Laboratory Work 3

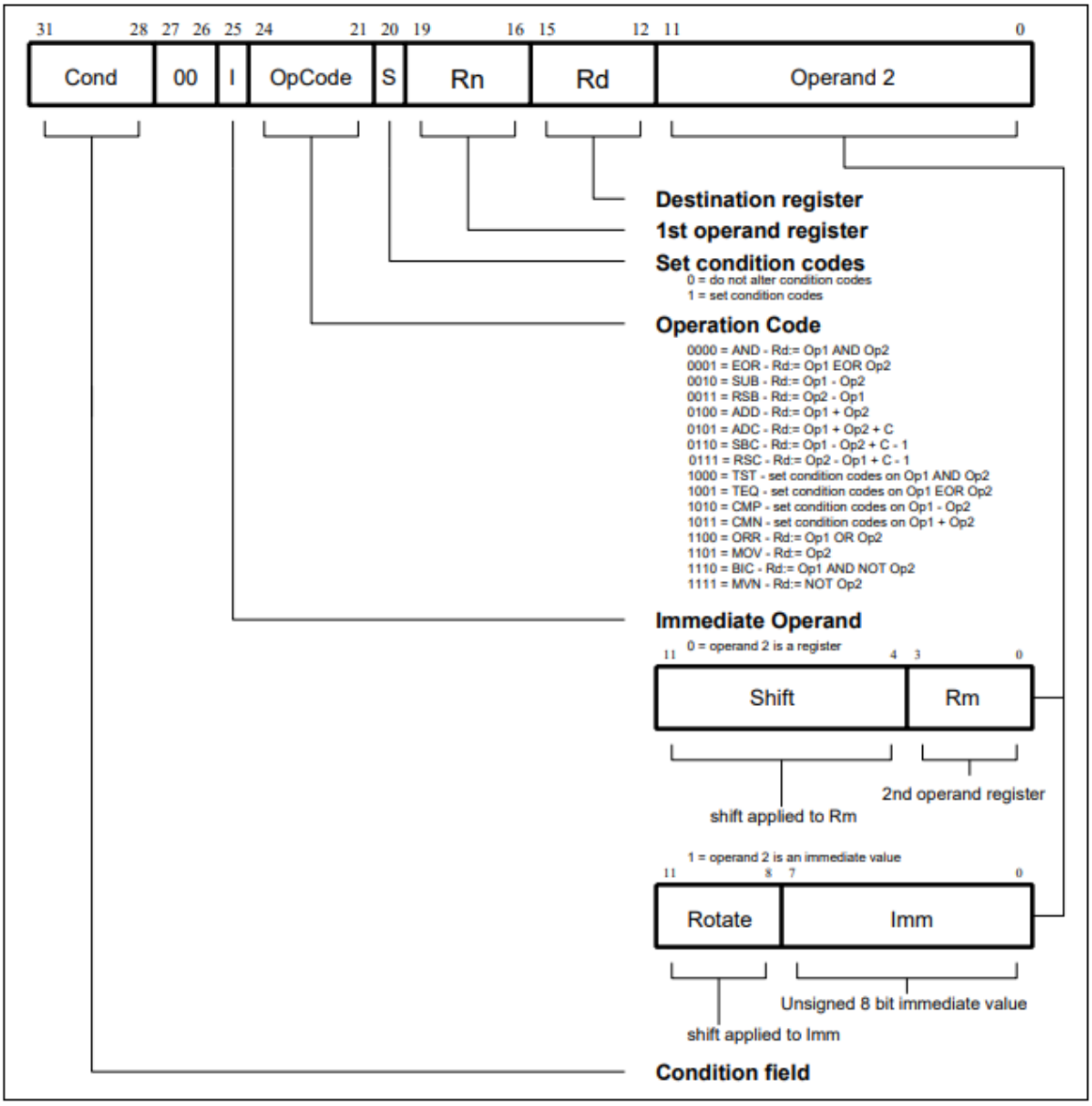
PRELIMINARY WORK

Ataberk ÖKLÜ
METU EEE | 2305142

Table of Contents

I. Datapath Design 2
A. Instruction Steps 3
1. DP (ADD/SUB/AND/OR/CMP) 3
2. DP (MOV: LSL LSR) 3
3. Memory (LDR) 3
4. Memory (STR) 3
II. Controller Design 6
A. Black Box Diagram 6
B. Changes in Base Implementation 6
C. Truth Table 6
1. Condition 6
2. ALU Operation 6
3. Operation Decoder 7
D. Controller Implementation 7
III. Parametrization 7

I. Datapath Design



A. Instruction Steps

1. DP (ADD/SUB/AND/OR/CMP)

- Fetch InstMem[PC]
- Increment PC by 4
- Decode Instruction -> DP + (ADD/SUB/AND/OR) + Rd + Rn + Rm
- Set AD3 = Rd and (AD1, AD2) as (Rn, Rm)
- Set Alu Source as Registers
- Set Alu Control as (ADD/SUB/AND/OR/CMP)
- Set Alu output to go register file WD3 port.
- Set Register Write 1
- Clock Rising Edge: PC and Register file is updated

2. DP (MOV: LSL LSR)

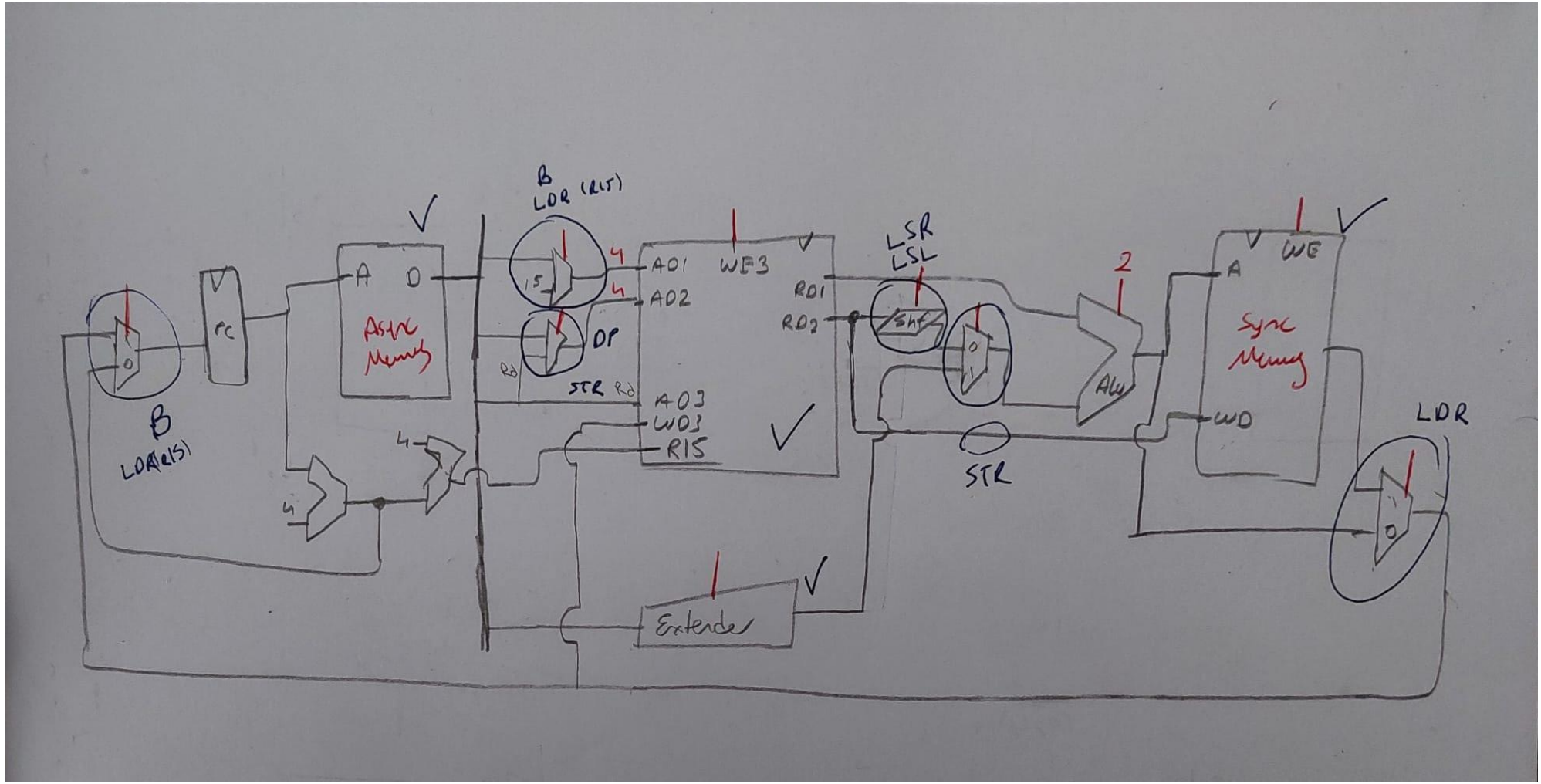
- Fetch InstMem[PC]
- Increment PC by 4
- Decode Instruction -> DP + (LSL/LSR) + Rd + Rm
- Set AD3 = Rd, AD2 = Rm
- Set Shifter Mode and Amount
- Set MemSrc as Shifter
- Forward Shifted Value to register file WD3 port by Mem2Reg.
- Set Register Write 1
- Clock Rising Edge: PC and Register file is updated

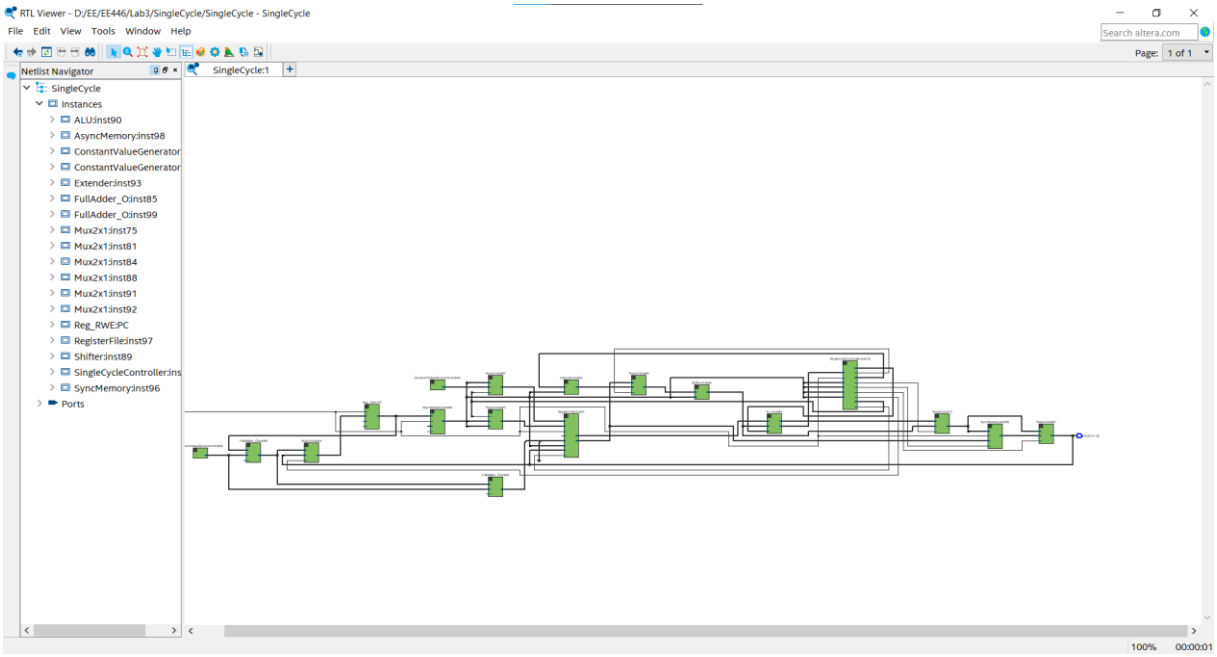
3. Memory (LDR)

- Fetch InstMem[PC]
- Increment PC by 4
- Decode Instruction -> MP + (LDR) + Rd + Rn + Imm12
- Set AD3 = Rd, AD1 = Rn
- Select Immediate (ALUSrc) and Set Extender to Imm12 mode
- Set MemSrc as ALU
- Set Mem2Reg = 1, forward read data to WD3 of Register File.
- Set Register Write 1
- Clock Rising Edge: PC and Register file is updated

4. Memory (STR)

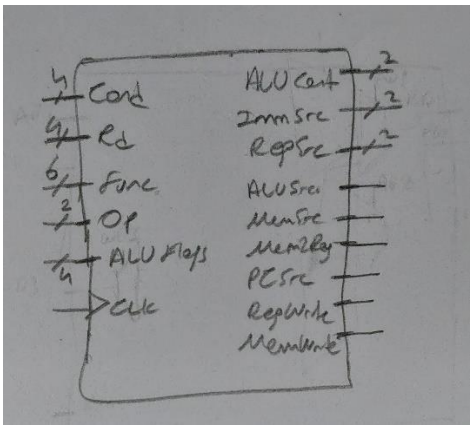
- Fetch InstMem[PC]
- Increment PC by 4
- Decode Instruction -> MP + (STR) + Rd + Rn + Imm12
- Set AD1 = Rn, AD2 = Rd
- Select Immediate (ALUSrc) and Set Extender to Imm12 mode
- Set MemSrc as ALU
- Set Mem2Reg = 1 to select AD of Data Memory.
- Set WD of Data Memory as Rd
- Set Memory Write 1
- Clock Rising Edge: PC and Data Memory is updated


B
LOR (RLS)
PC
A
O
Async Memory
B
LOR(RLS)
4
15
4
A01
WE3
RD1
RD2
AD2
DP
RD
STR Rd
A03
WD3
R15
LSR
LSL
Shf
2
ALU
STR
A
WE
Sync Memory
WD
LDR
Extender


RTL Viewer - D:/EE/EE446/Lab3/SingleCycle/SingleCycle - SingleCycle
File Edit View Tools Window Help
Search altera.com
Page: 1 of 1
Netlist Navigator
SingleCycle
Instances
ALU:inst90
AsyncMemory:inst98
ConstantValueGenerator
ConstantValueGenerator
Extender:inst93
FullAdder_0:inst85
FullAdder_0:inst99
Mux2x1:inst75
Mux2x1:inst81
Mux2x1:inst84
Mux2x1:inst88
Mux2x1:inst91
Mux2x1:inst92
Reg_RWE:PC
RegisterFile:inst97
Shifter:inst89
SingleCycleController:ins
SyncMemory:inst96
Ports
SingleCycle:1
100%
00:00:01

II. Controller Design

A. Black Box Diagram



B. Changes in Base Implementation

- To Implement both Data Processing and Memory Operation, Register Multiplexers are used. Their Control signals are RegSrc.
- To Implement Compare Operation, NoWrite logic is implemented in Controller in order not to update Flags
- All Other Entities are preserved from base implementation.

C. Truth Table

1. Condition

Bits	Mnemonic Extension	Meaning	Flag
0000	EQ	Equal	Z = 1
0001	NE	Not equal	Z = 0
0010	CS/HS	Carry Set/Higher or Same	C = 1
0011	CC/LO	Carry Clear/Lower	C = 0
0100	MI	Minus/Negative	N = 1
0101	PL	Plus	N = 0
0110	VS	V Set (Overflow)	V = 1
0111	VC	V Clear (No Overflow)	V = 0
1000	HI	Higher	C = 1 and Z = 0
1001	HS	Lower or Same	C = 1 and Z = 1
1010	GE	Greater than or Equal	N = V
1011	LT	Less than	N ≠ V
1100	GT	Greater than	Z = 0 and N = V
1101	LE	Less than or Equal	Z = 0 or N ≠ V
1110	AL	Always (unconditional)	
1111	---	Not Valid	

2. ALU Operation

Table 7.3 ALU Decoder truth table

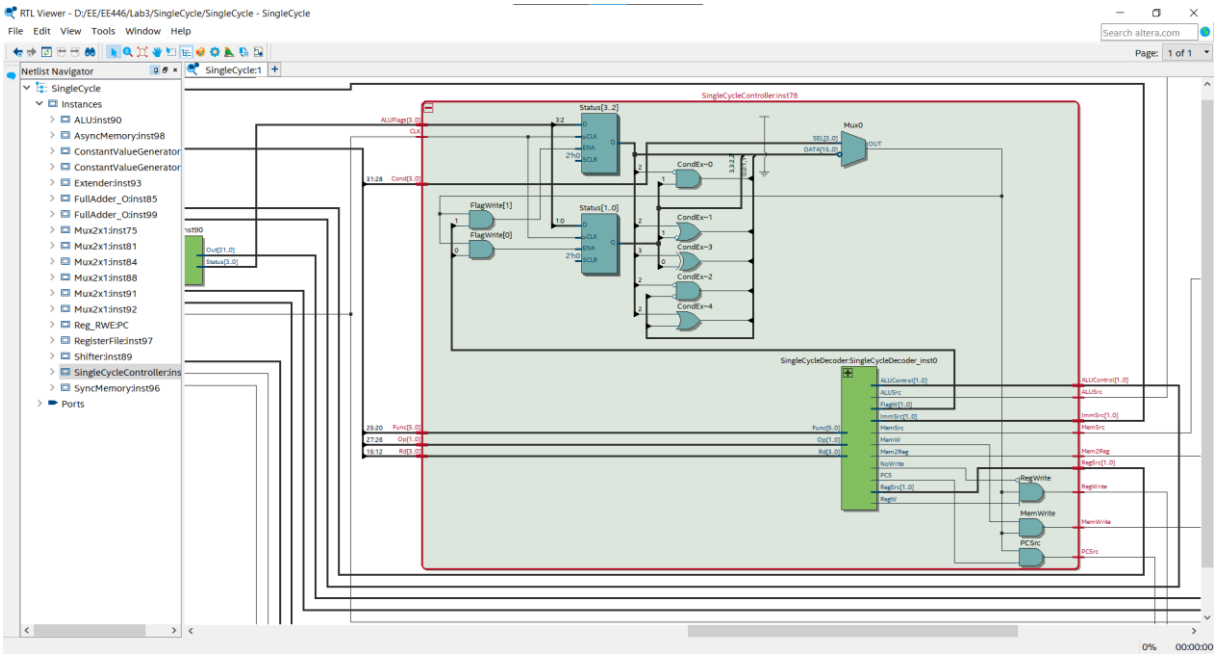
ALUOp	$Funct_{4:1}$ (*cmd*)	$Funct_0$ (*S*)	Type	$ALUControl_{1:0}$	$FlagW_{1:0}$
0	X	X	Not DP	00 (Add)	00
1	0100	0	ADD	00 (Add)	00
		1			11
	0010	0	SUB	01 (Sub)	00
		1			11
	0000	0	AND	10 (And)	00
		1			10
	1100	0	ORR	11 (Or)	00
		1			10

3. Operation Decoder

Table 7.2 Main Decoder truth table

Op	$Funct_5$	$Funct_0$	Type	Branch	MemtoReg	MemW	ALUSrc	ImmSrc	RegW	RegSrc	ALUOp
00	0	X	DP Reg	0	0	0	0	XX	1	00	1
00	1	X	DP Imm	0	0	0	1	00	1	X0	1
01	X	0	STR	0	X	1	1	01	0	10	0
01	X	1	LDR	0	1	0	1	01	1	X0	0
10	X	X	B	1	0	0	1	10	0	X1	0

D. Controller Implementation



III. Parametrization

- All Registers, Memories, Decoders and Muxers are implemented in the way that Data Width are parametric. Hence, we can scale each entity and overall design.
- Memory depth is configured by the help of parametrization so that data and instruction memory fits in the FPGA.
- Complications: Since the top layer (Datapath) is implemented as Board Design File, changing parameters requires manual settings of bus widths.
- This design should work various memory depths.