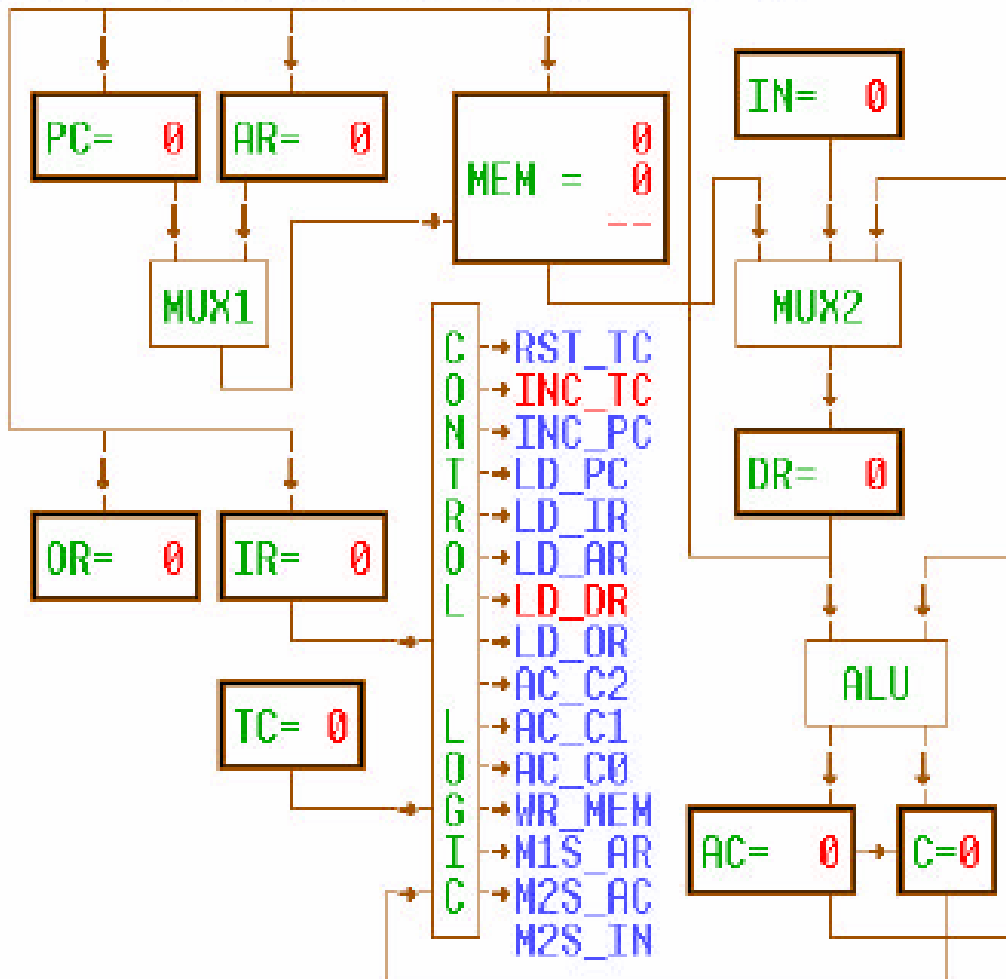


PSIM Architecture



Sequential Logic:

Program Memory (MEM)

Program Counter (PC)

Address Register (AR)

Data Register (DR)

Input Register (IN)

Output Register (OR)

Accumulator (AC)

ALU Carry Register (C)

Instruction Register (IR)

Timing Counter (TC)

Combinational Logic:

Control Logic

Arithmetic/Logic Unit (ALU)

Multiplexers 1&2 (MUX)

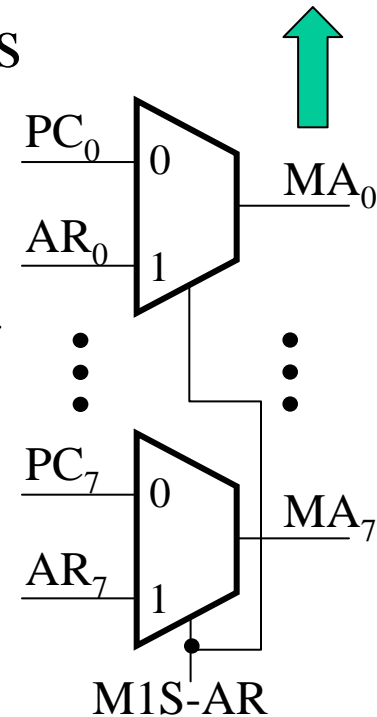
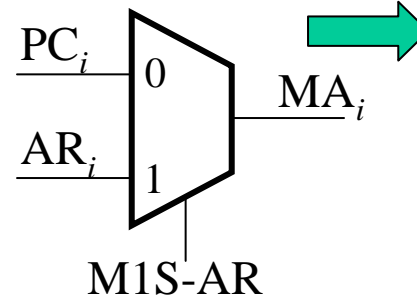
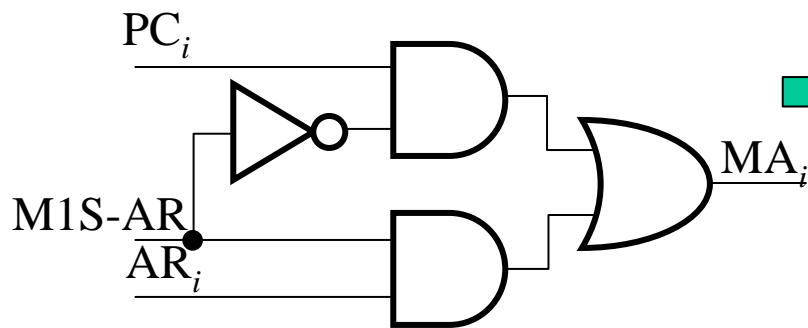
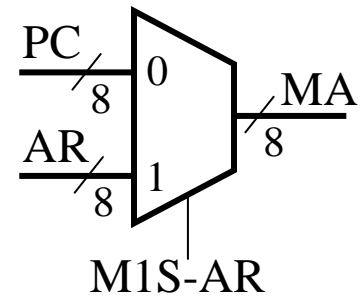
Combinational Logic in PSIM

- 4 combinational logic circuits
 - MUX1
 - Eight 2-to-1 MUXs
 - MUX2
 - Eight 3-to-1 MUXs
 - Control Logic
 - A decoder based on micro-operation sequence
 - Arithmetic Logic Unit (ALU)
 - Produces logic values to go into accumulator (AC) and carry (C) registers
- The rest is sequential logic

MUX1

- Selects memory address (MA) inputs from:
 - Program Counter (PC), when M1S-AR = 0
 - Address Register (AR), when M1S-AR = 1
- Need eight 2-to-1 MUXs for 8-bit address
 - With common select signal (M1S-AR)

bus notation

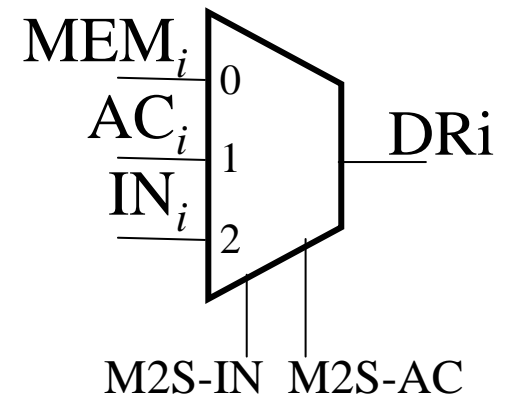
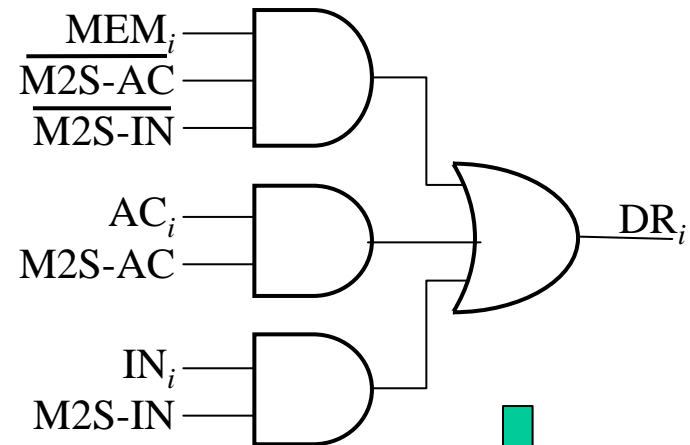
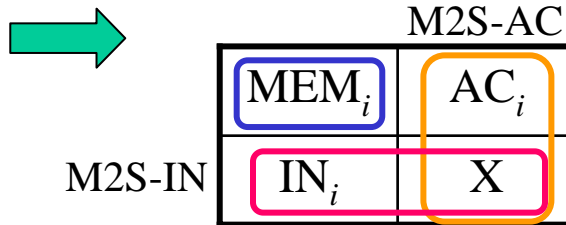


MUX2

- Eight 3-to-1 MUXs select
 - AC when M2S-AC = 1
 - IN when M2S-IN = 1
 - MEM when M2S-AC = 0 and M2S-IN = 0

$$Z = \overline{MEM}_i \cdot \overline{M2S-AC} \cdot \overline{M2S-IN} + AC_i \cdot M2S-AC + IN_i \cdot M2S-IN$$

M2S-IN	M2S-AC	Z
0	0	MEM _i
0	1	AC _i
1	0	IN _i
1	1	X

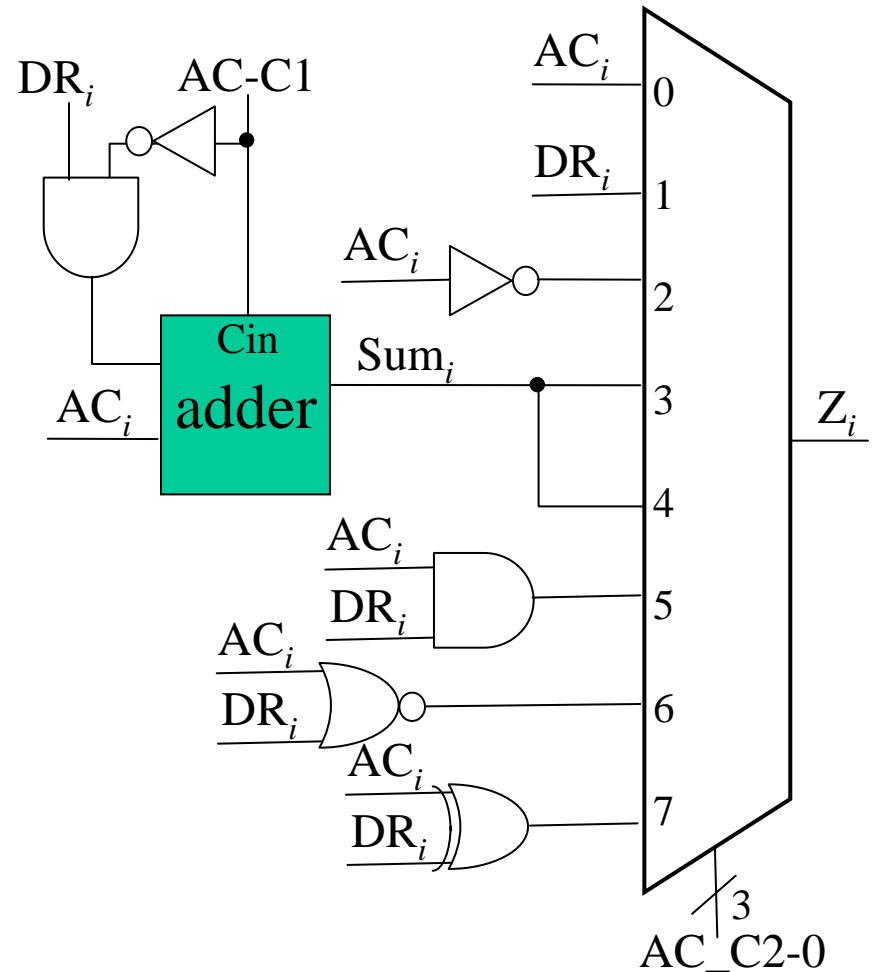


Arithmetic Logic Unit

AC- C2 C1 C0	Operation	Notation
0 0 0	Hold Data	$AC \leftarrow AC$
0 0 1	Load Data	$AC \leftarrow DR$
0 1 0	Complement	$AC \leftarrow AC'$
0 1 1	Increment	$AC \leftarrow AC+1$
1 0 0	Add	$AC \leftarrow AC+DR$
1 0 1	Logical AND	$AC \leftarrow AC \cdot DR$
1 1 0	Logical NOR	$AC \leftarrow (AC+DR)'$
1 1 1	Logical XOR	$AC \leftarrow AC \oplus DR$

8-to-1 MUX control

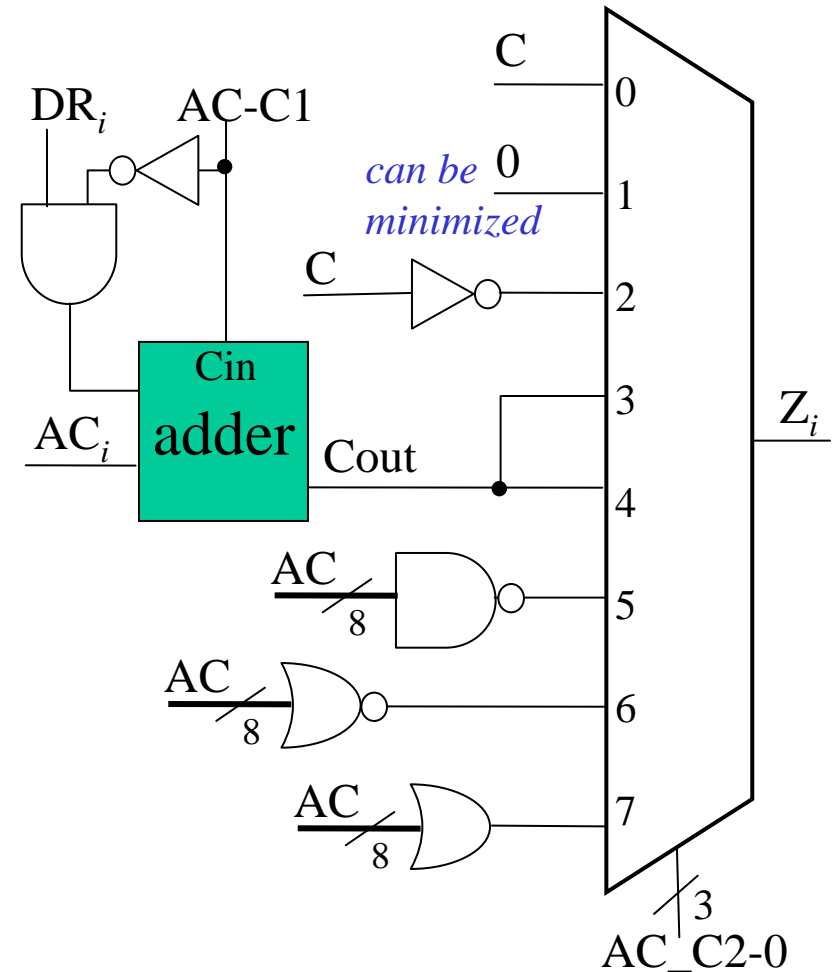
A similar approach taken for C



Arithmetic Logic Unit

AC- C2 C1 C0	Operation	Notation
0 0 0	Hold Data	$C \leftarrow C$
0 0 1	Load Data	$C \leftarrow 0$
0 1 0	Complement	$C \leftarrow C'$
0 1 1	Increment	$C \leftarrow \text{carry-out}$
1 0 0	Add	$C \leftarrow \text{carry-out}$
1 0 1	Logical AND	$C \leftarrow \text{NAND of AC}$
1 1 0	Logical NOR	$C \leftarrow \text{NOR of AC}$
1 1 1	Logical XOR	$C \leftarrow \text{OR of AC}$

8-to-1 MUX control



Control Logic Outputs

All control signals are active high

⇒ Logic 1 when active & logic 0 otherwise

Signal	Operation	Notation
RST_TC	Reset Timing Counter	$TC \leftarrow 0$
INC_TC	Increment Timing Counter	$TC \leftarrow TC+1$
INC_PC	Increment Program Counter	$PC \leftarrow PC+1$
LD_PC	Parallel Load Program Counter	$PC \leftarrow DR$
LD_IR	Parallel Load Instruction Register	$IR \leftarrow DR$
LD_AR	Parallel Load Address Register	$AR \leftarrow DR$
LD_DR	Parallel Load Data Register	$DR \leftarrow \text{MUX2 output data}$
LD_OR	Parallel Load Output Register	$OR \leftarrow DR$
AC_C2-0	ALU/Accumulator Control Signals	on previous slides
WR_MEM	Write Program Memory	$\text{MEM}[\text{MUX1 address}] \leftarrow DR$
M1S_AR	MUX 1 selects Address Register	$\text{MUX1} \leftarrow AR$
M2S_AC	MUX 2 selects Accumulator	$\text{MUX2} \leftarrow AC$
M2S_IN	MUX 2 selects Input Register	$\text{MUX2} \leftarrow IN$

Control Logic

- A big decoder

Condensed Truth Table

- 8 inputs:
 - Instruction Register bits: IR3-0
 - Timing Counter bits: TC2-0
 - Carry Register: C
 - Only for INC_PC
 - Don't care for others
- 15 outputs
 - RST_TC to MS2_IN
- Truth table taken from micro-operation sequence chart
 - example for RST_TC

RST_TC		TC							
Instr	IR	000	001	010	011	100	101	110	111
HLT	0000				X	X	X	X	X
NOP	0001			1	X	X	X	X	X
INA	0010			1	X	X	X	X	X
CMA	0011			1	X	X	X	X	X
ISZ	0100				1	X	X	X	X
LDI	0101				1	X	X	X	X
ADI	0110				1	X	X	X	X
BUN	0111				1	X	X	X	X
STA	1000							1	X
STI	1001							1	X
LDA	1010						1	X	X
LDO	1011						1	X	X
ADA	1100						1	X	X
AND	1101						1	X	X
NOR	1110						1	X	X
XOR	1111						1	X	X

Control Logic Minimization Example

Condensed Truth Table

RST_TC		TC							
Instr	IR	000	001	010	011	100	101	110	111
HLT	0000				X	X	X	X	X
NOP	0001			1	X	X	X	X	X
INA	0010			1	X	X	X	X	X
CMA	0011			1	X	X	X	X	X
ISZ	0100				1	X	X	X	X
LDI	0101				1	X	X	X	X
ADI	0110				1	X	X	X	X
BUN	0111				1	X	X	X	X
STA	1000							1	X
STI	1001							1	X
LDA	1010						1	X	X
LDO	1011						1	X	X
ADA	1100						1	X	X
AND	1101						1	X	X
NOR	1110						1	X	X
XOR	1111						1	X	X

7-variable K-map

IR\TC	000	001	011	010	110	111	101	100
0000			X		X	X	X	X
0001			X	1	X	X	X	X
0011			X	1	X	X	X	X
0010			X	1	X	X	X	X
0110			1		X	X	X	X
0111			1		X	X	X	X
0101			1		X	X	X	X
0100			1		X	X	X	X
1100					X	X	1	
1101					X	X	1	
1111					X	X	1	
1110					X	X	1	
1010					X	X	1	
1011					X	X	1	
1001					1	X		
1000					1	X		

$$\begin{aligned}
 \text{RST_TC} = & \text{TC}_2\text{TC}_1 + \text{TC}_2\text{TC}_0\text{IR}_1 + \text{TC}_2\text{TC}_0\text{IR}_2\text{IR}_1' + \\
 & \text{TC}_1\text{IR}_3'\text{IR}_2'\text{IR}_0 + \text{TC}_1\text{IR}_3'\text{IR}_2'\text{IR}_1 + \text{TC}_1\text{TC}_0\text{IR}_3'
 \end{aligned}$$

Control Logic ASL

```
ckt: cntlog in: ir3 ir2 ir1 ir0 c tc2 tc1 tc0
out: rst-tc inc-tc inc-pc ld-pc ld-ir ld-ar ld-dr ld-or
     wr-mem m1s-ar m2s-ac m2s-in ac-c2 ac-c1 ac-c0 ;
not: tc2n in: tc2 out: tc2n ;
not: tc1n in: tc1 out: tc1n ;
not: tc0n in: tc0 out: tc0n ;
not: ir3n in: ir3 out: ir3n ;
not: ir2n in: ir2 out: ir2n ;
not: ir1n in: ir1 out: ir1n ;
not: ir0n in: ir0 out: ir0n ;
not: cn in: c out: cn ;
and: a1 in: tc2 tc0 ir2 out: a1 ;
and: a2 in: a1 ir1 out: a2 ;
and: a3 in: a1 ir0 out: a3 ;
and: a4 in: tc2 tc0 ir2n ir1 ir0n out: a4 ;
and: a5 in: tc1 ir3n ir2n ir1 ir0n out: a5 ;
and: a6 in: tc1 tc0 ir3n ir1n ir0 out: a6 ;
and: a7 in: tc1 ir3n ir2n ir1 out: a7 ;
and: a8 in: tc1 tc0 ir3n ir1 ir0n out: a8 ;
or: ac-c0 in: a3 a4 a5 a6 out: ac-c0 ;
or: ac-c1 in: a2 a7 out: ac-c1 ;
or: ac-c2 in: a1 a8 out: ac-c2 ;
and: ld-ir in: tc2n tc1n tc0 out: ld-ir ;
and: a10 in: tc2n ir3 out: a10 ;
and: a11 in: tc1n tc0n out: a11 ;
and: a12 in: tc0n ir2 out: a12 ;
and: a13 in: tc1n ir2n ir1n out: a13 ;
and: a14 in: ir3n ir2n ir1n ir0n out: a14 ;
or: inc-tc in: ld-ir a10 a11 a12 a13 out: inc-tc ;
nor: rst-tc in: inc-tc a14 out: rst-tc ;
and: m2s-ac in: tc2 ir2n ir1n ir0n out: m2s-ac ;
and: m2s-in in: tc2 ir2n ir1n ir0 out: m2s-in ;
not: m1s-ar in: tc2n out: m1s-ar ;
and: a20 in: tc2n tc0 ir3 out: a20 ;
and: a21 in: tc0 ir3n ir1 ir0n out: a21 ;
and: a22 in: tc0 ir3n ir2 ir1n ir0 out: a22 ;
and: a23 in: tc1 ir3n ir2 ir1n ir0n cn out: a23 ;
or: inc-pc in: ld-ir a20 a21 a22 a23 out: inc-pc ;
and: ld-pc in: tc1 tc0 ir3n ir1 ir0 out: ld-pc ;
and: ld-ar in: tc1 tc0 ir3 out: ld-ar ;
and: a24 in: tc2n tc0n ir3 out: a24 ;
and: a25 in: tc0n ir3n ir2 ir1 out: a25 ;
and: a26 in: tc0n ir3n ir2 ir0 out: a26 ;
or: ld-dr in: a11 a24 a25 a26 out: ld-dr ;
and: ld-or in: tc2 tc0 ir2n ir1 ir0 out: ld-or ;
and: wr-mem in: tc2 tc0 ir2n ir1n out: wr-mem ;
```

Control Logic AUSIM Results

- Area Analysis:
 - $G = 43$
 - $G_{IO} = 177$
- Gate types and # of uses:
 - AND: 27
 - OR: 6
 - NOT: 9
 - NAND: 0
 - NOR: 1

*only a portion of
157 total input
vectors for design
verification*

```
# AUSIM (2.0) Simulation Results ;
#          rii          wmmm          ;
#          snnlllllr122aaa ;
#          tccddddd-sssccc ;
# iiii ttt -----m----- ;
# rrrr ccc ttpiadooaaiccc ;
# 3210c210 cccrrrrmrcn210 ;
# fetch ;
00002000 010000100000000
00012000 010000100000000
00102000 010000100000000
00112000 010000100000000
01002000 010000100000000
01012000 010000100000000
01102000 010000100000000
01112000 010000100000000
10002000 010000100000000
10012000 010000100000000
10102000 010000100000000
10112000 010000100000000
11002000 010000100000000
11012000 010000100000000
11102000 010000100000000
11112000 010000100000000
```

Simulating C input as a don't care →