FANTASTIC ACCUMULATOR

RED TEAM

Abi Naseem
Michelle Zhou
Darren Zhu
Joshua Giambattista

OUTLINE

1. Overall Architecture
2. Instruction Set
3. RTL
4. Final Datapath
5. Approach to Testing
6. Performance
7. What Makes Our Processor Unique
8. Good Design Decisions
9. Challenges
10. Improvements Needed

Overall Architecture

- A multi-cycle processor with 4 integration phases
- ~18 hardware implementations
- 8 registers and 32 instructions with 3 instruction types

Instruction Set

R-Type

D-Type

I/J-Type

Examples:

add $wr, $tp, 0
move $wr, $sp

pop $ra, 0
clrr $wr

addi 0x4
goto recpart
slti 0x0

RTL

Examples	add/sub/and/or	call	goto
Instruction fetch	IR = Mem[PC] PC = PC + 2		
Instruction decode	Imm = SE(IR[11:0])		
Reigister fetch	ALUOut = MUX[IR[7 : 5]] op MUX[IR[4 : 1]] (* op = +, -, &, |)	$ra = PC	PC = PC[15 : 13] || MUX([IR[11 : 0]) << 1
Depends on Instr	location = ALUOut	PC = PC[15 : 13] || MUX([IR[11 : 0]) << 1	

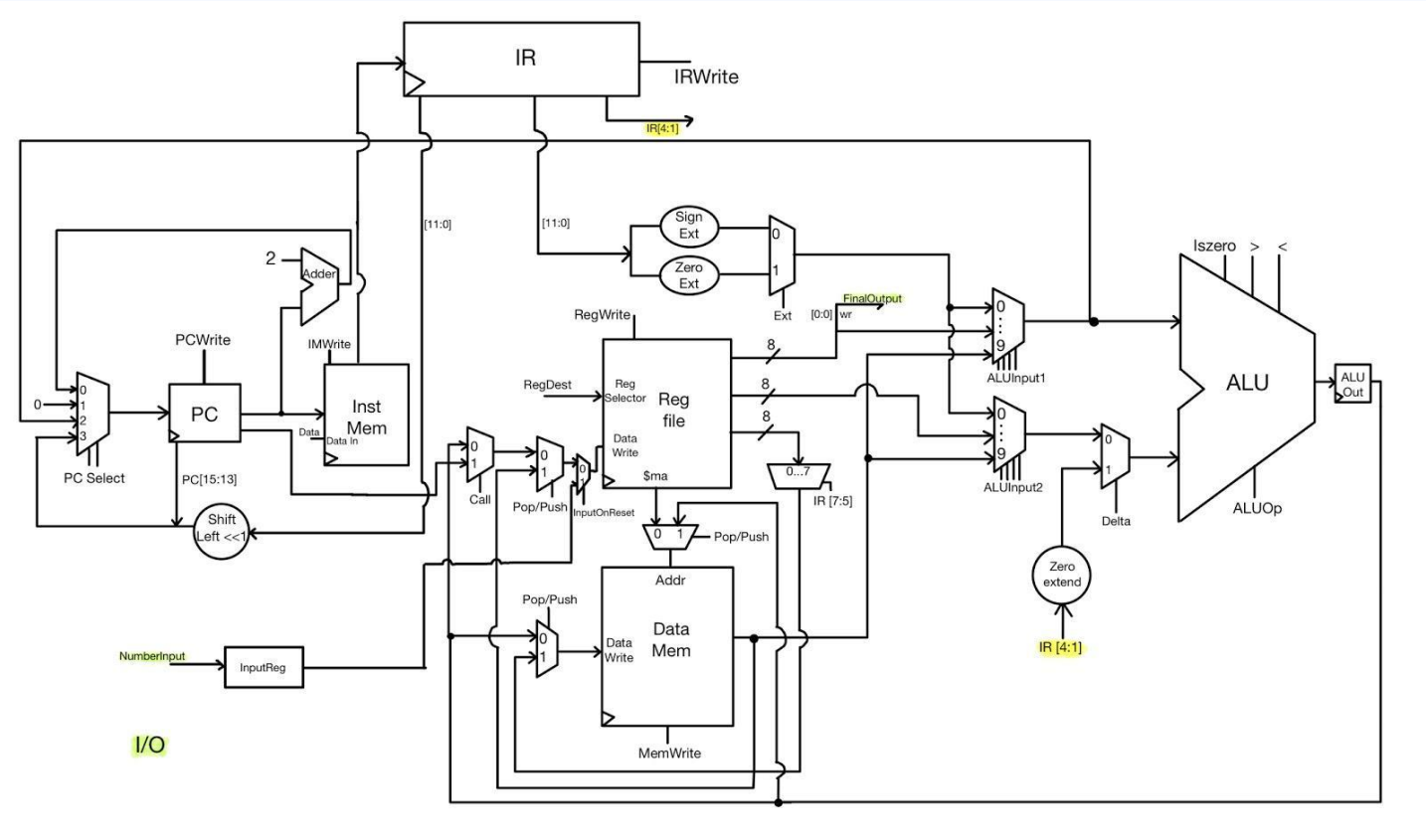
Examples	lui	luia	seq / sne	pop / push	jr
Instruction fetch	IR = Mem[PC] PC = PC + 2				
Instruction decode	Imm = SE(IR[11:0])				
Reigister fetch	ALUOut = Imm		ALUOut = $wr - MUX[IR[7:5]] isZero = Wire(ALUOut [0: 1])	ALUOut = $sp + Imm	PC = Reg[IR[7:5]]
Depends on Instr	$wr = ALUOut	$tp = ALUOut	If (isZero == 1): PC = PC + 2	C = Mem[ALUOut]	
Push/Pop Cycle				Reg[IR[7:5]] = C	

Examples	clrr	incr/decr	move	la
Instruction fetch	IR = Mem[PC] PC = PC + 2			
Instruction decode	Imm = SE(IR[11:0])			
Reigister fetch	ALUOut = 0	ALUOut = MUX[IR[7:5]] ± ZE(IR[4:1])	ALUOut = MUX[IR[4:1]]	ALUOut = MUX[IR[11:0]]
Depends on Instr	Reg[IR[7 : 5]] = ALUOut			
Push/Pop Cycle				

The multicycle RTL is broken down into five steps:

- Instruction fetch
- Instruction decode
- Register fetch
- (Optional steps)
- (Optional step only for push/pop)

Datapath


IR
IRWrite
IR[4:1]
[11:0]
[11:0]
Sign Ext
Zero Ext
Ext
FinalOutput
[0:0] wr
RegWrite
2
Adder
PCWrite
IMWrite
RegDest
Reg Selector
Reg file
Data Write
$ma
8
8
8
0...7
IR [7:5]
ALUInput1
ALUInput2
Iszero
>
<
ALU
ALU Out
ALUOp
Delta
Zero extend
IR [4:1]
PC
Inst Mem
Data
Data In
PC Select
PC[15:13]
Shift Left <<1
Call
Pop/Push
InputOnReset
Pop/Push
Addr
Pop/Push
Data Write
Data Mem
MemWrite
NumberInput
InputReg
I/O

Approach to Testing

1. Write the test benches for each hardware implementation. Test!
2. Put components together to form 4 integration phase. Test!
3. Combine integration phases together to form the overall datapath. Then test to see if we can successfully get the instruction information, run through the datapath, get the output, and then go to the next instruction accurately until all the instructions are done.

Performance

- **146 bytes** to store Euclid's algorithm and relPrime
- **8 bytes** of memory variables
- total number instructions executed when relPrime is called with 0x13B0: **153227**
- total number of cycles required to execute relPrime under the same conditions as Step 2 is **612908**
- average cycles per instruction is **4**
- cycle time is **34 ns**
- total execution time for relPrime under the same conditions as Step 2is **19 ms**

Performance

device utilization summary

Phase4_Verilog Project Status (02/19/2020 - 14:28:38)			
Project File:	Phase4.xise	**Parser Errors:**	No Errors
Module Name:	Phase4_Verilog	**Implementation State:**	Synthesized
Target Device:	xc3s500e-4fg320	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	51 Warnings (0 new, 0 filtered)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	652	4656	14%
Number of Slice Flip Flops	240	9312	2%
Number of 4 input LUTs	1250	9312	13%
Number of bonded IOBs	313	232	134%
Number of BRAMs	16	20	80%

What Makes Our Processor Unique

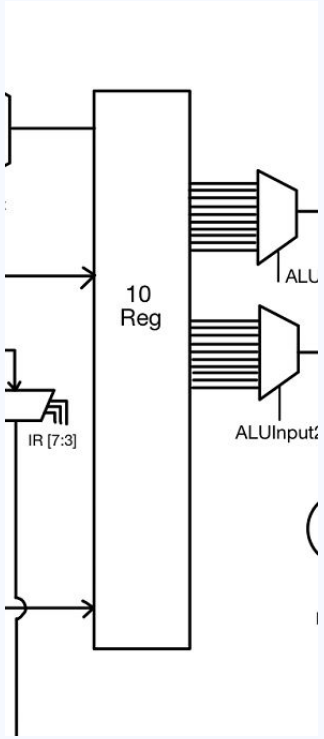
►The use of a memory address register that points to a place in data memory

► Processor supports full recursive procedures.

► If/else statements are extremely easy because we have a lot of instructions that can check for equality, inequality, etc. and will skip to the next instruction

Good Design Decisions

- instruction format
  - straight forward
  - can implement up to 32 instructions
- Deleting the wailing wall
  - the giant wall of 10 registers
- DJ
- Fire Assemblers

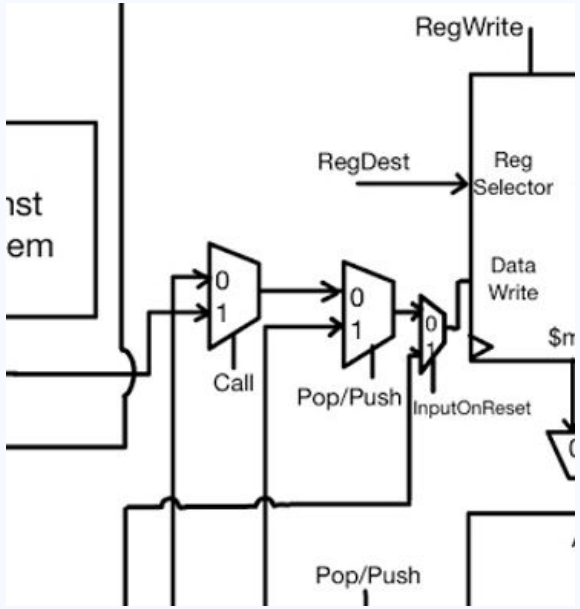


Challenges

- Memory Part (Datapath)
  - Figure out how to create a datapath that fit all 32 instructions
  - Way to initialize and reset PC
- Code part (Xilinx)
  - Finding errors in the schematic and Verilog files
- Git

Improvements Needed

- Mux way



- Big control

A control and a mini control with complicated code

Q & A