

Development of i960 Processor Models to Support Hardware/Software Codesign

George Powley
powley@ee.eng.ohio-state.edu

Ludovit Minjarik
minjarik@ee.eng.ohio-state.edu

Joanne DeGroat
degroat@ee.eng.ohio-state.edu

Department of Electrical Engineering
2015 Neil Avenue
The Ohio State University
Columbus, OH 43210

Abstract

The purpose of our research is to create two VHDL models of the i960 MX processor. These models will be used in the development of an embedded system running Ada programs on multiple i960 MX processors. The system designer will choose which model to use based on the amount of information required from the simulation. The two models produced will be a high-level model and a low-level model.

The low-level model will be an instruction level model of the i960 MX. It will provide information about the state of the processor every clock cycle. For this embedded system development, the low-level model will require the Ada code to be compiled to i960 executable form. This model will provide detailed information about the processor, but it will take a long time to simulate.

The high-level model will be used when the execution time is more important than the detail of the processor performance. For the high-level model, Ada code will be compiled to run on the host workstation. Links to VHDL code will provide i960 MX functions not supported by the workstation and the connection to other VHDL models. The combination of the host workstation and the high-level model will provide a “virtual” i960 MX, which will require less time to execute the Ada code.

This presentation will describe the modeling techniques used for the low-level model. A plan for the development of the high-level model will also be discussed.

Development of i960 Processor Models to Support Hardware/Software Codesign

October 19, 1992

George Powley
Dept of Electrical Engineering
The Ohio State University
powley@ee.eng.ohio-state.edu

Ludovit Minjarik
Dept of Electrical Engineering
The Ohio State University
minjarik@ee.eng.ohio-state.edu

Joanne DeGroat
Dept of Electrical Engineering
The Ohio State University
degroat@ee.eng.ohio-state.edu

Work is being done under US Air Force P.O.# A-054670
and supported in part by
NFS Grant 9112130
and
Vantage Analysis Systems, Inc.



DARL Design Automation Research Laboratory
Department of Electrical Engineering
The Ohio State University

1

Presentation Outline

- Purpose of Research Project
- Application Software Development
- Application Hardware Model
 - Low-Level Hardware Model
 - High-Level Hardware Model



DARL Design Automation Research Laboratory
Department of Electrical Engineering
The Ohio State University

2

Purpose of Research Project

- Develop a VHDL model of the i960[†] MX processor to be used in the design of a large embedded system running Ada code
- Model will provide means to verify hardware and software designs with embedded i960 MX processors
- Two models provide two levels of abstraction
 - Low-Level Model
 - * Provides instruction level model of the i960 MX
 - * More detailed model → long simulation times
 - High-Level Model
 - * Uses the host workstation and VHDL code to provide a "virtual" i960 MX
 - * Not as much detail → shorter simulation times

[†]i960 is a registered trademark of Intel Corporation



Design Automation Research Laboratory
Department of Electrical Engineering
The Ohio State University

3

Application Software Development

- The Low-Level model of the i960 MX requires i960 machine code
- An i960 MX Ada compiler was not available, so we need a method to generate i960 machine code from Ada code
- The High-Level model will use the STYX[†] VHDL - "C" interface, so the Ada code must be converted to "C" code
- A solution to both problems has been proposed

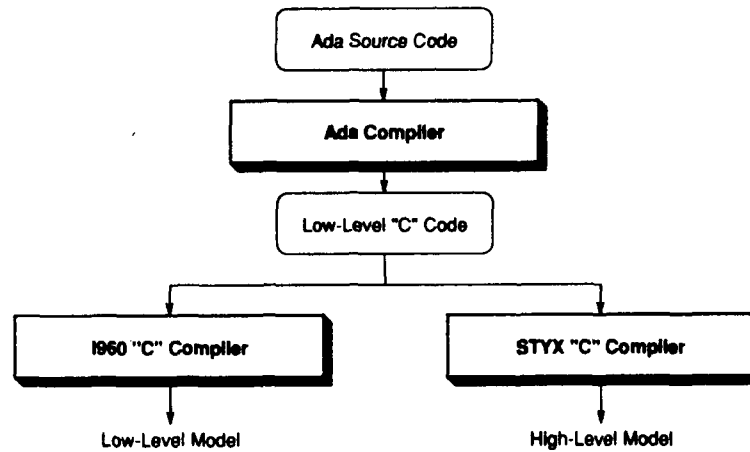
[†]STYX is a registered trademark of Vantage Analysis Systems



Design Automation Research Laboratory
Department of Electrical Engineering
The Ohio State University

4

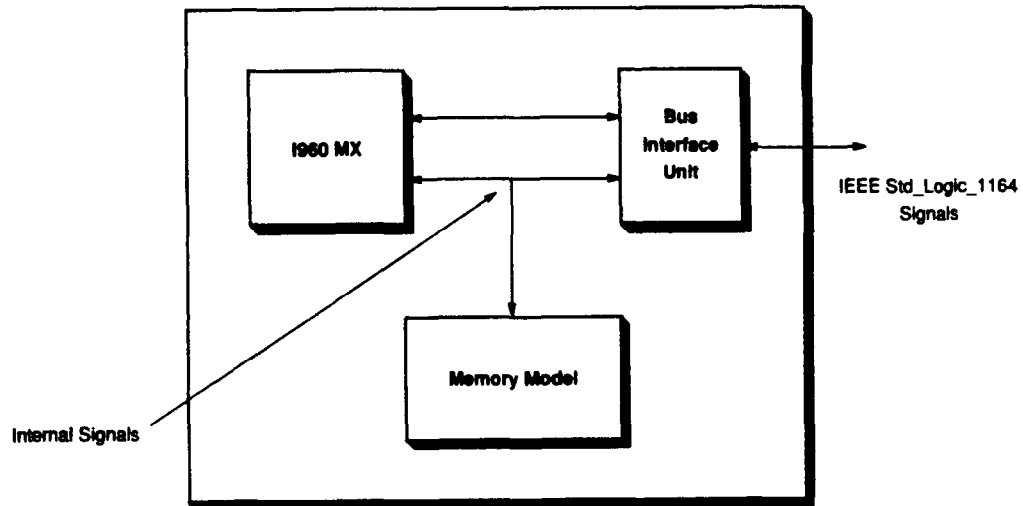
Ada to i960 Code



Low-Level Hardware Model

- The Low-level model will include the i960 MX processor model, a 4 GB memory model and a bus interface model
- This set of models will be referred to as an i960 Board
- Multiple i960 Boards can be used in the system development, each with its own application program
- The i960 Board will use IEEE Std.Logic_1164 to communicate with other VHDL models

i960 Board Block Diagram



i960 Board

- There are three entities on the i960 Board
 - i960 MX
 - Memory Model
 - Bus Interface Unit
- These entities communicate with each other via internal signals
- The Bus Interface Unit will be used to convert signals between the internal signal representation and IEEE Std_Logic_1164



Internal Signal Representation

- To represent a 32 bit word as a std_logicvector, VHDL requires 32 signals
- We defined an internal type that will allow us to represent a resolved 32 bit word with 4 signals

```
SUBTYPE wordhi IS integer RANGE 0 TO 2**16;
SUBTYPE wordlo IS integer RANGE 0 TO 2**16;

TYPE bus_state IS ( HIZ,           -- Hi Impedance
                   DRIVEN,        -- 1 Source Driving
                   ERROR);        -- More than 1 Source Driving

TYPE uwordbus IS
RECORD
  hi      : wordhi;
  lo      : wordlo;
  strength : bus_state;
END RECORD;
```

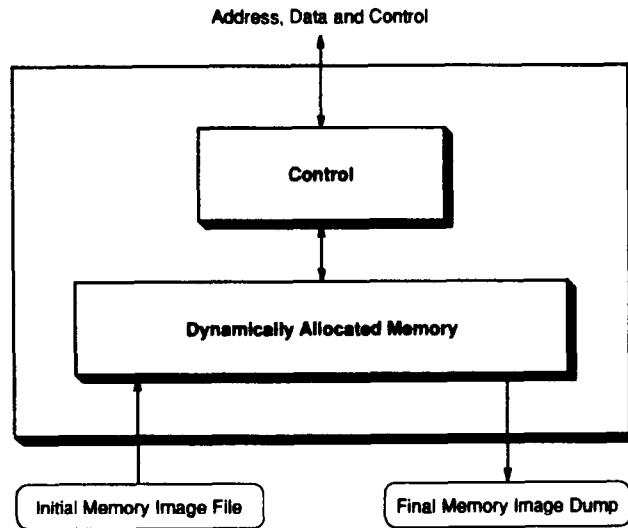


Internal Signal Representation (cont.)

- The 32 bit word is divided into two parts
 - word.hi = most significant 16 bits
 - word.lo = least significant 16 bits
- When a source is driving a signal, word.strength is set to DRIVEN
- A resolution function determines the number of signal drivers, and sets word.strength = ERROR if more than 1 source is driving the bus at a time
- This signal representation has many advantages
 - The number of signals is greatly reduced
 - Integer math can be performed on the 32 bit word (overload operators)
 - Type can be changed to represent any word length



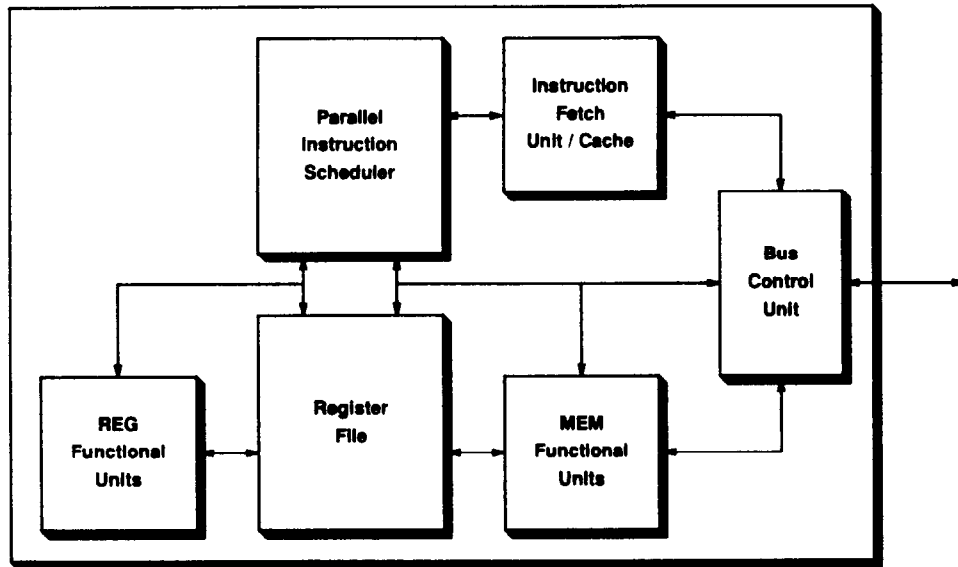
Memory Block Diagram



Memory Model

- The i960 MX has can address 2^{32} bytes of memory (4 GB)
- The memory model must support the entire address space
- A dynamic memory model is used
 - The memory is divided into 2^{20} 4 K pages
 - A page is allocated when the processor writes to that page
 - If the processor tries to read from an unallocated memory page, a predefined value is returned (ex. 0xFFFFFFFF)
- The memory model stores the memory as an array of type word
 - Type word is the same as type wordbus, without the strength
 - Allows 32 bits of memory to be stored as 2 integers
- The initial memory image is file programmable
- The memory model provides a dump of the memory at the end of the simulation

i960MX Block Diagram



i960 MX

- Some features of the i960 MX processor
 - Large register set (16 global, 16 local and 4 floating-point)
 - On-chip frame cache (8 sets of 16 registers)
 - 2K internal data cache
 - 2K instruction cache
 - 64 entry translation lookaside buffer
 - On-chip floating-point unit
 - On-chip memory management unit
 - Object-based memory organization
 - Three-fold protection system
- The i960 also supports faults, interrupts, multitasking and multiprocessing

Instruction Fetch Unit / Instruction Scheduler

- Instruction Fetch Unit / Instruction Cache
 - The fetch unit and cache are contained in one entity
 - This entity receives instruction requests from the scheduler
 - A fetch request is issued to the Bus Control Unit if the instructions are not in the cache
- Instruction Scheduler
 - The scheduler decodes and issues all instructions
 - Micro-flow instructions are coded in the scheduler, to reduce signal traffic during micro-flow execution
 - The scheduler calculates all branches, and supports branch prediction



Register File

- The multiport register file allows simultaneous access by different functional units
- The register scoreboard is part of the register file
 - Scoreboard is used to allow out-of-order instruction execution
 - If scoreboard bit is set for an operand or destination register, the instruction cannot execute
- Our model of the register file includes the frame cache
- The contents of the register file are dumped to a file at the end of simulation



Register-Side Functional Units

- The register-side functional units include
 - Execution Unit (EU)
 - Multiply Divide Unit (MDU)
 - Floating-Point Add Unit (FPA)
 - Floating-Point Multiply Unit (FPM)
- The EU and MDU perform all integer arithmetic, logical instructions and compare instructions
- The FPA unit performs adds for 32, 64 and 80 bit floating-point numbers
- The FPM unit performs multiplies and divides on 32, 64 and 80 bit floating-point numbers
- The STYX interface to "C" is used to perform transcendental instructions with 32 and 64 bit floating-point numbers



Memory-Side Functional Units

- The memory-side functional units include
 - Address Generation Unit (AGU)
 - Translation Lookaside Buffer (TLB)
 - Data Cache
 - Bus Control Unit (BCU)
- These units work together to provide on-chip memory management
- AGU and BCU models exist for the physical addressing mode
- Models for the virtual memory mode are under development



Future Research Goals

- Complete Low-Level Model of i960
- Setup environment to generate i960 machine code from Ada code
- Investigate High-Level Model of i960

