System-Level Parallelism and Throughput Optimization in Designing Reconfigurable Computing Applications

Esam El-Araby[1], Mohamed Taher[1], Kris Gaj[2], Tarek El-Ghazawi[1],
David Caliga[3], and Nikitas Alexandridis[1]
[1]The George Washington University, [2]George Mason University, [3]SRC Computers
{esam, mtaher}@gwu.edu, kgaj@gmu.edu,{tarek, alexan}@gwu.edu, caliga@srccomp.com

Abstract

Reconfigurable Computers (RCs) can leverage the synergism between conventional processors and FPGAs to provide low-level hardware functionality at the same level of programmability as general-purpose computers. In a large class of applications, the total I/O time is comparable or even greater than the computations time. As a result, the rate of the DMA transfer between the microprocessor memory and the on-board memory of the FPGA-based processor becomes the performance bottleneck. In this paper, we perform a theoretical and experimental study of this specific performance limitation. The mathematical formulation of the problem has been experimentally verified on the state-of-the art reconfigurable platform, SRC-6E. We demonstrate and quantify the possible solution to this problem that exploits the system-level parallelism within reconfigurable machines.

1. Introduction

Reconfigurable Computers combine the flexibility of traditional microprocessors with the power of Field Programmable Gate Arrays (FPGAs). The programming model is aimed at separating programmers from the details of the hardware description, and allowing them to focus on an implemented function. This approach allows the use of software programmers and mathematicians in the development of the code, and substantially decreases the time to the solution. The SRC-6E Reconfigurable Computer is one example of this category of hybrid computers [1].

In this paper we will discuss the existing limitations on the performance of reconfigurable computers, and propose an optimization technique that improves this performance. Our experimental results confirm the efficiency of the proposed solution.

2. SRC-6E Reconfigurable Computer

2.1. Hardware Architecture

SRC-6E platform consists of two general-purpose microprocessor boards and one MAP® reconfigurable processor board. Each microprocessor board is based on two 1 GHz Pentium 3 microprocessors. The SRC MAP board consists of two MAP reconfigurable processors. Overall, the SRC-6E system provides a 1:1 microprocessor to FPGA ratio. Microprocessor boards are connected to the MAP board through the SNAP® interconnect. SNAP card plugs into the DIMM slot on the microprocessor motherboard [1].

Hardware architecture of the SRC MAP processor is shown in Fig. 1. This processor consists of two programmable User FPGAs, six 4 MB banks of the on-board memory (OBM), and a single Control FPGA.

In the typical mode of operation, input data is first transferred through the Control FPGA from the microprocessor memory to OBM. This transfer is followed by computations performed by the User FPGA, which fetches input data from OBM and transfers results back to OBM. Finally, the results are transmitted back from OBM to microprocessor memory.

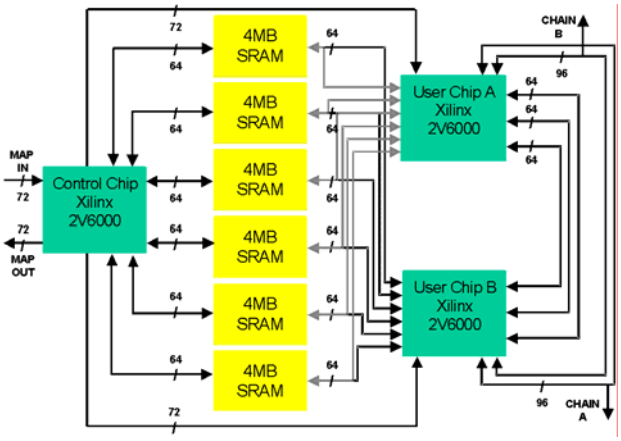


Figure 1. Hardware Architecture of SRC-6E

2.2. Programming Model

The SRC-6E has a similar compilation process as a conventional microprocessor-based computing system, but needs to support additional tasks in order to produce

logic for the MAP reconfigurable processor, as shown in Fig. 2. There are two types of application source files to be compiled. Source files of the first type are compiled targeting execution on the Intel platform. Source files of the second type are compiled targeting execution on the MAP reconfigurable processor. A file that contains a program to be executed on the Intel processor is compiled using the microprocessor compiler. All files containing functions that call hardware macros and thus execute on the MAP are compiled by the MAP compiler. MAP source files contain MAP functions composed of macro calls. Here, macro is defined as a piece of hardware logic designed to implement a certain function. Since users often wish to extend the built-in set of operators, the compiler allows users to integrate their own VHDL/Verilog macros.

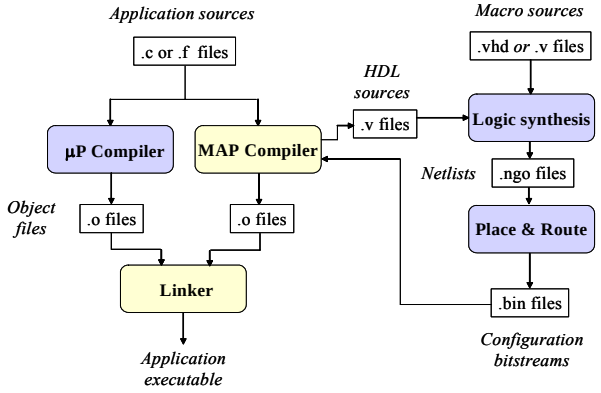


Figure 2. SRC Compilation Process

3. Current Performance Limitations

The total execution time for any application on a reconfigurable machine consists of the computations time and the total I/O time as shown in Fig. 3.

In a large class of applications, the total I/O time is comparable or even greater than the computations time. As a result, the rate of the DMA transfer between the microprocessor memory and the on-board memory becomes the performance bottleneck.

One possible solution is the redesign of the system hardware in such a way that it supports a higher data transfer rate. Taking into account the cost of the hardware system upgrade, this solution may not be practical. Additionally, even with the higher data transfer rate, there might be still applications in which the DMA time is comparable or even longer than the computations time.

Therefore, our goal has been to find a general solution to speed-up a large class of applications running on a reconfigurable computer without any changes to the system hardware. Our solution exploits the system-level parallelism within the SRC machine, and requires only small changes in the application code.

Figure 3. Execution time without overlapping

4. The Proposed Optimization Technique

4.1. Model Formulation

The objective of our optimization technique is to overlap computations with the data transfer which substantially reduces the total execution time.

This technique is constrained by both the machine and the nature of the application.

The machine constraints can be in terms of the I/O bandwidth, total number of concurrent DMA channels, the capability of overlapping the input with the output DMA channels, and the asymmetry between the input and output DMA channels bandwidths. In our model, we assume a generic hypothetical machine that has all the above mentioned constraints (see Fig. 4). In other words, we assume asymmetric I/O transfers, non-equal number of concurrent input and output DMA channels, and varying overlapping ability among the DMA channels.

On the other hand, the application forces some constraints, depending on its nature, which makes it difficult to model all the possible variations. Two essential variations in this context are the nature of data acceptance and data processing by the application. For the data acceptance, our model assumes that the application is periodic, i.e. data are fed into the application sequentially in fixed-size blocks. Periodicity, in general, accommodates for the special nature of pipelined applications as a subset of the range of applications it covers. For the nature of processing, we assume concurrent processing of multiple blocks of data as well as linear dependency between the computations time and the amount of data being processed. These assumptions are met by a large class of applications, including encryption [2, 3, 4], compression, and selected image and data processing algorithms [5, 6, 7].

The details of the presented technique are illustrated in Fig. 5. Both the DMA-IN and DMA-OUT transfers are divided into a sequence of n data transfers each. Each of these transfer parcels is further divided into a number of concurrent transfer parcels equal to the number of the DMA channels available in each direction. The computation period has been divided into a number of partial computation periods spanning the time interval between the end of the first DMA-IN transfer and the beginning of the last DMA-OUT transfer.

The first and the last data parcels are special, as no computations can be performed in parallel with these data transfers.

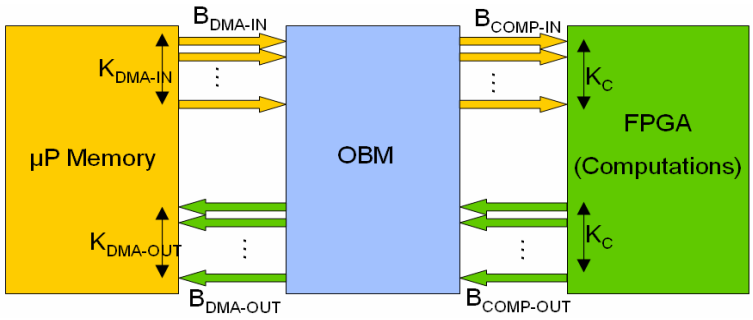
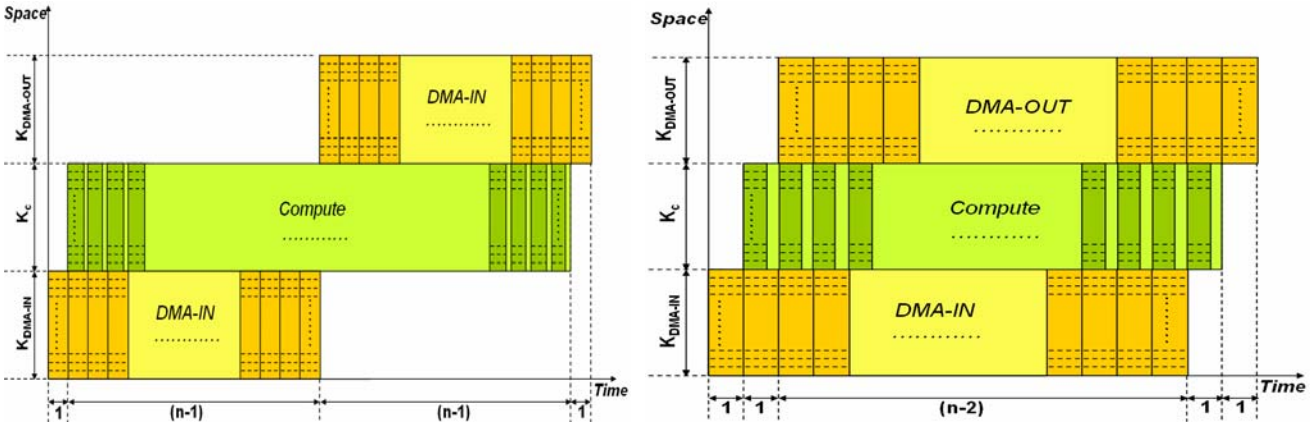


Figure 4. Model architecture



a) Non-overlapped DMA channels (*V*=0) **b) Overlapped DMA channels (*V*=1)**

Figure 5. Overlapping Computations with Data Transfers

4.2. Analysis

The following notation will be used in our mathematical model:

- $n_{DMA\text{-}IN}$ is the number of input data parcels
- n_{COMP} is the number of partial computations
- $n_{DMA\text{-}OUT}$ is the number of output data parcels
- $K_{DMA\text{-}IN}$ is the input transfer concurrency (multiplicity) factor, i.e. the number of concurrent input channels
- $K_{DMA\text{-}OUT}$ is the output transfer concurrency (multiplicity) factor, i.e. the number of concurrent output channels
- K_{DMA} is the total DMA concurrency (multiplicity) factor

$$K_{DMA} = K_{DMA\text{-}IN} + K_{DMA\text{-}OUT} \quad (1)$$

- K_C is the computations concurrency (multiplicity) factor, i.e. the number of concurrent processing units, it is also the number of independent data channels between the OBM and the computations on the FPGA in either direction (e.g. number of OBM memory banks)
- $B_{DMA\text{-}IN}$ is the bandwidth for the input data transfer from the microprocessor memory to the OBM per single DMA channel
- $B_{COMP\text{-}IN}$ is the bandwidth for the input data transfer between the OBM and a single computational unit
- $B_{COMP\text{-}OUT}$ is the bandwidth for the output data transfer from a single computational unit to the OBM
- $B_{DMA\text{-}OUT}$ is the bandwidth for the output data transfer from the OBM to the microprocessor memory per single DMA channel
- $D_{BLOCK\text{-}IN}$ is the data block size for each of the concurrent input parcels
- $D_{BLOCK\text{-}COMP}$ is the data block size for each of the concurrent computations
- $D_{BLOCK\text{-}OUT}$ is the data block size for each of the concurrent output parcels
- $D_{DMA\text{-}IN}$ is the total data size for the input transfer

$$D_{DMA-IN} = n_{DMA-IN} D_{BLOCK-IN} K_{DMA-IN} \quad (2)$$

- $D_{COMP\text{-}IN}$ is the total input data size for the computations and it is equal to the total data transferred in by the DMA

$$D_{COMP-IN} = n_{COMP} D_{BLOCK-COMP} K_C \quad (3)$$

$$D_{COMP-IN} = D_{DMA-IN} \quad (4)$$

- $D_{COMP\text{-}OUT}$ is the total output data size from the computations and it is equal to the total data to be transferred out by the DMA

$$D_{COMP-OUT} = \beta D_{COMP-IN} \quad (5)$$

$$D_{COMP-OUT} = D_{DMA-OUT} \quad (6)$$

- β is the data production-consumption factor; i.e. $\beta>1$ for data-producing applications, and $\beta<1$ for data-consuming applications
- $D_{DMA\text{-}OUT}$ is the total data size for the output transfer

$$D_{DMA-OUT} = n_{DMA-OUT} D_{BLOCK-OUT} K_{DMA-OUT} \quad (7)$$

- $T_{DMA\text{-}IN}$ is the single-channel DMA transfer time from the microprocessor memory to the on-board memory

$$T_{DMA-IN} = \frac{D_{DMA-IN}}{B_{DMA-IN}} \quad (8)$$

- $T_{DMA\text{-}OUT}$ is the single-channel DMA transfer time from the on-board memory to the microprocessor memory

$$T_{DMA-OUT} = \frac{D_{DMA-OUT}}{B_{DMA-OUT}} \quad (9)$$

- T_{DMA} is the single-channel total DMA transfer time

$$T_{DMA} = T_{DMA\text{-}IN} + T_{DMA\text{-}OUT} \quad (10)$$

- T_{COMP} is the total computations time for the case of no-overlapping
- $T_{NoOverlap}$ is the total execution time for the case of no-overlapping
- $T_{Overlap}$ is the total execution time for the case of overlapping
- V is the DMA channel-overlapping factor; i.e. V=0 for no overlapping between input and output DMA transfers, V=1 for maximum overlapping between input and output DMA transfers (see Fig. 5)

We also introduce the following notation for the ratios of respective times.

$$X_{in} = \frac{T_{DMA-IN}}{T_{DMA}}, \; X_{out} = \frac{T_{DMA-OUT}}{T_{DMA}}, \; X_c = \frac{T_{COMP}}{T_{DMA}} \quad (11)$$

Equations (1) to (9) show the sources of asymmetry in DMA transfer times. Asymmetry can be caused by difference in the number of channels, $K_{DMA\text{-}IN}$, $K_{DMA\text{-}OUT}$, and in bandwidths, $B_{DMA\text{-}IN}$, $B_{DMA\text{-}OUT}$, between the transfers, which are machine constraints. It can also be caused by transferring different data sizes in each direction depending on whether the application being either data-producing ($D_{DMA\text{-}OUT} > D_{DMA\text{-}IN}$; i.e. $\beta>1$) or data-consuming ($D_{DMA\text{-}OUT} < D_{DMA\text{-}IN}$; i.e. $\beta<1$). In addition to these factors, asymmetry can be caused by difference in the number of input parcels, $n_{DMA\text{-}IN}$, and the number of output parcels, $n_{DMA\text{-}OUT}$. To limit the asymmetry conditions to those factors which are only forced by the machine and/or application constraints, not by the proposed technique itself, we can deliberately select the number of transfer parcels in both directions to be equal; i.e. $n_{DMA\text{-}IN}=n_{DMA\text{-}OUT}= n$. In general, asymmetry can be caused by either one or all of the above three constraints. The resultant effects of all these asymmetry constraints are collectively modeled in equations (2), (7), (8), (9), (10), and (11).

As a conceptual representation [8] of the model, Fig. 5 suggests some bounds on the number of input channels, $K_{DMA\text{-}IN}$, and the number of output channels, $K_{DMA\text{-}OUT}$, when related to the number of concurrent processing units, K_C. The existence of OBM, which is very common in almost all types of RCs, serves as a buffering mechanism which relaxes any bounding limits on the relation between $K_{DMA\text{-}OUT}$ and K_C. In other words, these two factors can boundlessly be changed independently. On the other hand, there can be a lower bound for the relation between $K_{DMA\text{-}IN}$ and K_C. This lower bound is forced by the fact that the first block of transferred data should be at least the minimum amount of data necessary to start the processing; i.e. $(K_{DMA\text{-}IN} * D_{BLOCK\text{-}IN}) \geq (K_C * D_{BLOCK\text{-}COMP})$.

Equations derived to assess the effectiveness of the proposed overlapping technique are grouped together in Table 1. Based on Figs. 3 and 5, equations (12) and (13) have been derived to describe the total execution time for both cases of no-overlapping and overlapping. To evaluate the effectiveness of the technique, the speedup in the total execution time, S, is defined by equation (14).

Based on equations (12), (13), and (14), equation (15) gives a simplified formula for S for the different values of the X_c ratio. The upper limit on the speedup, and the asymptotic behavior of this limit, for both cases of non-overlapped DMA channels, V=0, and maximally overlapped DMA channels, V=1, are given in equation (16) under the conditions of symmetric DMA transfers.

In Fig. 6, the asymptotic dependence between the speedup, S, and X_c, is plotted for different values of the system parameters, $K_{DMA\text{-}IN}$, $K_{DMA\text{-}OUT}$, K_C, and V. Based on equation (15) and Fig. 6, our technique, for a given K_C, gives the best results for the case of $X_{in} = X_{out}$, $K_{DMA\text{-}IN}$ =

$K_{DMA-OUT}$, i.e., symmetric data transfer where data transfer-in and data transfer-out take the same amount of time. If this is not the case, the speedup, S, shifts downward, shaded areas in Fig. 6, from the peak values when X_c varies between X_{Cmin} and X_{Cmax}, where X_{Cmin} and X_{Cmax} are defined in Table 1, equation (15). In other words, when the DMA transfer-in time differs from the DMA transfer-out time, the maximum performance degrades from the peak value, i.e. the DMA asymmetry introduces some speedup losses.

An asymmetry between the DMA-IN throughput and DMA-OUT throughput exists in the current version of the SRC-6E system. The case of $X_{min}=X_{in}=0.4$ and $X_{max}=X_{out}=0.6$, shown in Fig. 6, corresponds to the experimentally measured difference between the DMA-IN and DMA-OUT times for the SRC system.

From Fig. 6 and equation (16), it can be seen that the change in the asymptotic maximum in speedup, $S_{\infty max}$, is in direct proportion to the change in the number of channels, K_{DMA}, while the relative change in its location shifts left to less X_c (faster computations), i.e. the computation for which this maximum is achieved, is in inverse proportion to the change in the number of channels. It can also be seen that as the level of DMA channel-overlapping, V, increases, the effect of the number of channels on the speedup increases, and the speedup loss, the shaded areas in Fig. 6, increases. In addition to this, the change in the asymptotic maximum in

Table 1. Equations describing the performance of the proposed technique

$$T_{NoOverlap} = T_{DMA-IN} + T_{COMP} + T_{DMA-OUT} = T_{DMA}(1+X_c) \tag{12}$$

$$T_{Overlap} = \frac{T_{DMA-IN}}{nK_{DMA-IN}} + [n-1-V(n-2)]\left[Max(\frac{T_{DMA-IN}}{nK_{DMA-IN}}, \frac{T_c}{K_c[2(n-1)-V(n-2)]}) + Max(\frac{T_c}{K_c[2(n-1)-V(n-2)]}, \frac{T_{DMA-OUT}}{nK_{DMA-OUT}})\right] + V(n-2)\left[Max(\frac{T_{DMA-IN}}{nK_{DMA-IN}}, \frac{T_c}{K_c[2(n-1)-V(n-2)]}, \frac{T_{DMA-OUT}}{nK_{DMA-OUT}})\right] + \frac{T_{DMA-OUT}}{nK_{DMA-OUT}} \tag{13}$$

$$S = \frac{T_{NoOverlap}}{T_{Overlap}} \tag{14}$$

$$S = \begin{cases} \dfrac{n(1+X_c)}{nX_{k\max} + [n-V(n-2)]X_{k\min}} & 0 \le X_c \le X_{c\min} \\ \dfrac{n(1+X_c)}{X_{k\min} + nX_{k\max} + \dfrac{n[n-1-V(n-2)]}{K_c[2(n-1)-V(n-2)]}X_c} & X_{c\min} \le X_c \le X_{c\max} \\ \dfrac{n(1+X_c)}{(X_{k\min} + X_{k\max}) + \dfrac{n}{K_c}X_c} & X_{c\max} \le X_c \le \infty \end{cases} \quad where \quad \begin{cases} X_{c\min} = \dfrac{K_c[2(n-1)-V(n-2)]}{n} X_{k_{\min}} = \dfrac{K_c[2(n-1)-V(n-2)]}{n} Min(\dfrac{X_{in}}{K_{DMA-IN}}, \dfrac{X_{out}}{K_{DMA-OUT}}) \\ X_{c\max} = \dfrac{K_c[2(n-1)-V(n-2)]}{n} X_{k_{\max}} = \dfrac{K_c[2(n-1)-V(n-2)]}{n} Max(\dfrac{X_{in}}{K_{DMA-IN}}, \dfrac{X_{out}}{K_{DMA-OUT}}) \end{cases} \tag{15}$$

$$S_{\infty\max} = \lim_{n\to\infty} S\Big|_{X_c = X_{c\max}} = \begin{cases} K_c + \frac{1}{2}K_{DMA} & V=0 \\ K_c + K_{DMA} & V=1 \end{cases} \quad and \quad X_{c\max} = X_{c\min} = \begin{cases} \dfrac{2K_c}{K_{DMA}} & V=0 \\ \dfrac{K_c}{K_{DMA}} & V=1 \end{cases} \tag{16}$$

$$where \quad \left\{ \begin{array}{c} K_{DMA-IN} = K_{DMA-OUT} = \frac{1}{2}K_{DMA} \\ X_{\min} = Min(X_{in}, X_{out}) = X_{\max} = Max(X_{in}, X_{out}) = \frac{1}{2} \end{array} \right\} \quad \text{Symmetric DMA Transfers}$$

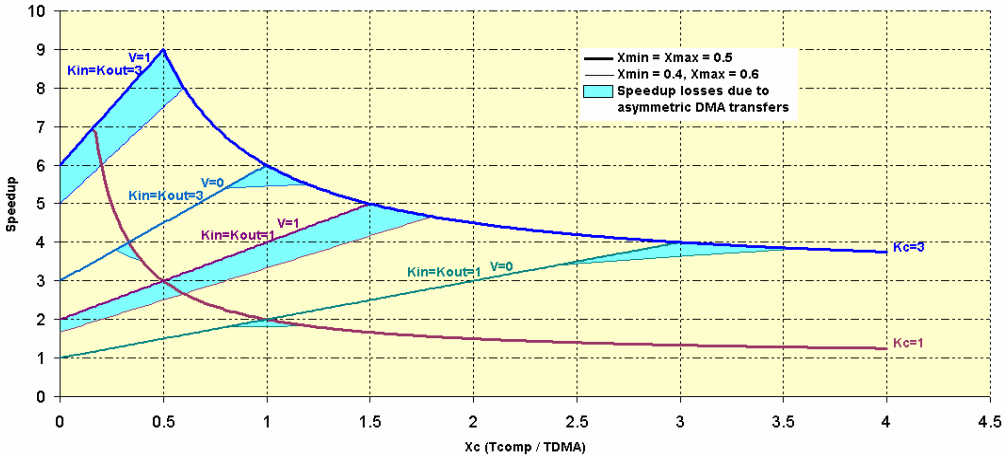


Figure 6. Theoretical asymptotic speedup ($n \to \infty$)

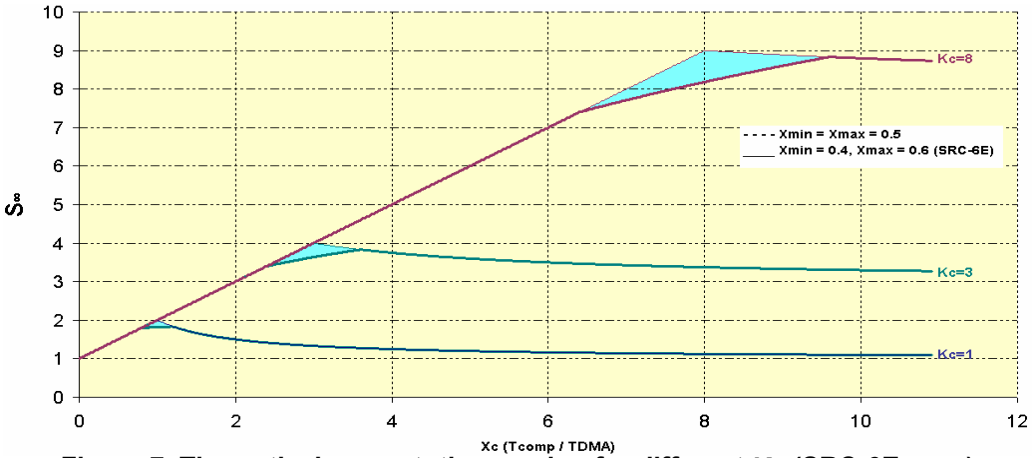


Figure 7. Theoretical asymptotic speedup for different *Kc* (SRC-6E case)

speedup, $S_{\infty max}$, is in direct proportion to the change in the number of concurrent processing units, K_C, and the shift in its location, right shift to larger X_c (slower computations), is also directly proportional to the change in K_C.

4.3. SRC-6E Case Study

4.3.1 Model Parameters

SRC-6E reconfigurable computer has been used as our testbed to verify our model.

To apply our model to SRC-6E system we set the model parameters with some experimentally measured values, and others from the machine specifications.

The machine parameters are set to the following values:

- X_{in}=0.4, X_{out}=0.6 (asymmetric DMA)
- $K_{DMA\text{-}IN}$= $K_{DMA\text{-}OUT}$=1 (single-channel DMA)
- V=0 (non-overlapped DMA channels)

Fig. 7 shows a plot of the asymptotic speedup against X_C for different K_C.

This figure shows the effect on speedup of the DMA asymmetry, as well as the effect of computations concurrency, i.e. the number of concurrent processing units and/or the number of independent data channels between the OBM and the FPGA (see Fig. 4). The peak speedup, $S_{\infty max}$, can be calculated from the following equation:

$$S_{\infty \max} = K_c + \frac{1}{2X_{\max}} = K_c + \frac{1}{2X_{out}} = K_c + 0.83 \qquad (17)$$

In the experimental verification of this model we investigated only applications with the parameter K_C set to 1.

4.3.2 The Design Problem

The design problem can be stated as follows: given the machine constraints and the application constraints, what is the minimum number of transfer parcels that achieves a speedup as close as possible to the asymptotic maximum for that application? In other words, given X_{in}, X_{out}, $K_{DMA\text{-}IN}$, $K_{DMA\text{-}OUT}$, V, and K_C, we are trying to find the minimum n, n_{min}, that gives speedup S very close to S_∞ with an

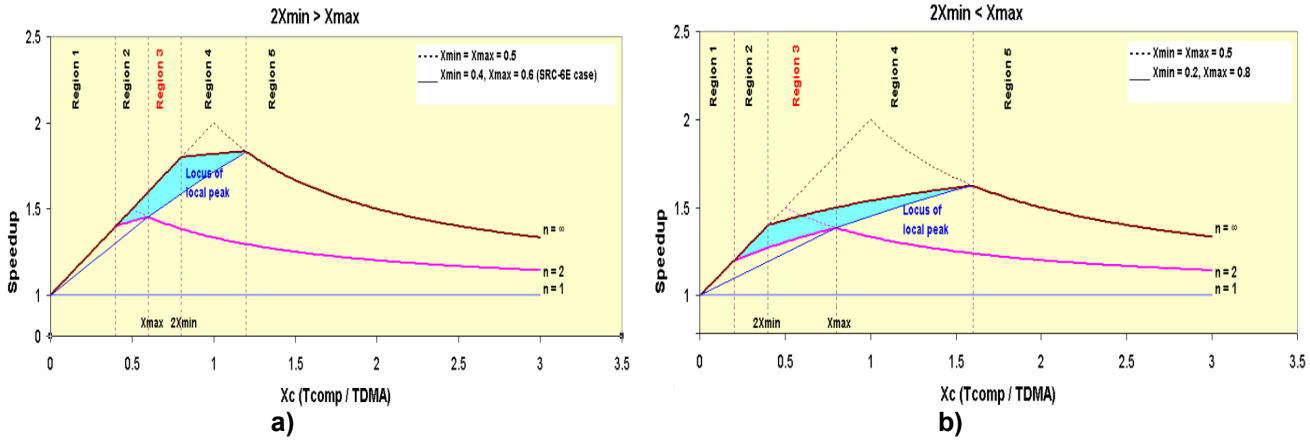


Figure 8. The design regions for different n (SRC-6E case)

Table 2. Design values for the minimum number of transfer parcels, n_{min}

		Region 1	Region 2		Region 3		Region 4		Region 5
			$2X_{min} > X_{max}$	$2X_{min} < X_{max}$	$2X_{min} > X_{max}$	$2X_{min} < X_{max}$	$2X_{min} > X_{max}$	$2X_{min} < X_{max}$	
		$0 \le X_c \le X_{min}$	$X_{min} \le X_c \le X_{max}$	$X_{min} \le X_c \le 2X_{min}$	$X_{max} \le X_c \le 2X_{min}$	$2X_{min} \le X_c \le X_{max}$	$2X_{min} \le X_c \le 2X_{max}$	$X_{max} \le X_c \le 2X_{max}$	$2X_{max} \le X_c < \infty$
S_∞		$1+X_c$	$1+X_c$		$1+X_c$	$\frac{1+X_c}{X_{max}+\frac{1}{2}X_c}$	$\frac{1+X_c}{X_{max}+\frac{1}{2}X_c}$		$1+\frac{1}{X_c}$
S_p		S_∞	$\frac{2(1+X_c)}{1+X_{max}+X_c}$		$\frac{1+X_c}{1+(1-\frac{1}{2X_{max}})X_c}$	$\frac{2(1+X_c)}{1+X_{max}+X_c}$	$\frac{1+X_c}{1+(1-\frac{1}{2X_{max}})X_c}$		$\frac{1+X_c}{\frac{1}{2}+X_c}$
E_p		1	S_p / S_∞		S_p / S_∞	S_p / S_∞	S_p / S_∞		S_p / S_∞
n_{min}	E=1	2	$\frac{1}{1-\frac{X_c}{2X_{min}}}$		$\frac{1}{1-\frac{X_c}{2X_{min}}}$	∞	∞		∞
n_{min}	$E=E_p$					$\frac{E_p X_{min}}{(1-E_p)(X_{max}+\frac{1}{2}X_c)}$	$\frac{E_p X_{min}}{(1-E_p)(X_{max}+\frac{1}{2}X_c)}$		$\frac{E_p}{(1-E_p)X_c}$

efficiency E near to 1, where S_∞ is the asymptotic value of S for this specific application, and E is the ratio between S and S_∞.

From Fig. 8 the design problem can be broken down into two cases, namely when $2X_{min} > X_{max}$ and when $2X_{min} < X_{max}$. Table 2 serves as a guideline to finding the required n_{min}.

5. Experimental Results

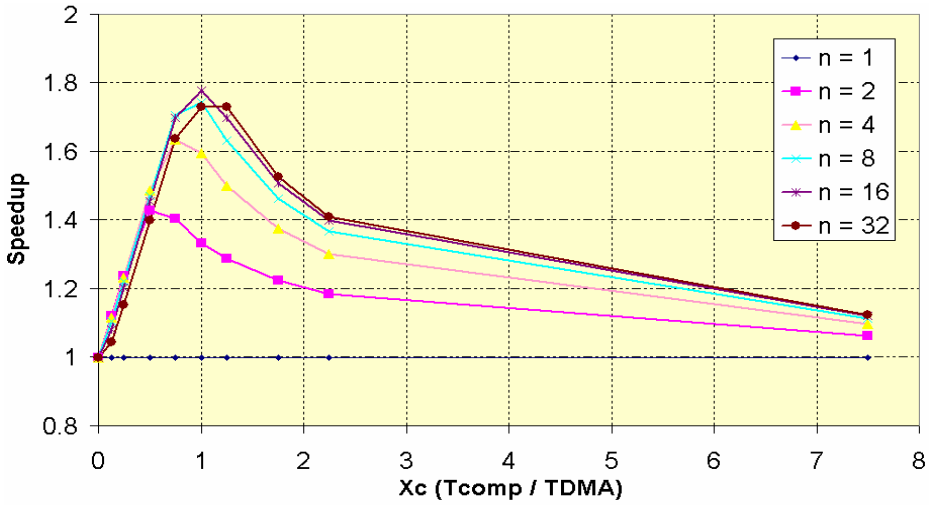
The experimental work has been performed, as mentioned earlier, on the SRC-6E.

In our experiments, we selected a certain value of X_c, and then repeated the experiment multiple number of times with the different number of transfer parcels, n. We started with n=1 (no overlap, speedup = 1), then n = 2, 4, 8, 16, 32. Then, we repeated the experiments for different values of X_c.

The results of experiments are summarized in Fig. 9. All curves, for any value of X_c, start with the unity speedup when n = 1 (no overlap case), then as n increases the speedup increases. After the specific number of stages the speedup starts to saturate.

In our experiments, we have obtained the maximum speedup when X_c was equal to one and n was equal to 16. The speedup obtained for these parameters was equal to 1.78, and was consistent with our theoretical predictions as in equation (17) for the case of X_{min}= 0.4, X_{max} = 0.6 and K_C= 1.

We also confirmed experimentally that the maximum performance could be accomplished when X_c was close to one. When X_c was larger or lower than one, the speedup deteriorated.



Figue 9. Experimental values of speedup

For the case of X_c larger than one, the only gain in speedup is to hide the DMA time within the computations time. When the DMA transfer is very short relative to the computations time, the gain will also be very small. Similarly, when X_c is smaller than one, the gain is to hide the computations time within the DMA time. So, the idea is always to hide the shorter time within the longer time, and when both times are close to each other, we can obtain a speedup close to 2.

6. Conclusions

In this paper, a technique for optimizing the performance of a reconfigurable computer is introduced. A mathematical model for this technique has been derived for a generic reconfigurable machine, taking into account the constraints imposed by both the system and the application. This technique depends on overlapping the computations on the User FPGAs with the I/O transfer. This overlapping requires dividing data transfers into multiple transfer parcels that can be overlapped with partial computations.

The presented technique has been implemented and experimentally verified on the SRC-6E reconfigurable computer. Both theoretical analysis and experimental results proved that this technique is efficient in speeding up the execution time. The maximum theoretical speedup was shown to be 2 for an application with one processing unit and a system with a single DMA channel perfectly balanced for DMA-IN and DMA-OUT transfers. For the current generation of the SRC system, the theoretical maximum speedup was shown to be 1.83, and the corresponding experimental maximum speed-up was 1.78.

References

[1] SRC-6E C-Programming Environment Guide, SRC Computers, Inc. 2003.

[2] Fidanci O. D. , Diab H. , El-Ghazawi T., Gaj K., and Alexandridis N., "Implementation Trade-offs of Triple DES in the SRC-6e Reconfigurable Computing Environment", Proc. MAPLD 2002.

[3] Fidanci O.D., Poznanovic D., Gaj K., El-Ghazawi T., and Alexandridis N., "Performance and Overhead in a Hybrid Reconfigurable Computer", Reconfigurable Architectures Workshop, RAW 2002, Proc. International Parallel and Distributed Processing Symposium (IPDPS) Workshops 2003, Nice, France, April 22-26, 2003, pp. 176 -183.

[4] Michalski A., Gaj K., El-Ghazawi T., "An Implementation Comparison of an IDEA Encryption Cryptosystem on Two General-Purpose Reconfigurable Computers", Proc. FPL 2003, Lisbon, Sept. 2003, pp. 204-219.

[5] Parhi K.K., "VLSI Digital Signal Processing Systems: Design and Implementation", John Wiley & Sons, NY, 1999.

[6] El-Ghazawi T. and Le Moigne J., "Multiresolution Wavelet Decomposition on the MasPar Massively Parallel System", International Journal of Computers and Their Applications, September 1994.

[7] Mallat S.G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 7, July 1989.

[8] Hwang K. and Xu Z., "Scalable Parallel Computing: Technology, Architecture, Programming", McGrawHill, 1998.