Relocation of FPGA Partial Configuration Bit-Streams for Soft-Core Microprocessors

Jeff Carver
Microsoft Research
One Microsoft Way
Redmond, WA 98052
t-jeffc@microsoft.com

Neil Pittman
Microsoft Research
One Microsoft Way
Redmond, WA 98052
pittman@microsoft.com

Alessandro Forin
Microsoft Research
One Microsoft Way
Redmond, WA 98052
sandrof@microsoft.com

ABSTRACT

With dynamic partial reconfiguration (PR) we can augment a soft-core with application specific blocks that may change, or reconfigure, during run-time. In this paper we address some of the inefficiencies in available FPGA tool flows by using bit-stream relocation.

Standard tool-flows from FPGA manufacturers require the creation of separate bit-streams for each PR region. The space and time complexities that this entails are undesirable, especially in an embedded system where storage is at premium. In this paper we introduce a run-time algorithm that allows the relocation of one bit-stream to any number of compatible regions, in linear time. The application loader running on the data path can perform the relocation as well as loading of the application code. We have implemented the algorithm on the eMIPS, a soft-core microprocessor of our own design, and on the MicroBlaze, an industrial production soft-core microprocessor. Evaluation of the algorithm shows a dependency on the composition of the stream and on the target region, as well as a strong dependency on the memory architecture of the system.

1. INTRODUCTION

The preferred model for dynamic partial reconfiguration of FPGAs (PR) is with one static region and one PR region. The static region guarantees the basic functionality and proper behavior during reconfiguration, especially with respect to the I/O signals. The single PR region is used to realize different temporal parts of the application, or alternate realization of certain (signal) processing, or to receive dynamic updates on deployed systems. Solutions that employ more than one PR region are described in the literature, but are not at all well supported by the tools. For example, if the user requires that any configuration may map to any region, it is currently required to synthesize each design repeatedly, once for each PR region. Each compilation can require hours of computer time. In addition, each of those long compilations produces a separate configuration file (bit-stream) for use with the given PR region and nowhere else. These bit-stream files are large even for the smallest FPGA models, in the order of hundreds of kilobytes. Further, all of the bit-streams must be present at run-time. These time and space inefficiencies lead to the desire to use a single bit-stream file that can be relocated to any one of a many PR regions. In this paper, we describe an algorithm for performing the *dynamic* relocation of bit-streams. We demonstrate relocatable bit-streams with two separate soft-cores. We used an extensible soft-core microprocessor of our own design, eMIPS, and an industrial production soft-core microprocessor, MicroBlaze. The two systems used in our tests have radically different architectures, resulting in drastically different performance results. We also found that the size, composition and target location of the bit-stream themselves all affected the time to relocate.

The rest of the paper is organized as follows. Section 2 presents background material and related work. Section 3 describes the bit-stream relocation algorithm, the tool flow, and their implementation. Section 4 presents our experiments and results. Section 5 concludes the paper.

2. BACKGROUND AND RELATED WORK

2.1 Partial Reconfiguration and Relocation

The ability to change portions of the FPGA configuration at run-time is called dynamic partial reconfiguration (PR). This entails modifying portions of the FPGA logic without affecting the remaining parts of the circuit, which continues to function unperturbed. Special support is needed in the FPGA chip for this process to execute flawlessly and without "glitches". Xilinx continues to support PR while Altera supported it in the past but has more recently dropped the feature. Currently the tool provided by Xilinx for doing dynamic partial reconfiguration is part of the Early Access Partial Reconfiguration, or EAPR, a flow that is found at [1]. Refer to [1] for additional information on PR. To perform on-chip reconfiguration on Xilinx devices, a designer instantiates a special macro for the Internal Configuration Access Port (ICAP), then sends the configuration data to it. For the Virtex-4 the ICAP can be implemented with either an 8-bit or 32-bit wide interface.

The bit-stream for configuring one PR region is tightly bound to the physical location of that region and cannot be used directly to reconfigure any other portion of the chip. In many cases though, it is possible to modify an existing bit-stream and adapt it to a different physical location. This process is termed bit-stream relocation and can be performed statically by tools operating on the designer's workstation, or dynamically by the agent that loads the bit-stream on chip. There are various works describing static or dynamic relocation of configurations of a PR region to another. The motivation is to reduce the number of partial bit-streams required if two or more PR regions use the same implementation. For example, if the FPGA had four target PR regions for the bit-stream, the FPGA could save the storage of three bit-streams copies in memory and reduce the compilation time by a factor of four. The savings are noticeable because bit-streams tend to be large and the compilation times are often measured in hours per design. The trade-off is the size of the software/hardware and some placement restrictions on the PR regions to enable relocation.

Becker et al [2] describe the building of bit-streams for a Virtex-4 FPGA designed for relocation. The approach does not allow any static logic in the PR regions. Manipulation of the bit-stream is performed to relocate a column (ex. CLB) to a non-identical column (ex. DSP) with respect to routing. The provided example of a software defined radio with two reconfigurable regions showed a reduction in the number of partial bit-streams by 50%

and compilation time by 43%. We implemented a similar baseline using the MicroBlaze and extended it for use on our eMIPS architecture.

Montminy et al.[3] show how to layout the redundant modules of a Triple Modular Fault Tolerant design in such a way that one module's configuration is relocated to correct the errors in another redundant module. A circuit automatically calculates the CRC value as the bit-stream is being relocated. Horta et al.[7] demonstrate relocatable bit-streams for a Virtex-E chip. They used Gaskets, similar to bus macros, to define the routing between the similar regions. Our FPGAs do not appear affected by the absences of a correct CRC. For this reason, we have delayed adding the CRC calculation in our tools.

Sedcole et al. [11] discuss relocation for a Virtex-4. The distinguishing features in this work includes the ability to route statically through a relocatable region by reserving routing lines for the static logic, and the ability to merge relocatable and static parts at run-time. A certain percentage of long lines are reserved for the static logic to cross over the PR region. The static design is then re-routed to use the reserved long lines. To merge parts at runtime the current configuration is read out, stripped of the previous configuration except the static logic, and merged with the new configuration. This process proved to be very time-consuming, with an increase of reconfiguration time from 6.2x to 11.4x. The example used the HWICAP provided by Xilinx. Due to the additional complexity and time overhead, we chose to exclude static routing from extensible regions.

Kalte et al. present REPLICA [12], a system with a zero-overhead cost in relocation of the partial bit-stream. This is accomplished with a hardware module capable of relocating CLB columns for a Virtex-E FPGA. The hardware module also computes the new CRC automatically. Additional hardware would be needed in a Virtex-4 setup to allow for bit-reversal of the frames. Ferrandi et al. present the Bit-stream Relocation Filter (BiRF) [13], a device similar to REPLICA, but for a Virtex-2 and with minimal area cost. Krasteva et al. describe the pBITPOS tool [14], to allow relocation for Virtex II (Pro) solutions. The additional feature in this work is the ability to relocate configurations that make use of BRAM/MULs. Additional hardware for supporting bit-stream relocation would improve performance; however our implementation is done completely in software at this time.

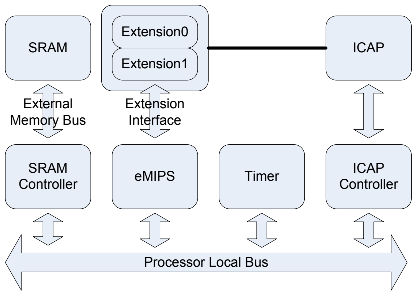


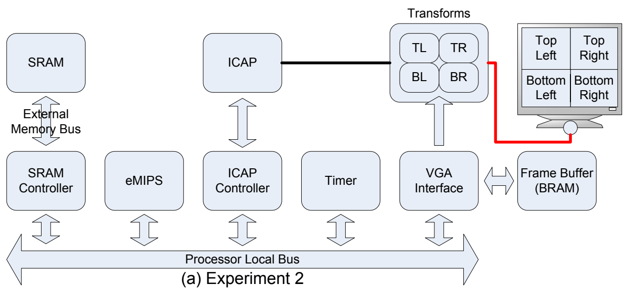
Figure 1. Experiment 1, eMIPS Extensions

2.2 The Extensible MIPS Processor

The eMIPS microprocessor system [8] provides a MIPS [10] RISC data path tightly integrated with Extension slots mapped to PR regions. During run-time the Extension slots implement hardware modules called Extensions. The system is available for download at [9]. Figure 1 and Figure 2(a) show block diagrams of two eMIPS based systems (Experiments 1 & 2). We implemented the eMIPS microprocessor using a Xilinx FPGA and the partial reconfiguration feature to change the state of the Extension slots at runtime. In Figure 1, the eMIPS data path interfaces to the ICAP, timer, General Purpose Input and Output (GPIO), and other peripherals on an internal bus. The eMIPS microprocessor uses the external SRAM for instruction and data memory. Refer to [8] for additional information on eMIPS.

2.3 The MicroBlaze Processor

MicroBlaze is a soft-core RISC processor provided by Xilinx 0. It can be implemented as a 3-stage or 5-stage pipeline. Figure 2(b) provides an overview of how the MicroBlaze interacts with the other components in the system used for our experiments (Experiment 3). Xilinx provides an IP core, the HWICAP, for interfacing the ICAP to the MicroBlaze internal bus. The Timer is free-running to measure the cycles it takes to configure a region. The application code and all the configuration data required for the experiment are stored on the internal block RAMS (BRAMS) of the FPGA.



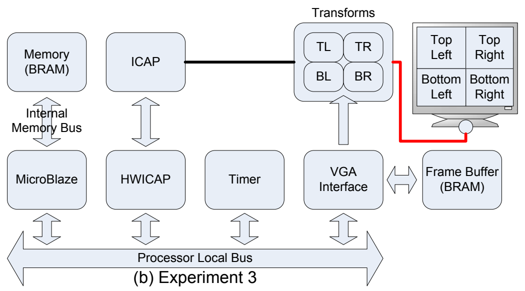


Figure 2. VGA Transform Experimental Setups

2.4 Virtex-4 Configuration Layout

The smallest unit of configuration on a Xilinx FPGA is the frame. A frame includes a fixed number of Configurable Logic Blocks (CLBs), physically laid out in fixed geometries. On the Virtex-4 FPGAs, frames span the height of 16 CLBs, which is one HCLK row. In previous Virtex FPGAs [15] a frame spans the entire height of the chip. Each frame in the Virtex-4 is composed of 1,312 bits. Frames are addressed in a 2-D fashion. A frame address command sets the starting destination of the configuration frames. A frame address on the Virtex-4 is composed of five parts: top/bottom of chip, block type, HCLK row, major column address, and minor column address. The top/bottom part specifies whether the target is at the top or at the bottom of the chip. The block type is one of three types: CLB/IOB/DSP/GCLK (0), BRAM interconnect (1), and BRAM content (2). The HCLK row indicates which row of the chip is targeted. Numbering of the

HCLK row starts from the middle of the chip outward. The major column address specifies which resource column to change. The number begins at zero at the far left of the chip and increments going to the right. The number of minor addresses for a given column depends on the type of resource targeted. There are 22 minor frames for CLBs, 21 for DSPs, 20 for BRAM interconnect, 64 for BRAM content, 30 for IOBs, and 2 for GCLK.

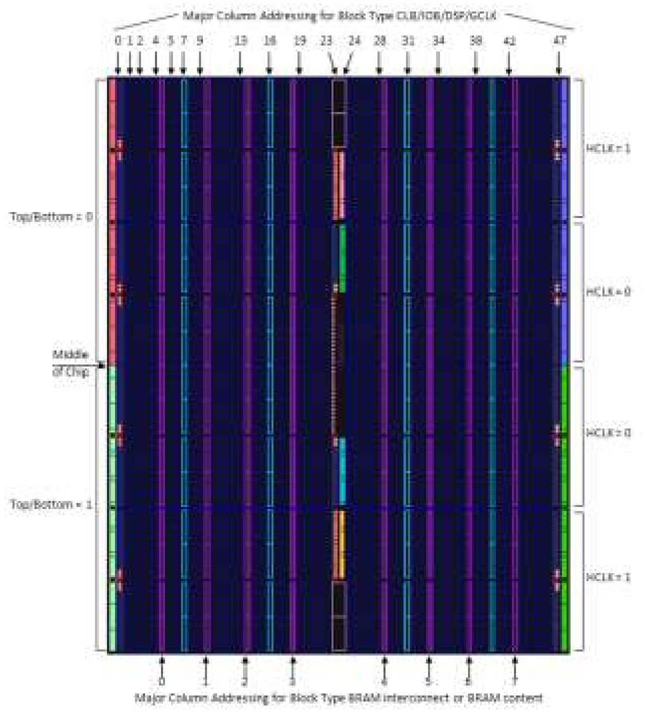


Figure 3. Frame Address Mapping Example on SX35

For example in Figure 3, to reach the first CLB column in the upper-left of the chip, the frame address would be as follows: top/bottom=0, block type=0, HCLK row=1, major column address=1, and minor column address=0-21. This correlation is critical to understand how to manipulate the bit-stream to target the desired PR region.

3. RELOCATING BIT-STREAMS

Relocation of the bit-stream is simple and efficient as long as the source and target PR regions meet a few constraints. In our work the regions must: (1) have the same pattern of resources, (2) span the entire height of the HCLK row (one entire frame), (3) encompass the same amount of area on the chip, and (4) use bus-macros in the same (relative) placement. When these constraints are valid, relocation consists mostly in adjusting the frame addresses in the source bit-stream to match the target PR region.

3.1 Relocation Tool-Flow

The complete tool-flow for performing relocation is shown in Figure 4 and includes both compile-time and run-time elements. The inputs required are a description of (some properties of) the Target FPGA and a description of the PR regions that are potential targets. Currently only the Virtex-4 LX and SX chips are supported because we do not have a complete understanding of exactly how the embedded PowerPC on the FX chip series affects the layout of the configuration frames. The FPGA Configuration Generator computes the pattern of resources and the number of HCLK rows on the target chip. The Adjustment Generator uses this information to compute the run-time parameters required for the Relocation of the Bit-streams stage. These parameters can be compiled in the relocation code, or provided as a data file. The relocation algorithm itself is therefore oblivious to the original inputs. Note that the parameters are specific to the FPGA and its PR configuration, which are also static elements in the overall design. The Adjustment Generator performs some additional checks to verify that the PR regions are relocatable to each other, e.g. that they obey the constraints previously defined.

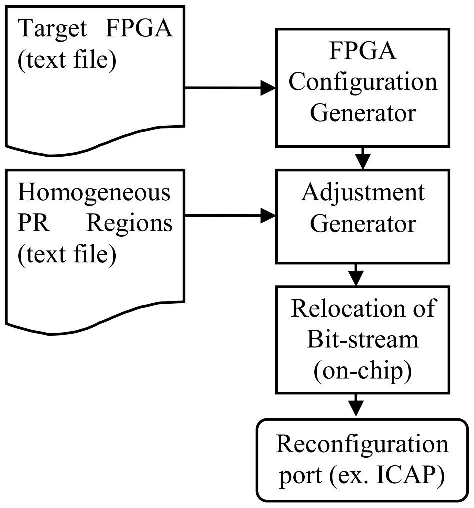

Target FPGA (text file)
FPGA Configuration Generator
Homogeneous PR Regions (text file)
Adjustment Generator
Relocation of Bit-stream (on-chip)
Reconfiguration port (ex. ICAP)

Figure 4. Tool Flow

3.2 Implementation Constraints

Before we discuss the run-time relocation algorithm, we need to consider a few practical problems that will strongly affect our results.

A set of configuration frames for a column of CLBs can be used as-is in another column of CLBs because it requires the exact same routing bits and configuration logic bits. This is the good case that we want to obtain at run-time, if at all possible, because the relocation can then proceed at full speed. A more difficult case is when a set of configuration frames is targeted for one side of the chip (ex. top) and the target is at the other side of the chip (ex. bottom). On the Virtex 4, this case requires a bit-reversal of the configuration frames, each frame being over a thousand bits long. Architectures that do not implement this mirroring from top to bottom (ex. Virtex 5) can remove this step. The architectural layout of frames on the Virtex-4 chips is such that frames on the top of the chip are mirror images of the frames on the bottom of the chip. Only the middle word in the frame is not mirrored. This word contains the global routing bits and other configuration data. Bit-reversal means, for example that the bit in position 0 is at position 1,311 on the other side of the chip. This requires that the frame is read from end to beginning, because the word at the end of the frame is now the first word that needs to be bit reversed and written to the ICAP. This can be cumbersome and costly to do in software. In our implementation the frame is already in memory and read backward from end to start. Each word is bit reversed using a lookup table at the byte level. This at minimum requires four loads from the LUT, three shifts and three ORs per word. Then the word is written to the ICAP. Testing shows that this algorithm is much better than others that use ALU instructions. Hardware support can eliminate the need for this costly reversal step through multiplexing the 32-bit signal with its bit reversed version and a control bit. Such implementation would allow the bit reverse case to match the performance of the case where bit reversing is not required.

The wires used for routing signals in/out of the PR region must match between the source and the target PR regions. The configuration data routes only to/from adjacent columns, and assumes that there actually is an external connection in the adjacent column at the periphery of the region. This requirement is currently handled by the bus macros on the Virtex-4. As long as the bus-macros have the same relative positioning between the two PR regions, this scheme correctly handles the relocation constraints between PR regions and our algorithm does not have to modify the routing information at all.

Using the available FPGAs, a transformation of the relative location of the bus macros is not possible. In practical applications it would be beneficial for PR regions on opposite sides of the FPGA to have their bus macro placements reflected across the middle to take advantage of symmetries in design. However, the bits encoding the routing data in the bit stream is not of a structure that can be readily transformed short of rerunning the routing algorithm.

For ease and speed of reconfiguration, we decided not to support static routing, e.g. the logic in the static region is not allowed to route through the PR regions. This can have an adverse effect on the design, for instance when that signal must route to an I/O pin. To handle this requirement, the static routing must either match across all of the PR designs, or we could use the scheme described in [11] and reserve some long lines for the static region inside the PR regions. We prohibited static routing from the PR region by setting the "ROUTING=CLOSED" constraint in the Xilinx ISE. Even though we used this constraint the router would still sometimes inexplicably route through the PR region when trying to get to the bus macros. To help the router route around the PR region, we created target LUTs to route to before routing to the troublesome bus macros.

3.3 Relocation Algorithm

Figure 5 shows the algorithm for relocating bit-streams at run-time. There are three possible destinations for a configuration:

1. Destination is where the bit-stream was generated for. No relocation is necessary.
2. Destination is not where it was generated for, but on the same side of the chip. Relocation consists only of the translation of frame addresses.
3. Destination is on the opposite side of the chip. Relocation involves both the translation of the frame addresses and bit-reversal of the frames.

It is actually possible to encounter a combination of cases two and three. The Multiple Frame Write (MFWR) command can write a single frame of data to multiple frames addresses [15] and this can create a problem. Suppose a bit-stream that is on the top of the chip covering two HCLK rows must relocate to a PR region that straddles the middle of the chip. In the target region, one HCLK is used for both the top and bottom of the chip. A MFWR command could correctly write the same configuration frame to both HCLK rows in the original configuration, but in the new configuration we now need two separate MFWR commands, one for the top and one for the bottom of the chip. The frame data is valid as-is for the HCLK on the top of the chip, but it must be flipped for the HCLK row on the bottom of the chip. The separation can be done at run-time. The cost is just some additional code because we need to buffer the frame data regardless. It is clearly easier to use a tool that expands the MFWR commands into multiple ones before deploying the bit-stream. Therefore the examples presented in Section 5 do not cover this case.

Based on this algorithm, the three key contributing factors in the time to relocate a bit-stream are (1) the bit-stream size, (2) the bit-stream composition, and (3) the location of the destination PR region on the chip. By composition we mean the relative count of commands that set the frame address and commands that write configuration frames. If the stream must be relocated, it would be best if we have very few frame addresses and do not need bit-reversal.

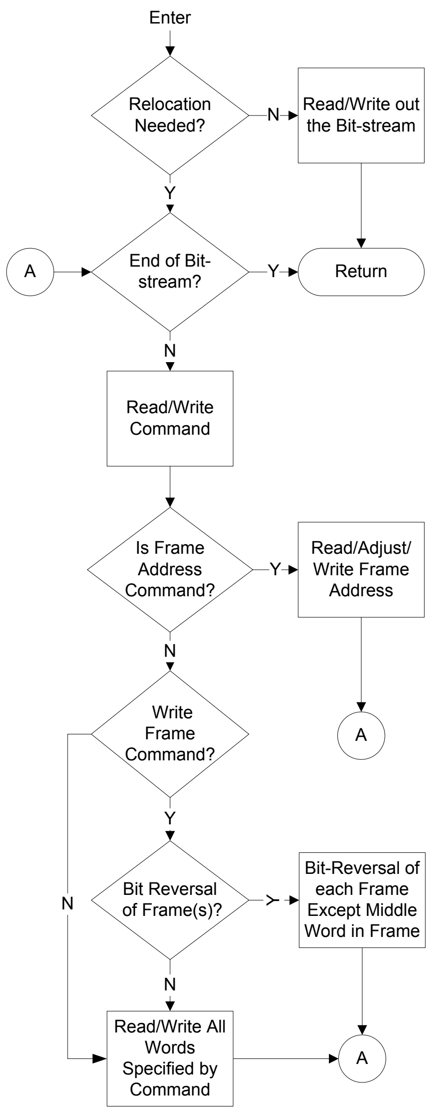

Enter
Relocation Needed?
N
Read/Write out the Bit-stream
Y
A
End of Bit-stream?
Y
Return
N
Read/Write Command
Is Frame Address Command?
Y
Read/Adjust/ Write Frame Address
N
A
Write Frame Command?
Y
N
Bit Reversal of Frame(s)?
Y
Bit-Reversal of each Frame Except Middle Word in Frame
N
Read/Write All Words Specified by Command
A

Figure 5. Relocation Algorithm

4. RESULTS

We ran our experiments on a 2.4 GHz Intel Dual core processor with 2 gigabytes of RAM using Xilinx ISE 9.2.4.PR7. As an example of the compilation time saved on map, place and route, a bit-stream with 99,528 bits took 28 minutes and 50 seconds to complete. This does not include the time to generate bit-streams. A bit-stream with 87,888 bits took 27 minutes and 5 seconds to complete. Those savings are multiplied by the number of PR regions that we do not need to compile for.

We implemented the relocation algorithm in C and evaluated the performance on two soft-cores, using the GCC compiler in both cases. We used two tests, in the three setups of Figure 1 and Figure 2(a,b).

In the first test, we used the setup of Figure 1 and implemented two eMIPS Extensions (mmldiv64 and ldret) for one of the two Extension slots only. We then used the relocation algorithm to relocate the Extension to the other slot, as part of the image activation by the RTOS.

In the second test, we implemented a simple image display system, using a VGA driver and four transformation modules that alter the outputs of the VGA on the path to the monitor. Figure 2 depicts this setup. The PR regions correspond to the four quadrants of the display, each one implements one transformation and are reconfigurable. We implemented four transformations: no transform (INIT), SIN, COS and MULT. We only synthesized for one of the PR regions and used the relocation algorithm to dynamically relocate this implementation for the other regions. We run the experiment with both the MicroBlaze and the eMIPS microprocessors.

Table 1 presents a breakdown of the configuration bit-streams sizes and frame address. The VGA experiment emphasizes small designs. By performing the same relocation algorithm with two different host microprocessors we can see the impact of the non-software aspects of the relocation process. The MicroBlaze uses exclusively the on-chip memory provided by the BRAMs. The eMIPS instead executes from the off-chip memory available in the SRAM of the ML40x. The eMIPS Extensions emphasize larger designs for which the algorithm will have a greater impact. All of the designs were clocked at 100 MHz.

Table 1. Configuration Bit-streams

Configuration Name	Size of Bit-stream (bytes)	#Frames Written	# FAR Commands
Blank (Extension)	26488	52	862
MMLDIV64 (Extension)	99528	559	367
LDRET (Extension)	87888	477	457
SIN (Transform)	11184	64	10
MULT (Transform)	11616	67	7
COS (Transform)	11652	67	9
INIT (Transform)	11076	62	22

The results from relocating different bit-streams on eMIPS are shown in Table 2, the results for MicroBlaze in Table 3.

Table 2. Relocation Timing Results Using eMIPS

Configuration Name	Relocate ?	Frame Bit Reversal?	Time (msec)	KB/sec
Blank (Extension)	N	N	83.1	318.7
	Y	N	212.5	124.6
	Y	Y	1488	79.97
MMLDIV64 (Extension)	N	N	312	318.9
	Y	N	466.6	213.2
	Y	Y	1709	58.23
LDRET (Extension)	N	N	275.5	318.9
	Y	N	427.4	205.6
	Y	Y	1488	59.05
SIN (Transform)	N	N	35.12	318.3
	Y	N	49.32	226.7
	Y	Y	191.5	58.38
MULT (Transform)	N	N	36.48	318.4
	Y	N	50.82	228.5
	Y	Y	199.6	58.17
COS (Transform)	N	N	36.59	318.4
	Y	N	51.21	227.5
	Y	Y	191.5	58.23
INIT (Transform)	N	N	34.78	318.3
	Y	N	50.25	220.3
	Y	Y	188.2	58.82

In the eMIPS measurements, the bit-streams are located in the SRAM section of the board, along with the code and data buffers for the relocation program itself. As can be readily seen comparing Tables 2 and 3, the memory type and parameters chosen for a design will impact the latency required to relocate the bit-stream. For the eMIPS setup the latency for accessing SRAM is five cycles. Using DDRAM would create more latency and FLASH would be even worse. At present, eMIPS does not use any caches or on-chip memory. This penalizes the results in Table 2 because they include not only the time to fetch the bit-stream from SRAM but also the instruction fetches and data load/stores. The temporary swap buffer is also located in SRAM.

Table 3. Relocation Timing Results Using MicroBlaze

Configuration Name	Relocate ?	Frame Bit Reversal?	Time (msec)	KB/sec
SIN (Tranform)	N	N	1.121	9971
	Y	N	1.190	9392
	Y	Y	3.174	3523
MULT (Transform)	N	N	1.165	9971
	Y	N	1.229	9445
	Y	Y	3.309	3509
COS (Transform)	N	N	1.168	9972
	Y	N	1.236	9419
	Y	Y	3.317	3512
INIT (Transform)	N	N	1.111	9966
	Y	N	1.204	9196
	Y	Y	3.148	3517

If the bit-stream does not require any modification the throughput achieved is about 318 kilobytes per second. If the bit-stream is relocated but does not require a bit reversal of the frames, the throughput is generally around 220 kilobytes per second. If the bit-stream requires a reversal in the bits in the configuration frames, the throughput is about 59 kilobytes per second.

The Blank bit-stream does not follow the trend of the other points due to its composition. The blanking bit-stream removes almost all the routing that was done in the PR region, which results in a large number of matching configuration frames. The bit-stream issues a large amount of MFWR commands to write the same configuration frame to multiple frame addresses. This bit-stream therefore contains an unusually large concentration of frame addresses relative to its size. For comparison, the LDRET bit-stream has only 457 frame address commands compared to 862 for the Blank bit-stream. This decreases the throughput for the relocation with no bit reversal because of the increased calls to translate the frame addresses. Similarly, this bit-stream performs better than average for the relocation with bit reversal of frames because it contains a low concentration of configuration frames compared, for instance, to the COS bit-stream. This results in reducing the penalty of calling the bit-reversal function. The throughput for the no-modification case is approximately the same for all the different bit-streams. This is expected, since the algorithm is just copying the bit-stream to the ICAP without any modification.

The results from relocating on the MicroBlaze setup (Figure 2(b)) are shown in Table 3. In this case, all of the bit-streams, code and data buffers are located in BRAM on the chip, which is 32-bit wide and accessible in a single cycle. If the bit-stream does not require modification the throughput achieved is about 10 megabytes per second. If the bit-stream is relocated but does not require a bit reversal of the frames, the throughput is between 9.2 to 9.5 megabytes per second. If the bit-stream requires a bit-reversal of the configuration frames, the throughput is about 1.4 megabytes per second.

The ICAP accepts one write per cycle, and in both experiments it was configured in a 32-bit width. Since the designs are run at 100 MHz, the maximum achievable throughput is 400 megabytes per second. The BRAMs provide the same throughput.

5. CONCLUSIONS

We have shown that relocatable bit-streams are beneficial in two dimensions: they reduce the number of bit-streams stored on a deployed system and they save compilation time during development. Both savings scale linearly with the number of PR regions used in a system.

We have presented an on-chip algorithm and the corresponding tool-flow for performing bit-stream relocation. The algorithm was implemented and evaluated in two different architectures, leading to different performance numbers. In both cases, the content and destination of the bit-stream have the same and very noticeable effect on the maximum achieved throughput. The span between maximum and minimum throughput can be up to a factor of four, and far from the maximum bandwidth of the configuration port.

6. REFERENCES

[1] Available at http://www.xilinx.com/support/prealounge/protected/index.htm

[2] Becker, T.; Luk, W.; Cheung, P.Y.K., "Enhancing Relocatability of Partial Bit-streams for Run-Time Reconfiguration," *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, vol., no., pp.35-44, 23-25 April 2007.

[3] Montminy, D.P.; Baldwin, R.O.; Williams, P.D.; Mullins, B.E., "Using Relocatable Bit-streams for Fault Tolerance," *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, vol., no., pp.701-708, 5-8 Aug. 2007.

[4] Note, J. and Rannaud, É. 2008. From the bit-stream to the netlist. In *Proceedings of the 16th international ACM/SIGDA Symposium on Field Programmable Gate Arrays* (Monterey, California, USA, February 24 - 26, 2008). FPGA '08. ACM, New York, NY, 264-264.

[5] Guccione, S., Levi, D. and Sundararajan, P. "JBits: Java based interface for reconfigurable computing", Xilinx Inc, San Jose, CA

[6] C. Claus, B. Zhang, M. Huebner, C. Schmutzler, J. Becker, W. Stechele, "An XDL-based busmacro generator for customizable communication interfaces for dynamically and partially reconfigurable systems", Workshop on Reconfigurable Computing Education at ISVLSI 2007, Porto Alegre, Brazil, May 12, 2007.

[7] Horta, E.L.; Lockwood, J.W.; Taylor, D.E.; Parlour, D., "Dynamic hardware plugins in an FPGA with partial run-time reconfiguration," *Design Automation Conference, 2002. Proceedings. 39th* , vol., no., pp. 343-348, 2002.

[8] Pittman, R. N., Lynch, N. L., Forin, A. eMIPS, A Dynamically Extensible Processor, MSR-TR-2006-143, Microsoft Research, WA, October 2006.

[9] Download at http://research.microsoft.com/research/EmbeddedSystems/eMIPS/eMIPS.aspx

[10] Kane, G., Heinrich, J. 1992. MIPS RISC Architecture. Prentice Hall, Upper Saddle River, NJ.

[11] Sedcole, P.; Blodget, B.; Becker, T.; Anderson, J.; Lysaght, P., "Modular dynamic reconfiguration in Virtex FPGAs," *Computers and Digital Techniques, IEE Proceedings -* , vol.153, no.3, pp. 157-164, 2 May 2006.

[12] Kalte, H.; Lee, G.; Porrmann, M.; Ruckert, U., "REPLICA: A Bit-stream Manipulation Filter for Module Relocation in Partial Reconfigurable Systems," *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International* , vol., no., pp. 151b-151b, 04-08 April 2005.

[13] Ferrandi, F., Novati, M., Morandi, M., Santambrogio, M. D., Sciuto, D. "Dynamic Reconfiguration: Core Relocation via Partial Bit-streams Filtering with Minimal Overhead," *System-on-Chip, 2006. International Symposium on* , vol., no., pp.1-4, Nov. 2006.

[14] Krasteva, Y.E.; de la Torre, E.; Riesgo, T.; Joly, D., "Virtex II FPGA Bit-stream Manipulation: Application to Reconfiguration Control Systems," *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on* , vol., no., pp.1-4, 28-30 Aug. 2006.

[15] Xilinx Inc. Virtex-4 Configuration Guide v1.10, April 2008.

Xilinx Inc. MicroBlaze Processor Reference Guide. URL: http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf.