BILKENT UNIVERSITY

COMPUTER ENGINEERING DEPT.

CS223 PROJECT REPORT

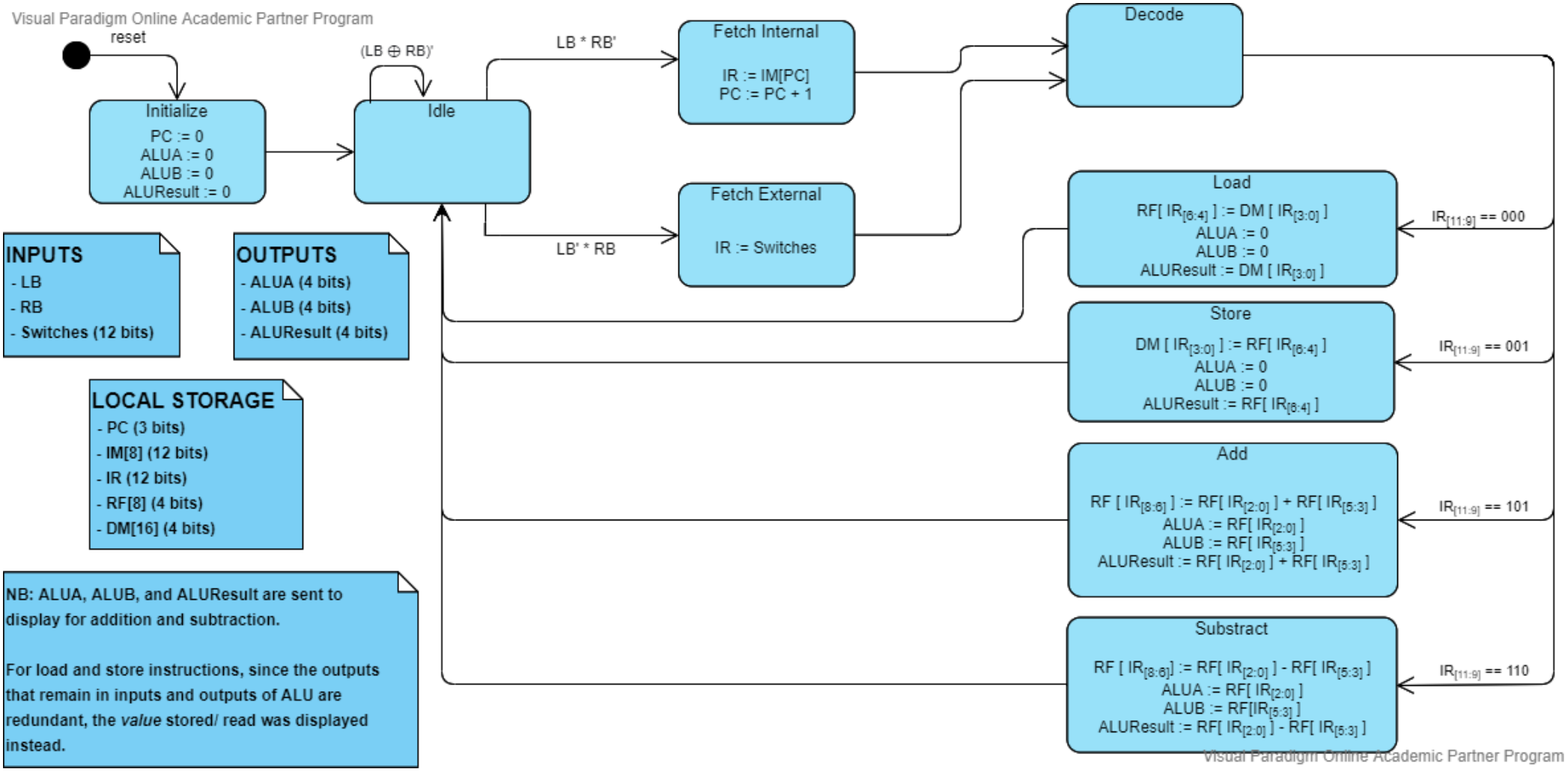
SIMPLE PROCESSOR

ZÜBEYİR BODUR

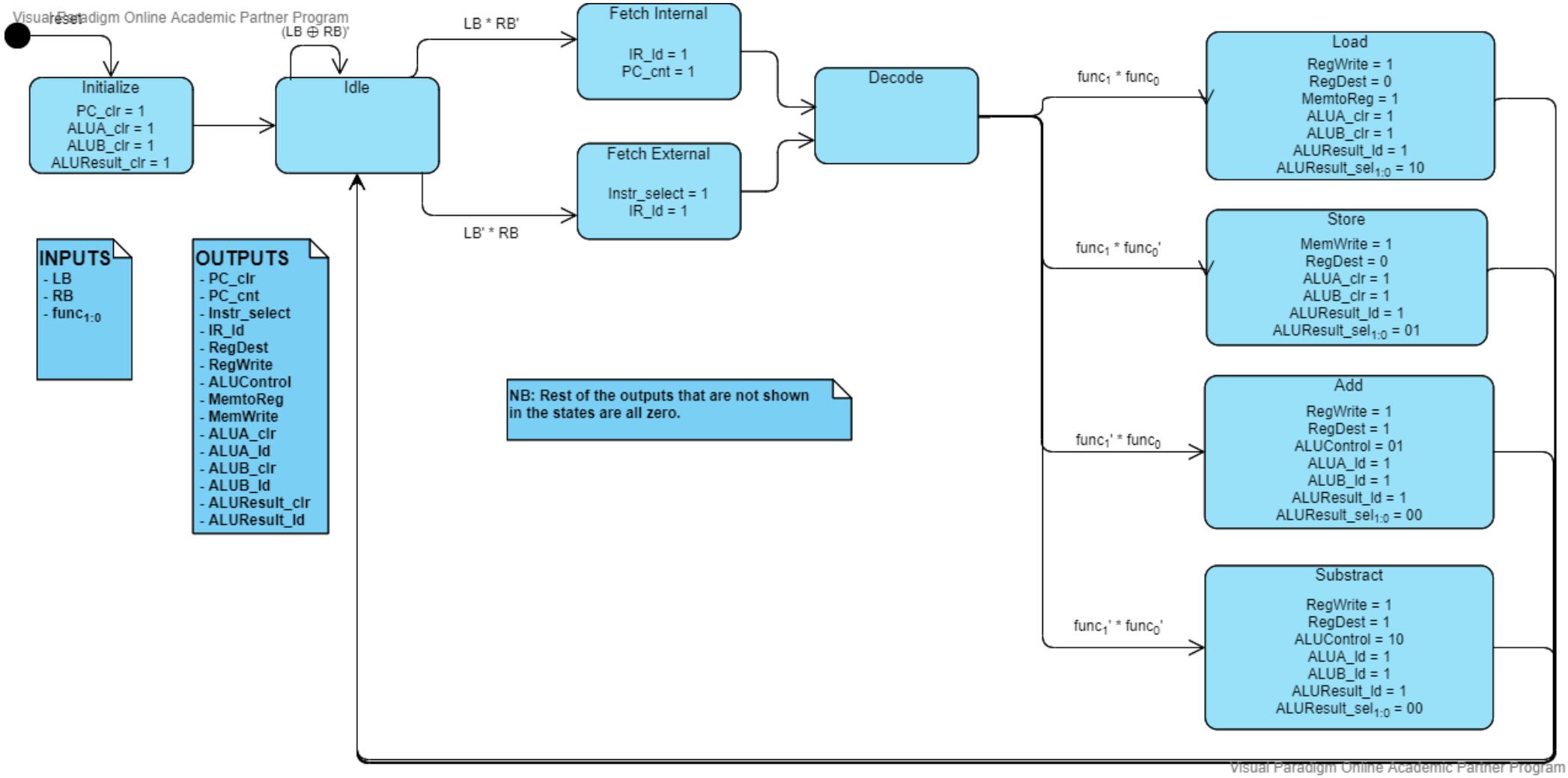
21702382

Spring 2021

1. HLSM Diagram



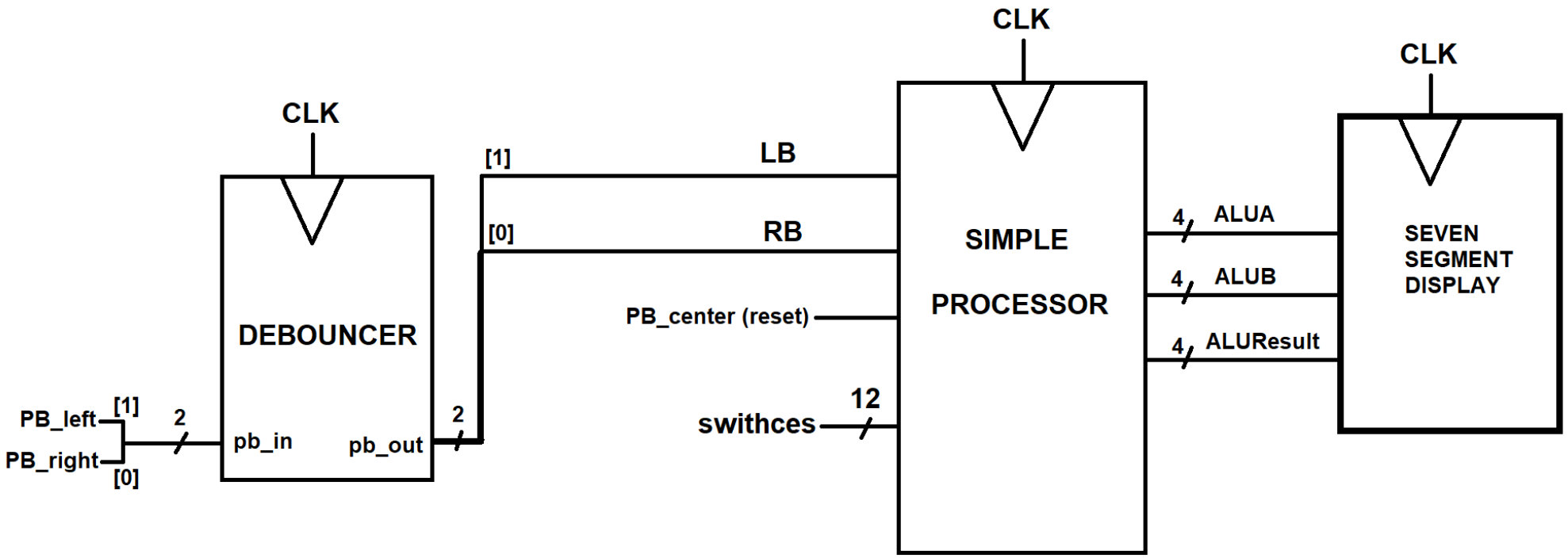
2. Controller FSM



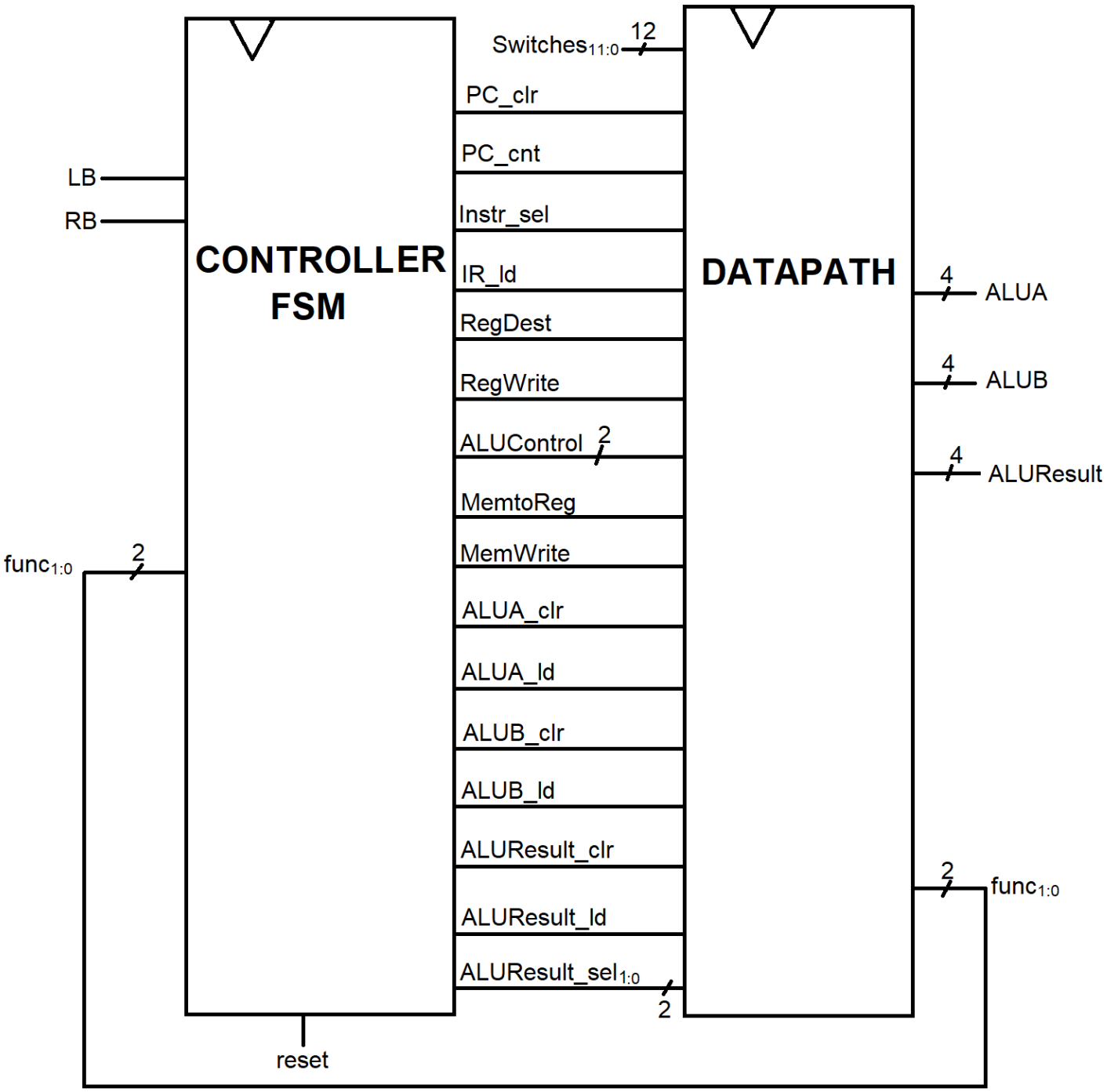
3. Top Module Block Diagram

a) Topmost Module

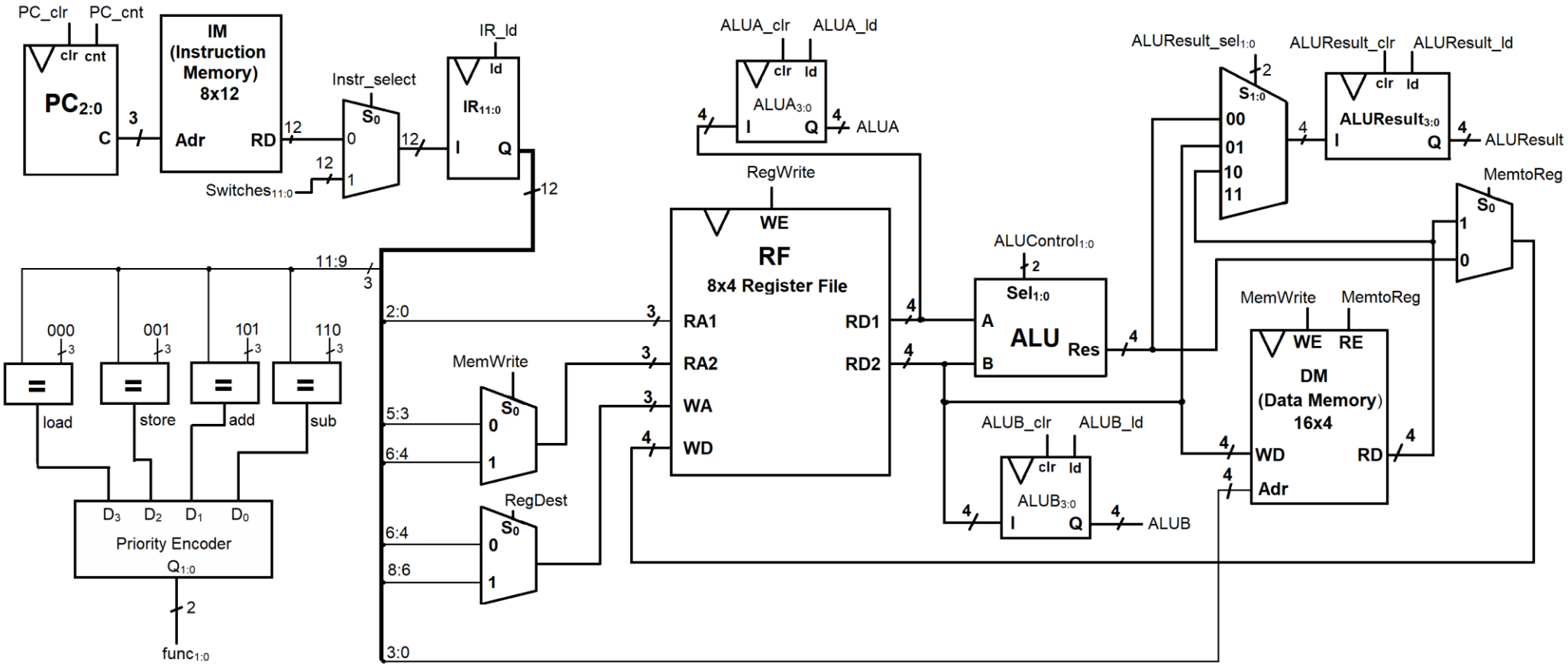
"Topmost" module refers to the module that will be used to run in Basys3 FPGA. The "2-bit" debouncer FSM actually does not accept 2 bit outputs, it's rather 2 debouncer FSMs connected in parallel. Seven segment display outputs are not shown for simplicity as well.



b) Top Module (Simple Processor)



4. Datapath Block Diagram



NB: Priority Encoder is used to encode load, store, add & sub bits into $func_{1:0}$. $func_{1:0}$ will always have a value because it's assumed that $IR_{11:9}$ is either 000, 001, 101, 110, meaning $D_{3:0}$ can't be 0000.

Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Priority Encoder Truth Table, black X's denote DON"T CARE while red X's denote contention

load => $func_{1:0} = 11$ store => $func_{1:0} = 10$

add => $func_{1:0} = 11$ sub => $func_{1:0} = 11$

5. Controller Block Diagram

Since the controller FSM has 9 states (encoded in 4 state bits), 4 inputs and 16 outputs, for the sake of simplicity, look up tables (LUT) are used. In addition, the next state logic truth table would have $2^{4+4} = 256$ rows, so don't cares were used whenever possible.

Below is the state encoding for the FSM.

State	$S_{3:0}$
Init	0000
Idle	0001
Fetch Internal	0010
Fetch External	0011
Decode	0100
Load	0101
Store	0110
Add	0111
Sub	1000

The truth tables below are the next state logic and output logic, respectively.

Current State	$S_{3:0}$	LB	RB	$func_{1:0}$	$S'_{3:0}$	Next State
Init	0000	X	X	X	0001	Idle
Idle	0001	0	0	X	0001	Idle
Idle	0001	0	1	X	0011	Fetch External
Idle	0001	1	0	X	0010	Fetch Internal
Idle	0001	1	1	X	0001	Idle
Fetch Internal	0010	X	X	X	0100	Decode
Fetch External	0011	X	X	X	0100	Decode
Decode	0100	X	X	11	0101	Load
Decode	0100	X	X	10	0110	Store
Decode	0100	X	X	01	0111	Add
Decode	0100	X	X	00	1000	Sub
Load	0101	X	X	X	0001	Idle
Store	0110	X	X	X	0001	Idle
Add	0111	X	X	X	0001	Idle
Sub	1000	X	X	X	0001	Idle

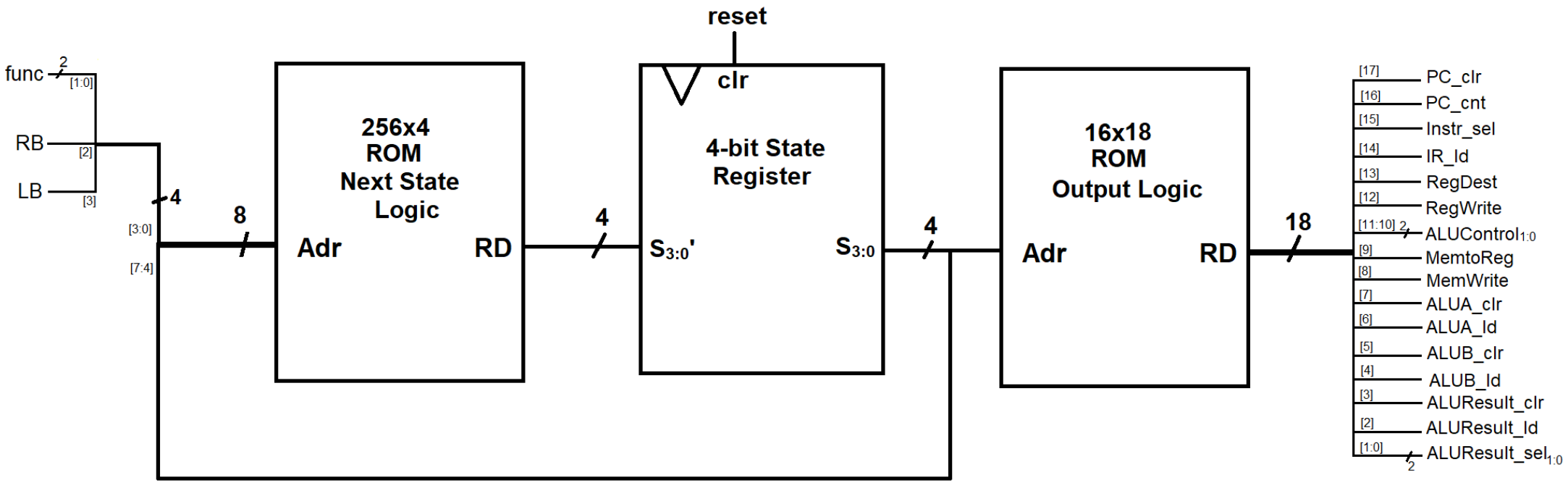
LUT for the Next State Logic

This LUT requires a 2^8 X 4 = 256 x 4 ROM, where the read output's bits are next state bits.

Current State	$S_{3:0}$	PC_clr	PC_cnt	Instr_ sel	IR_ld	RegDest	RegW rite	ALU Cont $rol_{1:0}$	Memto Reg	Mem Write	ALUA_ clr	ALUA _ld	ALUB_ clr	ALUB _ld	ALUResult_ clr	ALUResult _ld	$ALUResult\_sel_{1:0}$
Init	0000	1	0	0	0	0	0	00	0	0	1	0	1	0	1	0	00
Idle	0001	0	0	0	0	0	0	00	0	0	0	0	0	0	0	0	00
Fetch Internal	0010	0	1	0	1	0	0	00	0	0	0	0	0	0	0	0	00
Fetch External	0011	0	0	1	1	0	0	00	0	0	0	0	0	0	0	0	00
Decode	0100	0	0	0	0	0	0	00	0	0	0	0	0	0	0	0	00
Load	0101	0	0	0	0	0	1	00	1	0	1	0	1	0	0	1	10
Store	0110	0	0	0	0	0	0	00	0	1	1	0	1	0	0	1	01
Add	0111	0	0	0	0	1	1	01	0	0	0	1	0	1	0	1	00
Sub	1000	0	0	0	0	1	1	10	0	0	0	1	0	1	0	1	00

LUT for the Output Logic

This LUT requires a 2^4 X 16 = 16 x 16 ROM, where the read output's bits are PC_clr, PC_cnt, … $ALUResult\_sel_{1:0}$, from the most significant bit to the least.



Hence, the block diagram for the controller can be drawn as above, using the LUTs as ROM.