Retiming and Recycling for Elastic Systems with Early Evaluation

Dmitry E. Bufistov*
Universitat Politècnica de Catalunya,
Barcelona, Spain

Jordi Cortadella†
Universitat Politècnica de Catalunya
Barcelona, Spain

Marc Galceran-Oms‡
Universitat Politècnica de Catalunya
Barcelona, Spain

Jorge Júlvez§
University of Zaragoza
Zaragoza, Spain

Mike Kishinevsky
Strategic CAD Lab, Intel Corp.
Hillsboro, OR, USA

ABSTRACT

Retiming and recycling are two transformations used to optimize the performance of latency-insensitive (a.k.a. synchronous elastic) systems. This paper presents an approach that combines these two transformations for performance optimization of elastic systems with early evaluation. The method is based on Mixed Integer Linear Programming.

On a set of random benchmarks the proposed method achieves, in average, 14.5% performance improvement over min-delay retiming configurations.

Categories and Subject Descriptors: B.5.2 [Register-transfer-level implementation]: Design Aids.
General Terms: Design, Theory, Performance.
Keywords: Elastic systems, early evaluation, optimization.

1. INTRODUCTION

Latency-insensitive (a.k.a. synchronous elastic) systems tolerate changes in communication and computation latencies [3, 5]. The term "elastic system", ES, will be used in this paper.

An ES can be viewed as a composition of combinational blocks and elastic FIFOs connected by channels. A channel is comprised of data wires and a pair of handshake control signals: (valid, stop). The basic case of an elastic FIFO, called elastic buffer, EB, has a latency of one clock cycle and a capacity to store two pieces of information (*tokens*). An EB initially storing one token of information is an elastic equivalent of a synchronous register. An empty EB which contains no tokens is called a *bubble*.

The *valid* and *stop* bits in elastic channels implement a handshake protocol between the sender and the receiver. The valid bit, going in the forward direction, is used by the sender to indicate when useful data is being sent. The stop bit, going in the backward direction, is used for stalling the sender by propagating *back-pressure* when the receiver is not ready.

Any synchronous circuit can be transformed into an equivalent ES following a simple automatic flow [5, 4].

A key aspect of ESs is that they accept a set of valid transformations [7] that preserve the circuit behavior regardless the timing characteristics of its components.

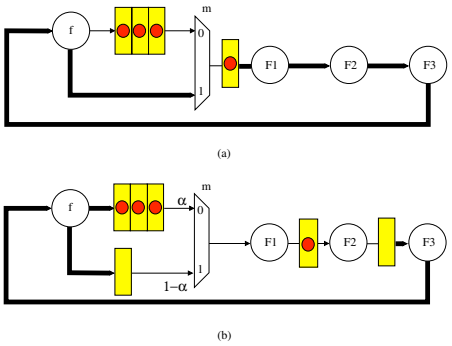

Figure 1: (a) A retiming and recycling graph, (b) Optimal retiming&recycling solution

1.1 Retiming and Recycling Graph

Retiming [8] is a well-known technique for sequential optimization. It moves registers across combinational blocks to minimize the clock cycle or area. It preserves the sequential behavior of a circuit. In ESs, retiming moves EBs instead of regular registers.

An ES is modeled by a *Retiming and Recycling Graph* (RRG). Each node of the graph is a combinational block with an associated combinational delay. Each edge represents a connection between combinational blocks labelled with EBs when needed. The RRG can be viewed as an extension of the retiming graph [8].

Figure 1(a) shows an example of an RRG. Only the datapath of the ES is drawn. Each box at the edges represents an EB. If the box is empty, the EB contains no valid information. If the box is marked with a dot, the EB contains one token. E.g., the top edge between nodes f and m has three EBs each labelled with a token. Multiplexors (such as node m) are drawn by using a different symbol than other nodes. Later we will see why.

Assuming that nodes F1, F2, F3 have unit combinational delays while other nodes have zero delays, the *cycle time* of the RRG in Figure 1(a) is equal to three time units, determined by the *critical combinational path* $F1, F2, F3, f, m$.

*Supported by FPU grant AP2005-4866, research projet CICYT TIN2007-66523
†Partially supported by a grant from Intel Corp., CICYT TIN2004-07925.
‡Supported by FI grant B1 00063
§Supported by Juan de la Cierva fellowship from the Spanish Ministry of Education and Science

DAC 2009, July 26 - 31, 2009, San Francisco, California, USA.

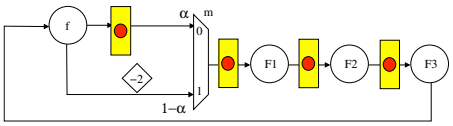

Figure 2: Optimal retiming&recycling with early evaluation

1.2 Retiming and Recycling (RR)

In ESs it is possible to insert an empty EB at any channel of the system preserving sequential behavior with respect to valid tokens of information. Empty buffer insertion is called *recycling*.

The directed cycle $F1, F2, F3, f, m$ in Figure 1(a) with a bottom edge f, m contains only one EB. Retiming preserves the total number of EBs at each directed cycle [8]. Thus, the moves of retiming cannot reduce the cycle time in this example: 3 is the minimal cycle time achievable by retiming. Despite, the RRG in Figure 1(b) obtained by applying one retiming move and inserting two bubbles has a smaller cycle time. It is equal to 1 time unit.

This reduction of cycle time is, however, useless, since the actual performance of the ES has not been improved. Indeed, two inserted bubbles reduce the *throughput* of the ES (defined as the amount of useful work done per cycle) to $\frac{1}{3}$. The multiplexor needs to wait for both valid inputs before computing a new token. This is the reason for throughput degradation.

To compare the performance of two ESs the *effective cycle time* metrics is used. The effective cycle time is the ratio of the cycle time to the throughput. Ignoring the delay overhead of inserting extra EBs the effective cycle time of both ESs shown in Figures 1 is the same. It is equal to 3 time units. Minimizing the effective cycle time of an ES by using RR is the main goal of this paper.

1.3 Early evaluation (EE)

Conventional ESs are based on *late evaluation*: the computation is initiated only when all inputs are available. Sometimes this requirement is too strict. For example, once a multiplexor received a select signal, it is sufficient to wait for the selected data channel to produce a token. The other data channel is a "don't care".

Early evaluation (EE) takes advantage of this flexibility to improve the performance of the ES. Care must be taken of the late arriving irrelevant tokens to avoid spurious enabling of functional units. The control logic for ESs with EE is presented in [4]. When EE occurs, a negative token, also called *anti-token*, is generated in the late channels that were not using for enabling the block. When an anti-token and a token meet in the same channel, they cancel each other. Anti-tokens can be *passive*, waiting for the token to arrive, or *active*, traveling backwards through the control until they meet a token. This paper considers only passive anti-tokens.

1.4 Motivational Example

Let us show how RR applied together with EE can improve performance of the RRG from Figure 1. Assume that the select channel of the multiplexor is always valid and it chooses the top data channel with probability $0 < \alpha < 1$, and the bottom channel with probability $(1 - \alpha)$.

The behavior of ESs with EE can be modeled using *Markov chains* [6]. Although this approach does not scale in general, it can be used for analysis of this small example to compute an exact expression for the throughput. Recall that with late evaluation the effective cycle time of the ES in Figure 1(b) is equal to 3. With EE, the throughput is 0.491 for $\alpha = 0.5$. Hence, the effective cycle time is $1/0.491 \approx 2.037$ time units. For $\alpha = 0.9$ the throughput is higher and is equal to 0.719 and the effective cycle time is approximately 1.39 time units.

Using RR it is possible to further improve the performance in the example. The obtained optimal solution is shown in Figure 2. Resolving the Markov chain for the ES in Figure 2, the following expression for the throughput is obtained: $1/(3-2\alpha)$. For $\alpha = 0.9$, the throughput is equal to $\frac{5}{6} \approx 0.833$ that is about 16% better than the throughput for the ES from Figure 1(b) with an EE mux.

The bottom channel coming to the multiplexor contains two anti-tokens (drawn in the rhombus). Note that the total sum of tokens is an invariant and is equal to four for the top cycle and to one $(3 - 2)$ for the bottom cycle.

The contributions. The first contribution of this paper is the demonstration, as shown in the introductory example, that allowing anti-tokens in initial configurations may help to achieve a better throughput. This is not the case for ESs without EE.

The second contribution is a precise marked graph model for performance estimation of ESs with EE.

The last contribution is a formal method for minimization of the effective cycle time of ESs with EE. The work is an extension of the paper about performance optimization of ESs with late evaluation [1].

2. BACKGROUND

This section formalizes basic concepts.

DEFINITION 2.1 (RRG). *A* Retiming and Recycling Graph *(RRG) is a tuple $\langle S, \beta, R_0, R, \gamma \rangle$, where:*

- *$S = (N, E)$ is the underlying multi-graph of the ES, N is the set of nodes and E is the set of edges. The set N is partitioned into N_1 and N_2: N_1 includes the simple combinational nodes and N_2 the EE nodes.*
- *$\beta : N \to \mathbb{R}^+$ assigns a combinational delay to each node.*
- *$R_0 : E \to \mathbb{Z}$ is the number of the tokens on each edge. If negative, R_0 is the number of anti-tokens. To ensure liveness the sum of tokens on each directed cycle of S must be positive.*
- *$R : E \to \mathbb{Z}^+$ is the number of EBs on each edge. Condition $R \geq R_0$ must hold.*
- *$\gamma : E \to \mathbb{R}^+ \backslash \{0\}$ is the branch selection probability for input edges of EE nodes $n \in N_2$. The sum of the probabilities for all inputs of an EE node $n \in N_2$ is equal to one, i.e.: $\sum_{e=(n_i,n) \in E} \gamma(e) = 1$.*

As an example, the values of R_0, R and γ of the top (bottom) edge (f, m) of the RRG in Figure 1(b) are 3, 3 and α (0, 1 and $1 - \alpha$).

Given an RRG, a *combinational path* is a sequence of adjacent edges $e_1, \ldots, e_k$ such that $R(e_i) = 0, 1 \leq i \leq k$. The delay of the combinational path is the sum of the delays of the corresponding nodes. For example, the path formed by the nodes $F1, F2, F3$ in Figure 1(a) is a combinational while the path $f, m, F1$ is not.

The *cycle time* of an RRG, τ(RRG), is the maximum delay of all combinational paths.

Let us assume that combinational delays of nodes $F1$, $F2$, $F3$ are equal to 1 time unit while the delays of the rest of the nodes are equal to 0. Then, the cycle time of the RRG in Figure 1(a) is equal to 3. The combinational path $F1, F2, F3, f, m$ is *critical*. Its delay is equal to the cycle time of the RRG.

The *throughput*, $\Theta(n)$, of node $n \in N$ of a RRG is defined as: $\Theta(n) = \lim_{t \to \infty} \frac{\sigma_n(t)}{t}$, where $\sigma_n(t)$ is the number of tokens produced by n till time stamp t. The throughput of every node is the same [6], i.e., $\Theta(n_i) = \Theta(n_j)$ for every $n_i, n_j \in N$. Thus, the throughput of any node can be denoted by Θ(RRG).

Notice that if an RRG has no bubbles (see Figure 1(a)), one token is produced by each EB each cycle, then $\Theta(\mathsf{RRG}) = 1$. The *effective cycle time* of a RRG, $\xi(\mathsf{RRG})$, is the ratio of the cycle time and the throughput.

A *retiming vector* $r \in \mathbb{Z}^{|N|}$ of a given RRG, is a map $N \to \mathbb{Z}$ that for every edge $e = (u, v)$ transforms R_0 to R'_0 as follows: $R'_0(e) = R_0(e) + r(v) - r(u)$.

In contrast to the classical definition in [8] this definition allows negative values for R_0. This is because in ESs EBs can keep anti-tokens [4].

DEFINITION 2.2 (RR CONFIGURATION). *Given an RRG, a RR configuration,* RC*, is a pair of vectors $R'_0 \in \mathbb{Z}^{|E|}, R' \in \mathbb{Z}^{+|E|}$ that satisfies the following constraints:*

$$\begin{aligned} &R'_0(e) = R_0(e) + r(v) - r(u), \\ &R'(e) \geq R'_0(e), \textit{ for each edge } e = (u, v), \end{aligned} \tag{1}$$

where r is a retiming vector.

An RRG has a lot of different RCs. For instance, the retiming vector: $r(m) = -2, r(F1) = -2, r(F2) = -1, r(f) = r(F3) = 0$, transforms the RC in Figure 1(a) to the RC in Figure 2.

Combinational path constraints. In order for an RC to meet a cycle time τ, the delay of every combinational path in the corresponding RRG must be smaller than or equal to τ. There are a set of linear constraints that guaranties this [1].

In the following, these constraints for a given RC and cycle time τ will be refereed as $\texttt{Path\_Constr}(\texttt{RC}, \tau)$.

3. THROUGHPUT OF RRG

The performance of an RRG can be estimated by using the result from [6] on performance analysis of *guarded marked graphs*.

A *Guarded Marked Graph* (GMG) is a tuple $\langle N, E, m_0, G\rangle$ where N is the set of nodes which is partitioned into subsets N_1 and N_2: N_1 includes the simple nodes and N_2 - the EE nodes; $E \subset N \times N$ is the set of edges; $m_0 : E \to Z$ assigns an initial number of tokens (possibly negative), $m_0(e)$, to each edge e; $G : N \to 2^{2^E}$ assigns a set of guards to every node, such that the following condition is satisfied. Let us denote the set of input and output edges of a node n_i as ${}^\bullet n_i = \{(n_j, n_i) | (n_j, n_i) \in E\}$ and $n_i^\bullet = \{(n_i, n_j) | (n_i, n_j) \in E\}$, respectively. Then for $n \in N_1$ the guards set $G(n)$ is one element set $\{\{{}^\bullet n\}\}$. This means that all input edges of the node n are in the same guard. For $n \in N_2$ the guards set has $|{}^\bullet n|$ elements, $G(n) = \{{}^\bullet n\}$.

The dynamic behavior of an GMG is determined by the following firing rules:

1. *Guard selection.* A guard $g(n) \in G(n)$ for the next firing of n is selected nondeterministically. The guard selection is trivial for simple nodes, since they only have one guard. For EE nodes any guard in $G(n)$ can be selected.
2. *Enabling*. If the guard $g(n)$ has been selected for the next firing of n, then the node n becomes enabled when corresponding input edge has positive marking.
3. *Firing*. An enabled node n at marking m can fire leading to another marking m' by removing one token from each input edge and adding one token to each output edge.

Timed guarded marked graphs. In order to carry out performance analysis on GMGs a timing interpretation must be added to it and each guard must be assigned the probability of being selected.

A *Timed Guarded Marked Graph* (TGMG) is a tuple $\langle N, E, m_0, G, \delta, \gamma\rangle$ where $\langle N, E, m_0, G\rangle$ is a GMG; $\delta : N \to \mathbb{R}^+$ assigns a nonnegative delay to every node; $\gamma : G \to \mathbb{R}^+ \backslash \{0\}$ assigns a strictly positive probability to each guard of $G(n)$. It must hold that: $\sum_{e \in G(n)} \gamma(e) = 1$.

For the time evolution of an TGMG it is assumed that the guard selection process has zero duration and that it respects the probabilities (γ) in any infinite execution.

The *throughput*, $\Theta(\mathcal{N})$, of an TGMG is defined as: $\Theta(\mathcal{N}) = \lim_{t \to \infty} \frac{\sigma(t)}{t}$, where t represents the time and $\sigma(t)$ is the firing count vector at time t, i.e., the j's component of $\sigma(t)$ corresponds to the number of times node n_j has fired till the time stamp t.

Notice, $\Theta(\mathcal{N})$ is defined as a vector. In [6] it is shown that all nodes of an TGMG have the same throughput. The throughput is upper bounded by the solution of the following LP problem:

$$\begin{array}{l} Maximize \quad \phi : \\ \quad \delta(n) \cdot \phi \leq \widehat{m}(e), \qquad n \in N_1, e \in {}^\bullet n \\ \quad \delta(n) \cdot \phi \leq \sum_{e \in {}^\bullet n} \gamma(e) \cdot \widehat{m}(e), \quad n \in N_2 \\ \quad \widehat{m}(e) = m_0(e) + \sigma(u) - \sigma(v), \quad e = (u, v). \end{array} \tag{2}$$

The vector σ is real in the constraints.

RRG throughput constraints. There is a simple procedure that constructs a TGMG model for an RRG. Because of the lack of the space the formal description is skipped. It can be found in [2]. For illustration the initial TGMG model for the RRG in Figure 1(b) is shown in Figure 3(a). Figure 3(b) shows the final version. Basically, a unit delay self loop for each EE node has been added and then TGMG was transformed to preserve the guard's set $G(n)$ [2].

Applying (2) to the TGMG model of an RRG it can be guaranteed with the linear constraints that given RC has throughput upper bound $1/x, x \geq 1$ [2]. Let us denote the set of this constraints as $\texttt{Thr\_Constr}(\texttt{RC}, x)$. The throughput upper bound of a given RC can be found as a minimum value of x subject to the $\texttt{Thr\_Constr}(\texttt{RC}, x)$. Let us denote this upper bound as $\Theta^{lp}(\texttt{RC})$ and the corresponding effective cycle time as $\xi^{lp}(\texttt{RC})$, i.e., $\xi^{lp} = \tau(\texttt{RC})/\Theta^{lp}(\texttt{RC})$, where $\tau(\texttt{RC})$ refers to the cycle time of the RC.

4. RETIMING AND RECYCLING

A straightforward method that combines the combinational path and throughput constraints for minimizing the effective cycle time leads to the following non-convex mixed integer quadratic programming problem:

$$\begin{array}{l} Minimize \quad x \cdot \tau, \\ \qquad R'_0(e) = R_0(e) + r(v) - r(u), \\ \qquad R' \geq R_0, R' \geq 0, \\ \qquad \texttt{Path\_Constr}(\texttt{RC}, \tau), \\ \qquad \texttt{Thr\_Constr}(\texttt{RC}, x), \\ \qquad R' \in INT, r \in INT. \end{array} \tag{3}$$

The exact solution of (3) is not necessarily the one with the minimum effective cycle time, $\texttt{RC}_{min}$, but it is a very good approximation. On the other hand, (3) represents a big challenge for existing solvers. [2] provides a heuristics based on a MILP to solve (3). This heuristics finds few *non-dominated* RCs and select one with the minimal effective cycle time, $\texttt{RC}^{lp}_{min}$.

5. EXPERIMENTAL RESULTS

A set of experiments was performed to verify the throughput model and to demonstrate optimization power of the algorithm for ESs with EE.

A set of random RRGs has been generated. The ISCAS89 circuits have been used to extract the underlying graph structures. All parameters have been set randomly, based on simple criterions. For each test case the RC with the minimal effective cycle time, $\texttt{RC}^{lp}_{min}$,

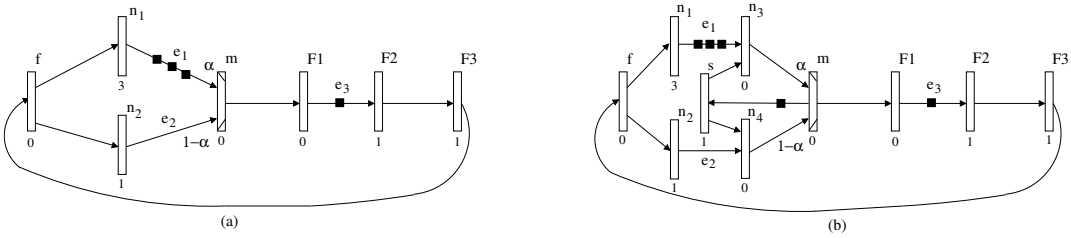

Figure 3: (a) Simple TGMG, (b) refined TGMG.

Table 1: All non-dominated RCs for the test case s526

Name	τ	Θ^{lp}	Θ	err(%)	ξ^{lp}	ξ	$\Delta(\%)$
s526	19.98	0.2500	0.2390	4.6025	79.9200	83.5983	
	24.10	0.3333	0.3050	9.2896	72.3000	79.0164	
	31.74	0.4936	0.4200	17.5219	**64.3041**	75.5714	
	56.54	0.8367	0.7910	5.7787	67.5742	**71.4791**	
	74.52	1.0000	1.0000	0.0000	74.5200	74.5200	5.4

was found. The Verilog representation of elastic controller was generated for each non-dominated RC. The actual throughput was calculated by performing intensive simulations.

Table 1 shows all non-dominated RCs for the test case s526. Rows of the table correspond to different RCs. The column τ provides the cycle time of the RC. The columns Θ^{lp} and Θ provide the throughput upper bound and the actual throughput of the RC (obtained by simulation) respectively. The column $err(\%)$ provides the relative difference between the throughput upper bound Θ^{lp} and Θ. The effective cycle times of $\mathtt{RC}^{lp}_{min}$ and $\mathtt{RC}_{min}$ are marked in bold in the columns ξ^{lp} and ξ respectively. The last column $\Delta(\%)$ is the relative difference between $\xi(\mathtt{RC}_{min})$ and $\xi(\mathtt{RC}^{lp}_{min})$, e.g., for s526 it is equal to $(75.5714-71.4791)/71.4791\cdot 100\% \approx 5.4\%$. It can be seen that the $\mathtt{RC}^{lp}_{min}$ and $\mathtt{RC}_{min}$ are different configurations in this case, however $\mathtt{RC}^{lp}_{min}$ has only 5.4% worse performance. Also the second best configuration returned by the algorithm does correspond to $\mathtt{RC}_{min}$.

Table 2 shows the obtained results. The first column is the name of the underlying ISCAS89 circuit. The next three columns are the number of simple nodes, EE nodes and edges respectively. The column ξ^* provides the cycle time before the optimization (it is equal to the effective cycle time because originally RRGs have no bubbles). The column ξ_{nee} provides the minimal effective cycle time of the RRG with all nodes being simple (late evaluation). It often coincides with the min-delay retiming cycle time (see [1] for details). In the experiments the ξ_{nee} was always provided by min-delay retiming configuration. The columns ξ^{lp}_{min} and ξ^{sim}_{min} show $\xi(\mathtt{RC}^{lp}_{min})$ and $\xi(\mathtt{RC}_{min})$, respectively. E.g., for s526 the corresponding values are equal to 75.57 and 71.48. The last column $I(\%)$ provides the performance improvement obtained by MIN_EFF_CYC using EE. It is calculated as follows: $I = ((\xi_{nee} - \xi^{sim}_{min})/\xi_{nee}) \cdot 100\%$.

CPLEX was used as an MILP solver. The timeout for integer optimization was set to 20 minutes in all experiments. For all MILPs the optimal solutions were always found.

Observation 1: The average effective cycle time improvement is equal to 14.5% (the average value of the column $I\%$). The improvement strongly depends from the position of EE nodes. The ξ_{nee} was not improved for s832, s1488, s1494. This is because some critical directed cycles (the cycles where bubbles have to be inserted) have no early evaluation nodes. The EE does not affect the performance of such ESs.

Observation 2: The $\mathtt{RC}^{lp}_{min}$ coincides with $\mathtt{RC}^{sim}_{min}$ in more than half of the examples. In s641,s386, s400,s526, s713, s953 the value of $\Delta(\%)$ is within 5%.

Table 2: Experimental results.

Name	$\lvert N_1\rvert$	$\lvert N_2\rvert$	$\lvert E\rvert$	ξ^*	ξ_{nee}	ξ^{lp}_{min}	ξ^{sim}_{min}	$I\%$
s641	206	15	270	183.15	109.62	93.72	89.98	17.9
s27	9	5	24	43.73	43.73	32.31	32.31	26.1
s444	45	13	82	174.88	106.75	92.50	92.50	13.3
s386	36	12	131	74.80	74.60	58.55	59.81	21.5
s344	122	13	176	130.63	114.19	90.79	82.89	27.4
s400	37	9	66	149.29	79.50	80.10	77.63	2.3
s526	43	7	71	144.47	74.52	75.57	71.48	4.1
s382	35	7	60	84.65	68.47	66.07	66.07	3.5
s420	7	1	9	76.70	76.70	59.78	59.78	22.1
s832	76	41	462	62.11	50.39	50.39	50.39	0.0
s1488	85	48	572	64.28	59.52	59.52	59.52	0.0
s510	63	40	407	116.63	116.63	73.26	73.26	37.2
s953	232	36	371	354.86	292.28	125.92	119.53	59.1
s713	229	27	341	119.15	96.63	99.13	95.96	0.7
s1494	88	48	572	61.97	55.80	55.80	55.80	0.0
s820	72	38	424	55.64	53.23	46.90	46.90	13.5

Observation 3: The average error $err(\%)$ of the throughput estimation is equal to 12.5%. The error achieves 35% for some configurations. Usually the error is proportional to the difference between throughputs of an RRG with and without EE nodes.

6. CONCLUSIONS AND FUTURE WORK

A MILP based algorithm for retiming and recycling of elastic systems with early evaluation has been presented. The proposed MILPs are difficult to solve exactly for circuit graphs with more than one thousand edges. However, there are simple and efficient heuristics for solving MILP problems. Exploring such heuristics is a part of the future work.

The proposed model can be extended to handle *telescopic* nodes (i.e., nodes with variable combinational delays).

7. REFERENCES

[1] D. Bufistov, J. Cortadella, M. Kishinevsky, and S. Sapatnekar. A general model for performance optimization of sequential systems. In *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 2007.

[2] D. E. Bufistov, J. Cortadella, M. Galceran-Oms, J. Júlvez, and M. Kishinevsky. Retiming and recycling for elastic systems with early evaluation. Technical Report LSI-09-11-R, 2009. http://www.lsi.upc.edu/~techreps/files/R09-11.zip.

[3] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli. Theory of latency-insensitive design. *IEEE Transactions on Computer-Aided Design*, 20(9):1059–1076, Sept. 2001.

[4] J. Cortadella and M. Kishinevsky. Synchronous elastic circuits with early evaluation and token counterflow. In *Proc. ACM/IEEE Design Automation Conference*, pages 416–419, June 2007.

[5] J. Cortadella, M. Kishinevsky, and B. Grundmann. Synthesis of synchronous elastic architectures. In *Proc. ACM/IEEE Design Automation Conference*, pages 657–662, July 2006.

[6] J. Julvez, J. Cortadella, and M. Kishinevsky. Performance analysis of concurrent systems with early evaluation. In *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 2006.

[7] T. Kam, M. Kishinevsky, J. Cortadella, and M. Galceran-Oms. Correct-by-construction microarchitectural pipelining. In *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 2008.

[8] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5–35, 1991.