

CompRegReg computer designment
Computer designment techniques

Moreno Anarte, Rafael

April 2016

Contents

1	Introduction	3
1.1	Objective	3
1.2	Used hardware	4
1.2.1	Basys2 Spartan 3E FPGA	4
1.2.2	Leds	4
1.3	Used software	4
1.3.1	Xilinx Webpack ISE 14.2	4
2	Requirements Specification	5
2.1	Memory programming	5
2.2	Registers bank	5
2.3	Instruction structure	5
3	Desing	6
3.1	TOP	6
3.1.1	Dependencies:	6
3.1.2	Signals	6
3.2	Control Unit	7
3.2.1	States	7
3.2.2	ControlWords	7
3.3	Data Path	9
3.3.1	Dependencies:	10
3.3.2	Signals	10
3.4	Display7Seg-4ON	11
3.4.1	Dependencies:	11
3.5	Debounce	12
3.5.1	States	12
3.5.2	Dependencies:	12

1 Introduction

1.1 Objective

Creation of a computer with the following characteristics:

- Architecture or execution mode: Register-Register
- Same memory for data and program code. (Von Neumann).
- Includes a registers bank
- Instruction Set Architecture (ISA) 7 Instructions:
 - LD dir,R2 – Load memory data into a register.
 - ST dir,R2 – Stores register data into memory.
 - ADD R1,R2 – Makes add operation between R1 and R2.
 - SUB R1,R2 – Makes subtraction operation between R1 and R2.
 - INC inm,R2 – Makes add operation between immediate and R2
 - DEC inm,R2 – Makes subtraction operation between immediate and R2
 - BeQ dir,XX – Jumps to dir if FZ = '1'
- Operation codes:
 - LD 000
 - ST 001
 - ADD 010
 - SUB 011
 - INC 100
 - DEC 101
 - BeQ 110
- Instruction lenght: 10 bits.
- Instruction format:
 - 1º Field: Operation code (3bits).
 - 2º Field: Memory address (4bits) / Immediate (4bits) / Register1 address (3bits).
 - 3º Field: Register2 address.

1.2 Used hardware

List and basic description of the elements ultimately used in the final design

1.2.1 Basys2 Spartan 3E FPGA

The Basys 2 FPGA development board is a circuit design and implementation platform for beginners to gain experience building real digital circuits. Built around a Xilinx® Spartan®-3E FPGA and an Atmel® AT90USB2 USB controller, the Basys 2 development board provides complete, ready-to-use hardware suitable for hosting circuits ranging from basic logic devices to complex controllers. A large collection of on-board I/O devices and all required FPGA support circuits are included, so countless designs can be created without the need for any other components.

1.2.2 Leds

- x3 Green leds.
- x1 Red led.

1.3 Used software

1.3.1 Xilinx Webpack ISE 14.2

<http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools/v142.html>

2 Requirements Specification

2.1 Memory programming

Memory should have a program for properly testing all instructions.

- Use the RAM(0) to RAM(10) for instructions
- Use RAM(11) to RAM(15) for data.

2.2 Registers bank

Registers bank should have the following characteristics:

- 2 inputs for R1 and R2 addresses.
- 2 output for R1 and R2 content
- 1 input for R2 data-writing

2.3 Instruction structure

Most follow the following rules:

- All instruction will cross ALU, with exception of BeQ
- On LD/ST the "dir" refers the address.
- INC and DEC are using immediates of 4bits
- BeQ will jump if the previous instruction who modifies FZ, modified it to '1'.

3 Desing

CompRegReg machine is made with the conjunction of a few modules. For a better understanding see the explanation below:

3.1 TOP

Top module of the architecture, unifies all modules and connects the results with the outside.

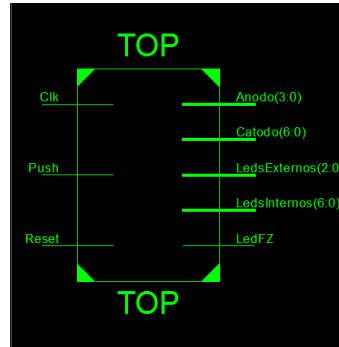


Figure 1: Module TOP from CompRegReg architecture.

3.1.1 Dependencies:

- Control Unit.
- Data Path.
- Display7Seg4ON.
- Debounce.

3.1.2 Signals

- signal-filteredPush connects Debounce's filteredPush with Control-unit's push and is used as datain on five and gates (CW2/3/4/7/8).
- signal-fz connects Data-Path's SalFZ with Control-unit's FZ.
- signal-controlWords connects Control-unit's ControlWord with DataPath CWs and five and gates (CW2/3/4/7/8).
- signal-busRam connects Data-Path's BusRam with Display7Seg-4ON's Dato1.
- signal-sal-opeA connects Data-Path's sal-opeA with Display7Seg-4ON's Dato2.
- signal-sal-opeB connects Data-Path's sal-opeB with Display7Seg-4ON's Dato3.
- signal-salALU connects Data-Path's salALU with Display7Seg-4ON's Dato4.
- signal-instruccion connects Data-Path's instruccion with Control-unit's opCode and Led-sExternos with the 3 MSB and the rest with LedsInternos.

3.2 Control Unit

Its the responsible of all control words, i could say its the brain. Its a state machine who decides the control words output looking at an operation code.

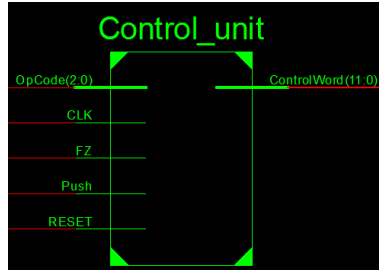


Figure 2: Module Control Unit from CompRegReg architecture.

3.2.1 States

Control Unit has nine states, every push = '1' will provoke a change on states.

- Idle: Initial state
- LoadInst: Loading instruction state
- Deco: Decodifying state (takes decision looking at operation code)
- Load: Load operation state
- Store: Store operation state
- AaddB: Add operation state
- AsubB: Substraction state
- INC: Immediate add state
- DEC: Immediate subtraction state
- BeQ: BeQ state

3.2.2 ControlWords

Control word output will be define with the following parameters:

- constant Outputs-Idle : std-logic-Vector(11 downto 0):="000000000000"
- constant Outputs-LoadInst : std-logic-Vector(11 downto 0):="000010001000"
- constant Outputs-Deco : std-logic-Vector(11 downto 0):="000000000000"
- constant Outputs-Load : std-logic-Vector(11 downto 0):="000001010010"
- constant Outputs-Store : std-logic-Vector(11 downto 0):="001000000110"
- constant Outputs-AaddB : std-logic-Vector(11 downto 0):="010100110000"

- constant Outputs-AsubB : std-logic-Vector(11 downto 0):="011100110000"
- constant Outputs-BincA : std-logic-Vector(11 downto 0):="100100010000"
- constant Outputs-BdecA : std-logic-Vector(11 downto 0):="101100010000"
- constant Outputs-BeQ : std-logic-Vector(11 downto 0):="110010000001"

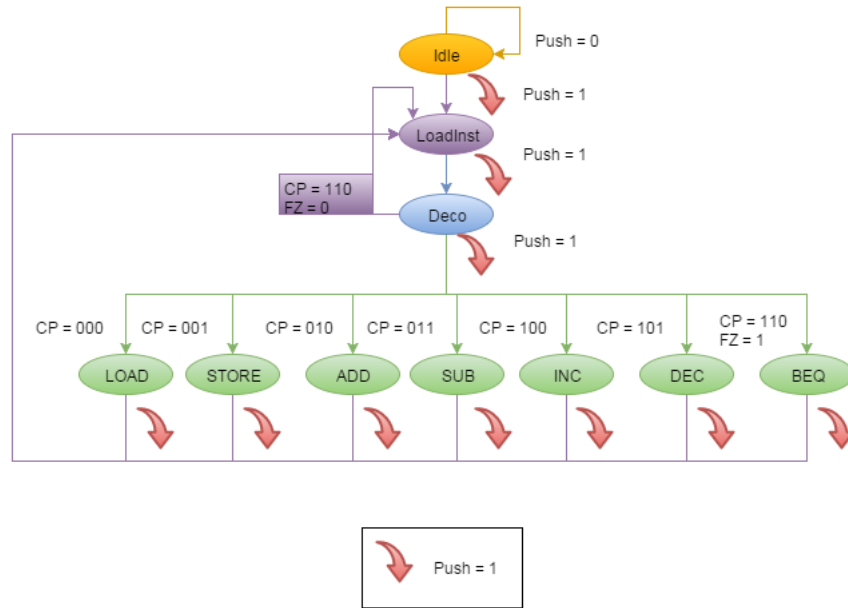


Figure 3: States changes from Control Unit.

The hearth of CompRegReg machine. Will receive information from Control Unit and will process the data on his ALU (exception with BeQ).

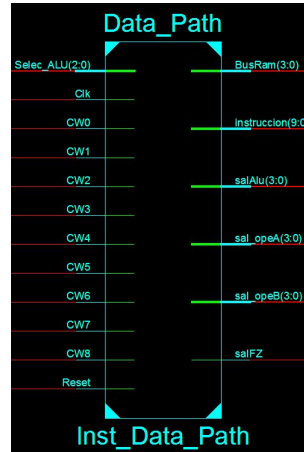


Figure 4: Module Data Path from CompRegReg architecture.

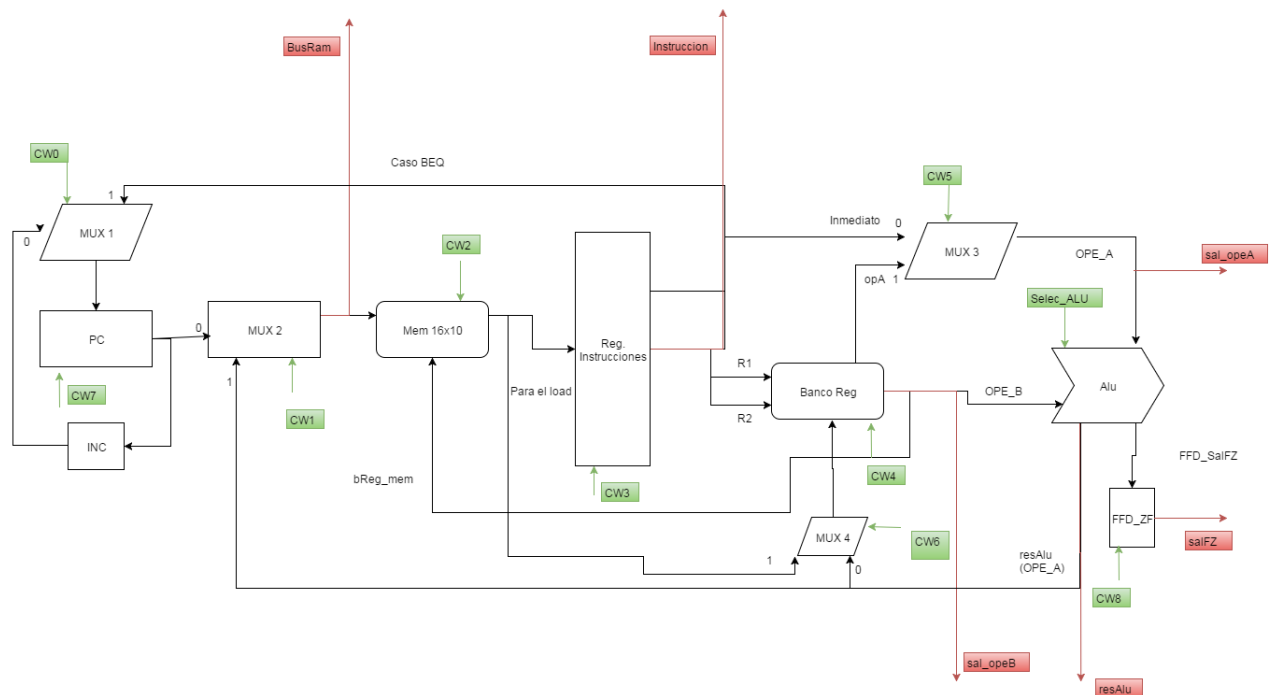


Figure 5: Module Data Path from CompRegReg architecture.

Looking deep-inside the module. We could see there are 4 multiplexors, a memory ram module, the program counter...etc.

Red cables are outputs, and green signals are controlwords.

3.3.1 Dependencies:

- Mux2-4bits.
- Ram16x10.
- Inc1-4bits.
- Reg-4bits.
- ALU-4bits
- FFD-Basic.
- Banco Registros8x4.

3.3.2 Signals

- opA connects Banco-registros-8x4's DataOut-reg1 with MUX3's B
- OpB connects Banco-registros-8x4's DataOut-reg2 with ALU-4bits's OPE-B in.
- resALU connects ALU's output with MUX2's B and MUX4's A and with the output salALU.
- mux3-alu connects MUX3's output with ALU-4bits's OPE-A in.
- mux4-bReg connects MUX4's output with Banco-registros-8x4's DataIn-reg2.
- mem-Reginst connects RAM-16x10's dataout with Reg-10bits's DataIn.
- sal-regInst connects Reg-10bits's Dataout with MUX1 B and MUX3 A and Banco-registros-8x4's Address-reg1/2 and the output instruccion.
- sal-regInst connects Reg-10bits's Dataout with MUX1 B and MUX3 A and Banco-registros-8x4's Address-reg1/2 and the output instruccion.
- bReg-mem connects 4 LSB from opeB + 0x000.. with RAM-16x10's DataIn.
- signal-salFZ connects ALU-4bits's salZF with FFD-Basic's D (datain).
- FFD-SALFZ connects FFD-Basic's Q (dataout) with salFZ output.
- pc-inc-mux2 connects Reg-4bits's dataout (PC dataout) with MUX2's A and Inc1-4bits's value (datain).
- inc-mux1 connects Inc1-4bits's value-inc (dataout) with MUX1's A.
- mux1-pc connects MUX1's Z (dataout) with Reg-4bits's datain.
- mux2-mem connects MUX2's Z (dataout) with RAM-16x10's address and output BusRam

3.4 Display7Seg-4ON

Our machine will use a seven segments display for showing the following:

- Display AN3: RAM Address.
- Display AN2: First operand in ALU.
- Display AN1: Operando B de la ALU.
- Display AN0: Salida de la ALU.

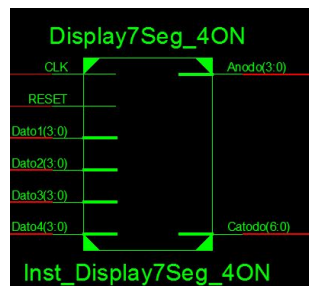


Figure 6: Module Display7Seg-4ON from CompRegReg architecture.

3.4.1 Dependencies:

- Mux4-4bits.
- Disp7Seg.
- YourTurn.
- CLK-1Khz.

3.5 Debounce

States machine used to increase resilience of ComRegReg against mechanic errors, like the problem with pushing on mechanical push buttons.

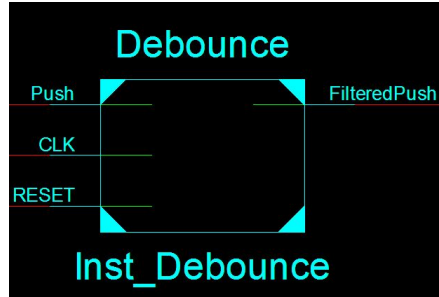


Figure 7: Module Debounce from CompRegReg architecture.

3.5.1 States

Inside Debounce, there is a module called debounce FSM who is the one who has the states. In fact, it has four states. The states are:

- Inic: Initial state. It will change if receives a '0', and will stay on this state if receives '1'.
- S0: It will stay on this state if receives '0', and it will swap if receives '1'.
- S01: This state will only work for one cycle, doesn't matter what it receives. Will put the enable timer in '1'.
- Espera: It will stay on this state if receives '0' otherwise it will swap.

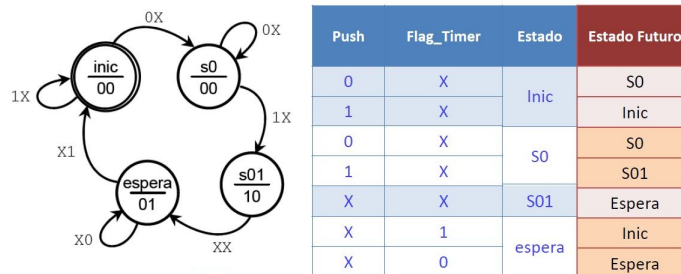


Figure 8: Image taken from practice 5 page 7.

3.5.2 Dependencies:

- Mux4-4bits.
- Disp7Seg.
- YourTurn.
- CLK-1Khz.

