High Speed Partial Run-Time Reconfiguration Using Enhanced ICAP Hard Macro

Simen Gimle Hansen, Dirk Koch and Jim Torresen
Department of Informatics, University of Oslo, Norway
{simenha, koch, jimtoer}@ifi.uio.no

***Abstract*—Achieving high speed run-time reconfiguration is important for the adaptation of partial reconfiguration in many applications. The reconfiguration speed that is currently available today is somehow artificially limited by the FPGA vendors, while the fabrication process technologies used for building the latest devices today are capable of achieving much higher reconfiguration speed. In this paper we will present a new design and implementation method for achieving high speed partial run-time reconfiguration that exceeds the specified reconfiguration speed of today's FPGAs. By adding custom logic around the Internal Configuration Access Port (ICAP) to implement an enhanced ICAP hard macro, we will investigate the partial run-time reconfiguration speed and explore the limits of the ICAP interface. This is done by using overclocking of the ICAP. Compared with previously work on high-speed reconfiguration, using the enhanced ICAP hard macro will significantly increase the reconfiguration speed.**

***Keywords*-Field Programmable Gate Array (FPGA), High Speed Partial Run-Time Reconfiguration, Internal Configuration Access Port (ICAP), Enhanced ICAP Hard Macro.**

I. Introduction

The adaptation into using partial run-time reconfiguration of FPGAs have up till now been slow in the industry. Although partial run-time reconfiguration of FPGAs has been available for more than a decade, there is still not any wide spread usage of it. There are two main reasons for this. The lack of good commercial design tools supporting partial run-time reconfiguration, and the low recommended reconfiguration speed that limits the use of partial run-time reconfiguration in many applications. Achieving high speed run-time reconfiguration is important for the adaptation of partial reconfiguration in many applications. The reconfiguration speed that is currently available today is somehow artificially limited by the FPGA vendors, while the fabrication process technologies used for building the latest devices today are capable of achieving much higher reconfiguration speed. For example, Xilinx, the main vendor in providing run-time reconfiguration devices, has through its last three generations of run-time reconfigurable devices kept the reconfiguration speed at the same level. While their Virtex-4 and Virtex-5 devices are manufactured in a 90 and a 65 nm process [15], [14], the recent Virtex-6 generation is based on a 40 nm process [17].

Partial run-time reconfiguration has been an active research field for more than a decade. Much research has been done on increasing the reconfiguration speed and reducing the reconfiguration time. Many different approaches and ideas have been proposed. In [5], the concept of configuration prefetching was introduced to hide and improve the reconfiguration time by swapping modules in the background. [4], uses configuration preemption of modules together with reconfiguration port scheduling to improve the efficiency of the reconfiguration interface and to decrease the reconfiguration time under real-time constraints. In [9], the concept of hyper reconfigurable architectures was introduced to reduce the amount of configuration data need to perform reconfiguration, which results in lower reconfiguration time and faster reconfiguration speed. Bitstream decompression has been investigated in [10], [6], [7] to reduce the bandwidth requirements of configuration storage and buses during reconfiguration, and to achieve lower reconfiguration time while still being able to achieve full reconfiguration speed. Several different types of reconfiguration controllers, with or without Direct Memory Access (DMA) capabilities, have been investigated in [2], [12], [11] to improve the reconfiguration speed and reduce the reconfiguration time.

Common for all the research mentioned above, is that the research have mostly taken place within the limits of the recommended default reconfiguration speed. So far, only [1] has gone beyond the recommended default reconfiguration speed limit specified by Xilinx, by overclocking the configuration interface beyond the specified operation frequency.

In this paper, we will present a new design and implementation method for achieving even higher speed during partial run-time reconfiguration that exceeds the recommended reconfiguration speed of FPGAs from the vendor Xilinx multiple times. We will investigate the run-time reconfiguration speed and explore the limits of the ICAP interface.

This paper is organized as follows: In Section II, we will first give a short introduction to the Xilinx Internal Configuration Access Port (ICAP) interface which is used for dynamic run-time reconfiguration. Then, we will present and describe our enhanced ICAP hard macro module. The enhanced ICAP hard macro module is based on an enhanced extension of the Xilinx's native ICAP primitive. Section III describes the two high-speed partial run-time reconfiguration architectures we have implemented for testing and verification of the enhanced ICAP hard macro module. We will also describe how the verification and testing of the enhanced ICAP hard macro module has been performed on the Xilinx XUPV5-LX110T development board. Experimental performance measurement results for the enhanced ICAP hard macro module will be presented in Section IV and compared with previously related

work. Finally, the paper conclusions and a discussion of future work are presented in Section V.

II. Internal Configuration Access Port (ICAP)

The Xilinx ICAP primitive provides internal access to the configuration logic of the FPGA from within the FPGA fabric. Through the ICAP interface, the configuration data can be dynamically loaded into the configuration memory of the FPGA at run-time. It is also possible to perform a readback of the configuration data from the configuration memory or to read status registers of the configuration logic with the ICAP interface. As shown in Figure 1, the ICAP primitive interfaces to the configuration memory which provides access to the configurable resources of the FPGA fabric.

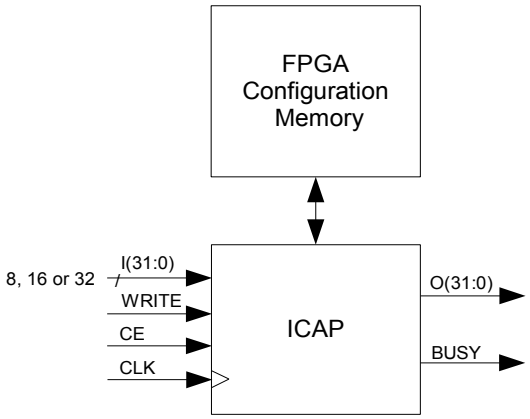

Fig. 1. Xilinx ICAP Primitive.

The ICAP interface consists of separate data ports for reading (O) and writing (I) configuration data. The width of the configuration data ports can be configured to be 8 or 32 bit wide for Virtex-4, and 8, 16 or 32 bit wide for Virtex-5 and Virtex-6, respectively. The ICAP interface also provides a chip enable (CE) and a write enable (WRITE) input signals, a busy/ready (BUSY) output signal, and a clock (CLK) input. For Virtex-5 and Virtex-6 devices, the BUSY output signal is only used during read operations. During write operations the BUSY signal is set to low [16], [18]. Writing is done when WRITE is set low and reading is done when WRITE is set high, respectively. Configuration data is written to the device at the rising clock edge and if the ICAP port is enabled. Consequently, the writing of configuration data can be controlled by the clock as well as by setting the enable signal while connecting the ICAP primitive to a fixed clock.

For Virtex-4, Virtex-5, and Virtex-6, Xilinx specifies that the ICAP interface is not to be clocked faster than the recommended default frequency of 100 MHz. With a frequency of 100 MHz and a 32 bit wide data interface, the ICAP interface will then provide a maximum default reconfiguration speed of 3200 Mbits/s or 400 MBytes/s. For comparison, Table I lists the recommended default reconfiguration speed of the different configuration interfaces for Xilinx Virtex-4 and later FPGAs.

Although Xilinx by the default recommendation limits the reconfiguration speed, the fabrication process technologies used for building Virtex-4 and later, should be capable of running the ICAP interface at a higher clock frequencies. For example, the DSP48 slices of the Virtex-4 are capable of running at a frequency of up to 500 MHz. This indicates that there should be headroom for clocking the ICAP at a much higher frequency than the default speed, and possibly achieve a much higher maximum reconfiguration speed. We will in this paper take advantage of this and explore this headroom to achieve high speed partial reconfiguration.

Configuration Interface	Type	Bit Width	Frequency MHz	Configuration Speed Mbits/s / MBytes/s
Serial Configuraton Port	External	1	100	100 / 12.5
Boundary Scan Port	External	1	66	66 / 8.25
SelectMap Port	External	8 16 32	100 100 100	800 / 100 1600 / 200 3200 / 400
ICAP	Internal	8 16 32	100 100 100	800 / 100 1600 / 200 3200 / 400

TABLE I
Configuration speed for the different interfaces on Xilinx FPGAs.

A. Enhanced ICAP Hard Macro

Implementing FPGA designs by just using the default design tool chain flow are sometimes not enough to achieve timing requirements and timing closure for the high speed parts of a design. To be able to achieve the maximum possible performance with regard to clock frequency and operational speed, it is necessary to be in full control of the placement and routing of the logic elements and the signal wires in the design. One way of achieving this with the Xilinx design tools chain flow is to implement the high speed parts of a design as a pre-placed and pre-routed hard macro. Using a hard macro provides the designer with the ability to maintain and define the exact placement and routing of a design. The Xilinx vendor tools, allows a designer to implement hard macros basically in two different ways. Firstly, they can be directly generated as netlists using the Xilinx Design Language (XDL). This was used in [3] and [8] to implement On-FPGA communication architectures for integrating reconfigurable modules into a system. Secondly, macros can be implemented with a graphical tool from Xilinx called FPGA Editor. For this work, we used the second option, as the XDL-flow is not fully documented.

The external SelectMAP interface and the internal ICAP interface provides the fastest possible default reconfiguration speed of the different configuration interfaces that recent Xilinx Virtex series FPGAs support. Both interfaces are specified to provide a reconfiguration speed of 400 MBytes/s. Of these two, the ICAP interface is the key component for achieving high speed partial run-time reconfiguration by being one of the fastest available configuration interfaces in Xilinx FPGAs, and also the only way to access the configuration interface from *within* the FPGA fabric. We investigated the internal

configuration port as we expected faster on-chip operation then off-chip operation via external pins. To be able to achieve reconfiguration speed that goes beyond the recommended default speed, and to explore the limit of what the ICAP can sustain, it is necessary to enhance the default ICAP primitive with additional logic. By adding our own custom additional logic in front of the default ICAP primitive we have created and implemented an enhanced ICAP hard macro. Its function is to widen the data path size of the original ICAP primitive from 32 bit to 64 bit in order to gain configuration throughput. The enhanced ICAP hard macro can thus be seen as a 64-to-32 bit data multiplexer that takes 64 bit data input at one data rate and multiplexing it out as two 32 bit data outputs at twice the data rate.

The enhanced ICAP hard macro (ICAP_64) primitive is shown in Figure 2. The ICAP_64 primitive consists of two 32 bit wide data ports for writing (L and H) configuration data and one 32 bit wide data port for reading (O) configuration data. In addition to the original set of control signals of the ICAP (WRITE and BUSY), the ICAP_64 primitive requires two clock inputs, CLK1X and CLK2X. The two clock inputs CLK1X and CLK2X must be in phase with each other and the CLK2X clock must also operate at twice the frequency of that of the CLK1X clock.

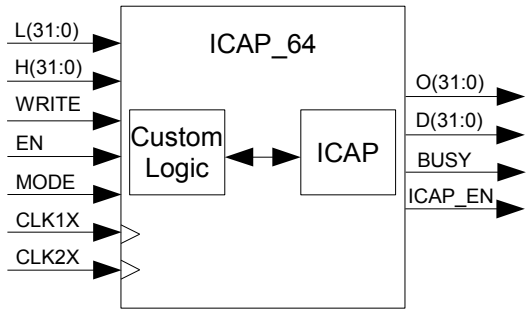

Fig. 2. Enhanced ICAP Hard Macro (ICAP_64) Primitive.

The ICAP_64 primitive also provides an enable (EN) and a mode (MODE) input signals. The MODE input signal is used for controlling how the ICAP_64 hard macro operates. The ICAP_64 hard macro can operate in two modes. A high speed mode (MODE set to low) where the configuration data is written to the ICAP at each rising edge of CLK2X when EN is set high, and a low speed mode (MODE set to high) where the configuration data is written to the ICAP at the first rising edge of CLK2X after a rising edge of the EN signal. In other words, for the low speed mode the EN signal must be toggled for each 32 bit configuration data word to be written to the ICAP. The low speed mode is only used for debug purpose. Two output signals for debugging (D and ICAP_EN) are also provided by the ICAP_64 primitive.

A block diagram of the ICAP_64 is shown in Figure 3. The CLK1X clock is used for sampling the two 32 bit wide data ports L and H into two 32 bit registers, L_REG and H_REG. The EN input signal is also sampled into the EN_REG register by the CLK1X clock. The two 32 bit registers L_REG and H_REG are connected to a multiplexer (MUX) which multiplexes between the two registers starting with the L_REG register. The output of the multiplexer is connected to a 32 bit wide pipeline register ICAP_DATA. The multiplexer MUX and the pipeline register ICAP_DATA is clocked and controlled by the CLX2X clock. The output of the ICAP_DATA register is connected to the input data port (I) of the Xilinx's ICAP primitive and clocked into the ICAP primitive by the CLK2X clock.

B. Resource Utilization of the Enhanced ICAP Hard Macro

The implementation of the enhanced ICAP hard macro requires a total of 27 slices, using 104 slice Flip Flops and 99 slice LUTs. We were careful to fit the logic in a low number of slices, as partly filled slices within a macro will be left unused by the Xilinx technology mapping tool (*map*). The resource utilization of the enhanced ICAP hard macro is summarized in Table II.

# Slices	# LUTs	# Flip Flops
27	99	104

TABLE II
RESOURCE UTILIZATION OF THE ENHANCED ICAP HARD MACRO.

Figure 4 shows the physical layout of the enhanced ICAP hard macro in the Xilinx FPGA Editor. Timing analysis done with Xilinx FPGA Editor, reports a minimum delay path for the enhanced ICAP hard macro to be 0.280 nanoseconds. The maximum delay path is reported to be 1.232 nanoseconds. Assuming a setup time of 0.49 ns for CLB flip-flops, this gives a maximum possible operation frequency for the enhanced ICAP hard macro of 580 MHz. However, this is beyond the operating frequency of what the Xilinx ICAP primitive is capable of operating at. We will in section IV reveal the maximum operating frequency of the Xilinx ICAP primitive found by our experiments.

III. EXPERIMENTAL SETUP

This section describes the experimental test system and the test experiments that have been developed for testing and verification of the enhanced ICAP hard macro. All of the experiments have been performed on at least three different Xilinx XUPV5-LX110T development boards with a Virtex-5 XC5VLX110T-1C (slowest speed grade) FPGA.

A. Experimental Test System

A test system has been developed to prove and verify the concept behind the enhanced ICAP hard macro. The test system was made as small as possible to leave most of the FPGA area unused. The reason for this is that we want to test and verify that the enhanced ICAP hard macro can read and write configuration data to the configuration memory in the FPGA, at as many as possible different locations with different resources, and with a high as possible frequency. By maximizing the empty area of the test system, we are also maximizing the available unused configuration memory that

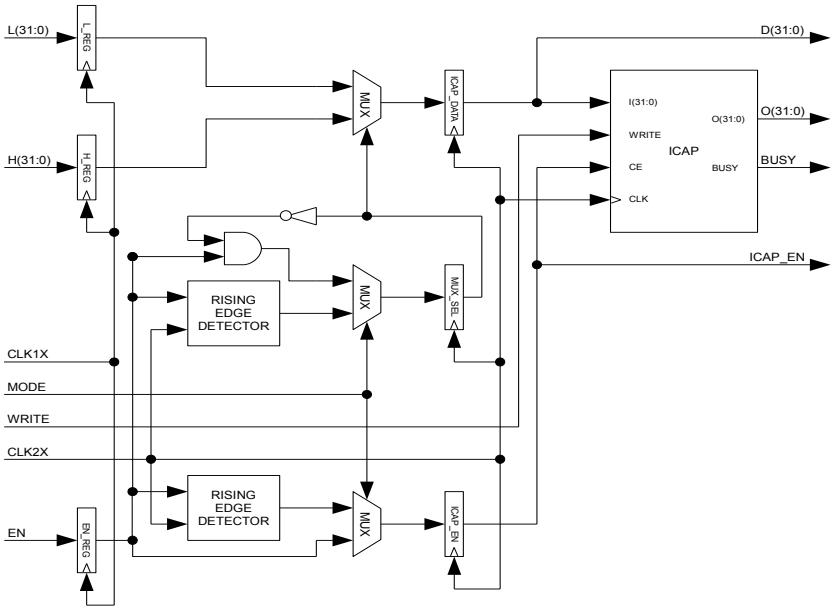

Fig. 3. Enhanced ICAP Hard Macro (ICAP_64) Block Diagram.

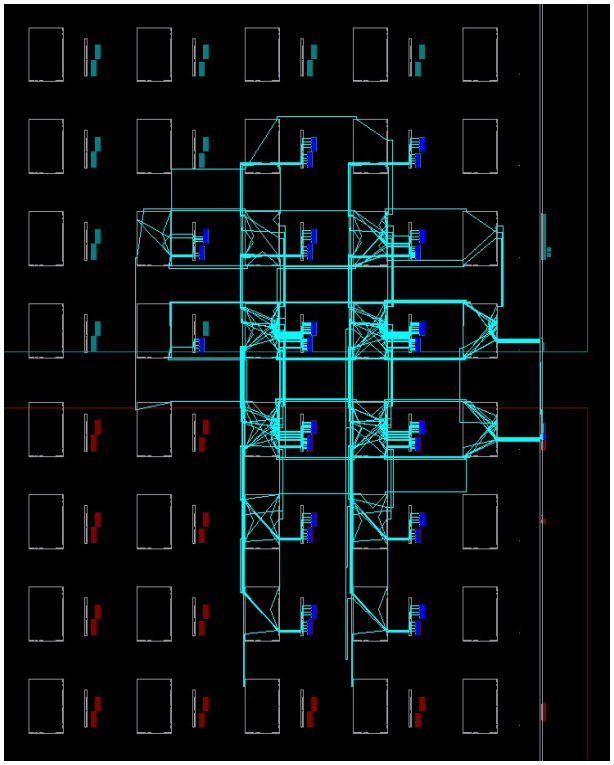

Fig. 4. Floorplan layout of the Enhanced ICAP Hard Macro (ICAP_64) taken from Xilinx FPGA Editor.

can be used for reconfiguration to test and verify the enhanced ICAP hard macro.

The structure of the test system is depicted in Figure 5. The test system consists of a RS-232 UART communication module, a FIFO for temporary storage or caching/prefetching of configuration data, a reconfiguration control module, a LCD control module and the enhanced ICAP hard macro primitive, ICAP_64. The RS-232 UART module receives the configuration data over the serial link from the host PC and stores the configuration data in the FIFO. Thus, the test system only provides a low speed connection to the host PC for transmission of the reconfiguration data. The reconfiguration control module reads the configuration data from the FIFO, and writes it to the enhanced ICAP hard macro primitive, ICAP_64. The reconfiguration control module is controlled with the push buttons on the XUPV5-LX110T development board. The LCD module is used for displaying status information during operation.

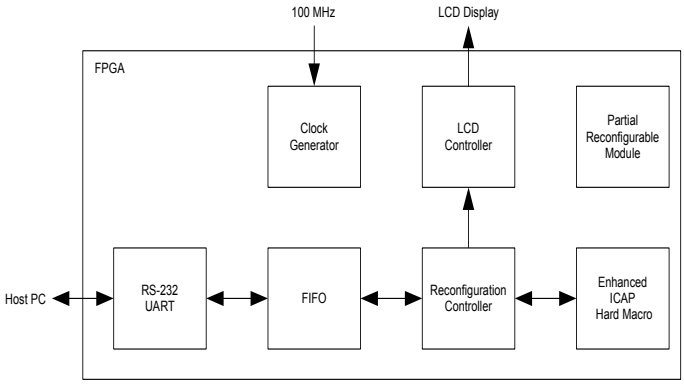

Fig. 5. Structure of the test system.

Two different clocking strategies were used to clock the test system. In the first clocking strategy, we implemented a Direct Digital Synthesizer (DDS) clock system. The DDS clock system consists of a Digital Clock Manager (DCM) module, a 32 bit phase accumulator and a Phase Locked Loop (PLL) module. Figure 6 shows the DDS clock system. The

DCM takes the input clock (100 MHz) and multiplies it by 2 to generate a 200 MHz clock output. The 200 MHz clock is used to clock the phase accumulator. The input to the phase accumulator is a 32 bit input word that can be adjusted. By increasing or decreasing the value of the input to the phase accumulator, the most significant bit of the accumulator will produce the variable frequency being synthesized. The synthesized frequency is in the range of 25 to 35 MHz. The adjustment of the input word is tuned to have a frequency change on the output clock in steps of around 50 KHz. The adjustment of the input word to the phase accumulator is controlled by two push buttons on the XUPV5-LX110T development board. The most significant bit output of the phase accumulator is used as the clock input to the PLL module. The PLL is configured to create two different clocks with a fixed ratio (8x and 16x with respect to the input clock of the PLL) between them. With this clock system (DDS) we are able to provide clocks with adjustable frequencies.

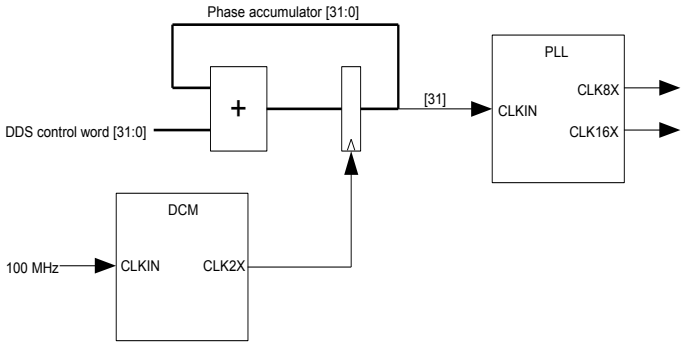

Fig. 6. DDS clock system.

The second clocking strategy, that we implemented, is based on a DCM module cascade coupled to a PLL module. The DCM module filters the input clock to reduce the clock jitter. The filtered output clock of the DCM is connected to the PLL. The PLL is configured to multiply the input clock with a fixed ratio to generate the desired clock frequencies needed and to create two different output clocks with a fixed ratio (1x and 2x) between them. With this clock system (DCM_PLL) we are able to provide clocks with fixed frequencies and a very low clock jitter. For both clocking strategies, the default 100 MHz system clock equipped on the Xilinx XUPV5-LX110T development board was used as the base system clock for our test system.

The FIFO used for temporary storage of the configuration data has a 64 bit wide write port and a 64 bit wide read port. The size of the FIFO was 64 KBytes. This limits and restricts the size of the reconfigurable modules that we can use in our experiments to only 64 KBytes.

B. Test Experiments and Verification

To test and verify that the enhanced ICAP hard macro operates correctly, the test system described in the previous section has been used to write and read partial reconfigurable modules at different locations on the FPGA fabric (i.e. different address locations in the configuration memory of the FPGA). Since we are primarily interested in verifying that the enhanced ICAP hard macro can write data to and read data from the configuration memory of the FPGA, and not that particular interested in the functionality of the partial reconfigurable module, both functional and non-functional partial reconfigurable modules were used.

Several different partial reconfigurable modules were made, where the partial reconfigurable modules contain different types of resources like BRAMs, CLBs and DSP48 blocks. For each set of resources, at least two different partial reconfigurable modules were made, such that the configuration bitstreams of each partial reconfigurable module differ from each other. For example, the initialization values of a BRAM were changed from one value to another value, or an incrementor was changed to a decrementor in a block of CLBs. The two different partial reconfigurable modules were then used to generate two different static test systems, one for each of the two different partial reconfigurable modules. The test bitstreams have been generated based on an incremental design flow, where an initial system has been extended with a partial module. This partial module has been constrained in a bounding box. The partial bitstream was generated by taking the difference between the full initial system bitstream and the extended system bitstream (with the *bitgen -r* option). The process described above was used to generate the partial bitstream files for all of the partial reconfigurable modules that we used in our experiments. With the two partial bitstream files for the partial reconfigurable module and the two configuration bitstream files for the static test system, we are able to do both reconfiguration and readback of the partial reconfigurable module in the static test system.

The readback of the partial reconfigurable module is done by setting the ICAP primitive in read mode and sending a readback command sequence to it. During readback operations the clocks to the enhanced ICAP hard macro are set to the default recommended operational frequency of 100 MHz. This is done to ensure that the configuration memory of the FPGA is read correctly. The readback operation is controlled with the reconfiguration controller.

The bitstream read from the configuration memory is sent to the RS-232 UART module and captured. Alternatively, a CRC checksum is calculated on the bitstream for the partial reconfigurable modules that is read from the configuration memory. The CRC checksums for the partial reconfigurable modules are used to verify the correct operation of the enhanced ICAP hard macro.

All tested partial reconfigurable modules where placed at several different locations on the FPGA and with various distance from the enhanced ICAP hard macro. This was achieved by manipulating the address inside the partial bitstreams. The placement of the partial reconfigurable modules was controlled by using area and location constrains. The configuration size of the tested partial reconfigurable modules varied from 1 KByte and up to 42 KBytes.

Clock System	System Clocks MHz	ICAP Frequency MHz	Reconfiguration Throughput Speed Mbits/s / MBytes/s
DDS	266, 533	533	17056 / 2132
DCM_PLL	275, 550	550	17600 / 2200

TABLE III
MEASURED MAXIMUM RECONFIGURATION SPEED FOR THE ENHANCED ICAP HARD MACRO.

IV. EXPERIMENTAL RESULTS

In this section, the experimental results obtained from experimenting with the operation frequency and overclocking the enhanced ICAP hard macro are presented and compared with previously reported results in literature.

A. Reconfiguration Speeds

Table III presents the results from measuring the maximum reconfiguration speed that has been achieved with the enhanced ICAP hard macro with the help of overclocking. For the experimental test system based on the DDS clocking strategy, the maximum operation frequency for the enhanced ICAP hard macro is 533 MHz, which gives a maximum reconfiguration throughput speed of 17056 Mbits/s or 2132 MBytes/s. The enhanced ICAP hard macro was tested and verified to operate correctly in the frequency range from 200 MHz to 533 MHz with the test system based on the DDS clocking strategy. For the experimental test system based on the cascade coupled DCM and PLL clocking strategy, the maximum operation frequency for the enhanced ICAP hard macro is 550 MHz. This gives a maximum reconfiguration throughput speed of 17600 Mbits/s or 2200 MBytes/s. The enhanced ICAP hard macro was tested and verified to operate correctly in the frequency range from 200 MHz to 550 MHz with the test system based on the cascade coupled DCM and PLL clocking strategy.

Clocking the enhanced ICAP hard macro at frequencies above 534 MHz for the test system based on the DDS clock strategy, and above 570 MHz for the test system based on the cascade coupled DCM and PLL clocking strategy, results in an instant freeze of the FPGA and subsequently reset of the FPGA by the XUPV5-LX110T development board for both test systems. For the test system based on the cascade coupled DCM and PLL clocking strategy, operating the enhanced ICAP hard macro at frequency above 555 MHz and below 570 MHz, results in an unstable and unpredictable system that sometimes is capable of doing reconfiguration only one time before it freezes. This indicates that the maximum operational frequency limit for the ICAP primitive is around 550 MHz.

In Table IV, the ICAP reconfiguration throughput speed reported and presented by different authors previously in the literature, is depicted and compared with the results obtained with our approach.

As can be seen in Table IV, our approach achieves a reconfiguration throughput of 2200 MBytes/s which is 5.5 times higher than the default reconfiguration throughput speed specified by Xilinx (400 MBytes/s). Compared with previously reported reconfiguration throughput, our approach achieves a throughput which is over 83 % higher than any other previously reported reconfiguration throughput in literature so far (1200 MBytes/s) [1]. By using our approach with the enhanced ICAP hard macro, it should be possible to reconfigure a fully featured Microblaze soft-core processor within 100 μs.

Source	Device Type	Bit Width	ICAP Frequency MHz	Measured Throughput MBytes/s	Theoretical Throughput MBytes/s
[12]	Virtex-4	32	100	350	400
[13]	Virtex-4	32	144	219.31	576
[11]	Virtex-4	32	100	371.4	400
[1]	Virtex-4	32	140	560	560
[1]	Virtex-5	32	300	1200	1200
Authors	Virtex-5	32	500	2000	2000
Authors	Virtex-5	32	533	2132	2132
Authors	Virtex-5	32	550	2200	2200

TABLE IV
RECONFIGURATION THROUGHPUT SPEED OF THE ICAP PORT.

V. CONCLUSION AND FUTURE WORK

In this paper we have presented a new design and implementation method for achieving high speed partial run-time reconfiguration. By extending the Xilinx ICAP primitive with our custom logic into an enhanced ICAP hard macro, we are able to achieve reconfiguration speed that exceeds the recommended reconfiguration speed from Xilinx by more than five times. The enhanced ICAP macro makes it possible to overclock the native ICAP primitive in a more controlled manner. The enhanced ICAP hard macro has been tested and verified to operate correctly by doing reconfiguration of different types of logic resources at different locations on the FPGA fabric. Through the experiments that have been performed, we have shown that it is possible to overclock the ICAP interface at up to 550 MHz without malfunction. At present we can achieve a maximum reconfiguration speed of up to 2200 MByte/s by using the enhanced ICAP hard macro.

A. Future Work

To achieve high speed partial run-time reconfiguration on FPGAs it is necessary to solve two main issues. The first issue is how to write the configuration bitstream to the configuration memory at a high data rate. The second issue is how to deliver the configuration data to the configuration interface on the FPGA and sustain a high speed configuration data rate. In this paper we have restricted our experiments to small reconfigurable modules that have been able to fit inside a 64 KByte FIFO just to prove our approach. To be able to take advantage of the enhanced ICAP hard macro in more useful high speed partial run-time reconfiguration systems, it is necessary to have a much better subsystem for delivering the configuration data than what we have used. The best way to achieve this is to build a partial reconfiguration memory subsystem based on Xilinx's MPMC with DDR2 memory. With the XUPV5-LX110T development board, it should be possible to implement a reconfiguration memory subsystem based on MPMC and DDR2 memory that can deliver a

memory throughput of up to 1600 MBytes/s. This is of cause not enough memory bandwidth to sustain the maximum reconfiguration speed throughput that the enhanced ICAP hard macro is capable of. To be able to reach and sustain the maximum reconfiguration speed throughput of the enhanced ICAP hard macro, it will be necessary to also apply bitstream decompression to increase the memory throughput. With a bitstream decompression, it should be possible to construct a reconfiguration memory subsystem capable of sustaining the maximum reconfiguration speed throughput of the enhanced ICAP hard macro.

We will also investigate other FPGAs, like Spartan-6 devices that are manufactured in a 45 nm process but specified to be configured with at maximum 100 MHz over a 16 bit wide configuration port (200 MBytes/s).

ACKNOWLEDGMENTS

This work is supported by the Norwegian Research Council under grant 191156V30.

REFERENCES

[1] C. Claus, R. Ahmed, F. Altenried, and W. Stechele. Towards Rapid Dynamic Partial Reconfiguration in Video-Based Driver Assistance Systems. In *Reconfigurable Computing: Architectures, Tools and Applications, 6th International Symposium, ARC 2010, Bangkok, Thailand, March 17-19, 2010. Proceedings*.

[2] C. Claus, F. H. Müller, J. Zeppenfeld, and W. Stechele. A New Framework to Accelerate Virtex-II Pro Dynamic Partial Self-reconfiguration. In *Proceedings of the IEEE 21th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–7, Long Beach, California, USA, March 2007.

[3] C. Claus, B. Zhang, M. Hübner, C. Schmutzler, J. Becker, and W. Stechele. An XDL-based Busmacro Generator for Customizable Communication Interfaces for Dynamically and Partially Reconfigurable Systems. In *Workshop on Reconfigurable Computing Education at ISVLSI*, Porto Alegre, Brazil, May 2007.

[4] F. Dittmann and S. Frank. Caching in Real-Time Reconfiguration Port Scheduling. In *Proceedings of the International Conference on Field-Programmable Logic and Applications (FPL 2007)*, pages 740 –744, 2007.

[5] S. Hauck. Configuration Prefetch for Single Context Reconfigurable Coprocessors. In *Proceedings of the 6th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 65–74, New York, NY, USA, 1998. ACM Press.

[6] M. Hübner, M. Ullman, F. Weissel, and J. Becker. Real-time Configuration Code Decompression for Dynamic FPGA Self-Reconfiguration. *11th Reconfigurable Architectures Workshop (RAW 2004), Santa Fe, New Mexico, USA*, April 2004.

[7] D. Koch, C. Beckhoff, and J. Teich. Bitstream Decompression for High Speed FPGA Configuration from Slow Memories. In *Proceedings of the 2007 International Conference on Field Programmable Technology (ICFPT 2007)*, pages 161 –168, 2007.

[8] D. Koch, C. Beckhoff, and J. Teich. ReCoBus-Builder– a Novel Tool and Technique to Build Statically and Dynamically Reconfigurable Systems for FPGAs. In *Proceedings of the International Conference on Field-Programmable Logic and Applications (FPL 2008)*, pages 119–124, Heidelberg, Germany, Sept. 2008.

[9] S. Lange and M. Middendorf. Hyperreconfigurable Architectures for Fast Run Time Reconfiguration. *Proceedings of the 12th Annual International IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2004)*, 0:304–305, 2004.

[10] Z. Li and S. Hauck. Don't Care Discovery for FPGA Configuration Compression. In *Proceedings of the 7th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA 1999)*, pages 91–98, New York, NY, USA, 1999. ACM Press.

[11] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch. Run-Time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration. In *Proceedings of the International Conference on Field-Programmable Logic and Applications (FPL 2009)*, pages 498 –502, 2009.

[12] P. Manet, D. Maufroid, L. Tosi, G. Gailliard, O. Mulertt, M. Di Ciano, J.-D. Legat, D. Aulagnier, C. Gamrat, R. Liberati, V. La Barba, P. Cuvelier, B. Rousseau, and P. Gelineau. An Evaluation of Dynamic Partial Reconfiguration for Signal and Image Processing in Professional Electronics Applications. *EURASIP Journal on Embedded Systems 2008*, November 2008.

[13] M. Shelburne, C. Patterson, P. Athanas, M. Jones, B. Martin, and R. Fong. MetaWire: Using FPGA Configuration Circuitry to Emulate a Network-on-Chip. *Computers Digital Techniques, IET*, 4(3):159 –169, May 2010.

[14] Xilinx Inc. Virtex-5 Family Overview Data Sheet, Feb. 2009. Available online: `http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf`.

[15] Xilinx Inc. Virtex-4 Family Overview Data Sheet, Aug. 2010. Available online: `http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf`.

[16] Xilinx Inc. Virtex-5 FPGA Configuration User Guide, Aug. 2010. Available online: `http://www.xilinx.com/support/documentation/user_guides/ug191.pdf`.

[17] Xilinx Inc. Virtex-6 Family Overview Data Sheet, Jan. 2010. Available online: `http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf`.

[18] Xilinx Inc. Virtex-6 FPGA Configuration User Guide, Nov. 2010. Available online: `http://www.xilinx.com/support/documentation/user_guides/ug360.pdf`.