An FPGA-powered, Pipelined Microprocessor Based Solution to Ubiquitous Computing for Next Generation Personal Devices

1st Haoyan Qi
School of Computer Science
Faculty of Engineering (of the University of Sydney)
Sydney, Australia
haqi6150@uni.sydney.edu.au

2nd Ive Zhang
School of Electrical and Information Engineering
Faculty of Engineering (of the University of Sydney)
Sydney, Australia
szha8538@uni.sydney.edu.au

3rd Lucas English
School of Electrical and Information Engineering
Faculty of Engineering (of the University of Sydney)
Sydney, Australia
leng5171@uni.sydney.edu.au

4th Eric Zhao
School of Electrical and Information Engineering
Faculty of Engineering (of the University of Sydney)
Sydney, Australia
yzha2452@uni.sydney.edu.au

***Abstract*—The demand of ubiquitous computing devices coincides with the ongoing global development and success in research into Field Programmable Gate Arrays (FPGAs). We propose a unique 16 bit, pipelined microprocessor developed on a Terasic DE1-SoC Development Kit to bridge these two fields.**

***Index Terms*—FPGA, ubiquitous computing, microprocessor**

I. Introduction

We have constructed a state-of-the-art microprocessor based on FPGA technology, which utilizes our custom instruction set dubbed *Y68*. The processor's opcodes are 4 bytes and the addressing space allows for 64KB of virtual memory.

II. Datapath

- The Instruction Pointer holds a 16 bit address in virtual memory to an instruction.
- The two Instruction Registers hold a 32 bit instruction to be decoded.
- The Decoder and Control Unit are strongly coupled and provide the functionality of decoding 32 bit instructions and outputting the corresponding signals to: the ALU, the ALU Buffer, the Accumulator, the General Purpose Registers, the Instruction Pointer, the two Instruction Registers, the RAM, the MMU, and the I/O controller.
- The 16 Bit System Data Bus facilitates transmission of 16 Bits of data per clock cycle between two registers.
- The 16 General Purpose Registers store data and act as an interface for the CPU to communicate with I/O and memory.
- The ALU Buffer stores intermediate values for computation in the ALU.
- The Arithmetic Logic Unit (ALU) is a combinational logic block which facilitates division, multiplication, addition and subtraction.
- The Accumulator stores the answer to the last computation performed in the ALU.
- The I/O controller interfaces between the CPU and external hardware (namely a screen).
- The Memory Management Unit (MMU) facilitates the movement of data between the internals of the CPU (including its General Purpose Registers and the ALU) and virtual memory.

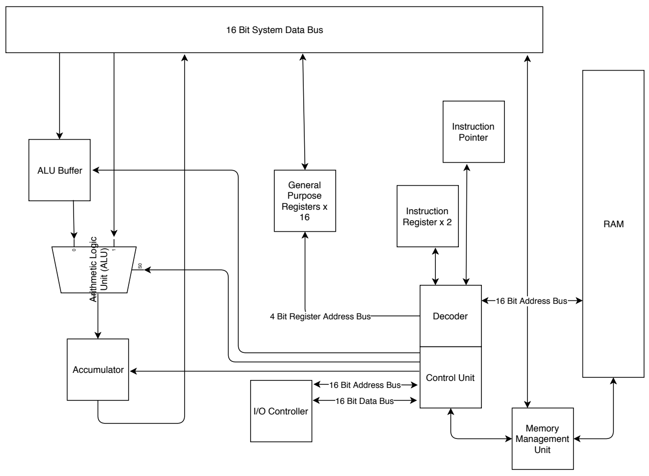


Fig. 1. The microprocessor's datapath

A. Virtual Memory

The processor utilizes a Memory Management Unit (MMU) which controls the address mapping of memory. The first 8 kilobytes of virtual memory maps to high-speed RAM. The next 6 bytes of memory map to the FPGA board's Hex displays, LEDs and buttons (4 bytes, 1 byte and 1 byte respectively). The next 512 bytes map to the internal video RAM of the screen.

III. Y68 Instruction Set

Instructions consist of 4 bytes, whose syntax is shown below, with each row representing one byte of the instruction:

Opcode
Register Address
Immediate High
Immediate Low

The first byte consists of the instruction's opcode. This opcode byte is of the following layout:

P	M	I	D	x	x	x	x

1) The first bit of the opcode - which will hereafter be referred to as the P bit - details whether the instruction will change the program counter. This is called P mode.
2) The second bit of the opcode - the M bit - details whether the instruction will require a memory operation. This is alled M mode.
3) The third bit of the opcode - the I bit - details whether the instruction will perform an I/O operation. This is called I mode.
4) The fourth bit of the opcode - the D bit - details whether the operation will use an immediate value (1 for true, 0 for false). This is called D mode
5) The fifth through to the eighth bits are reserved, and their function depends on the operation mode.

Operation modes are set by a 1 in their corresponding bits. P mode, M mode and I mode are mutually exclusive such that an operation is illegal if any two or more of those bits are set to true. However, the D bit can be set to true in combination with any one of the P, M or I bits.

The last 4 reserved bits for each mode are in the following layout:

P Mode

S	C	E	x

1) The S bit determines whether the operation will use the stack (used in a call and jump operations)
2) The C bit is a carry flag.
3) The E bit is a zero flag.
4) The last bit is unused.

M Mode

W	S	x	x

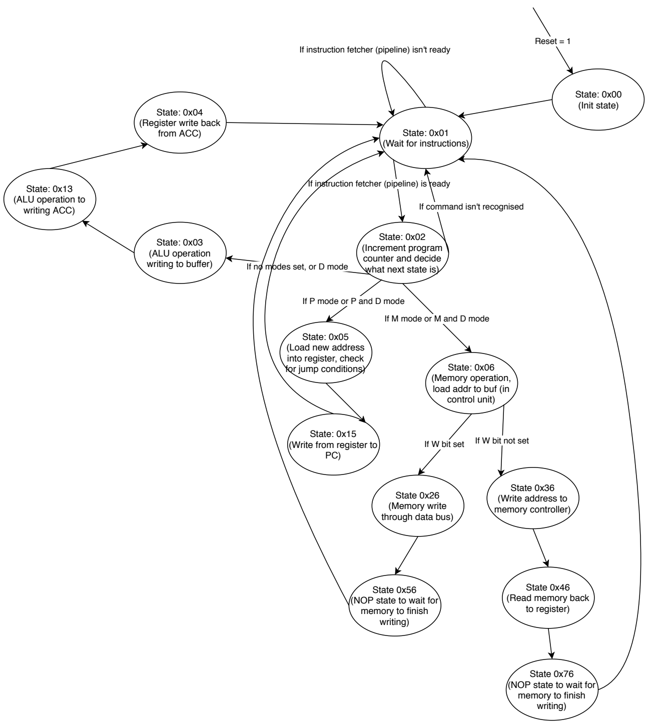
1) The W bit determines whether the operation will write to memory.
2) The S bit determines whether the operation is a stack operation.
3) The last two bits are unused.

I Mode and D Mode

x	x	x	x

1) No bits are used in I Mode, or D Mode (if it is combined with other modes).

IV. Finite State Machine



Reset = 1
If instruction fetcher (pipeline) isn't ready
State: 0x00 (Init state)
State: 0x04 (Register write back from ACC)
State: 0x01 (Wait for instructions)
State: 0x13 (ALU operation to writing ACC)
If instruction fetcher (pipeline) is ready
If command isn't recognised
State: 0x03 (ALU operation writing to buffer)
State: 0x02 (Increment program counter and decide what next state is)
If no modes set, or D mode
If P mode or P and D mode
If M mode or M and D mode
State: 0x05 (Load new address into register, check for jump conditions)
State: 0x06 (Memory operation, load addr to buf (in control unit)
State: 0x15 (Write from register to PC)
If W bit set
If W bit not set
State 0x26 (Memory write through data bus)
State 0x36 (Write address to memory controller)
State 0x56 (NOP state to wait for memory to finish writing)
State 0x46 (Read memory back to register)
State 0x76 (NOP state to wait for memory to finish writing)

Fig. 2. The control unit's FSM

Acknowledgment

Thanks to Professor David Boland for being a wonderful lecturer.