CS6814 **Research Seminar on Reconfigurable Hardware** Fall 2003

Prof. John W. Lockwood

http://www.arl.wustl.edu/~lockwood/class/cs6814/

Review of:

A Self-reconfiguring Platform

•Paper by:
- Brandon Bloget, Scott McMillan, Prasanna Sundararajan (Xilinx, San Jose, CA)
- Philip James-Roxby, Eric Keller (Xilinx, Longmount, CO)

•Published in:
- FPL
- Lisbon, Portugal
- September 1-3, 2003

•Review by:
- Chris Neely

Brandon Blodget

Introduction

- Motivation for the work
  - Create a general purpose tool for:
    - "Fast reconfiguration"
    - "Bitstream manipulation"
    - "An infrastructure to make on-chip reconfiguration easier to use"
    - "Intelligent control of reconfiguration via an embedded processor"
    - "C based API to update FPGA resources"
  - Who could benefit
    - Any application that could benefit from SR (self-reconfiguration) (using the Virtex II/II Pro)
    - MGRs, high density crossbars (Young et al.), embedded operating systems for FPGAs

Style of Writing

- Original work: Presents the first detailed description of their SR platform.
- Evaluates two implementations of SR platform (current and planned)
  - Both methods use ICAP specific to Virtex II/II Pro architectures
  - Performance comparison
- Motivates self-reconfiguration
  - Short survey of previous work

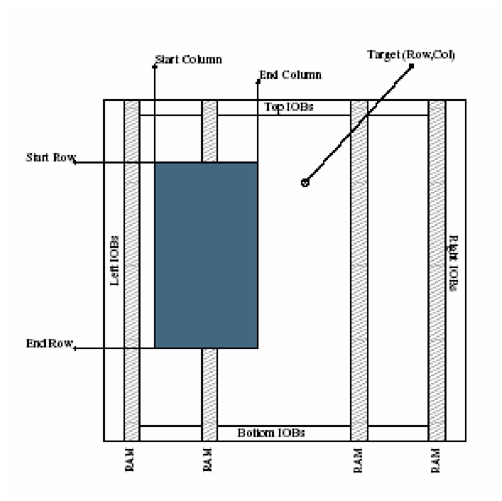
Research Approach

- Use experience from prior work → propose a SR platform specific to Virtex II
- Lightweight, layered infrastructure
  - Development and Testability
    - Layers separate HW dependent details
    - Each layer is light → easier to use
  - Performance
    - C rather than Java
    - Avoids using external datapaths
  - Features
    - Reconfigure individual frames
    - Embedded controller provides flexibility (compressed, encrypted bitstreams)
- Benefits from FPL
  - "No external configuration cache required"
- Disadvantages from FPL
  - "Slower. Must do a read first."
  - "SRL16s and LUT RAMs can cause problems."

Types of Reconfiguration

- Full
  - All devices require a full configuration of the chip resources at start time
- Partial
  - Configuration of a subset of the chip resources
- Dynamic
  - Configuration of a subset of the resources by an external device, while rest of chip maintains correct operation
- Self-reconfiguration
  - Configuration of a subset of the resources by another subset of the chip resources, while rest of chip maintains correct operation
  - How much is practical?

Frames_Sz: Smallest Reconfiguration Unit



(Virtex-E figure from DHP DAC'02 paper)

Virtex I/II/II Pro require a pad frame after configuration data to flush reconfiguration pipeline.

Frame Sizes

Q. How much configuration data needs to be sent to the device to make the smallest possible change?
A. This varies with the frame size, it requires 384 bits + 1 frame + 1 dummy frame for a write operation. For a XCV300 this would be 1784 bits for a single frame write.

Q. What is the fastest time in which a change can be performed?
A. Using the previous example and the SelectMAP port at 50MHz it would require
1784/8*20ns =4.32uS.

Device	Frame Size	Time (50 MHz)*
XC2V40	104 bytes	6us
XC2V500	344 bytes	16us
XC2V2000	584 bytes	26us
XC2V6000	984 bytes	41us
XC2VP7	424 bytes	20us
XC2VP70	1064 bytes	45us

(table from FPL slide)

*includes header + data frame + pad frame

Virtex II architecture: SelectMap vs. ICAP

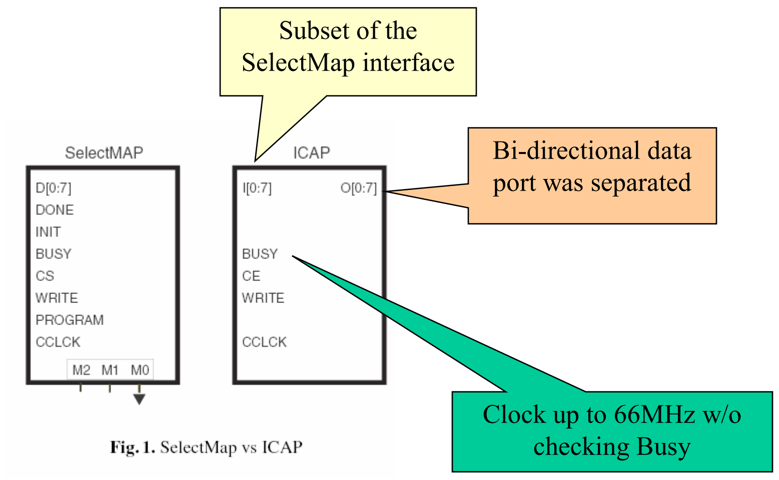
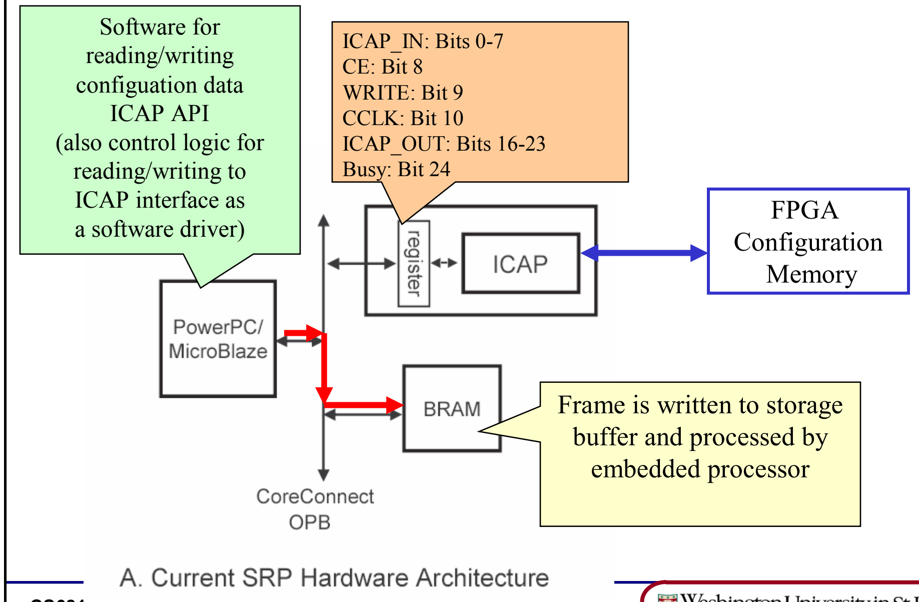

Subset of the SelectMap interface
Bi-directional data port was separated
Clock up to 66MHz w/o checking Busy
SelectMAP
D[0:7]
DONE
INIT
BUSY
CS
WRITE
PROGRAM
CCLCK
M2 M1 M0
ICAP
I[0:7]
O[0:7]
BUSY
CE
WRITE
CCLCK

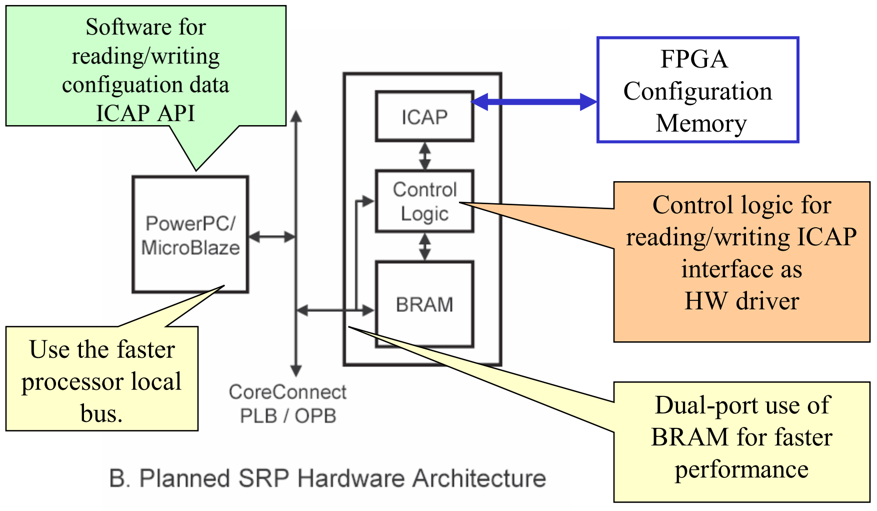
Fig. 1. SelectMap vs ICAP

Current SRP Hardware Architecture


Software for reading/writing configuation data ICAP API (also control logic for reading/writing to ICAP interface as a software driver)
ICAP_IN: Bits 0-7
CE: Bit 8
WRITE: Bit 9
CCLK: Bit 10
ICAP_OUT: Bits 16-23
Busy: Bit 24
register
ICAP
FPGA Configuration Memory
PowerPC/ MicroBlaze
BRAM
Frame is written to storage buffer and processed by embedded processor
CoreConnect OPB

A. Current SRP Hardware Architecture

Next Gen. (Planned) SRP Hardware Arch.


Software for reading/writing configuation data ICAP API
PowerPC/ MicroBlaze
Use the faster processor local bus.
ICAP
Control Logic
BRAM
FPGA Configuration Memory
Control logic for reading/writing ICAP interface as HW driver
Dual-port use of BRAM for faster performance
CoreConnect PLB / OPB

B. Planned SRP Hardware Architecture

SRP Software Layers

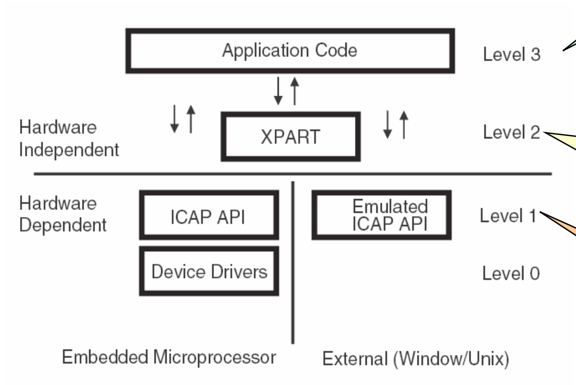


Fig. 3. SRP Software Layers

i.e. Install new accel. function

- Bitstream abstraction
- Re-locatable modules

Read/write frames

XPART Example Methods

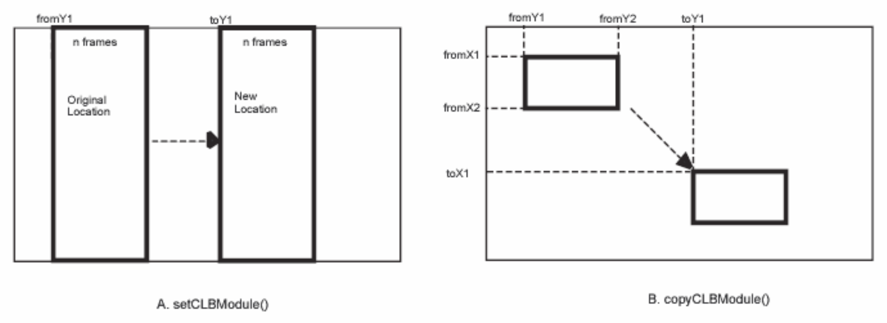


Fig. 5. Illustration of Module Methods

Software Layers (cont'd)

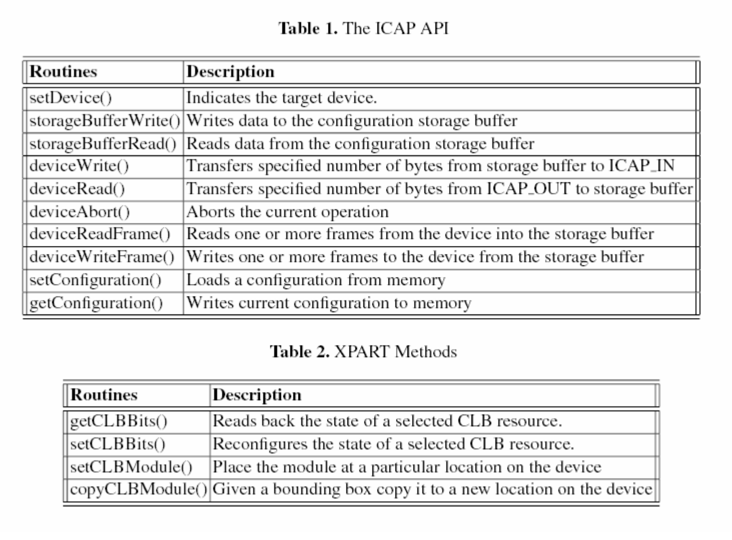


Table 1. The ICAP API

Routines	Description
setDevice()	Indicates the target device.
storageBufferWrite()	Writes data to the configuration storage buffer
storageBufferRead()	Reads data from the configuration storage buffer
deviceWrite()	Transfers specified number of bytes from storage buffer to ICAP_IN
deviceRead()	Transfers specified number of bytes from ICAP_OUT to storage buffer
deviceAbort()	Aborts the current operation
deviceReadFrame()	Reads one or more frames from the device into the storage buffer
deviceWriteFrame()	Writes one or more frames to the device from the storage buffer
setConfiguration()	Loads a configuration from memory
getConfiguration()	Writes current configuration to memory

Table 2. XPART Methods

Routines	Description
getCLBBits()	Reads back the state of a selected CLB resource.
setCLBBits()	Reconfigures the state of a selected CLB resource.
setCLBModule()	Place the module at a particular location on the device
copyCLBModule()	Given a bounding box copy it to a new location on the device

Read/modify/write Configuration Data

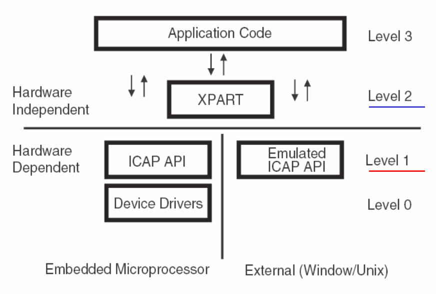


Fig. 3. SRP Software Layers

XPART::setCLBBits() {

1. Calculate target frame
2. Find LUT bits in target frame
3. Read target frame from device and put in storage buffer—deviceReadFrame().
4. Modify the LUT bits in the storage buffer—writeStorageBuffer()
5. Write new LUT bits (reconfigures device)—deviceWriteFrame()

}

Read/modify/write Configuration Data

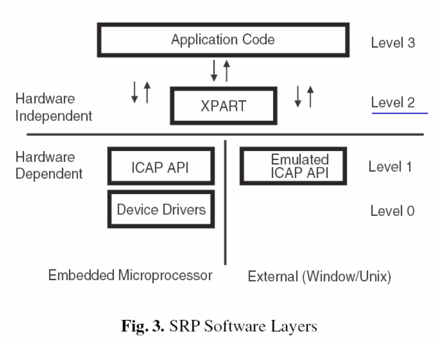


Fig. 3. SRP Software Layers

XPART::setCLBModule() {

1. Generate partial reconfiguration packet containing module
2. Modify packet to a different frame address header

}

C code example: Update LUT

```
#include <XPART.h>
#include <LUT.h> /* Bitstream resource library for LUTs */

int main(int argc, char *args[]) {
  char* value;
  int error, i, row, col, slice;
  setDevice(XC2VP7); // Set the device type
  ... [initialize row,col,slice and value to desired values] ...
  error = setCLBBits(row, col, LUT.RES[slice][LE_F_LUT], value,16);
  return error;
} /* end main() */
```

Reconfig. 1 LUT Comparison from Paper

- Current SRP (SW driver)
  - Virtex II, XCV2V1000
  - Read,write,modify:10 ms.
  - MicroBlaze core @50 MHz
  - ICAP handshake @50MHz
- Planned SRP (HW driver)
  - Virtex II, XCV2V1000
  - Read,write,modify: 26us
  - Processor core not specified
  - ICAP handshake@66 MHz
  - Speedup: ~400X

Performance Comparison from FPL

- System1
  - XC2VP7 – CPU@50MHz, OPB @100 MHz
  - BRAM read/write: 1.5 us
  - Frame read/write: 27us
- System2
  - XC2VP7 – CPU@300MHz, OPB @100 MHz
  - BRAM read/write: 600 ns
  - Frame read/write:15us
  - Speedup: ~2X

Implementation Details from FPL

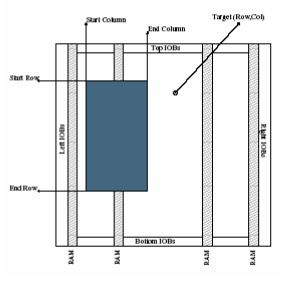
- Size
  - OPB SRP Peripheral – 125 Slices 2% of XC2VP7
  - Lightweight SRP Peripheral – 26 slices
  - SRP library (ICAP API + XPART) - ~12KB
    - Bit files for various resources

Related Work

- Our work:
    - Taylor, Turner, Lockwood: "Dynamic hardware plugins (DHP): exploiting reconfigurable hardware for high-performance programmable routers" (IEEE Open Network Programming Proceedings, 2001)
    - Horta, Lockwood, Taylor, Parlour: "Dynamic hardware plugins in an FPGA with partial run-time reconfiguration" (DAC02)
    - Horta, Lockwood: "PARBIT: a tool to transform bitfiles to implement partial reconfiguration of FPGAs" (WUCS Tech. Report, 2001)

Compare to our PARBIT and DHPs work

- XPART is similar to PARBIT:
  - Arbitrary block regions of a compiled design can be re-targeted into any similar size region of the FPGA
- DHPs work examines interface between the DHP plugins and the infrastructure
  - Gaskets – static route interface between dynamic regions and infrastructure logic
- Their solution requires Virtex II arch. Virtex-E would require using SelectMap interface and external configuration datapath.



MGTs, Xilinx App. Note 662

- Available online at:
    - http://direct.xilinx.com/bvdocs/appnotes/xapp662.pdf
  - Virtex-II Pro devices contain RocketIO MGTs
    - Number of MGTs between four and 20
    - Create highspeed serial links between devices
    - Up to 3.125 Gb/s per channel
  - The MGTs
    - Four levels of pre-emphasis
    - Five levels of differential swing control
    - Selectable by RocketIO primitive attributes.
  - These attributes can be modified to optimize the transmission of high-speed serial signals
    - Initially set in the bitstream by BitGen
    - Reconfigurable using ICAP

Other Related Work

- Dynamic Reconfiguration:
    - Compton, K., Hauck, S. Reconfigurable computing: A survey of systems and software (ACM Comp. Surv.)
- Self-reconfiguration:
    - Formal definition
        - Sidhu and Prasanna, Efficient metacomputation using self-reconfiguration (FPL02)
    - Case study
        - McGregor, G. Lysaght, P: Self controlling dynamic reconfiguration: A case study (FPL99)

CS6814: Fall 2003 Washington University in St.Louis

Other Related Work

- Applications improved by self-reconfiguration
    - First Self-Reconfiguration Work
        - French., P.C., Taylor, R.W.: A self-reconfiguring processor. In: IEEE Symposium (FCCM93).
    - Example application
        - Sidhu, Mei, Prasanna, V.K: Genetic programming using self-reconfigurable FPGAs (FPL99)
- Flexible URISC
    - Donlin, A.: Self modifying circuitry- a platform for tractable virual circuitry (FPL 98)
- High IO Reconfigurable Crossbar Switch
    - Young, S., Alfke, P., Fewer, C., McMillan, S., Blodget, B., Levi, D.: “A high i/o reconfigurable crossbar switch.” (FCCM03)

CS6814: Fall 2003 Washington University in St.Louis

Your Conclusions

- Summary: Authors were successful
  - Created simple platform, multiple layers accessible for debugging, exploits new architecture features
  - The SR platform moves frame operations (programming circuit) within the device, allows sending different stream formats
  - XPART repeats functionality of PARBIT and JBitsDiff, but it fits well in their layered infrastructure
  - Optimized platform for speed afterwards
  - Re-presented work in dynamic and SR, sited appropriate work

Additional References:

- Xilinx Partial Reconfiguration FAQ:
  - http://www.xilinx.com/xilinxonline/partreconfaq.htm