Behavioral Synthesis of High Performance, Low Cost, and Low Power Application Specific Processors for Linear Computations

Miodrag Potkonjak
C&C Research Laboratories, NEC USA, Princeton, NJ

Mani B. Srivastava
AT&T Bell Laboratories, Murray Hill, NJ 07974

Abstract

Throughput has been widely traditionally recognized as the most popular performance metric for implementation of application specific computations. However, increasingly applications such as embedded controllers impose constraints on both throughput and latency as important metrics of speed. Although throughput alone can be arbitrarily improved for several classes of systems using previously published techniques, none of those approaches are effective when latency constraints are considered.

DSP, communications, and control systems are often either linear, or have subsystems that are linear. Recently, an optimal technique for simultaneous optimization of throughput and latency of linear computations was introduced in [Sri94]. However, in many cases this technique introduces significant area overhead. In this paper we apply certain key aspects of that technique (on-arrival-processing and maximally fast implementation of linear computations) with exploration of state-space based transformations to develop four synthesis techniques which generate high throughput, low latency, low area, and low power application specific processors for the special case of single input linear computations. The new transformation techniques can also be used to increase the implementation efficiency while achieving the same latency and throughput as the original design - we obtained large improvements in area and power on many benchmarks when using the proposed transformations in this alternate role.

1.0 Introduction

Throughput has been widely traditionally recognized as the most popular performance metric for implementation of application specific computations. Many algorithm transformation techniques have been developed to obtain high-throughput implementations of such computation [Kun88, Par89a, Par89b, Par89c, Fet91, Pot92]. Transformations such as lookahead, unfolding, retiming, and algebraic manipulations restructure the original computation algorithm to an equivalent algorithm that can achieve higher throughputs - in some cases even arbitrarily high throughput - although most often at the cost of increased implementation cost and latency [Par89c, Pot91].

However, increasingly application specific computation are used in systems where both throughput and latency are of importance [Par93, Sta88]. An example is DSP subsystems used in hard real-time embedded systems where the reactive nature of the system imposes hard constraints on throughput as well as latency. For instance, in an embedded system controlling an industrial process, not only does it have to keep up with the rate of arrival of sensory data, but also has to generate the corresponding output within a certain period of time. A proper addressing of these hard real-time constraints is widely recognized as one of the most important research issues [Par93, Sta88].

Although throughput alone can be improved using previously published techniques, only one of them [Sri94] is effective when latency constraints are simultaneously considered. However, the technique in [Sri94], while general in its scope, often results in high area overhead. In this paper we present new techniques for the special case of single-input linear time-invariant systems with zero initial state. These techniques produce solutions with good latency-throughput characteristics, while also significantly reducing area and power requirements.

1.1 Related Work

Transformations have been used for optimization of a variety of performance parameters in a number of engineering and scientific domains, including databases, compilers, and numerical and DSP and communication related computations [Par89c, Fis91]. Two main streams of efforts can be recognized in the application of transformations for optimization of application specific processors. The first group of researchers, primarily from the VLSI DSP community is mainly targeting the application of sophisticated and hand tailored transformations procedures on important instances of application specific computations [Fet91, Par89a, Par89b]. The second group addresses the problem from the high level synthesis viewpoint, and aims for the development of generic algorithms and complete software platforms which automate the transformational synthesis process in a wider class of computations [Wal91, Pot92].

While transformations have been used for the optimization of a number of goals, including throughput, area, power, permanent and transient fault-tolerance, and testability overhead, only recently, with a growing widespread recognition of the importance of embedded computations, has joint optimization of latency and throughput been addressed.

2.0 Problem Description

The starting point for the optimization of latency and throughput is the following set of equations which describes an arbitrary linear computation. Note that any linear computation can be easily transformed to this form using, for example, the procedure for maximally fast implementation of linear computations [Pot92].

$$\begin{aligned} S[n] &= AS[n-1] + BX[n] \\ Y[n] &= CS[n-1] + DX[n] \end{aligned} \qquad \text{(EQ 1)}$$

$$n \in \{0, 1, 2, 3, \dots\} \quad X[n] \in \Re^{P \times 1} \quad S[n] \in \Re^{R \times 1} \quad Y[n] \in \Re^{Q \times 1}$$

$$A \in \Re^{R \times R} \quad B \in \Re^{R \times P} \quad C \in \Re^{Q \times R} \quad D \in \Re^{Q \times P} \quad S[-1] \in \Re^{R x 1}$$

We also assume, without loss of generality, that addition takes one control steps, while each multiplication takes m control steps for execution. In a recent paper [Sri94] we described a technique to transform a linear time invariant control data flow graph (LTI CDFG) to satisfy, if feasible, arbitrary simultaneous constraints on latency and throughput. While the technique presented in [Sri94] is completely general in that it produces guaranteed results and is applicable to any LTI CDFG, the technique, and the theory behind it, do not make use of the values of the coefficients in the initial state-space equation - the four coefficient matrices A , B , C , and D in (EQ 1) were assumed to be absolutely arbitrary. In the work presented here our goal is to take advantage of the coefficient values. In particular, coefficients that are 0 need not be considered, and coefficients that have magnitude 1 need not be multiplied. Consequently, it may indeed be possible to achieve combinations of throughput (T_S) and latency (T_L) that the technique from [Sri94] fails to achieve. Further, the knowledge about 0 and 1 coefficients may also be used to obtain more cost efficient implementations - for example, it may be possible to achieve a combination of T_S and T_L with a smaller unfolding factor than is predicted by the algorithm in [Sri94], or it may be possible to use simpler transformations.

Taking advantage of coefficients with values 0 and 1 is, unfortunately, extremely difficult - the mathematical analysis becomes intractable - unless there is some regularity and mathematical structure to the location of these coefficients in the matrices A, B, C, and D. Of course there is no such regularity or structure in the general case of a LTI CDFG. However it is well known in DSP and Linear System Theory that a special case of these LTI CDFGs, namely LTI CDFGs with single-input and zero initial state can always be transformed to certain standard (canonical) CDFG structures [Del88, Rob87]]. Since throughput and implementation cost (area, number of operations) have been the more popular metrics in traditional DSP, these standard CDFG structures have been developed and analyzed with those two metrics in mind - latency has been overlooked. Some of these standard forms either have good latency and throughput characteristics at low cost, or are good starting points to apply some of the techniques of [Sri94], such as algebraic transformation, unfolding, on-arrival processing, maximally fast computation, and skew between the arrival of the input and the state, to obtain solutions with low latency and high throughput at a low cost. On-arrival processing is a technique were input samples corresponding to different iterations after loop unfolding are processed as soon as they arrive [Sri94], unlike what is done in block processing were sample are initially buffered [Par89a, Par89b, Par89c].

In this paper we describe four techniques that are based on the approach of first converting a single-input single-output LTI CDFG with zero initial state (a single-input multiple-output CDFG can be treated as a collection of multiple single-input single-output CDFGs) into one of the standard forms, and then applying a fixed sequence of transformations to yield cost efficient solutions with good (often optimal) throughput and latency characteristics. Since a large fraction of applications (e.g. many filters and controllers) are single-input, these techniques can indeed be useful in a variety of designs.

3.0 Transformation Techniques for Simultaneous Improvements in Latency, Throughput, Area and Power

Technique #1: Modified Direct Form II

We begin by describing a technique that is based on the well known standard form: *Direct Form II*. Any single-input single-output LTI CDFG with zero initial state can be transformed to the Direct Form II (shown in Figure 1(a)) using the following steps:

Step 1: Transform the given single-input single-output zero initial state LTI CDFG into state space equations, as in (EQ 1), characterized by P=1, Q=1, R, A, B, C, and D. P is the number of inputs, Q is the number of outputs, and R is the of states in the linear computation.

Step 2: Taking the Z-Transform of the two state space equations, calculate the *Transfer Function* H(z) = Y(z)/X(z). The zero initial state condition is required for this step.

$$H(z) = \frac{Y(z)}{X(z)} = C(zI-A)^{-1}B + D = \frac{Cadj(zI-A)B}{det(zI-A)} + D \qquad \text{(EQ 2)}$$

It is clear that H(z) will be a ratio of two polynomials in z, and can be expressed as:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N} \alpha_i z^i}{\sum_{i=0}^{N} \beta_i z^i} \qquad \beta_N \neq 0, N \leq R \qquad \text{(EQ 3)}$$

Step 3: Taking Inverse Z-Transform, the time-domain equations corresponding to (EQ 3) are:

$$Y[n] = \sum_{i=0}^{N}\left(\frac{\alpha_{N-i}}{\beta_N}\right)X[n-i] + \sum_{i=1}^{N}\left(-\frac{\beta_{N-i}}{\beta_N}\right)Y[n-i] \qquad \text{(EQ 4)}$$

Comparing (EQ 8) to the CDFG for the Direct Form II structure in Figure 1:(a) it follows that the coefficients in the Direct Form II structure can be expressed in terms of the coefficients of the numerator and denominator polynomials of H(z) as below:

$$a_i = -\frac{\beta_{N-i}}{\beta_N} \qquad b_j = \frac{\alpha_{N-j}}{\beta_N} \qquad i \in 1..N, j \in 0..N \qquad \text{(EQ 5)}$$

The Direct Form II structure is widely recognized as a low-throughput, high-latency, high-cost structure, and does not appear to be particularly useful - in the general case of arbitrary coefficients this structure has latency $T_L = 2m + N + 1$ and sample period $T_S = m + N$. However, we found that a simple and fixed sequence of transformation steps can yield a modified structure that has latency $T_L = m + 1$ and sample period $T_S = m + 2$. By way of comparison, the general technique presented in [Sri94] can achieve $T_L = m + 1$ for all $T_S \geq 2$, and

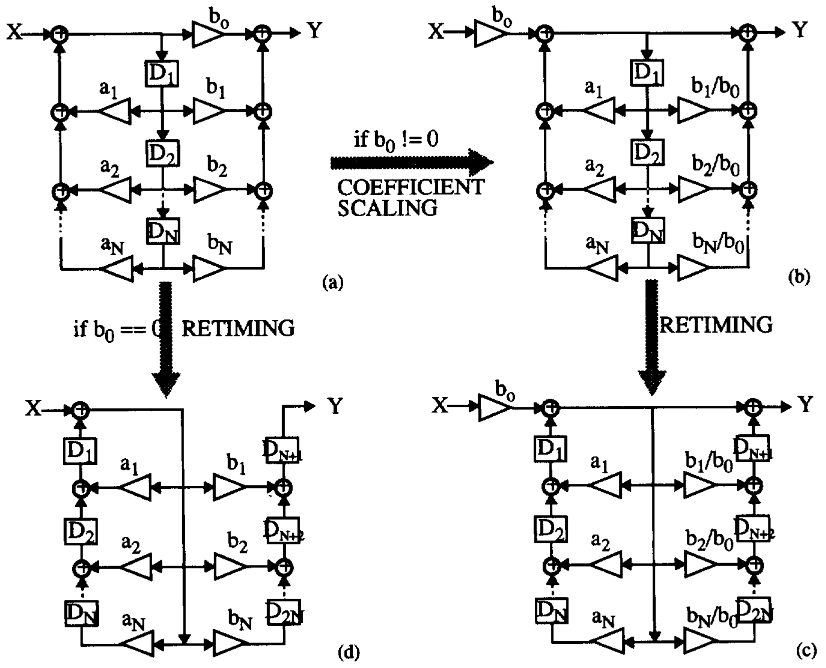


FIG. 1. Transforming *Direct Form II* to a Low Latency and High Throughput Structure

$T_L = m+2$ at $T_S = 1$ for the single-input case. The algorithm to modify the Direct Form II structure is as below.

Step 1: If the coefficient b0 in Figure 1(a) is not equal to 0, we first apply a coefficient scaling transformation to obtain the CDFG in Figure 1(b), and then apply a sequence of retiming steps that move the delay nodes in the middle branch of the CDFG to two sides as in Figure 1(c).

If the coefficient b0 in Figure 1(a) is equal to 0, we remove the corresponding multiplication node and directly apply a sequence of retiming steps to move the delay nodes in the middle branch of the CDFG to the two sides as shown in Figure 1(d).

Step 2: We convert the CDFG obtained in Step 1 into the state space representation. The state space equations for the case $b_0 \neq 0$ are:

$$S[n] = \begin{bmatrix} a_1 & 1 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ a_2 & 0 & 1 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_N & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ (b_1/b_0) & 0 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 \\ (b_2/b_0) & 0 & 0 & \dots & 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ (b_N/b_0) & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} S[n-1] + \begin{bmatrix} b_0 a_1 \\ b_0 a_2 \\ \dots \\ b_0 a_N \\ b_1 \\ b_2 \\ \dots \\ b_N \end{bmatrix} X[n] \qquad \text{(EQ 6)}$$

$$Y[n] = \begin{bmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix} S[n-1] + \begin{bmatrix} b_0 \end{bmatrix} X[n]$$

and, the state space equations for the case $b_0 = 0$ are:

$$S[n] = \begin{bmatrix} a_1 & 1 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ a_2 & 0 & 1 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_N & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ b_1 & 0 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 \\ b_2 & 0 & 0 & \dots & 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ b_N & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} S[n-1] + \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \\ b_1 \\ b_2 \\ \dots \\ b_N \end{bmatrix} X[n] \qquad \text{(EQ 7)}$$

$$Y[n] = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix} S[n-1] + \begin{bmatrix} 0 \end{bmatrix} X[n]$$

Step 3: Apply maximally fast linear computation transformation [Pot92] - the resulting latency and sample period will be $T_L = m+1$ and $T_S = m+2$ for the case $b_0 \neq 0$, and $T_L = 0$ and $T_S = m+2$ for the case $b_0 = 0$.

Note that most multiplications in the structures in Figure 1(c) (for $b_0 \neq 0$) and in Figure 1(d) (for $b_0 = 0$) are of different constant coefficients with the same variable. By transforming those multiplications to the bit-level, and assuming that word length is W, the multiplications with the various coefficients can all be done using only W shifts regardless of the values of the coefficients.

Technique #2: Modified Direct Form II with One Level of Unfolding + On-Arrival Processing

This is an extension to technique #1 where we unfold once the system obtained in technique #1, and then use maximally fast computation, on-arrival processing, and state arrival skew T_j=0 to get $T_L = m+1$ and $T_S = m+1$. Following are the steps corresponding to this technique:

Step 1: Unfold once the modified Direct Form II structure that was obtained in technique 1. We consider only the case $b_0 \neq 0$ to illustrate this technique, and obtain the following state space representation of the once unfolded system using (EQ 8):

$$S[n+1] = \begin{bmatrix} a_1^2+a_2 & a_1 & 1 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ a_1a_2+a_3 & a_2 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_1a_N & a_N & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ (a_1b_1+b_2)/b_0 & b_1/b_0 & 0 & \dots & 0 & 0 & 0 & 1 & \dots & 0 \\ (a_1b_2+b_3)/b_0 & b_2/b_0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ (a_1b_N)/b_0 & b_N/b_0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} S[n-1] + \begin{bmatrix} b_0\left(a_1^2+a_2\right) \\ b_0(a_1a_2+a_3) \\ \dots \\ b_0a_1a_N \\ a_1b_1+b_2 \\ a_2b_2+b_3 \\ \dots \\ a_Nb_N \end{bmatrix} X[n] + \begin{bmatrix} b_0a_1 \\ b_0a_2 \\ \dots \\ b_0a_N \\ b_1 \\ b_2 \\ \dots \\ b_N \end{bmatrix} X[n+1] \quad \text{(EQ 8)}$$

$$Y[n] = \begin{bmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix} S[n-1] + \begin{bmatrix} b_0 \end{bmatrix} X[n]$$

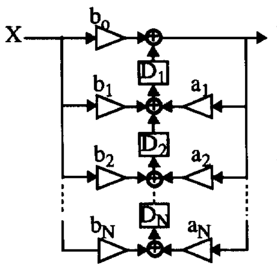
$$Y[n+1] = \begin{bmatrix} (a_1+b_1/b_0) & 1 & \dots & 0 & 0 & 1 & \dots & 0 \end{bmatrix} S[n-1] + \begin{bmatrix} b_0a_1+b_1 \end{bmatrix} X[n] + \begin{bmatrix} b_0 \end{bmatrix} X[n+1]$$

Step 2: Now use a combination of maximally fast linear expression computation and on-arrival processing. Note that $X[n]$ and $S[n-1]$ arrive at nT_S, $X[n+1]$ arrives at $(n+1)T_S$, $S[n+1]$ must be calculated by $(n+2)T_S$, $Y[n]$ must be calculated by nT_S+T_L, and $Y[n+1]$ must be calculated by $(n+1)T_S+T_L$. Using (EQ 8) it is easy to show that the smallest sample period at which this system is feasible are $T_S = m+1$, and that the corresponding latency is $T_L = m+1$ (which, according to Section 6.2, is the minimum latency for a general 1-input system). As a comparison, recall that the general technique of Section 6.3 can achieve $T_L = m+1$ for all $T_S \geq 2$, and $T_L = m+2$ at $T_S = 1$ for the single-input case.

Compared to technique #1, this technique reduces the sample period by 1, achieves the same latency, and has (8N+4) coefficients as opposed to (4N+1) for significant increase in coefficient memory.

Technique #3: Transposed Direct Form II

This technique is based on the observation that another standard form known as *Transposed Direct Form II* (also known as the *Companion Form*) has good latency throughput characteristics. This form is shown in Figure 2:, and has sample period $T_S = m+2$ and latency $T_L = m+1$ (which is equal to the minimum latency for a general single-input system). The coefficient matrices for the corresponding state space representation are also shown in the figure. The coefficients in Figure 2 for *Transposed Direct Form II* are equal to the corresponding coefficients in Figure 1:(a) for *Direct Form II*, and can therefore be calculated for any single-input single-output zero initial state LTI CDFG using (EQ 8) under technique #1. The advantage of this structure over that obtained in technique #1 is that there are only N state nodes, as opposed to



$$A = \begin{bmatrix} a_1 & 1 & 0 & \dots & 0 \\ a_2 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{N-1} & 0 & 0 & \dots & 1 \\ a_N & 0 & 0 & \dots & 0 \end{bmatrix} \quad B = \begin{bmatrix} b_1 + a_1 b_0 \\ b_2 + a_2 b_0 \\ . \\ b_{N-1} + a_{N-1} b_0 \\ b_N + a_N b_0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \quad D = \begin{bmatrix} b_0 \end{bmatrix}$$

FIG. 2. The *Transposed Direct Form II* (also called *Companion Form*) with $T_L = m+1$ and $T_S = m+2$

2N. Same throughput and latency is obtained as in technique #1 with twofold savings in number of registers used to store state variables, and only 2N+1 coefficients being required as opposed to 4N+1 resulting in significant savings in coefficient memory as well.

Technique #4: Transposed Direct Form II with One Level of Unfolding + On-Arrival Processing

Similar to the approach used in technique #2, a latency of $T_L = m+1$ and a sample period of $T_S = m+1$ are obtained if the *Transposed Direct Form II* of Figure 2 is unfolded by 1, and then implemented using on-arrival processing and maximally fast linear computation. We omit the details since they are similar to those of technique #2 although this technique has at twofold advantage in the number of states and a twofold advantage in the size of coefficient memory over technique #2 for the same latency and sample period.

4.0 Experimental Results

We implemented the four techniques presented in the previous section using a software platform which combines power of a high level synthesis system with a computer algebra system. We used Maple V, a computer algebra system originally from University of Waterloo [Cha91], with HYPER 2.1 [Rab91], a high level synthesis system from University of California, Berkeley. While HYPER is used for simulations and high-level synthesis tasks such as module selection, various transformations, scheduling, and allocation, MAPLE executes a script which performs a number of transformations (such as unfolding, minimum latency transformation, conversion to standard forms) in the described order.

Using the HYPER-Maple based software platform described in the previous section, we tested the effectiveness of the transformation technique presented in [Sri94] and the new heuristic techniques on a number of examples. The latency-throughput transformation techniques can be used for two distinct purposes: to transform a CDFG to meet joint constraints on latency and throughput, and to transform a CDFG so as to improve the cost of implementation at the same latency and throughput as that of the initial CDFG. We tested new techniques in both these modes, and the results are reported later in this section. Since we are not aware of any previous work on algorithm transformations that simultaneously addresses latency and throughput, we are unable to compare our results when using our transformations in the first mode, i.e. to meet constraints on latency and throughput. When using the transformations in the second mode - to improve the cost of implementation for unchanged latency and throughput requirements - we compare the cost of implementing the initial design and the final design using the HYPER synthesis system.

Design Name	Description	Characteristics of the Initial CDFG				
		P	Q	R	T_L [cycles]	T_S [cycles]
mat	3-state 1-input linear controller	1	1	3	4	4
ellip	4-state 1-input linear controller	1	1	4	5	5
lin4	5-state 1-input linear controller	1	1	5	6	6
lin5	5-state 1-input linear controller	1	1	5	6	6
iir5	5th order low pass elliptic wave digital IIR filter	1	1	5	7	9
iir6	6th order lowpass elliptic cascade IIR filter	1	1	6	8	7
iir8	8th order bandpass direct form IIR filter	1	1	8	11	10
iir10	10th order bandstop Butterworth cascade IIR filter	1	1	10	17	15
iir11	11th order low pass Chebyshev cascade IIR filter	1	1	11	19	19
iir12	12th order bandpass Chebyshev cascade IIR filter	1	1	12	20	18
steam	power plant controller	1	1	5	6	6

Table 1: Characteristics of the benchmark examples

The characteristics of the examples that we tested our techniques on are shown in Table 1. All the results in this section were obtained under assumption that both addition and multiplication take one control step each. The area numbers were obtained using the HYPER [Rab91] synthesis system which implements the designs as a custom chip made using a set of communicating word-parallel dedicated datapaths that are controlled by a central finite-state machine controller. HYPER generates the physical layout of the chips by using the LAGER silicon compilation system [Bro92] at the backend - datapaths are generated using the datapath compiler in LAGER, the FSM controller using the logic synthesis and tiling tools in LAGER, the datapath control logic using standard cells, and the overall chip using the macro-cell place-and-route and pad ring generator tools that are part of LAGER. We used a 1.2 micron feature size technology for the designs in this paper.

Table 2 presents the results obtained when the transformations described in this paper were used in the first mode mentioned above - to simultaneously reduce latency and sample period. The table contains the latency, sample period, and chip area for the initial CDFG as given by the user, and the corresponding numbers obtained by the four new heuristic techniques and the optimum technique from [Sri94]. The results show that both types of techniques are successful at achieving many factors of improvement in latency and throughput, although often at an increased implementation cost. However, the new heuristic techniques result in significant area savings compared to the previously published techniques. For example both the new technique H4 and the previous technique O2 produce implementation with identical latency and through-

Design Name	Initial Design		[Sri94] Technique[a]		Heuristic Techniques[b]	
	T_L, T_S [cycles]	Area [mm^2]	T_L, T_S [cycles]	Area [mm^2]	T_L, T_S [cycles]	Area [mm^2]
mat	4, 4	14.07	2, 3	15.51	2, 3	13.93
					2, 2	15.59
			2, 1	35.29	2, 3	11.26
					2, 2	10.52
ellip	5, 5	52.58	2, 3	90.02	2, 3	36.33
					2, 2	65.41
			2, 1	159.77	2, 3	21.65
					2, 2	36.83
lin4	6, 6	67.59	2, 3	183.85	2, 3	50.00
					2, 2	73.14
			2, 11	242.97	2, 3	27.74
					2, 2	44.11
lin5	6, 6	223.31	2, 3	614.47	2, 3	162.77
					2, 2	234.42
			2, 11	805.25	2, 3	91.64
					2, 2	141.10
iir5	7, 9	28.92	2, 3	138.16	2, 3	73.34
					2, 2	115.54
			2, 11	215.89	2, 3	40.25
					2, 2	63.11
iir6	8, 7	10.90	2, 3	71.83	2, 3	32.33
					2, 2	47.27
			2, 11	218.79	2, 3	22.06
					2, 2	33.23
iir8	11, 10	29.25	2, 3	253.82	2, 3	36.74
					2, 2	43.02
			2, 11	424.01	2, 3	24.47
					2, 2	28.86
iir10	17, 15	22.69	2, 3	259.80	2, 3	78.21
					2, 2	117.63
			2, 11	566.60	2, 3	49.66
					2, 2	81.09
iir11	19, 19	20.58	2, 3	166.94	2, 3	70.15
					2, 2	106.63
			2, 11	466.36	2, 3	45.57
					2, 2	71.15

Table 2: Improvements in latency and throughput of the initial design using the new heuristic techniques of, and the optimum technique from [Sri94].

Design Name	Initial Design		[Sri94] Technique[a]		Heuristic Techniques[b]	
	T_L, T_S [cycles]	Area [mm^2]	T_L, T_S [cycles]	Area [mm^2]	T_L, T_S [cycles]	Area [mm^2]
iir12	20, 18	25.73	2, 3	317.02	2, 3	104.13
					2, 2	175.51
			2, 11	886.13	2, 3	69.96
					2, 2	106.28
steam	6, 6	82.34	2, 3	184.00	2, 3	35.98
					2, 2	88.10
			2, 11	370.30	2, 3	34.00
					2, 2	54.28

Table 2: Improvements in latency and throughput of the initial design using the new heuristic techniques of, and the optimum technique from [Sri94].

a. For each example the first set of numbers corresponds to the case when the technique from [Sri94] is used to achieve a minimum latency system; the second set of number corresponds to the case when it is used to achieve a maximum throughput system.

b. The four sets of numbers for each example correspond to the four special-case heuristic techniques #1, #2, #3, and #4 respectively in the paper.

put. But, the implementation produces using the new technique are, on average, smaller by factor 3.41. The median difference is by factor 2.98 times.

Design Name	T_L, T_S [cycles]	Initial Design Area [mm^2]	Optimum Technique[a] Area [mm^2]		Heuristic Techniques[b] Area [mm^2]			
			O1	O2	H1	H2	H3	H4
mat	4, 4	14.07	9.29	11.15	9.62	9.84	5.31	6.47
ellip	5, 5	52.58	38.69	37.40	23.79	34.07	15.20	18.37
lin4	6, 6	67.59	41.14	42.39	31.55	33.24	16.03	19.53
lin5	6, 6	223.31	126.34	118.39	77.35	108.3	28.70	56.68
iir5	7, 9	28.92	28.81	44.62	27.77	34.63	18.16	34.63
iir6	8, 7	10.90	21.53	26.29	16.66	20.41	10.90	13.90
iir8	11, 10	29.25	21.53	26.29	14.18	20.22	12.95	15.34
iir10	17, 15	22.69	39.10	47.75	23.19	32.22	19.36	26.06
iir11	19, 19	20.58	43.86	44.44	20.65	28.81	15.12	22.25
iir12	20, 18	25.73	57.64	59.41	25.04	46.83	22.39	29.99
steam	6, 6	82.34	41.14	46.79	23.79	25.29	14.87	16.37

Table 3: Improvements in area over the initial design using the new heuristic techniques, and the technique [Sri94] when the transformed design is scheduled for the same latency and throughput as the original design (using $m = 1$)

a. O1 corresponds to the case when the optimum technique of Section 6.3 is used to achieve a minimum latency system; O2 corresponds to the case when it is used to achieve a maximum throughput system.
b. H1, H2, H3, and H4 correspond to techniques #1, #2, #3, and #4 respectively of Section 7.0.

Table 3 presents the data obtained when the transformation techniques were used in the second mode: to improve the cost of implementation for the same latency and throughput requirements as for the original CDFG. Again, the data is presented for the four special-case heuristic techniques, and for two extreme cases of the optimum technique. The data shows that substantial reduction in the area is obtained in many cases. For example when we compare the initial implementation and the implementations under the same initial timing constraints using technique H3, the area of all benchmark design was reduced. The average and the median reduction in area were by factors 2.93 and 2.26 respectively. When the approach H4 is used, then the average and median improvements were by factors 2.15 and 1.91 respectively.

As the data in Table 2 (where the transformed designs are scheduled under the new, and significantly stricter, latency and throughput constraints) suggests, a sharp reduction in the latency and sample period of designs is often achieved simultaneously with reduction, or a very limited increase, Table 3 in the number of operations and the size of final implementation. An intuitive generalization of the results of [Cha92] to the case where both throughput and latency are constrained suggests that the preceding set of conditions is sufficient for power reduction. Table 3 shows the reduction in power between the initial designs and designs obtained using the H4 technique. The power was reduced in all the examples. The highest reduction was for the *lin5* linear controller by factor of 16.8 times, while the smallest improvement was for the *iir6* filter, by 86%. The average and the median reduction in power were by factors 6.15 and 3.58 times. Although the average increase in area was by 88.8%, this number is biased by the sharp increase in areas for a few examples - in more than half the examples both area and power were simultaneously reduced, resulting in an overall median decrease in area by 1.3%.

Design Name	Initial Energy [nJ / sample]	H4 Energy [nJ / sample]	Improvement Factor
mat	59.2	16.8	3.52
ellip	222	45.7	4.86
lin4	288	30.2	9.54
lin5	756	45.0	16.8
iir5	118	33.0	3.58
iir6	40.2	21.6	1.86
iir8	119	14.1	8.44
iir10	96.7	28.8	3.35
iir11	81.3	27.1	3.00
iir12	89.2	28.3	3.15
steam	377	39.2	9.62

Table 4: Improvements in energy per sample over the initial design using H4, the technique #4.

5.0 Conclusion

Meeting simultaneous constraints on throughput and latency while synthesizing from a high-level description is an important unsolved problem. We presented four techniques which do not just simultaneously reduce latency and increase throughput in linear systems to a provably optimal point, but can also be used effectively to reduce area and power requirements. Improvements in all four parameters - latency, throughput, area, and power - have been demonstrated on a number of industrial and academic benchmarks.

6.0 References

[Bro92] R. W. Brodersen, editor. *Anatomy of a Silicon Compiler*. Kluwer Academic Publishers, 1992

[Cha91] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Managan, and S. M. Watt. *Maple V: The Future of Mathematics*. Springer-Verlag, 1991.

[Cha92] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R. Brodersen. "HYPER-LP: A System for Power Minimization Using Architectural Transformations." In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 300–303, 1992.

[Cha93] A. Chatterjee and R. Roy. "ARTIST: An Architectural Transformation Program for Optimization." In *Proceedings of ACM/IEEE Design Automation Conference*, pages 343–348, 1993.

[Dav88] J. H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra - Systems and Algorithms for Algebraic Computation*. Academic Press, London, UK, 1988.

[Del88] D. F. Delchamps. *State-Space and Input-Output Linear Systems*. Springer-Verlag, 1988.

[Fet91] A. Fetweis, H. Meyr, and L. Thiele. "Algorithm Transformations for Unlimited Parallelism." In *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 1756–1759, 1990.

[Fis91] C. N. Fischer and J. R. J. LeBlanc. *Crafting a Compiler*. The Benjamin/Cummings Publishing Co. Inc., 1991.

[Fri89] B. Friedland. *Control System Design: An Introduction to State-Space Methods*. McGraw-Hill, Inc., 1986.

[Kun88] S. Y. Kung. *VLSI Array Processors*. Prentice-Hall, 1988.

[Par89a] K. K. Parhi and D. Messerschmitt. "Pipeline Interleaving and Parallelism in Recursive Digital Filters - I: Pipelining using Scattered Look-ahead and Decomposition." *IEEE Transactions on Audio, Speech, and Signal Processing*, 37(7):1099–1117, 1989.

[Par89b] K. K. Parhi and D. G. Messerschmitt. "Pipeline Interleaving and Parallelism in Recursive Digital Filters - II: Pipelined Incremental Block Filtering." *IEEE Transactions on Audio, Speech, and Signal Processing*, 37(7):1118–1134, 1989.

[Par89c] K. K. Parhi. "Algorithm Transformation Technique for Concurrent Processors." *Proceedings of the IEEE*, 77(12):1879–1895, 1989.

[Par93] T. Parks. "Prototyping Real-Time Systems in Ptolemy." Handout at The 1993 ILP Conference at the University of California at Berkeley, 1993.

[Pot92] M. Potkonjak and J. Rabaey. "Maximally Fast and Arbitrarily Fast Implementation of Linear Computations." In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 304–308, November 1992.

[Rab91] J. M. Rabaey, C.-M. Chu, P. D. Hoang, and M. Potkonjak. "Fast Prototyping of Datapath-Intensive Architectures." *IEEE Design & Test of Computers*, 8(2):40–51, June 1991.

[Rob87] R. A. Roberts and C. T. Mullis. *Digital Signal Processing*, chapter 8-9. Addison-Wesley Publishing Company, 1987.

[Sim90] B. Simmon. "Four Computer Mathematical Environments." *Notices of the American Mathematical Society*, 37(7):861–868, 1990.

[Sta88] J. A. Stankovic. "Real-Time Computing Systems: The Next Generation." In J. A. Stankovic and K. Ramamritham, editors, *IEEE Tutorial on Hard Real-Time Systems*, pages 14–38. Computer Society Press of the IEEE, 1988.

[Sri94] M.B. Srivastava, M. Potkonjak: "Transforming Linear Systems for Joint Latency and Throughput Optimization", EDAC-94, paper 5B-2, 1994.

[Wol88] S. Wolfram. *Mathematica - A System for Doing Mathematics by Computer*. Addison-Wesley Publishing Company, 1988.