Ain Shams University
Faculty of Engineering
Computer & Systems Engineering Department

Configurable Application Specific Microprocessors

Thesis

Submitted in partial fulfillment of the Requirements for the M.Sc. Degree
in Electrical Engineering
(Computer and Systems)

By

Nahla Ahmed Mohamed Ahmed El-Araby
B.Sc in Computer and Systems Engineering ٢٠٠٢

Supervised By

Dr. Hassan Shehata Bedor
Computer & Systems Engineering Dept.
Faculty of Engineering, Ain Shams University

Dr. Ayman Mohamed Wahba
Computer & Systems Engineering Dept.
Faculty of Engineering, Ain Shams University

Dr. Mohamed Shaalan
Computer & Systems Engineering Dept.
Faculty of Engineering, Ain Shams University

Cairo ٢٠٠٧

Acknowledgements

Thank ALLAH at first then I'm thanking my supervisors **Dr. Hassan Shehata, Dr. Ayman Wahba.**

I am really so grateful and appreciating the great support and encouragement of **Dr. Ayman Wahba,** who helped me so much, taught me many things. He was my support and guide in this study.
Also I would like to thank **Prof.Dr. Shawki Eid** for his support and encouragement.

And I must thank my great, loving, caring and sweet parents as they are the owners of any success in my life.

ABSTRACT

Name: Nahla Ahmed Mohamed Ahmed El-Araby.

Thesis title: Configurable Application Specific microprocessors.

A thesis for M.Sc. degree. Ain Shams University, Faculty of Engineering, Computer and Systems Engineering Department, ٢٠٠٧.

The limitation of using Field Programmable Gate Arrays (FPGAs) as a platform for application specific processors is the lack of hardware resources available for specialized instruction sets. With the new FPGA's it is now possible to reconfigure the hardware dynamically *on-the-fly*. Partial reconfiguration allows configuring a section of the FPGA while the remaining logic is not affected.

A Dynamic Instruction Set Computer (DISC) processor uses Run Time Reconfiguration (RTR) to overcome FPGA hardware limitations and provide an essentially limitless application-specific instruction set.

The aim of this work is to use the idea of DISC to implement a processor which serves more than one application domain; saving area and decreasing configuration time. In this work an Automatic DISC processor is implemented; it automatically detects the type of running application and reconfigures itself to add the required circuitry needed by the running application. The design consists of a processor core and a partially reconfigurable module that can be changed according to the running application. A controller module is used to test the available hardware and download appropriate partial reconfiguration bit stream to update the processor hardware according to the running application. The processor

can run four groups of applications: Image Processing, Digital Signal Processing, Advanced Mathematical Operations, and Encryption applications.
Key Words: Dynamic reconfiguration, DISC, Appliacation-specific processors.

TABLE OF CONTENTS

Title **Page No.**

LIST OF TABLES

LIST OF FIGURES

List of Abbreviations

ASIC	Application Specific Integrated Circuit
BC	Bit Correlation
CPLD	Complex Programmable Logic Device
DISC	Dynamic Instruction Set Computer
FF	Flip Flop
FPGA	Field Programmable Gate Array
GPP	General Purpose Processor
LUT	Look Up Table
MPGA	Mask Programmable Gate Array
PAL	Programmable Array Logic
PLA	Programmable Logic Array
PROM	Programmable Read Only Memory
RTR	Run Time Reconfiguration
SPLD	Simple Programmable Logic Device
SRAM	Static Random Access Memory

Chapter ١: Introduction

Application Specific Integrated Circuits (ASICs) are very fast and efficient when executing the exact computation for which they were designed. This is due to the operators are tailored to the required function, and there is direct wire connection between operators. However, after fabrication the circuit cannot be changed. If any part of the circuit needs to be changed, the ASIC must be redesigned, and refabricated, which is not an easy task; it costs too much and increases time-to-market.

On the other hand General purpose processors are very flexible; they support a variety of problems. The same processor can run different applications. The down side of this flexibility is that the performance can suffer. The processor must read each instruction from memory, decode its meaning, and then adjust the functional units in order to execute it. This result in a high execution overhead for each operation, so the execution using general purpose processors is slower. Another draw back is area usage; too much area is needed to store instructions, data, and control execution. So the area usage is inefficient.

So General Purpose Processors delivers higher flexibility while custom – application specific – circuits offers higher efficiency.

Building Application specific or customized stored-program processors that are designed to run a specific type of applications can combine the enhanced performance of application-specific circuits with the flexibility of general purpose processors.

Configurable hardware especially Field Programmable Gate Arrays (FPGAs) offer a fast development time because they can be tested and reprogrammed in field, also they provide a simplified design process so they represent an excellent platform for application specific processors.

But one limitation of building application specific processors on FPGAs is the lack of hardware resources available for specialized instruction sets. Specialized instructions require too much hardware so a few hardware intensive instruction modules can quickly consume all the resources of even the largest FPGAs available today.

The emergence of high capacity reconfigurable devices opens new horizons in the field of general-purpose processing. With the new FPGA's it is now possible to reconfigure the hardware dynamically *on-the-fly* to tailor its functional units and interconnect according to the target application. Those new FPGAs can be partially reconfigured during run time. Many areas, such as cryptography, signal processing, and searching, require this capability. An increase of ١٠x – ١٠٠x in the functional density can be achieved, with reduced latency compared with conventional processor solutions [١][٢].

Run-Time Reconfiguration (RTR) allows system designers to virtually have more hardware than they actually have on a given chip. This works very well especially when there are parts of the hardware that are occasionally idle. For example, one application is a smart portable phone that supports multiple communication and data protocols, but only one protocol is active at a time. When the phone passes from a geographic region that is served by one protocol into a region that is served by

another protocol, the hardware is automatically reconfigured. Using this approach, it is possible to design systems that do more, cost less, and have shorter design and implementation cycles.

Another application that demands real-time computation is the automatic target recognition. The challenge addressed by these systems [٣] is the rapid comparison (correlation) of an input image to thousands of target templates with large number of pixels per image. The correlation of incoming images with the existing target templates is the computational bottleneck in the system, involving data rates and computational requirements that exceed, by several orders of magnitude, the processing load in any other steps in the algorithm. Correlation may be accelerated, if we have a hardware specially designed for each type of the available target templates.

Run-Time Partial Reconfiguration allows configuring a section of the FPGA while the remaining logic is not affected, which means that only the part that needs to be changed is being reconfigured while the rest of the design is still functioning, so the system state can be saved on the FPGA during configuration. The configuration time is reduced dramatically as only the requested part is configured, not the whole FPGA.

A Dynamic Instruction Set Computer (DISC) processor uses RTR to overcome FPGA hardware limitations and provide an essentially limitless application-specific instruction set.

In [٤] attempts in modifying a processor instruction set involved a writable control store and generating custom micro-code for each application.

The PRISM project [٥] extended this idea by augmenting the instruction set of a standard RISC processor with application-specific instructions on a tightly coupled FPGA. Hardware images of these instructions are extracted and compiled from the source code transparent to the user.

The WASMII project discusses a more dynamic approach that involves swapping hardware compute configurations in and out of the FPGA resource as demanded by the data-flow token [٦].

The work in [٧] presents a design of a DISC processor, the processor implements each instruction in the instruction set as an independent circuit module. The individual instruction modules are paged onto the hardware in a demand-driven manner as dictated by the application program. Hardware limitations are eliminated by replacing unused instruction modules with usable instructions at run-time. An application running on DISC contains source code, indicating instruction ordering, and a library of application-specific instruction circuit modules.

In [٨] a model of a runtime reconfigurable processor based on partial and dynamic reconfiguration is presented. This processor implements the ARM Thumb Instruction Set Architecture as a pipelined superscalar microarchitecture. A variable number of execution units which are dynamically updated according to the behavior of the program at runtime. Hence, no additional software tools are needed, so that compatibility with software and hardware systems based on the ARM Thumb ISA is assured. The functional units which are not reconfigured (Instruction Memory, Trace Cache, Fetch Unit/Predecoder, Decoder, RUU, Register File and Configuration Manager) build up a fixed module. Sets of