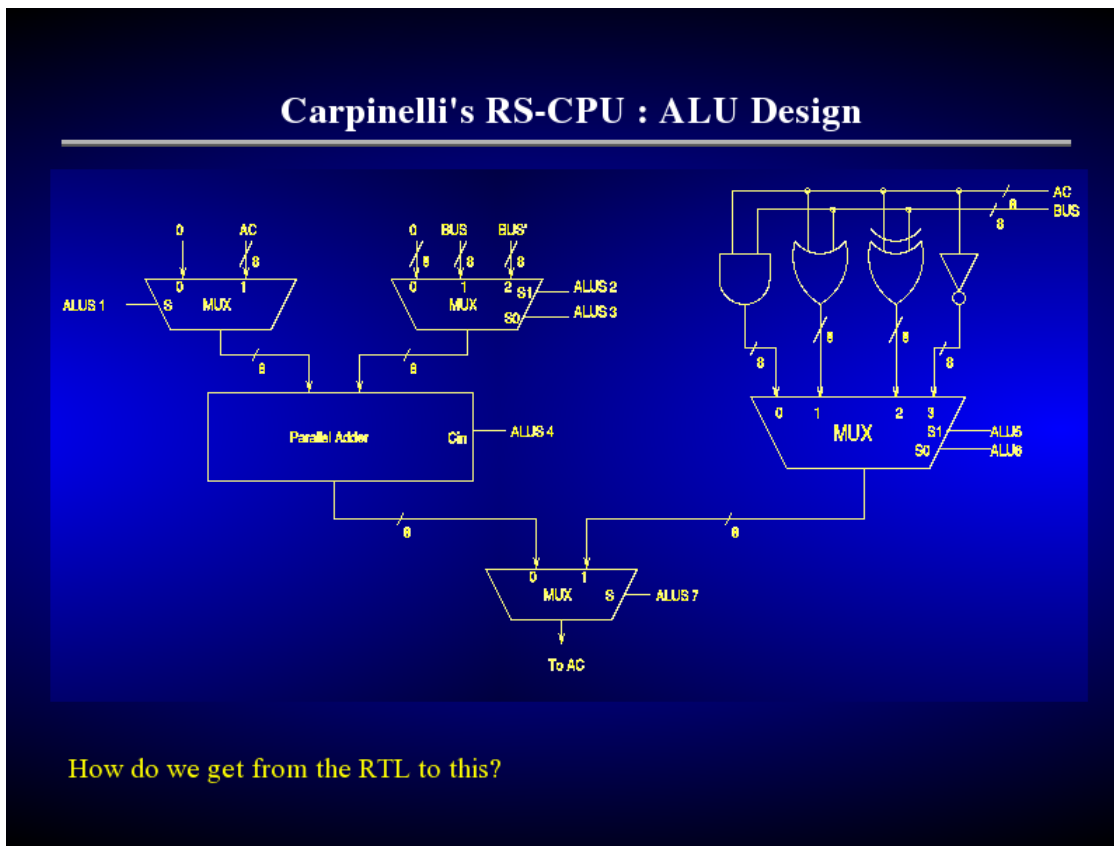


[\[index\]](#) [\[text page\]](#) [[<<start](#)] [[<prev](#)] [[next>](#)] [[last>>](#)]

Page 1: Carpinelli's RS-CPU : ALU Design



Generated by [MagicPoint](#)

Carpinelli's RS-CPU : ALU Design

First collate all transfers that modify the Accumulator

LDAC5:	AC	←	DR		AC	←	BUS
MOVR1:	AC	←	R		AC	←	BUS
ADD1 :	AC	←	AC + R		AC	←	AC + BUS
SUB1 :	AC	←	AC - R		AC	←	AC - BUS
INAC1:	AC	←	AC + 1		AC	←	AC + 1
CLAC1:	AC	←	0		AC	←	0
AND1 :	AC	←	AC \wedge R				
OR1 :	AC	←	AC \vee R				
XOR1 :	AC	←	AC \oplus R				
NOT1 :	AC	←	AC'				

Arithmetic instructions
showing operand source

[\[index\]](#) [\[text page\]](#) [\[<<start\]](#) [\[<prev\]](#) [\[next>\]](#) [\[last>>\]](#)

Page 3: Carpinelli's RS-CPU : Arithmetic Unit

Carpinelli's RS-CPU : Arithmetic Unit

Rewrite each operation as the sum of two values and a carry

LDAC5: AC \leftarrow 0 + BUS + 0

MOVR1: AC \leftarrow 0 + BUS + 0

ADD1 : AC \leftarrow AC + BUS + 0

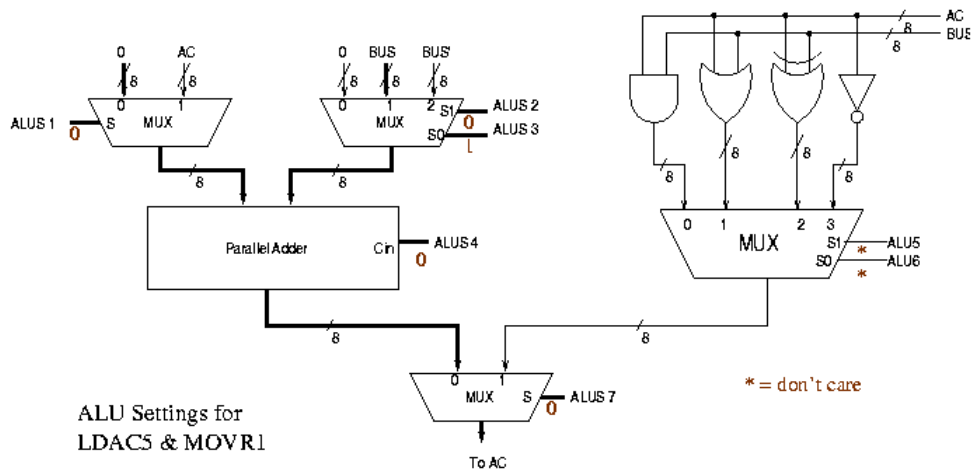
SUB1 : AC \leftarrow AC + BUS' + 1

INAC1: AC \leftarrow AC + 0 + 1

CLAC1: AC \leftarrow 0 + 0 + 0

Generated by [MagicPoint](#)

Implementing ALU datapath for LDAC5 and MOVR1



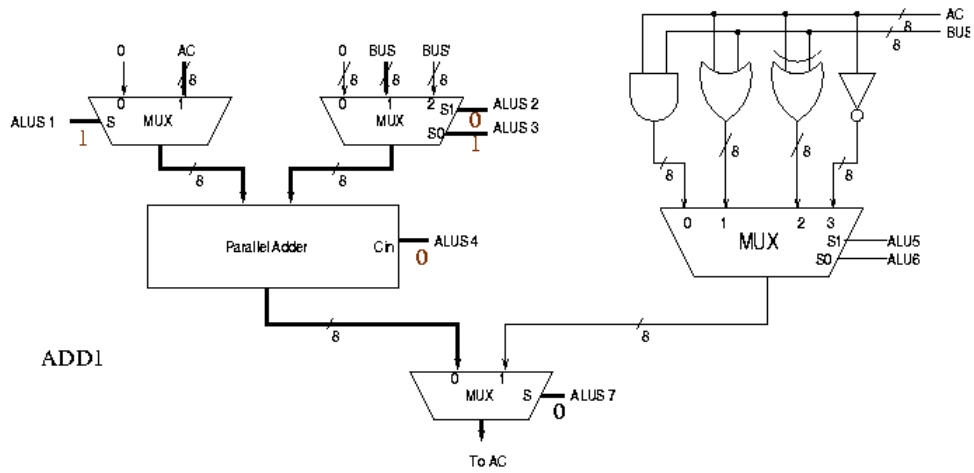
Gotchas :

- Alu signals numbered ALUS[1 .. 7]
- ALUS3 ALUS6 assigned to mux input S0
- ALUS2 ALUS5 assigned to mux input S1

[\[index\]](#) [\[text page\]](#) [\[<<start\]](#) [\[<prev\]](#) [\[next>\]](#) [\[last>>\]](#)

Page 5: Implementing ALU datapath for ADD1

Implementing ALU datapath for ADD1

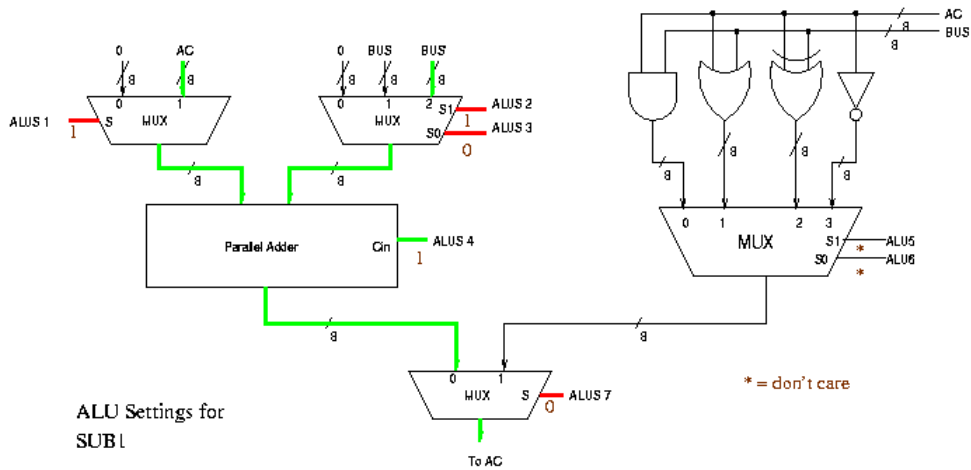


Generated by [MagicPoint](#)

[\[index\]](#) [\[text page\]](#) [\[<<start\]](#) [\[<prev\]](#) [\[next>\]](#) [\[last>>\]](#)

Page 6: Implementing ALU datapath for SUB1

Implementing ALU datapath for SUB1



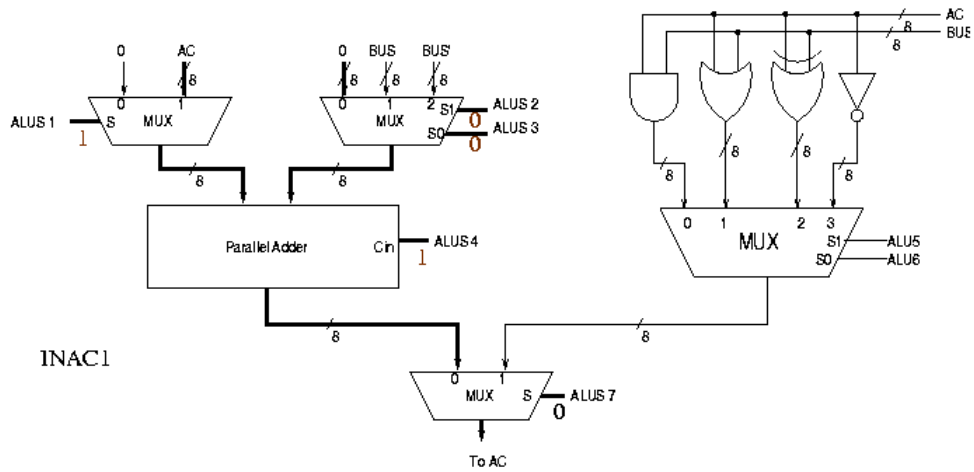
Note that the 'Carry In' is set to '1'.
2's complement !!

Generated by [MagicPoint](#)

[\[index\]](#) [\[text page\]](#) [\[<<start\]](#) [\[<prev\]](#) [\[next>\]](#) [\[last>>\]](#)

Page 7: Implementing ALU datapath for INAC1

Implementing ALU datapath for INAC1

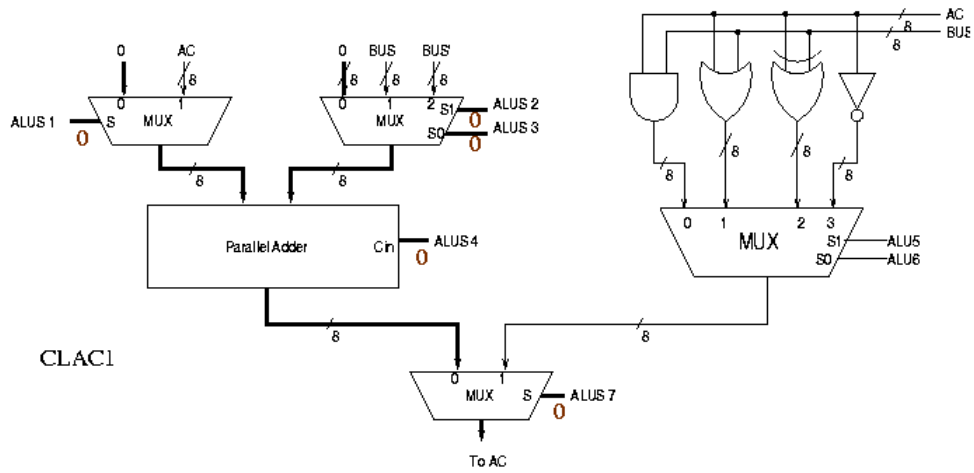


Generated by [MagicPoint](#)

[\[index\]](#) [\[text page\]](#) [\[<<start\]](#) [\[<prev\]](#) [\[next>\]](#) [\[last>>\]](#)

Page 8: Implementing ALU datapath for CLAC1

Implementing ALU datapath for CLAC1

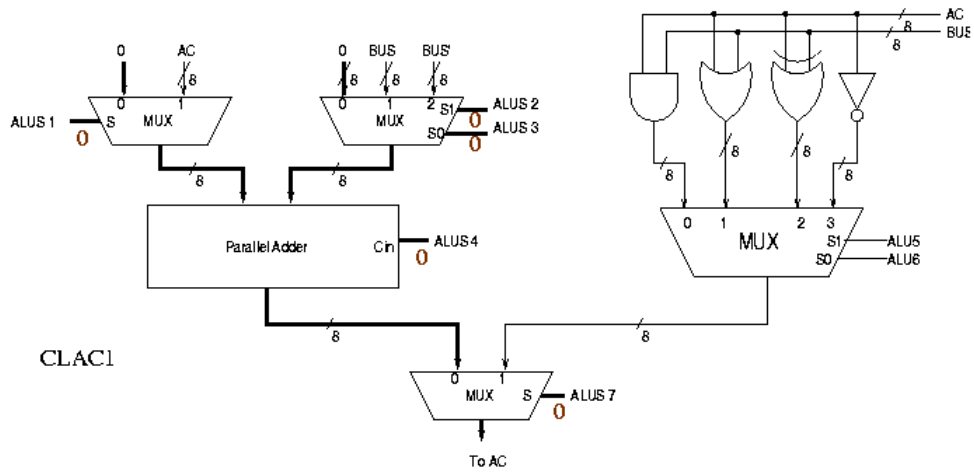


Generated by [MagicPoint](#)

[\[index\]](#) [\[text page\]](#) [\[<<start\]](#) [\[<prev\]](#) [\[next>\]](#) [\[last>>\]](#)

Page 8: Implementing ALU datapath for CLAC1

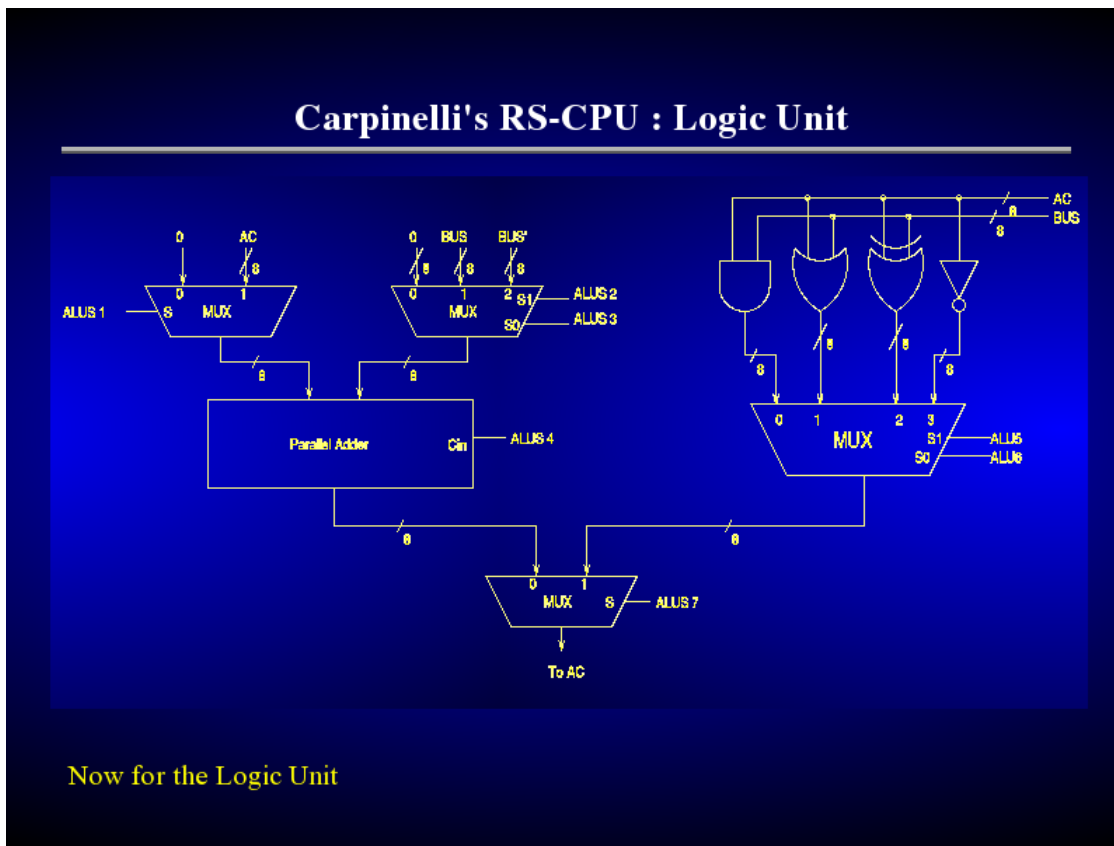
Implementing ALU datapath for CLAC1



Generated by [MagicPoint](#)

[\[index\]](#) [\[text page\]](#) [\[<<start\]](#) [\[<prev\]](#) [\[next>\]](#) [\[last>>\]](#)

Page 9: Carpinelli's RS-CPU : Logic Unit



Now for the Logic Unit

Generated by [MagicPoint](#)

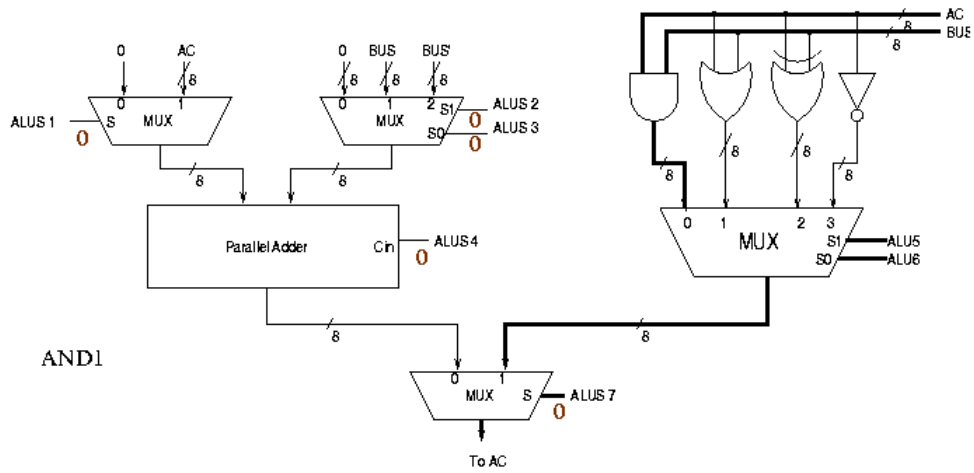
Carpinelli's RS-CPU : Logic Unit

Again, collate the operations by destination

AND1	:	AC	←	$AC \wedge R$	AC	←	$AC \wedge BUS$
OR1	:	AC	←	$AC \vee R$	AC	←	$AC \vee BUS$
XOR1	:	AC	←	$AC \oplus R$	AC	←	$AC \oplus BUS$
NOT1	:	AC	←	AC'	AC	←	AC'

Logic instructions
showing operand source

Implementing ALU datapath for AND1



Generated by [MagicPoint](#)