that has been described in terms of the behavior of its constituent elements at a high level of abstraction. It is part of the broader field of high-level synthesis (HLS) and is often used to implement digital signal processing (DSP) applications.

DSP applications can be classified as *single rate* and *multirate*, based on the sample rates of data flowing between the different functional elements. For the purpose of analysis and modeling, they are often represented using dataflow computation models such as synchronous dataflow (SDF) [1]. This model uses the concepts of *consumption* and *production* parameters, which allow convenient representation of multiple sample rates. A special case of SDF is the case of *homogeneous* graphs, where production and consumption parameters on all edges are equal to 1. This model has been widely used to study DSP graphs, and several techniques have been developed for mapping graphs represented in this model to both hardware and software architectures [1]–[4]. Most analytical results that are known for graph performance metrics have been derived for homogeneous graphs.

For the purpose of system synthesis, we need to first ascertain the functionality of the system, followed by timing and performance analysis. For this timing analysis, it is necessary to use a timing model to indicate how long the different computations will take. The conventional model for describing timing in dataflow systems is derived from the method used in combinational logic analysis. Here, each vertex is assigned a "propagation delay" value that is treated as the execution time of the associated subsystem. That is, once all the inputs are provided to the system, this propagation delay is the minimum amount of time required to guarantee stable outputs.

An important requirement of a timing description is the ability to represent systems hierarchically. For example, Fig. 1 shows the circuit of a full adder. By using the block model on the right, we obtain a huge reduction in the size of the graph representing a circuit containing full adders. In large systems, the savings offered by using hierarchical representations are essential to retaining tractability.

A major disadvantage of the conventional timing model is that it does not allow a hierarchical description of iterative systems (containing delay elements) and multirate systems. Most approaches to analyzing multirate SDF systems require transforming them to the equivalent expanded homogeneous graph, which is a potentially huge size increase.

Other models also exist that deal with the problem of synthesis for multirate systems. Synchronous reactive systems [5] and cyclostatic dataflow (CSDF) [6] provide more of a front-end view of the problem: They deal with logical modeling of the timing and are more concerned with verifying the functionality. The RT-level model proposed in [3] provides an interesting approach to the problem of multirate timing and interfacing but requires a master clock that is potentially much faster than any of the data rates in the system in order to synchronize the interfaces. Models such as the processor timing data used in [7] capture the effects of real system parameters and latency

Manuscript received December 10, 2001; revised May 22, 2003. This work was supported in part by the U.S. National Science Foundation under Grant 9734275 and NSF NYI Award MIP9457397. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chaitali Chakrabarti.

N. Chandrachoodan was with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA. He is now the Department of Electrical Engineering, Indian Institute of Technology, Madras, India.

S. S. Bhattacharyya is with the Department of Electrical and Computer Engineering and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 USA (e-mail: ssb@umd.edu).

K. J. R. Liu is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: kjrliu@umd.edu).

Digital Object Identifier 10.1109/TSP.2004.826178

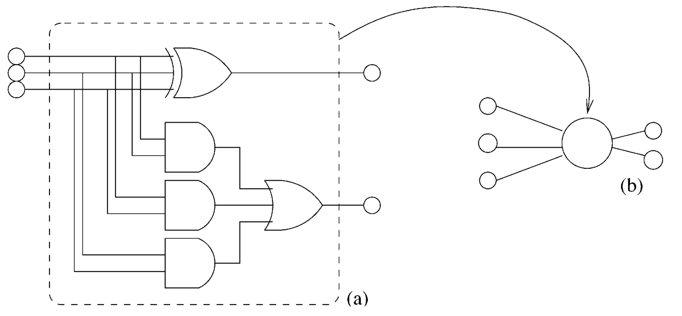


Fig. 1. (a) Full adder circuit. (b) Hierarchical block view.

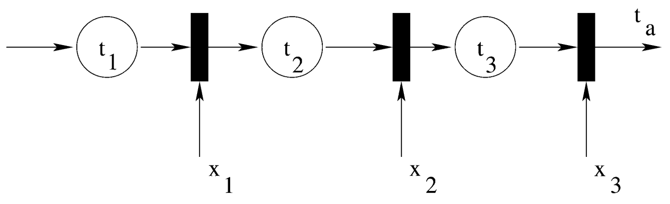
for single rate systems, but they do not provide ways to take advantage of skewed clock phases or multirate graphs directly. The discrete-time domain in Ptolemy II [8] introduces the concept of token time-lines, which helps to achieve greater understandability and analyzability of SDF graphs but does not provide the complete hierarchical timing analysis for which we aim.

In this paper, we propose a different timing model that overcomes these difficulties for dedicated hardware implementations of the dataflow graph. By introducing a slightly more complex data structure that allows for multiple input–output paths with differing numbers of delay elements, we are able to provide a single timing model that can describe both purely combinational and iterative systems. For purely combinational systems, the model reduces to the existing combinational logic timing model. For multirate systems, the new model allows a treatment that is very similar to that for normal homogeneous systems, while still allowing most important features of the multirate execution to be represented. The model also allows analytical results for homogeneous 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 hierarchical timing model to compute timing parameters of the ISCAS benchmarks, which are homogeneous systems. We have also used the 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. These results show a large savings in complexity by using the new approach.

In the next section, we discuss the requirements that a timing model for dataflow systems must meet and examine some of the shortcomings of the conventional model. Section III then presents a new model that overcomes these defects and shows how to compute the timing parameters according to our model. Section IV describes the requirements of timing models for multirate systems and shows how our model can be extended to these systems. Section V presents results of applying the model to several examples from signal processing and the ISCAS benchmark circuits. Finally, we present our conclusions and some interesting directions for further work.

Preliminary versions of the results in this paper were published in [9] and [10] for the timing pair model and its extension to multirate systems, respectively.



Single phase clock: x1 = x2 = x3 = 0: ta >= 3 * max (t1, t2, t3)

Multi-phase clock: x1 = t1, x2 = t2, x3 = t3, ta >= (t1 + t2 + t3)

Fig. 2. Ripple effects with clock skew (multiple phase clocks).

II. Requirements of a Timing Model for Hierarchical Systems

We focus on single-input single-output (SISO) systems. For general multiple-input multiple-output (MIMO) systems, each input/output pair can have different path lengths, resulting in different values for the longest combinational path between them. We assume, as is common in models of timing, that buses and other wide datapaths can be treated as single edges collapsed into relatively few timing parameters. However, this assumption is not fundamentally necessary, as we could equally well consider a system where the timing pairs are computed for each input–output pair. More accurate "bit-level" timing models can, if necessary, be implemented with the same basic constructs we propose.

One major difference between the model used in dataflow scheduling and in circuit level timing regards the treatment of delays on edges. In sequential circuits, the most common policy is to treat all delays as *flip-flops* that are triggered on a common clock edge. In high-level scheduling, we assume no such restriction on the timing of delays [11]–[14]. Each functional unit can be started at any time and signals its completion using some means. Because of this, as shown in Fig. 2, a signal applied to a dataflow graph can ripple through the graph much faster if appropriate *phase shifts* are used to trigger the flip-flops on the edges. This is because, in general, the propagation times through different elements can differ quite a bit from one another, but a single-phase clock has to take into account the worst-case value. As mentioned before, this assumption is common in high-level synthesis.

A. Timing Equivalence

We now try to clarify what is implied when we say that two descriptions of a system are equivalent for timing. Note that we are not trying to define the equivalence of circuits in the general case, as this is a considerably more complex problem. Our model also does not provide a general way to compute the timing behavior of a circuit: It assumes that there is some known way of computing the timing information for basic blocks and provides a way to combine these to get the timing information for hierarchical blocks (that may contain sequential elements).

The timing information associated with a block is used primarily for the purpose of establishing constraints on the earliest time that the successors of the block can start operating (i.e., when its outputs are ready and stable), that is, the edges of the dataflow graph imply the existence of constraints on the earliest

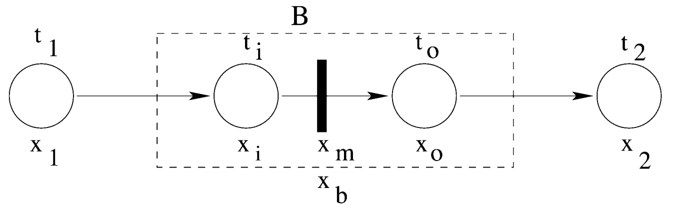


Fig. 3. Timing of complex blocks.

time that a given vertex can obtain all its inputs and start executing its function.

Using these constraints, additional metrics can be obtained relating to the throughput and latency of the system. These constraints are used to determine the feasibility of different schedules of the system, where a schedule consists of an ordering of the vertices on processing resources. An important metric of this kind is the *iteration period bound* [15], which is the minimum time within which the graph can complete a full cycle of execution. For homogeneous SDF graphs, this bound is known to be equal to the *maximum cycle mean* (MCM) of the graph (the maximum over all cycles of the sum of execution times of the vertices divided by the total number of delays on the cycle).

III. Hierarchical Timing Pair Model

Having identified the requirements of a timing model and the shortcomings of the existing model, we can now use Fig. 3 to illustrate the ideas behind the new model for timing. In this figure, we use t_i to refer to the propagation delay of block i and x_i to refer to the *start time* of the block. We use T to denote the iteration interval (clock period for the delay elements).

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. To clarify this idea, consider the block in Fig. 3. If we were to write the constraints in terms of the internal blocks x_i and x_o, we would obtain

$$x_i - x_1 \geq t_1, \quad (1)$$
$$x_o - x_i \geq t_i - 1 \times T \quad (2)$$
$$x_2 - x_o \geq t_o. \quad (3)$$

Note that the second constraint equation in the list above has the term $(-1 \times T)$ because of the delay element on the edge. Because of this delay, the actor at the output of the edge actually has a dependency on the sample produced in the previous iteration period rather than the current one. This fact is captured by the constraint as shown.

We can combine and rewrite these constraints as follows:

$$x_b - x_1 \geq t_1, \quad (4)$$
$$x_2 - x_b \geq t_i + t_o - 1 \times T. \quad (5)$$

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

In general, consider a path from input $v_i = v_1$ to output $v_o = v_k$ through vertices $\{v_1, \ldots, v_k\}$ given by $p : v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ with edges $e_i : v_i \rightarrow v_{i+1}$. Let t_i be the execution time (propagation delay assuming it is a simple combinational block) of v_i, and let d_j be the number of delays on edge e_j. Now, we can define the *constraint time* of this path as

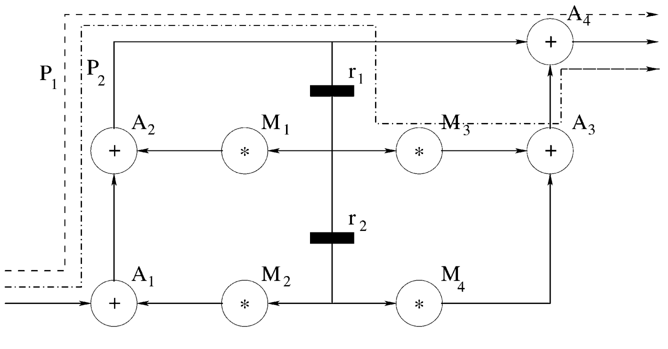


Fig. 4. Second-order filter section.

$$t_c(p) = \sum_{i=1}^{k} t_i - T \times \sum_{j=1}^{k-1} d_j. \quad (6)$$

We use the term "constraint time" to refer to this quantity because it is in some sense very similar to the notion of the execution time of the entire path but at the same time is relevant only within the context of the constraint system it is used to build. The term c_p is used to refer to the sum $\sum_{i=1}^{k} t_i$, and m_p refers to the sum $\sum_{j=1}^{k-1} d_j$. The ordered pair (m_p, c_p) is referred to as a *timing pair*.

We therefore see that by using the pair (m_p, c_p) (in the example of Fig. 3, $c_p = t_i + t_o$ and $m_p = 1$), we can derive the constraints for the system without needing to know the internal construction of B. The constraint time associated with the complex block B is now given by

$$t_c(B) = c_p - m_p \times T. \quad (7)$$

The intuition behind constraint time is that if we have a SISO system with an input data stream $x(n)$ and an output data stream $y(n) = 0.5 \times x(n-1)$, the constraint time through the system is the time difference between the arrival of $x(0)$ on the input edge and the appearance of $y(0)$ on the corresponding output edge. This is very similar to the definition of *pairwise latencies* in [16]. It is obvious that $y(0)$ can appear on its edge before $x(0)$ since $y(0)$ depends only on $x(-1)$, which (if we assume that the periodicity of the data extends backward as well as forward) would have appeared exactly T time-units before $x(0)$. Therefore, the constraint time through this system is $(t_m - T)$, where t_m is the propagation delay of the unit doing the multiplication by 0.5, and T is the iteration period of the data on the system.

We now need to extend the timing pair model to handle multiple input-output (I–O) paths, as seen in Fig. 4, which shows a second-order filter section [12]. Here, P_1 and P_2 are distinct I-O paths. Let the execution time for all multipliers be two time units and, for adders, one time unit, except for A_3, which has an execution time of two time units. In this case, for an iteration period $T \in [3, 4]$, P_2 is the dominant path, whereas for $T > 4$, P_1 is the dominant path. Therefore, we now need to store both these (m_p, c_p) values. We therefore end up with a *list*

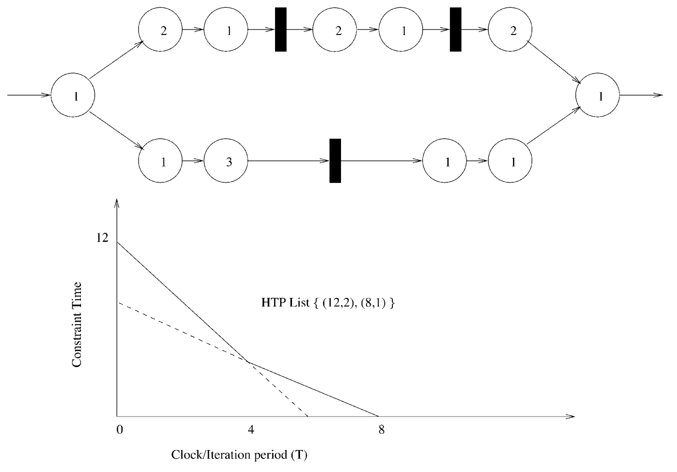


Fig. 5. Constraint time as a function of the system clock is governed by the timing pair list.

of timing pairs. The actual constraint time of the overall system can then be readily computed by traversing this list to find the maximum path constraint time. The size of the list is bounded above by the number of delays in the system ($|D|$).

Fig. 5 shows an example of how the constraint time of a system with multiple paths between input and output varies as a function of the system clock. The final constraint time, which is the only relevant figure as far as constraint analysis of the block is concerned, is given by the piecewise linear function indicated by the solid line in the figure.

Note that in addition to the timing pairs, we also need to specify a minimum clock period for which the system is valid, that is, just specifying the timing pairs could result in the erroneous impression that the system can execute at any clock period. In reality, the minimum period for the system depends on the internal minimum iteration bound of the hierarchical subsystem, or it could be set even higher by the designer to take into account safety margins or other constraints that do not derive directly from the dataflow representation.

We now have a model where the *timing pairs* that we defined above can be used to compute a *constraint time* on a system, which can be used in place of the execution time of the system in any calculations. This model is now capable of handling both combinational and iterative systems and can capture the hierarchical nature of these systems easily. We therefore refer to it as the *hierarchical timing pair (HTP) model*. The constraint time that is derived from the timing pairs can now be used in place of the execution time of the block. Note that the actual value of the constraint time will still need to be obtained by other means, such as profiling the individual elements of the block.

This definition of constraint time also results in a simple method for determining the iteration period or maximum cycle mean of the graph. It is obvious that the constraint time around a cycle must be negative to avoid unsatisfiable dependencies. In addition, note that for a fixed value of T, the constraint time of each subsystem becomes a fixed number rather than a list of timing pairs. Because of this, any algorithm that iterates over different values of T in order to determine the best value that is feasible for the graph will only have to deal with the final constraint time values and not the timing pair lists. Lawler's method [17] provides an efficient way of doing this. It performs a sequence of successive approximations to find a close approximation to the iteration period T. Since we have shown [18] the efficiency of Lawler's algorithm on graphs of bounded degree, this algorithm provides an effective way of computing the iteration period for graphs described using the hierarchical model. It may be possible to find other algorithms that can operate directly on the timing pair lists and compute a closed-form analytical expression for the maximum cycle mean of the system. However, since Lawler's method is already known to be efficient in practice, this is not a very urgent requirement.

TABLE I
TESTS FOR DOMINANCE OF A PATH

	Condition	Dominant path
1.	$m_{p_1} = m_{p_2},\, c_{p_1} < c_{p_2}$	P_2
2.	$m_{p_1} > m_{p_2},\, c_{p_1} > c_{p_2}$	$T_0 \leq T < \frac{c_{p_1}-c_{p_2}}{m_{p_1}-m_{p_2}} \Rightarrow P_1$
		$T \geq \frac{c_{p_1}-c_{p_2}}{m_{p_1}-m_{p_2}} \Rightarrow P_2$
3.	$m_{p_1} > m_{p_2},\, c_{p_1} < c_{p_2}$	P_2

A. Computation of HTP Parameters

As seen in the previous section, it is possible to have multiple I–O paths in the system, each with different numbers of delay elements. When two paths differ in the number of delay elements, the actual constraint time due to them depends on the iteration period T. For a given value of the delay count, however, only one path can dominate (the one with the longest execution time). The overall timing of the system is therefore best represented by a list of timing pairs, which have the property that for certain values of T, one or other of the pairs will dominate the timing of the system. Since each I–O path corresponds to one timing pair, we can compute a reduced set of these pairs by checking which of them are "dominant." The criteria for dominance are shown in Table I and correspond to the different possibilities of the relative values of T.

For the example of Fig. 4, P_1 has the timing pair (0, 3), whereas P_2 has (1, 7) with timing as assumed in Section III. Thus, from condition 2 above, P_2 will dominate for $3 \leq T < 4$, and P_1 will dominate for $T \geq 4$.

By combining these conditions with the Bellman–Ford algorithm for shortest paths through a circuit [19], it is possible to derive a simple algorithm that can compute the timing pairs between any two I–O points in the circuit. As we have already shown, as long as we restrict our attention to T in the valid range (namely $T > T_{\min}$), we will not encounter positive weight cycles in the graph. Recall that a positive constraint time around a cycle corresponds to an unsatisfiable constraint set, which in turn would correspond to a choice of T that is outside the feasible range for the system.

Using the above algorithm, the timing pairs for a single rate graph are easily computed. The complexity of the overall algorithm is $O(|D||V||E|)$, where $|D|$ is the number of delay elements in the graph (a bound on the length of a timing pair list of a vertex), $|V|$ is the number of vertices, and $|E|$ is the number of edges in the graph. Note that $|D|$ is quite a pessimistic estimate

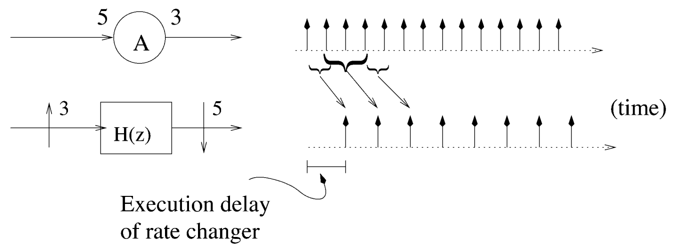


Fig. 6. Sample rate changer (3 : 5).

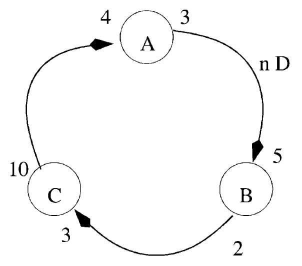


Fig. 7. Deadlock in multirate SDF system: If $n < 10$, the graph deadlocks.

since it is very rare for all the delays in a circuit to be on any single dominant path from input to output.

IV. 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 assumptions that make it easier to handle these systems mathematically.

Conventionally, multirate SDF semantics imply that tokens are produced and consumed in "bursts," with the number of tokens being given by the production and consumption parameters on the edges. This is useful for analyzing certain properties of the system but is unsatisfactory from the viewpoint of a hardware implementation. In fact, in most DSP applications, it is reasonable to consider the samples as occurring at a fixed rate in a periodic manner. Exceptions include block operations such as the discrete cosine transform (DCT) or block coders. Even in these cases, by treating a whole block of data as a token, we find that it is possible to treat the system as having periodic token flow.

A more important problem is with regard to the criterion used for firing vertices. Consider the example of the 3 : 5 rate changer shown in Fig. 6. According to the SDF interpretation, this vertex can only fire after five tokens are queued on its input and will then instantaneously produce three tokens on its output. However, a real rate changer need not actually wait for five tokens before producing its first output. In fact, in cases where such rate changers form part of a cycle in the graph, the conventional interpretation can lead to deadlocked graphs due to insufficient initial tokens on some edge or even due to the distribution of tokens among edges. The cyclostatic dataflow (CSDF) [6] model mentioned in the introduction provides a way around this by introducing the concept of execution phases.

Fig. 6 illustrates the above ideas. This is an implementation of a 3 : 5 fractional rate conversion that is implemented using an efficient multirate filtering technique (as used in the finite impulse response (FIR) filter implementation provided with Ptolemy [20]). We have assumed a seven-tap filter $(H(z))$ used for the interpolation, which results in the I–O dependencies, as shown in the figure. 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 two inputs only, the second depends on inputs two to four, and the third depends on inputs four and five. 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 five inputs to be consumed before starting to produce the outputs. One point to note is that because of the time-varying nature of the underlying implementation, we need to exercise care in deriving the timing for a multirate system.

The implementation we considered avoids unnecessary computations; therefore, it is possible to save power by either turning off the filters when they are not needed (using the clock inputs) or by using an appropriate buffering and delayed multiplication that will allow the multipliers to operate at $1/5$th of the rate of the input stream, using the observation that only one of the polyphase components [21] needs to operate for each output sample. This tradeoff would depend on whether we are considering an implementation with dedicated multipliers for each coefficient or shared multipliers. Real hardware implementations of multirate systems must resort to such efficient realizations as the performance penalties can otherwise be large.

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. 7 would be deadlocked if the edge AB has less than ten delays on it. On the other hand, six delays are sufficient on edge BC, whereas 16 delays are required on edge CA in order to prevent deadlock. The CSDF interpretation tries to avoid these difficulties by prescribing different token consumption and production phases but introduces further complexity and does not provide a complete solution to the timing problem. However, under our 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 homogeneous graphs.

It is important to understand that this interpretation of multirate SDF execution is useful because dedicated hardware implementations of real multirate DSP systems rarely require the interpretation in terms of token consumption of the conventional SDF model. Typical multirate blocks in DSP applications are decimators and interpolators (rate changers), multirate filters (very similar to rate changers), serial-to-parallel converters and vice versa, block coders and decoders, etc. A notable feature of these applications is that few of the applications actually require a consumption of c tokens before starting to produce p tokens. Even for block coders, in most implementations, for interoperating with the rest of the system, the data are produced in a periodic stream at a constant sample rate rather than in large bursts. As a result, the alternative interpretation of SDF execution suggested above is acceptable in most cases, especially when targeting fixed hardware architectures.

A. HTP Model for Multirate Systems

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 $e : u \longrightarrow v$ in the graph is given by

$$p_e \times q_u = c_e \times q_v \tag{8}$$

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 of the edge. Let T denote the overall iteration period of the graph. This is the time required for each actor x to execute q_x times (q_x is the repetition count of the actor). Therefore, the sample period on edge e is given by

$$T_e = \frac{T}{q_u \cdot p_e} = \frac{T}{q_v \cdot c_e}. \tag{9}$$

Now, extending the analogy of the homogeneous case, we define the *constraint time* on a path as

$$t_c(p) = \sum_{i=1}^{k} t_i - \sum_{j=1}^{k-1} (d_j \times T_j) \tag{10}$$

where T_j is the sample period on edge j. By noting that the effect of a delay on any edge (in both the homogeneous 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. In addition, 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.

Fig. 8 gives an example that may help to illustrate the ideas here. The consumption and production parameters help to establish the relative sample rates on the different edges so that the period between samples on edge AB is half that on edge BC. Based on our model, actor B can begin execution immediately, based on the initial token on edge AB, which is shown by the dotted arrow on the timeline for edge AB. The time intervals t_A, t_B, and t_C are the delays experienced by each respective element before it starts generating outputs. We can therefore compute the overall period by inspecting the token flow timelines, and we obtain

$$t_A + t_B + t_C = T_{AB} = T_{CA} = \frac{T_{BC}}{2} = \frac{T}{2}$$

where T is the overall iteration period required to return the system to its initial state; in this case, it is the same as T_{BC}. We can obtain the value of T by using the mathematical formulation discussed below as well.

The above example illustrates the ideas behind the new interpretation of execution time and fractional delays based on the clock used on the edge and shows how a multirate system can be treated consistently in a similar manner to a single rate 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

$$d_n(e) = \frac{d_e}{q_u \cdot p_e} = \frac{d_e}{q_v \cdot c_e}. \tag{11}$$

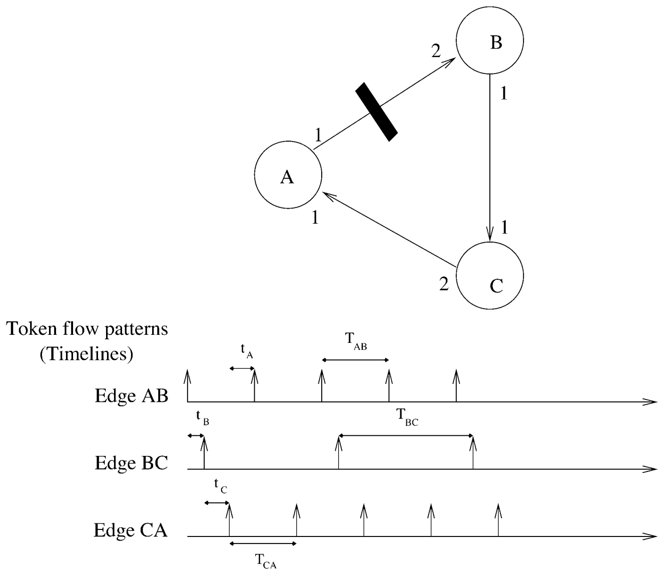


Fig. 8. Timing patterns for a multirate graph.

In terms of the normalized delays, the expression for constraint time becomes the same as that for the homogeneous case.

For homogeneous 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) [15], [22]. Thus far, no such tight bound is known for multirate SDF graphs that does not require the costly conversion to an expanded homogeneous equivalent graph. However, some good approximations for multirate graphs have been proposed [23]. Under our model, it is easy to determine an exact bound that is similar to the bound for homogeneous graphs but does not require the conversion to a homogeneous equivalent expanded graph. By considering the cumulative constraints around a loop for the single rate case, we can easily obtain the iteration period bound

$$T_{\min} = \max_{c \in C} \frac{\sum_c t_u}{\sum_c d_e} \tag{12}$$

where C is the set of all directed cycles in the graph.

Similarly, for the multirate case, we can obtain the result

$$T_{\min} = \max_{c \in C} \frac{\sum_c t_u}{\sum_c d_n(e)} \tag{13}$$

where $T_{\min}$ is the minimum admissible 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.

Note that to bound the number of timing pairs, we can no longer just count the number of delay elements in the circuit. Instead, the number of timing pairs is only bounded by the least common multiple of the denominators of the normalized delays on the edges.

One possible source of misunderstanding in this context is the use of fractional normalized delays in the computation. It may appear at first sight that the HTP model allows fractional delays to be used in the graph, even though such delays have no physical meaning in the context of signal processing. In this context,

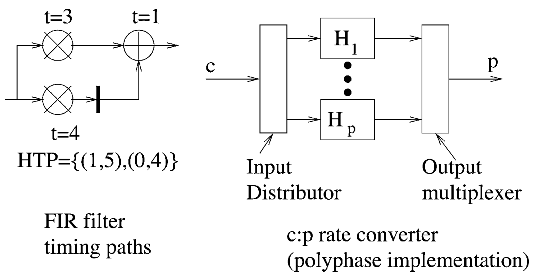


Fig. 9. Multirate FIR filter structure.

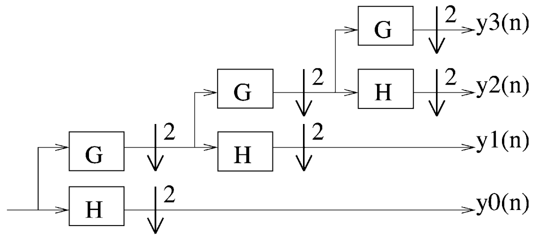


Fig. 10. Binary tree structured QMF bank.

it is important to remember that the HTP model only specifies information about the *timing* parameters of the graph. The functional correctness of the graph must be verified by other means. In particular, any fractional normalized delays only refer to the fact that the resulting timing shift is a fraction of an iteration period interval and does not indicate the use of actual fractional delays in a logical sense.

V. Examples and Results

A. 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 IV. As noted there, this must be treated as a primitive element of multirate systems. As shown in Fig. 9, the implementation uses a certain number of internal filters corresponding to the polyphase decomposition of the interpolating filter. We assume that these are implemented in a manner similar to the filter shown on the left of Fig. 9 and that the overall rate converting filter also therefore has similar timing parameters. In particular, we assume, for the sake of the other multirate examples, that any rate conversion is performed using a filter that has the timing parameters $\{(1,5),(0,4)\}$.

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 [20] (CD-DAT, DAT-CD converters and two channel nonuniform filterbank) and from [21, p. 256] [tree-structured quadrature mirror filter (QMF) bank; see Fig. 10].

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 II.

B. Single-Rate Systems

We have run the algorithm described in Section III.A on the ISCAS 89/93 benchmarks. A total of 44 benchmark graphs were considered. For this set, the average number of vertices is 3649.86, and the average number of output vertices in these circuits is 39.36.

First, we consider the case where synchronizing nodes were used to convert the circuit into an SISO system. We are interested in the number of elements that the final timing list contains since this is the amount of information that needs to be stored. Table III shows the breakup of the number of list elements. We find that the average number of list elements is 1.89.

Table IV shows some parameters obtained for the ten largest ISCAS benchmark circuits. The column "#HTP elements" refers to the number of dominating paths in the circuit. To understand these numbers, it is important to have in mind the goal of the HTP model, which is to represent a large circuit by a small number of parameters for the purpose of performance evaluation. The HTP parameters aim to provide a replacement for all the detail implied by the overall circuit.

Next, instead of assuming complete synchronization, we considered the case where inputs are synchronized and measured the number of list elements at each output. The number of distinct values obtained for this was an average of 14.73. Again, from Table IV, we see that for the circuit `s15850`, the number of distinct output elements is 36.

If we make an additional assumption that two list elements with the same m_p are the same, this number drops to 3.68 on average (two for the example `s15850`). This assumption makes sense when we consider that several outputs in a circuit pass through essentially the same path structures and delays but may have one or

TABLE II
TIMING PAIRS FOR MULTIRATE SYSTEMS

Benchmark	Timing pairs
Multirate FIR	$\{(1,5),(0,4)\}$
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 III
NUMBER OF DOMINANT TIMING PAIRS COMPUTED FOR ISCAS BENCHMARK CIRCUITS

# timing pairs	1	2	3	4	5
# circuits	21	13	5	4	1

TABLE IV
HTP PARAMETERS FOR TEN LARGEST ISCAS BENCHMARK CIRCUITS

Benchmark	#HTP list elts.	#outs w/ (m_p, c_p) diff.	#outs w/ m_p diff.	#Vertices
s38417	1	16	1	23843
s38584	1	88	1	20717
s35932	1	2	1	17828
s15850	2	36	2	10383
s13207	1	90	1	8651
s9234	1	13	1	5844
s6669	3	22	2	3402
s4863	2	11	2	2495
s3330	3	29	2	1961
s1423	1	5	2	748

two additional gates in their path that creates a slight and usually ignorable difference in the path length. For example, the circuit `s386` has six outputs. When we compute the timing pairs, we find that three have an element with one delay, and the corresponding pairs are $(1, 53)$, $(1, 53)$, and $(1, 57)$. Thus, instead of three pairs, it is reasonable to combine the outputs into one with the timing pair $(1, 57)$ corresponding to the longest path.

In order to compare these results, note that if we did not use this condensed information structure, we would need to include information about each vertex in the graph. In other words, in the best case, if we accept the (in most cases justifiable) penalty for synchronizing inputs and outputs, we need to store an average of 1.89 terms instead of 3649.86.

We have not considered the case of relaxing the assumptions on the inputs as well. This would obviously increase the amount of data to be stored, but as we have argued, our assumption of synchronized inputs and outputs has a very strong case in its favor.

We have also computed the timing parameters for HLS benchmarks such as the elliptic filter and 16-point FIR filter from [12]. These are naturally SISO systems, which makes the synchronizing assumptions unnecessary. If we allow the execution times of adders and multipliers to vary randomly, we find that the FIR filter has a number of different paths that can dominate at different times. The elliptic filter tends to have a single dominant path, but even this information is useful since it can still be used to represent the filter as a single block.

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 toward a dominant constraint time in the forward direction.

VI. Conclusion

We have presented a timing model for dataflow graphs (the hierarchical timing pair model) and associated data structures and algorithms to provide timing information for use in the analysis and scheduling of dataflow graphs.

For homogeneous graphs, the HTP model allows hierarchical representations of graphs. This results in reducing the amount of information to be processed in analyzing a graph. Alternately, by using this hierarchical representation, the size of the graph that can be analyzed with a given amount of computing power is greatly increased.

The HTP model is able to efficiently store information about multirate graphs and allows the computation of important system parameters such as the iteration period bound easily. Exact schedules for multirate systems can also be obtained as a solution to the constraints that can be set up using this model. We have shown that the HTP model overcomes many limitations of the conventional timing models while incurring a negligible complexity increase.

We have considered several typical multirate DSP applications and computed timing pairs for these models. The results demonstrate the power of our approach. We have also considered several homogeneous graphs and shown that the hierarchical aspects of 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 kinds of circuits, which include some fixed phase registers along with other nodes, where the delay can be adjusted.

References

[1] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proc. IEEE*, vol. 75, pp. 1235–1245, Sept. 1987.

[2] V. Zivojnovic, S. Ritz, and H. Meyr, "Optimizing DSP programs using the multirate retiming transformation," in *Proc. EUSIPCO Signal Process. VII, Theories Applicat.*, 1994.

[3] J. Horstmannshoff, T. Grötker, and H. Meyr, "Mapping multirate dataflow to complex RT level hardware models," in *Proc. ASAP*, 1997.

[4] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, *Software Synthesis from Dataflow Graphs*. Boston, MA: Kluwer, 1996.

[5] A. Benveniste and G. Berry, "The synchronous approach to reactive and real-time systems," *Proc. IEEE*, vol. 79, pp. 1270–1282, Sept. 1991.

[6] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cyclo-static dataflow," *IEEE Trans. Signal Processing*, vol. 44, pp. 397–408, Feb. 1996.

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

[8] C. Fong, "Discrete-time dataflow models for visual simulation in Ptolemy II," Master's thesis, Univ. California, Berkeley, CA, Dec. 2000.

[9] N. Chandrachoodan, S. S. Bhattacharyya, and K. J. R. Liu, "The hierarchical timing pair model," in *Proc. Int. Symp. Circuits Syst.*, vol. V, Sydney, Australia, May 2001, pp. 367–370.

[10] ——, "An efficient timing model for hardware implementation of multirate dataflow graphs," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Salt Lake City, UT, May 2001.

[11] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASIC's," *IEEE Trans. Computer Aided Design*, vol. 8, pp. 661–679, June 1989.

[12] S. M. H. de Groot, S. H. Gerez, and O. E. Herrmann, "Range-chart-guided iterative data-flow graph scheduling," *IEEE Trans. Circuits Syst. I*, vol. 39, pp. 351–364, May 1992.

[13] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Comput.*, vol. 39, pp. 945–951, July 1990.

[14] H. V. Jagadish and T. Kailath, "Obtaining schedules for digital systems," *IEEE Trans. Signal Processing*, vol. 39, pp. 2296–2316, Oct. 1991.

[15] R. Reiter, "Scheduling parallel computations," *J. ACM*, vol. 15, pp. 590–599, Oct. 1968.

[16] M. Potkonjak and M. Srivastava, "Behavioral optimization using the manipulation of timing constraints," *IEEE Trans. Comput. Aided Design*, vol. 17, pp. 936–947, Oct. 1998.

[17] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rhinehart, and Winston, 1976.

[18] N. Chandrachoodan, S. S. Bhattacharyya, and K. J. R. Liu. (1999, Sept.) Negative cycle detection in dynamic graphs. Inst. Adv. Comput. Studies, Univ. Maryland, College Park, MD. [Online]. Available: http://www.ece.umd.edu/DSPCAD/papers/CONTENTS.html

[19] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990.

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

[21] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[22] K. Ito and K. K. Parhi, "Determining the minimum iteration period of an algorithm," *J. VLSI Signal Process.*, vol. 11, pp. 229–244, 1995.

[23] R. Schoenen, V. Zivojnovic, and H. Meyr, "An upper bound of the throughput of multirate multiprocessor schedules," in *Proc. IEEE ICASSP*, 1997.

Nitin Chandrachoodan was born on August 11, 1975 in Madras, India. He received the B.Tech. degree in electronics and communications engineering from the Indian Institute of Technology (IIT), Madras, in 1996 and the M. S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1998 and 2002, respectively.

He is an Assistant Professor with the Department of Electrical Engineering, IIT Madras. His interests include analysis and representation techniques for system level synthesis of DSP dataflow graphs.

Shuvra S. Bhattacharyya received the B.S. degree from the University of Wisconsin, Madison, and the Ph.D. degree from the University of California, Berkeley.

He is an Associate Professor with the Department of Electrical and Computer Engineering and the Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park. He is also an Affiliate Associate Professor with the Department of Computer Science. He is the coauthor of two books and the author or coauthor of more than 50 refereed technical articles. His research interests center around architectures and computer-aided design for embedded systems, with emphasis on hardware/software co-design for signal, image, and video processing. He has held industrial positions as a Researcher at Hitachi, San Francisco, CA, and as a Compiler Developer at Kuck & Associates, Champaign, IL.

Dr. Bhattacharyya is a recipient of the NSF Career Award.

K. J. Ray Liu (F'03) received the B.S. degree from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1983 and the Ph.D. degree from the University of California, Los Angeles, in 1990, both in electrical engineering.

He is Professor with the Electrical and Computer Engineering Department and Institute for Systems Research, University of Maryland, College Park. His research interests span broad aspects of signal processing algorithms and architectures; multimedia communications and signal processing; wireless communications and networking; information security; and bioinformatics, in which he has published over 300 refereed papers.

Dr. Liu is the recipient of numerous honors and awards including IEEE Signal Processing Society 2004 Distinguished Lecturer, the 1994 National Science Foundation Young Investigator Award, the IEEE Signal Processing Society's 1993 Senior Award (Best Paper Award), and the IEEE 50th Vehicular Technology Conference Best Paper Award (Amsterdam ,The Netherlands, 1999). He also received the George Corcoran Award in 1994 for outstanding contributions to electrical engineering education and the Outstanding Systems Engineering Faculty Award in 1996 in recognition of outstanding contributions in interdisciplinary research, both from the University of Maryland. Heis the Editor-in-Chief of the IEEE SIGNAL PROCESSING MAGAZINE and was the founding Editor-in-Chief of the *EURASIP Journal on Applied Signal Processing*. He has served as an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, a Guest Editor of special issues on Multimedia Signal Processing of the PROCEEDINGS OF THE IEEE, a Guest Editor of the special issue on Signal Processing for Wireless Communications of the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS, a Guest Editor of the special issue on Multimedia Communications over Networks of the IEEE SIGNAL PROCESSING MAGAZINE, a Guest Editor of the special issue on Multimedia over Image Processing of the IEEE TRANSACTIONS ON MULTIMEDIA, and an editor of the *Journal of VLSI Signal Processing Systems*. He is on the Board of Governors and has served as Chairman of the Multimedia Signal Processing Technical Committee of the IEEE Signal Processing Society.