

Review of:
**Sequencing of Run-Time Reconfigured Hardware
with Software**

- **Paper by:**
 - Michael J. Wirthlin - Brigham Young University
 - Brad L. Hutchings – Brigham Young University
- **Presented at:**
 - FPGAs 1996: Monterey, California
 - February 11 - 13
- **Survey by:**
 - Todd Sproull

The Challenge

- **Demonstrate a Run-Time Reconfigured (RTR) system in hardware**
 - Most RTR systems are difficult to build
 - Reconfiguration times are high
- **Demonstrate the ease of software development**
 - ‘C’ like syntax available
 - Retargetable compiler allows for development of new instructions

Style of the Paper

- A Demonstration of DISC Hardware/Software
 - Demo of Image Processing Application
 - Image Filtering
 - Image Threshold
 - Image Thinning
 - Gives a few performance numbers
 - Benchmarks performed against a 66MHz 486
 - Describes its reconfiguration abilities
 - Describes how to partially reconfigure the FPGA
 - Discuss time needed to reconfigure

Runtime Reconfiguration -RTR

- Goal
 - To provide additional hardware resources for systems based on FPGAs
- Previous RTR work
 - Neural network systems developed that required very few FPGAs
 - Wireless video coding system constructed using FPGAs to be reconfigured during the image coding stages
- Limitations
 - Few tools available to break up design into equally sized partitions
 - Run-time configuration is slow
 - Some degree of communication is needed between partitions

Introduction to DISC

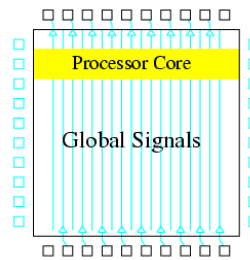
- DISC – Dynamic Instruction Set Processor
 - A RTR processor designed with FPGAs
 - Developed to avoid the previously mentioned problems with RTR
- Advantages
 - Allows for a user-defined application-specific instruction set
 - Instructions are designed in hardware
 - Application specific functions are developed into instruction modules
 - These modules are called in software

DISC Architecture

- DISC is divided into two main components
 - Processor Core
 - Handles Instruction Sequencing
 - Remains static on hardware during execution
 - Custom Instruction Space
 - Reserved for hardware instruction modules
 - Potentially reconfigured many times during execution

Processor Core

- Simple Design
 - Sequences program instructions
 - Provides Interface for external memory
 - Handles instruction swapping
 - Intermodule communication
- Global Accumulator register and state signals
 - Signals run across entire chip
 - Made available to all modules

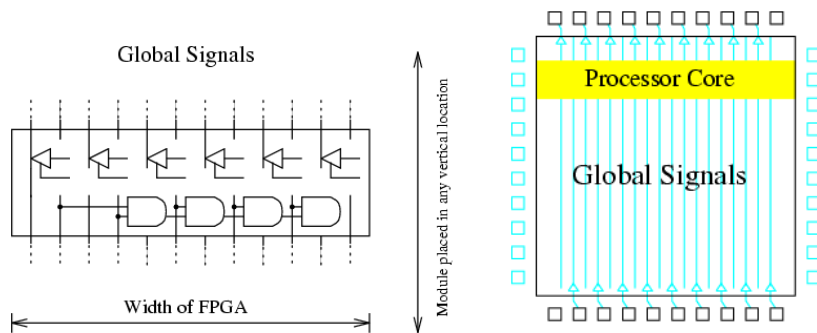


Custom Instruction Space

- Consists of custom instruction modules
 - Allowed to be placed anywhere within the FPGA
 - Location determined at run-time, not at design time
- Modules must be designed in a global context
 - Instructions must appear the same at every location
 - All modules are connected to the global signals
 - Makes interchanging modules very simple

Location of DISC Instructions

- DISC Instruction Module placed horizontally across entire FPGA
 - Allows access to global signals
 - Modules can be as “tall” as necessary

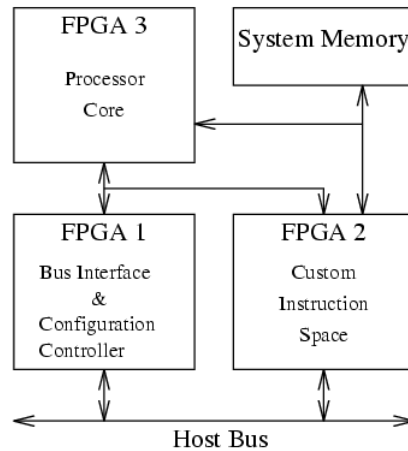


DISC Limitations

- Some problems exist with DISC architecture
 - Designed with wirewrapped prototype board
 - Single FPGA left little room for large instruction modules
 - 8-bit core provided very few built-in instructions and had a limited address range

DISC-II

- Addressed many of the problems with DISC
 - Developed on a FPGA-based board by National Semiconductor
 - Contained 3 FPGAs
 - Dedicated host interface
 - Additional memory



DISC Design Environment

- Designs take place in both Hardware and Software
 - Application Specific Instructions designed in hardware
 - Instructions are sequenced by a software program

Hardware Design

- Handles parts of the application that can exploit
 - Circuit-level parallelism
 - Custom I/O interfaces
 - Hardware Control
- Hardware modules designed under global constraints
 - Structural modules are used to ensure that the module conforms to the global context protocol
- Once a module is designed it may be used in any application program

Software Design

- Organizes the application into specific control sequences
 - Sequences vary from assembly language to complex C program
- To support this custom instruction set a retargetable assembler is used
 - Allows users to add and use new instructions as they are developed
- New instructions are typically defined by
 - Instruction mnemonic, type, opcode, and control parameters

Image Processing Application

- Created a image processing library for DISC
- 3 main operations
 - Pre-filtering
 - Thresholding
 - Region Thinning
- Source Code

```
main () {  
    image *image1, *image2;  
    histogram *hist;  
    int thresh;  
    int skel;  
  
    image2 = native_lowpass(image1);  
    image1 = native_clear(image2);  
  
    ...  
    for( skel = 0; ske != 0; ) {  
        skel= native_skeleton(image1, image2);  
        skel = native_skelton2(image2, image1);  
    }  
}
```



Prefilter

- Apply a low-pass filter to remove high-frequency noise
Image2 = native_lowpass(image1);



Threshold

- Converts grey-scale image into simple binary image
 - Places objects in foreground and background based on its image intensity

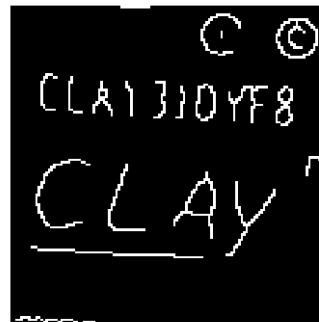
```
Image2=native_lowpass(image1);  
Image1=native_clear(image2);  
Hist = native_histogram(image2);  
Thresh = peakthresh(hist);  
Image1 = native_threshold(thresh, image2);
```



Object Thinning

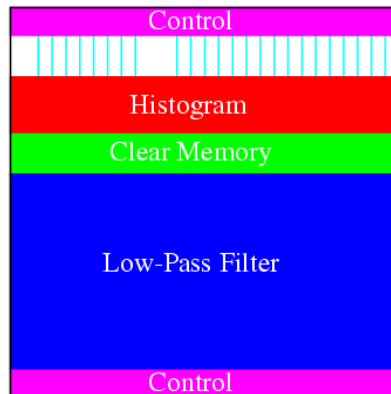
- Reduces redundant shape information from image to simplify object recognition process
 - Single pixel lines to represent objects are often called the “skeleton”

```
For(skel=0; skel!=0;){  
  skel = native_skeleton1(image1,image2);  
  skel += native_skeleton2(image2,image1);  
}
```



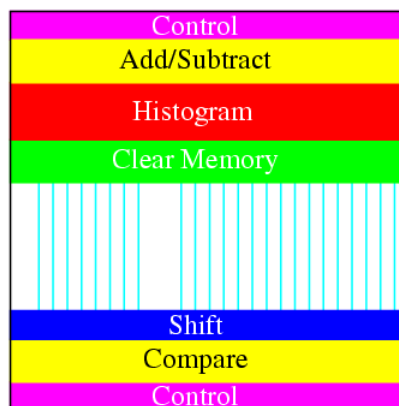
State of DISC during Processing

- After histogram DISC contains



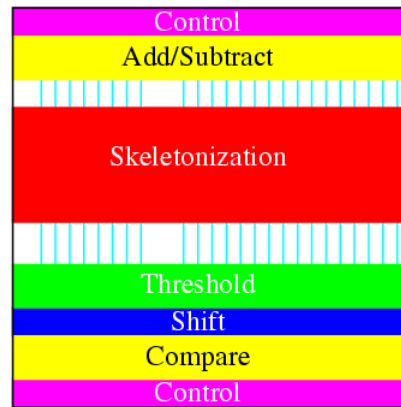
State of DISC during Processing

- After C Code DISC contains



State of DISC during Processing

- After skeletonization DISC contains



Performance

- Performance vs a 66 MHz 486 PC

IMAGE	486	DISC	Speedup
Silk screen	2.17s	.29s	6.5
Block letter	3.90s	.53s	6.4
News print	9.90s	.89s	10.1

Conclusion

- Demonstrates advantages of software sequencing support for run-time reconfigured hardware
- Provides infrastructure for reusable hardware modules
- Development of RTR applications is simplified
 - Resources are easily reused
 - DISC resources are automatically reclaimed as necessary