Dynamically Reconfigurable Coprocessors in FPGA-based Embedded Systems

Ph.D. Thesis
March, 2006

Student: Ivan Gonzalez
Director: Franciso J. Gomez

Ivan.Gonzalez@uam.es

Agenda

- Motivation and Thesis Goal
- Dynamic Partial Reconfiguration
- Cryptographic Coprocessors in FPGA
- Cryptographic Systems based on FPGA
- Self-Reconfigurable Systems
- Conclusions and Future Work

Motivation

- Embedded systems are currently an integral part of many communications devices
  - Actual communication applications require new capabilities
    - Pervasiveness
    - Mobility
  - Data processing in real time
    - Reach the transmission rates of the communication links
    - Guarantee a high security level is a remarkable challenge
- FPGA-based SoC solutions improve the versatility and performance of traditional embedded systems.

Thesis Goal

- *Develop FPGA-based embedded systems that make use of partial reconfiguration.*
  - Cryptographic coprocessors in FPGA.
  - CSoCs based on FPGA-embedded processors.
  - Self-reconfigurable CSoCs.
  - An application that makes use of the self-reconfigurable capabilities.

Dynamic Partial Reconfiguration

Reconfiguration Basics

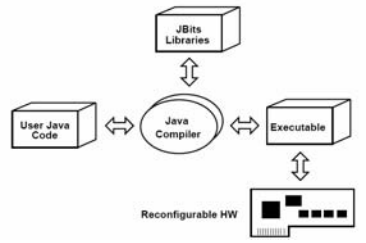
- Starting from 1998 Virtex family, all Xilinx FPGAs are partially reconfigurable
- Partial reconfiguration
  - Small parts of the design (a frame in Virtex) can be reconfigured without affecting the rest of the device.
  - *Dynamic Partial Reconfiguration:* the device is partially reconfigured while the rest of the device continues working.
  - *Self-Reconfiguration:* the device can control its own reconfiguration.

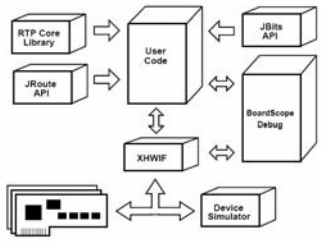
Tools for Reconfiguration

- Partial Reconfiguration is poorly supported at both design tools and documentation levels
- Research tools
  - Like JBits, PARBIT and others
- Xilinx-recommended, via bitgen –r:
  - Altering small portions of the FPGA configuration like LUTs or memories using FPGA Editor
  - Working with Modular Design to create reconfigurable areas
  - Plenty of bugs

JBits

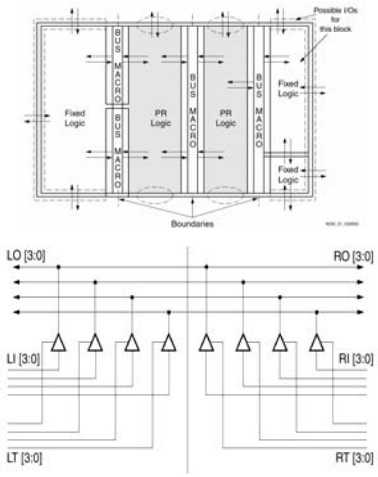
- Java API
- Dynamic design and modification of FPGA circuits in Virtex / Virtex-II families.
- Support for partial reconfiguration
  - Low level bitstream modification.
  - Automatically infers the differences between bitstreams
    - Reconfigure only the frames that have changed





Module-Based Design Flow

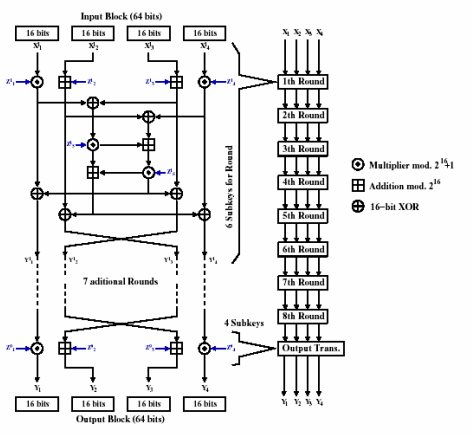
- It is a design flow that permits building the final FPGA layout from partial layouts
- Bus Macro: It is used to implement the routing between reconfigurable areas
- *bitgen –r* is used to create the bitstream that changes from one design to the other



Cryptographic Coprocessors in FPGA

IDEA Algorithm

- Symmetric algorithm.
- Encrypt 64-bit blocks with a 128-bit key, using three simple operations:
  - or-exclusive module 2^{16}
  - addition module 2^{16}
  - multiplication module $2^{16}+1$
- 8 similar stages, called rounds
- One stage called output transformation.



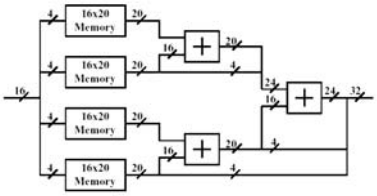
Design methodology

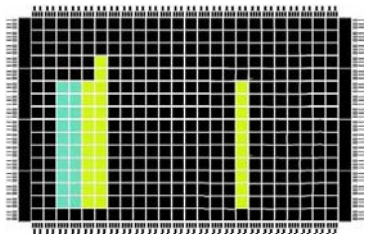
- Replace all the operational units involving the key with their constant-operand equivalents
  - Addition module 2^{16}
  - Multiplication module $2^{16}+1$
- Fundamentals
  - Arithmetic and logic operators by constant are faster and occupy less area
  - Changing of the key using partial reconfiguration
- This methodology will be useful when
  - the modifications in the circuit are easy to perform
  - the reconfiguration time remains acceptable

IDEA using JBits
Multiplier module $2^{16}+1$

- Low-High Algorithm

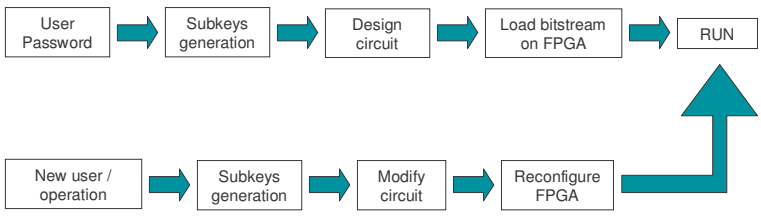
```
uint16 modmult(uint16 operand) {
   word32 p;
   uint16 c, d;

   if(k) {
      if(operand) {
         p = (word32)k * operand;
         c = low16(p);
         d = p >> 16;
         return c - d + (c < d);
      }
      else return 1 - k;
   } else return 1 - operand;
}
```





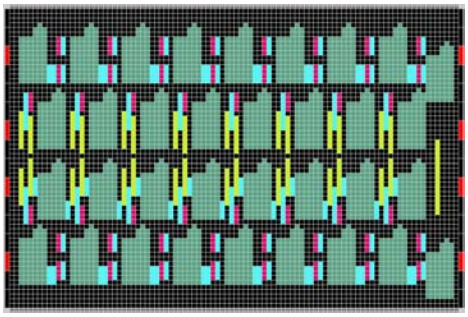
IDEA using JBits
Implementation

- Reconfigurable Platform
  - RC1000PP Celoxica
- Hardware/Software Procedure



IDEA using JBits Results

- Area (XCV1000)
  - Combinational design: 2708 CLBs (44%)
  - Segmented design 16 stages 3156 CLBs (51%)



- Speed
  - 20 MHz multiplier module $2^{16}+1$
    - Combinational IDEA: 0.8 MHz
    - Segmented IDEA: near 20 MHz

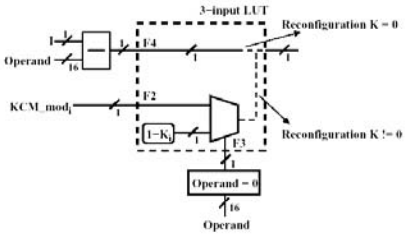
Custom Methodology

- Design with JBits is too difficult for complex circuits.
  - Partial Reconfiguration is well-supported in JBits
- New methodology
  - Improve the performance.
  - The circuit can be created using traditional tools, for example VHDL.
  - The changes must involve only the contents of the LUTs
  - The partial reconfiguration tool selected is JBits.

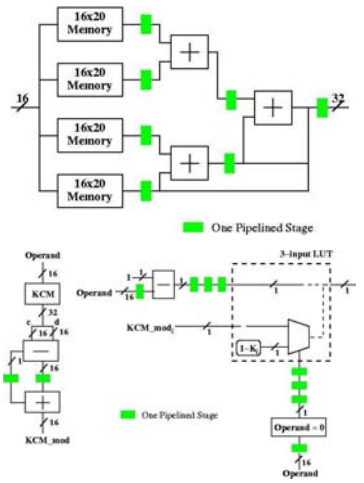
Custom Methodology
Multiplier module $2^{16}+1$

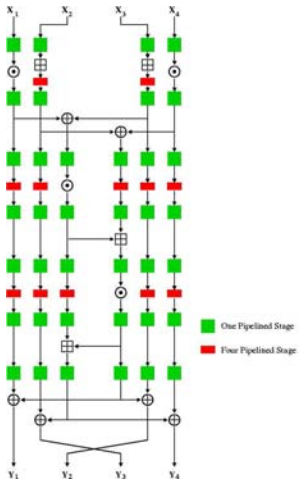
- New design:

```
uint16 modmult(uint16 operand) {
   word32 p;
   uint16 c, d;

   if(k) {
      if(operand) {
         p = (word32)k * operand;
         c = low16(p);
         d = p >> 16;
         return c - d + (c < d);
      }
      else return 1 - k;
   } else return 1 - operand;
}
```



Custom Methodology
Pipelined Algorithm





IDEA & Partial Reconfiguration

Implementation	Slices	Device	Frequency	Throughput
Combinational-JBits	2708	44% XCV1000-6	0.8 MHz	0.5 GBits/sec
Segmented-JBits	3156	51% XCV1000-6	20 MHz	1.2 GBits/sec
New Methodology	6078	87% XCV600-6	131.1 MHz	8.3 GBits/sec
Virtex-II Pipelined-VHDL	4573 (34 Mult.)	44% XC2V2000-6	140.6 MHz	9.0 GBits/sec
New Methodology Virtex-II	6128	79% XC2V1500-6	151.7 MHz	9.7 GBits/sec
Deeply Pipelined [1]	9052	73% XCV1000-6	105.9 MHz	6.8 GBits/sec
Cheung et al. [2] 2 round cores	2444	79% XCV300-6	82 MHz	1.2 GBits/sec
Combinational-VHDL	6863	55% XCV1000-6	1.4 MHz	0.8 GBits/sec
Pipelined-VHDL	7410	60% XCV1000-6	24.5 MHz	1.5 GBits/sec

[1] Antti Hämäläin et al.: "*6.78 Gigabits per Second Implementation of the IDEA Cryptographic Algorithm*", Proc. FPL 2002, pp. 760-769, Montpellier, France, September 2002.

[2] O.Y.H. Cheung, K.H.T. Soi, P.H.W. Leong, and M.P. Leong, "Tradeoffs in Parallel and Serial Implementations of the International Data Encryption Algorithm IDEA", *Proceedings of CHES*, pp. 333-347, Paris, 2001

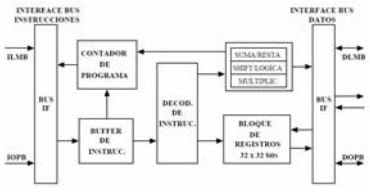
Cryptographic Systems based on FPGA

The Question: Soft or Hard Processors?

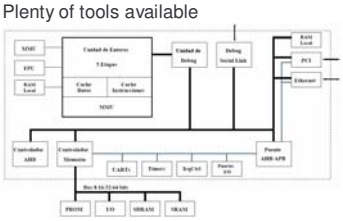
- A soft core is just a netlist that is implemented using the general-purpose resources of the FPGA
  - MicroBlaze, LEON, OpenRISC, NIOS, etc.
- A hard core is one embedded in the FPGA silicon
  - Altera Excalibur (ARM)
  - Xilinx Virtex-II Pro or Virtex-4 FX (PowerPCs)
- FPGA-based systems
  - Soft Core Processors MicroBlaze and LEON2
  - Hard Core Processor PowerPC

Soft Cores: MicroBlaze and LEON2

- Embedded soft RISC Processor (32-bit)
- HW Support for instructions
  - Mult, BS, Div, FPU
- Optional and parameterizable instruction cache and data cache
- Plenty of tools available

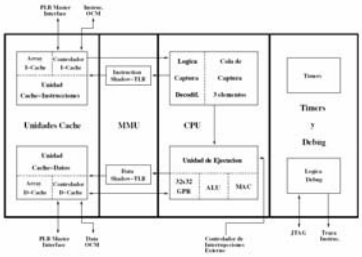


- Synthesisable VHDL model of a 32-bit SPARC V8 processor
- Integer unit with 5-stage pipeline
  - Hardware multiply, divide and MAC units
  - FPU and custom co-processors
  - Advanced on-chip debug unit
  - MMU optional
- Plenty of tools available



Hard-Core: PowerPC 405

- The IBM PowerPC 405 core is integrated into Xilinx Virtex-II Pro devices.
- 32-bit implementation of the PowerPC architecture.
- Features:
  - 5- stage data path pipeline
  - Hardware multiply and divide units
  - 32 x 32-bit general purpose registers
  - 16 KB 2-way set-associative instruction and data cache
  - Memory Management Unit (MMU)



- The PowerPC 405 core also supports IBM CoreConnect bus architecture

Cryptographic Systems based on FPGA

Ciphering Algorithms in FPGA based Embedded Systems

Ciphering Algorithms Experiments

- This work is focused in IDEA, DES, 3DES, Blowfish and AES symmetric-key algorithms and MD5 and SHA-1 hash algorithms.

Algorithm	XOR AND OR	Add Sub Mod.	Fixed Shift	MULT Mod.	MULT GF Const.	LUT	Internal Block
DES/3DES	•		•			6-to-4	32
IDEA	•	2^{16}		$2^{16}+1$			16
BLOWFISH	•	2^{32}				8-to-32	32
AES	•		•		$GF(2^8)$	8-to-8	32
MD5	•	2^{32}	•				32
SHA-1	•	2^{32}	•				32

- Measure the execution time consumed by the ciphering process.
  - Software Implementation
  - Custom Hardware

MicroBlaze-based Systems

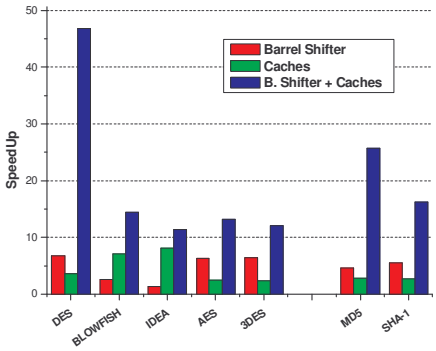
- Several architectures are proposed to take advantage of the parameterizable architectural features of MicroBlaze.
- Dedicated multiplication instruction is not available.

50 MHz MBlaze-E

Architecture	Barrel Shifter	Mult	Divisor	Cache	FPGA Slices	BRAM Blocks
MBlaze-A					1102	32
MBlaze-B	•				1213	32
MBlaze-C				•	1216	74
MBlaze-D	•			•	1321	74
MBlaze-E	•		•	•	1385	74

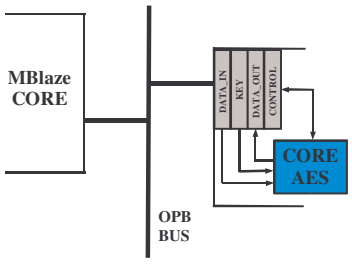
MicroBlaze Performance

- The different internal features increased the performance of the algorithms
  - C language implementation of the algorithms
  - Depend of the operational units involved in the algorithm
  - B-Shifter is the best feature
  - Division is not useful

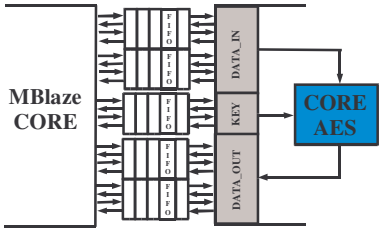


Coprocessors on MicroBlaze

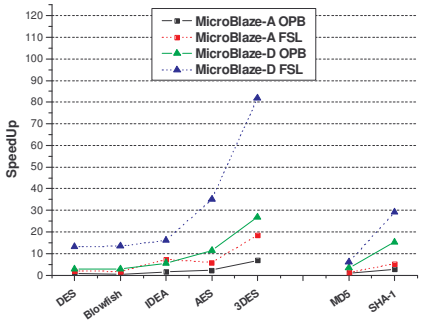
- The first alternative is to map them as a peripheral
  - In MicroBlaze, OPB bus will be used
  - Memory-mapped interface
    (Shared-memory communication)



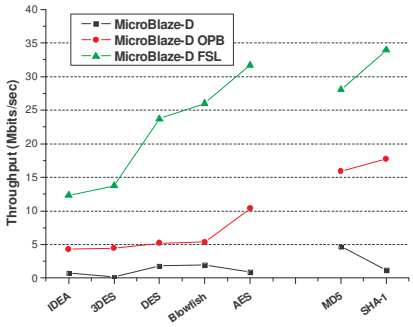
- The second alternative is to use it as a coprocessor
  - The processor must have a dedicated link
  - In MicroBlaze, this link is FSL



Cryptographic Coprocessors on MicroBlaze



Improvement Internal Features



Final Performance

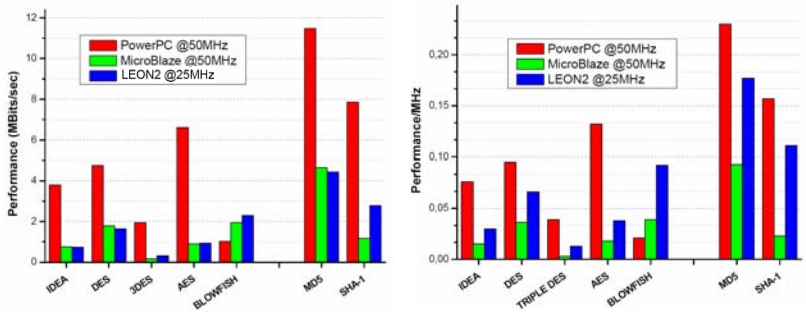
Ciphering Algorithms on LEON-based Systems

- LEON2 is configured similar than MicroBlaze-D
  - Cache organization scheme (Direct-Mapped) with an 8-Kbyte data and instruction cache
  - The hardware multiplier and the hardware division units are not used
  - The remainder features are used by default
- To connect ciphering cores, the LEON2 Custom Coprocessor Interface is evaluated
  - After de OPB results, the APB bus is not a well-suited interface

PowerPC 405 Systems

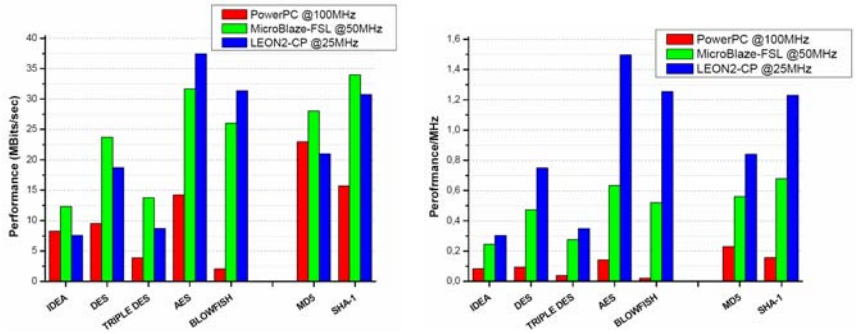
- Two PowerPC systems were implemented with similar configuration than the MicroBlaze system
  - SDRAM memory is accessed using a 64-bit data bus (PLB)
    - 32-bit data bus used in MicroBlaze and LEON2
  - Two systems working at
    - 50 MHz to compare with SCP systems without HW cores
    - 100 MHz to compare with SCP systems using HW cores
- In all cases, the internal features available in PowerPC were used.

Pure-Software Results



Comparison of performance and performance/MHz using a C implementation of the algorithms

Coprocessor-based Results



Comparison of performance and performance/MHz using Custom Coprocessors

Self-Reconfigurable Systems

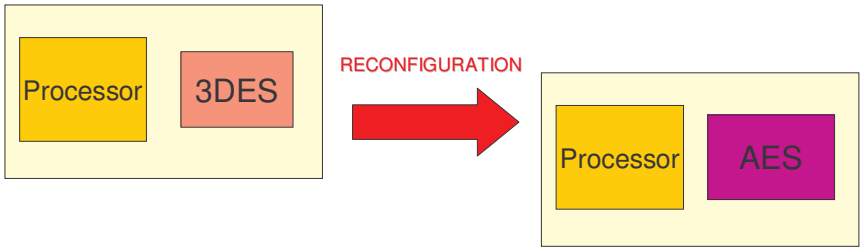
Hardware-Accelerated SSH

Coprocessing Drawbacks

- Two main problems:
  - Lack of versatility
  - Area occupied
- The flexibility of pure software is lost
  - Part of the algorithm is being executed in a fixed hardware
  - It is not possible to add new tasks at run time
  - The system cannot be updated
- Huge area requirements
  - When many HW-accelerated algorithms need to be supported
  - A general-purpose coprocessor might be used, but the performance will not be the same
    - Example: FP coprocessor vs dedicated FFT algorithm

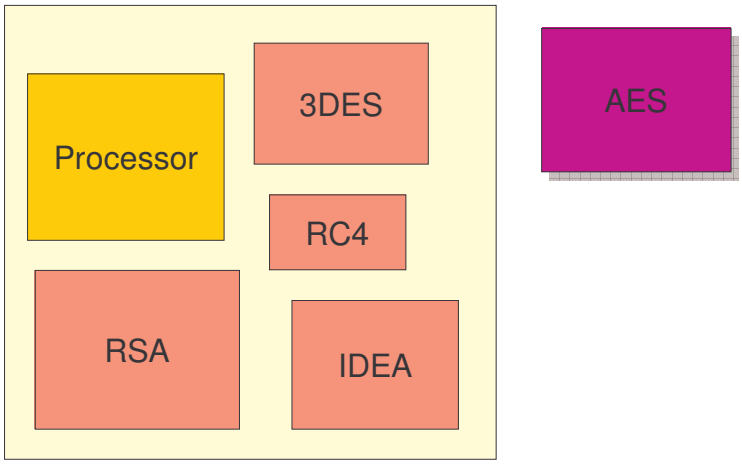
The Solution: Reconfigurable Coprocessors

- The FPGA is partially reconfigured to change the coprocessor it uses:



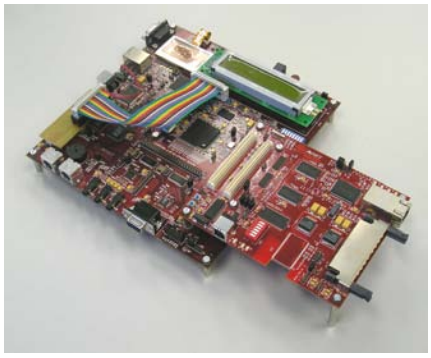
- Run-time reconfiguration, the processor keeps running
- It solves both the versatility and area problems

Hardware-Accelerated SSH



The Challenge

- Develop a hardware-accelerated version of SSH
  - Implement reconfigurable coprocessors for cryptography
- MicroBlaze is the selected SCP
  - FSL coprocessors
- The development platform will be Avnet's Spartan-3 board

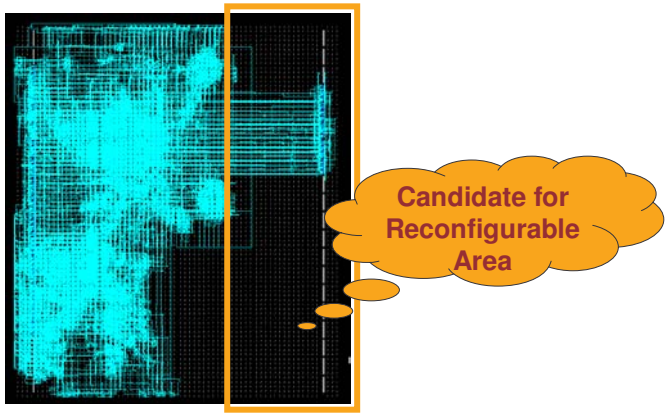


Spartan-3 Limitations

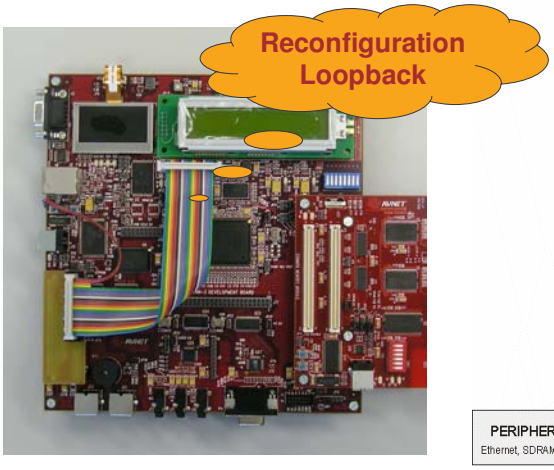
- The minimal reconfiguration unit is a whole CLB column
  - The reconfigurable area must be comprised of whole CLB columns, from top to bottom of the device
- There are no bus macros for Spartan-3
- Self-Reconfiguration requires an internal access to the configuration port.
  - Such access is the ICAP, but it is only available in Virtex-II and Virtex-4

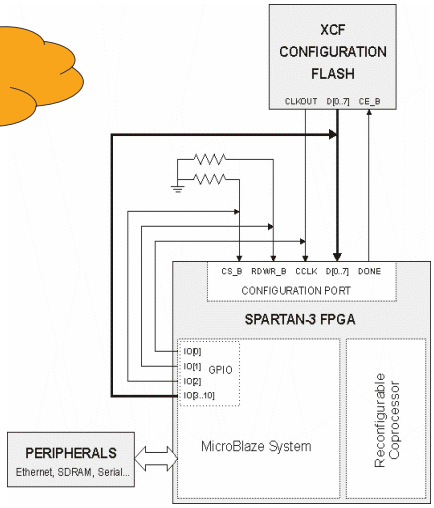
Choosing the Reconfigurable Area

- The FPGA pinout, there must be no used pins in the area dedicated to the reconfigurable coprocessor
  - The area must be pinless



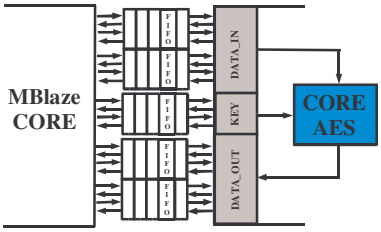
Solving Self-Reconfiguration External Connection



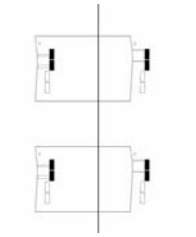


Dedicated Macros

- Instead creating generic macros, like the Bus Macro, we have created 3 different macros
  - One for each interface

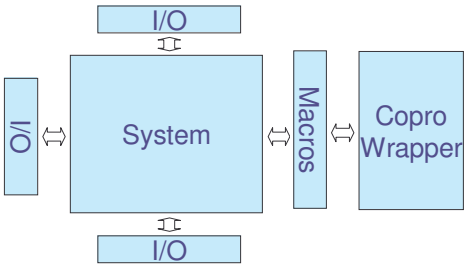


- LUT-based macros
  - The left slice is in the fixed area, the right one is in the reconfigurable area



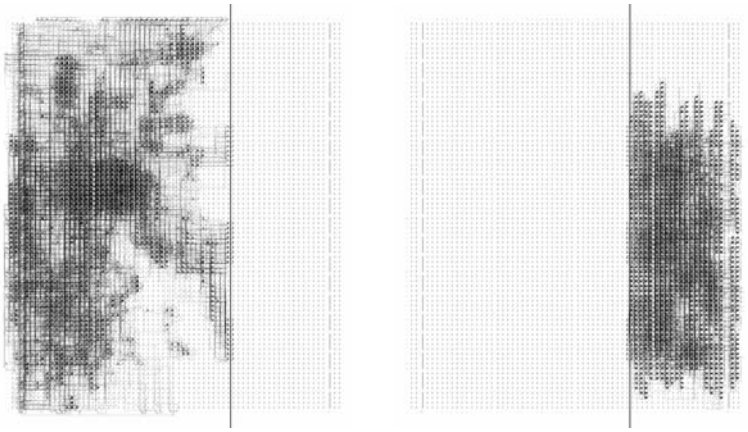
Creating the Design
Initial Budgeting Phase

- It is not possible to create the design in EDK
- A new EDK core has been developed to bring the FSL busses to the design pins
- The coprocessor and the macros have to be added afterwards



Creating the Design
Active Module Phase

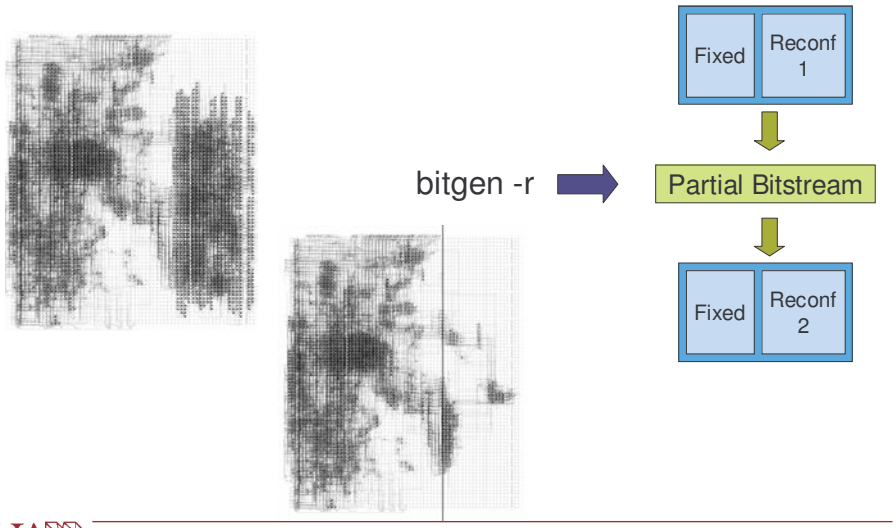
- The results of this stage of Modular Design are the actual layouts:



Creating the Design
Final Assembly Phase

- In theory, the module layouts are assemble to obtain the final layout
- But in practice, it fails due to bugs in the modular design flow
  - Plenty of Answer Records on Xilinx's website, none provides a satisfactory solution
  - Only simple designs can be finished
- But apparently the partial layouts have been correctly generated
  - The solution: develop our own tool to merge the layouts

Example of Final Layout

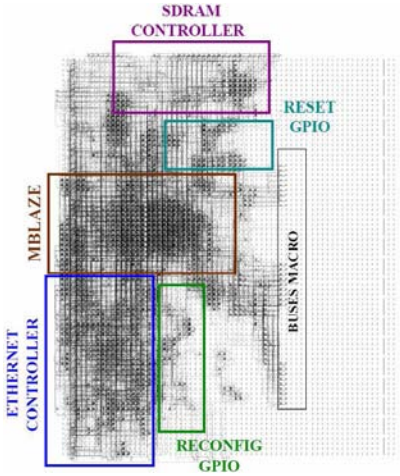


SSH on uCLinux

- The OpenSSH open source implementation of SSH available on uCLinux distribution has been used.
  - SSH permits secure login connections and file transfers over the Internet or other entrusted networks.
  - Cryptographic algorithms:
    - SSH1: DES, 3DES, IDEA, Blowfish and RSA
      - CRC for data integrity protection.
    - SSH2 : AES (Rinjdael), Twofish, CAST and DSA
      - HMAC-MD5 or HMAC-SHA for data integrity protection.
  - uCLinux is a port of regular Linux to microprocessors lacking a memory management unit

SSH on uCLinux

- Self-reconfigurable MicroBlaze-uCLinux System on Spartan-3
- Two symmetric-key algorithms were selected:
  - Default AES128-CBC
  - Optional 3DES-CBC.
- Partial bitstreams are available in a NFS server



SSH on uCLinux Results

- Implementation Results

	FPGA Slices	LUTs	FF /Latches	18x18 Mult	BRAM
MicroBlaze System	4198	4809	3618	3	20
AES Coproc.	4326	4632	1802		
3DES Coproc.	5424	4492	5825		

- Reconfiguration time: 166 ms
- Evaluation
  - Copy a set of files through secure channel.
  - Bottleneck:
    - FTP shows better performance
    - Not transfer bandwidth problem
    - **MD5 algorithm in SW**

Protocol	Algorithm	File 1 335 KB	File 2 2410 KB	File 3 8266 KB
SSH in Software	AES	27.8	48.1	52.7
	3DES	16.7	26.6	30.3
SSH in Hardware	AES	33.2	60.4	66.4
	3DES	32.0	60.4	67.7
FTP	None	877.3	696.1	683.2

Throughput in Kbytes/sec.

Conclusions

- CSoC that include processors and reconfigurable logic are good solutions to speed up embedded software
  - Critical parts of the code can be implemented as application-specific coprocessors.
  - Good examples of this methodology are cryptography applications
- Self-reconfiguration
  - The area savings are remarkable because only the coprocessor for the currently running algorithm has to be loaded in the FPGA.
- Hardware-accelerated SSH
  - Allow us to validate the feasibility of using self-reconfiguration to accelerate secure communication protocols.

Conclusions and Future Work

Conclusions

- **Dynamic partial reconfiguration of cores**
  - Reduce the logic resources employed.
    - Up to 20% of reduction
    - Allow to remove parts of the algorithms.
  - Increase the throughput.
    - IDEA: 8.3 GBits/sec in Virtex and 9.7 GBits/sec in Virtex-II
  - The presented results show the potential of partial reconfiguration to optimise hardware cores.

Conclusions

- **Dynamically reconfigurable systems**
  - The high flexibility offered by SCPs is combined with run-time partial reconfiguration capabilities to develop CSoC solutions
  - Ability to reconfigure the coprocessor unit depending on the task being executed in the processor.
  - A self-reconfigurable platform based on a commercial low-cost FPGA
    - Versatility and Upgradeability (also it solves the area problems)

Conclusions

- **Final conclusion of this thesis**
  - Partial reconfiguration of FPGAs has been successfully evaluated
    - Dynamically reconfigurable cores and Dynamically reconfigurable systems can be combined.
    - Solution to improve the lack of potential and flexibility of actual embedded systems.
  - The complexity of using partial reconfiguration is the main drawback to adopt this technology.
  - The results obtained in this thesis allow to be optimistic
    - Most probably in a few years partial reconfiguration will be used in future electronic systems.

Future Work

- **Secure applications**
- **Operative systems for reconfigurable hardware**
- **Multiprocessor systems on FPGA**
- **Pervasive computing**
- **Modular robotics**

Dynamically Reconfigurable Coprocessors in FPGA-based Embedded Systems

Ph.D. Thesis
March, 2006

Student: Ivan Gonzalez
Director: Franciso J. Gomez

Ivan.Gonzalez@uam.es