Final Project Presentation

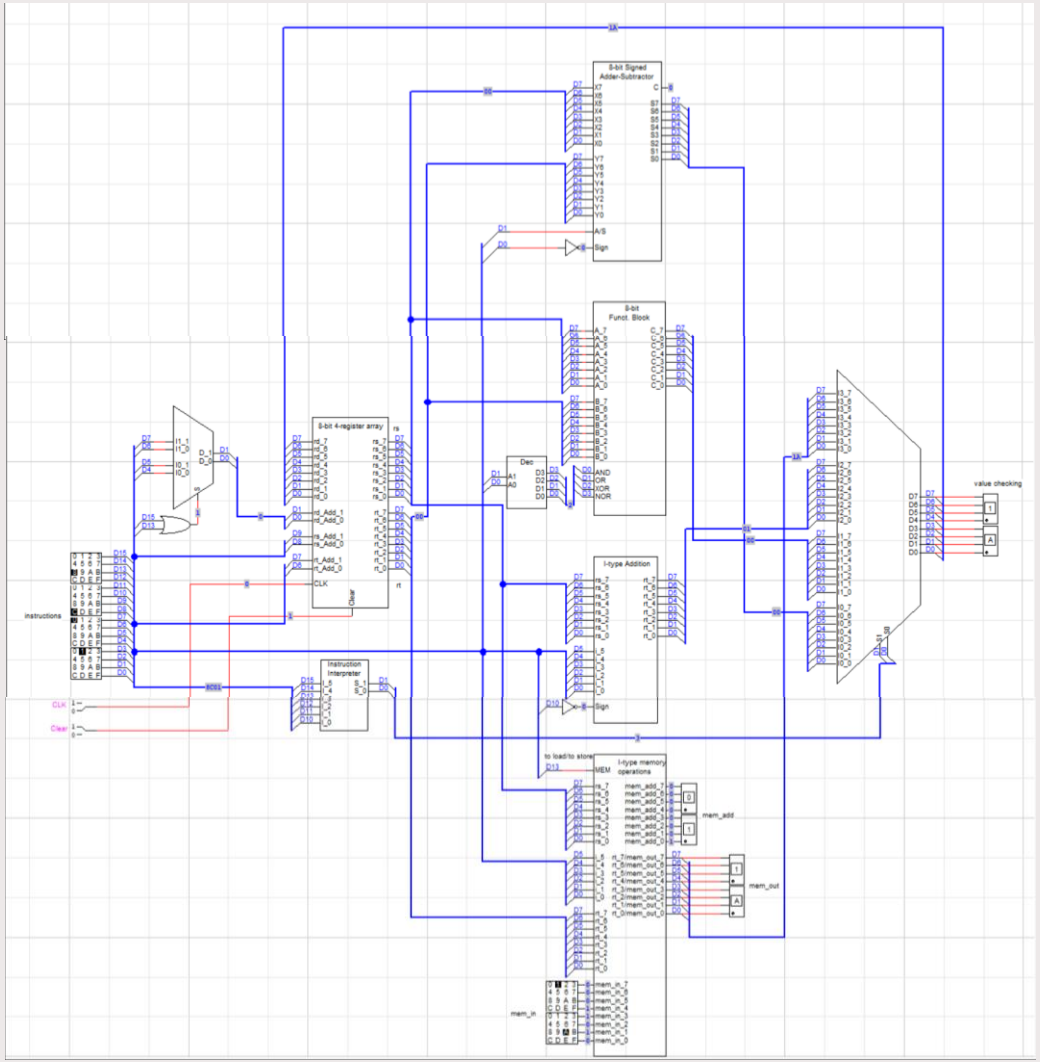
By: Jennifer Olive Tanojo (811601)

Contents

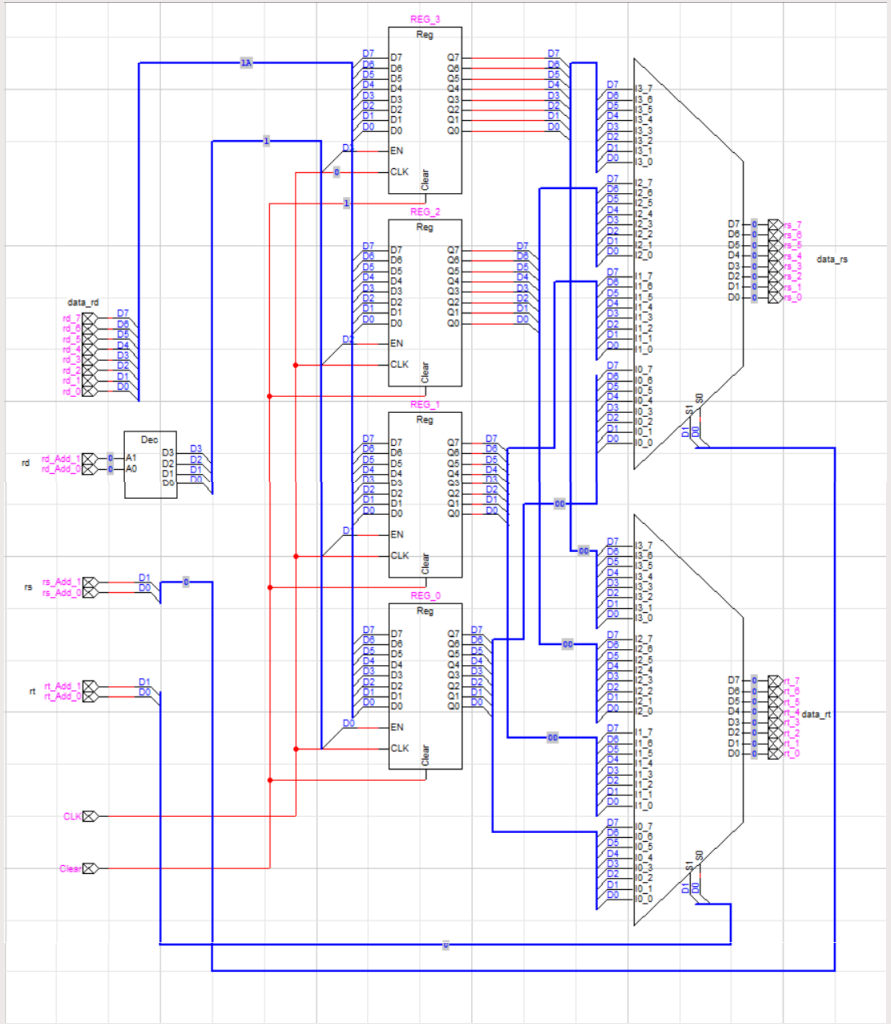
01

Circuit Design Overview and Register Array

Circuit Design

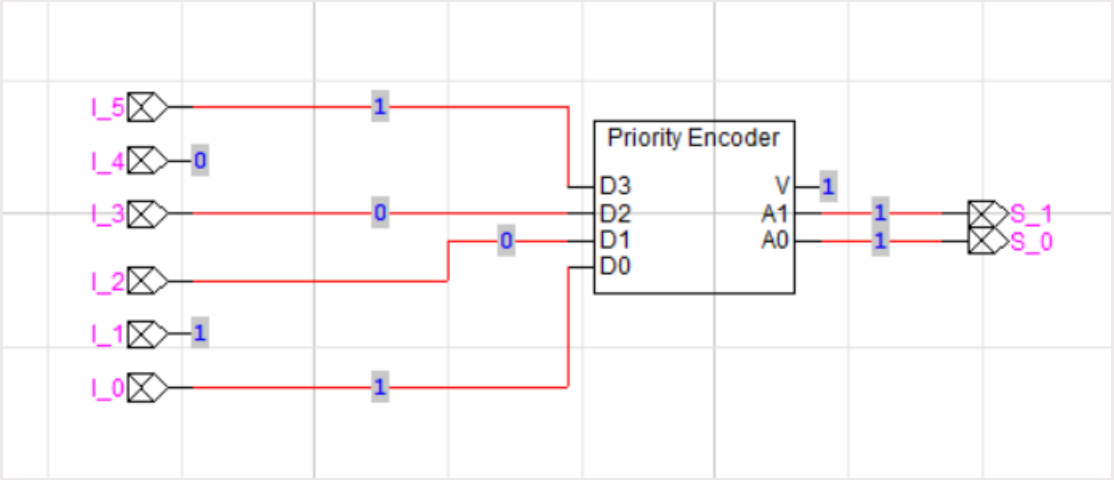


Register Array



02

Instruction Interpreter



Utilizing 4-to-2 Encoder to interpret instruction (opcode) to the multiplexer

Encoder Truth Table

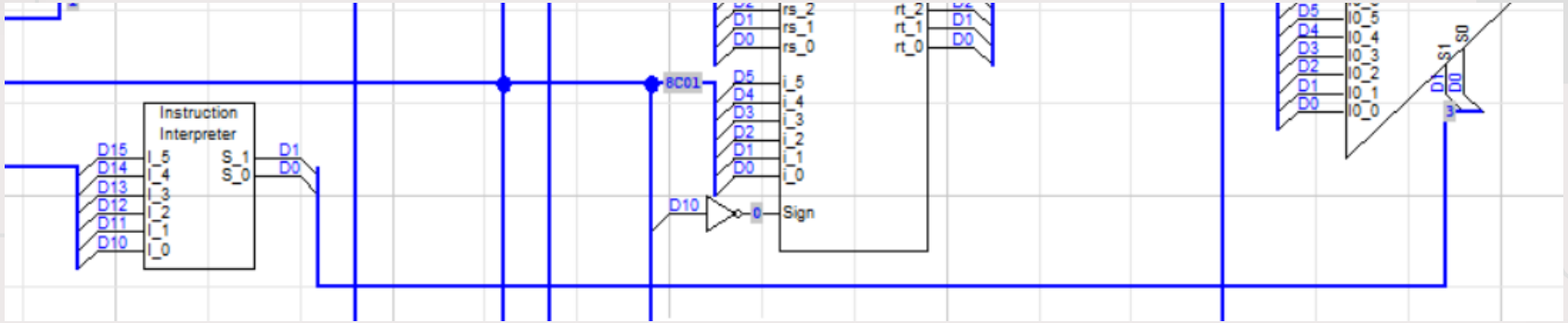
I_5	I_3	I_2	I_0	A_1	A_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

If opcode: $\mathbf{0}0\mathbf{0}0\mathbf{0}\mathbf{0}$, then $A_1A_0 = 00$

If $opcode$: $\mathbf{0}0\mathbf{0}1\mathbf{0}\mathbf{0}$, then $A_1A_0 = 01$

If $opcode$: $\mathbf{0}0\mathbf{1}0\mathbf{0}\mathbf{0}$ or $opcode$: $\mathbf{0}0\mathbf{1}0\mathbf{0}\mathbf{1}$, then $A_1A_0 = 10$

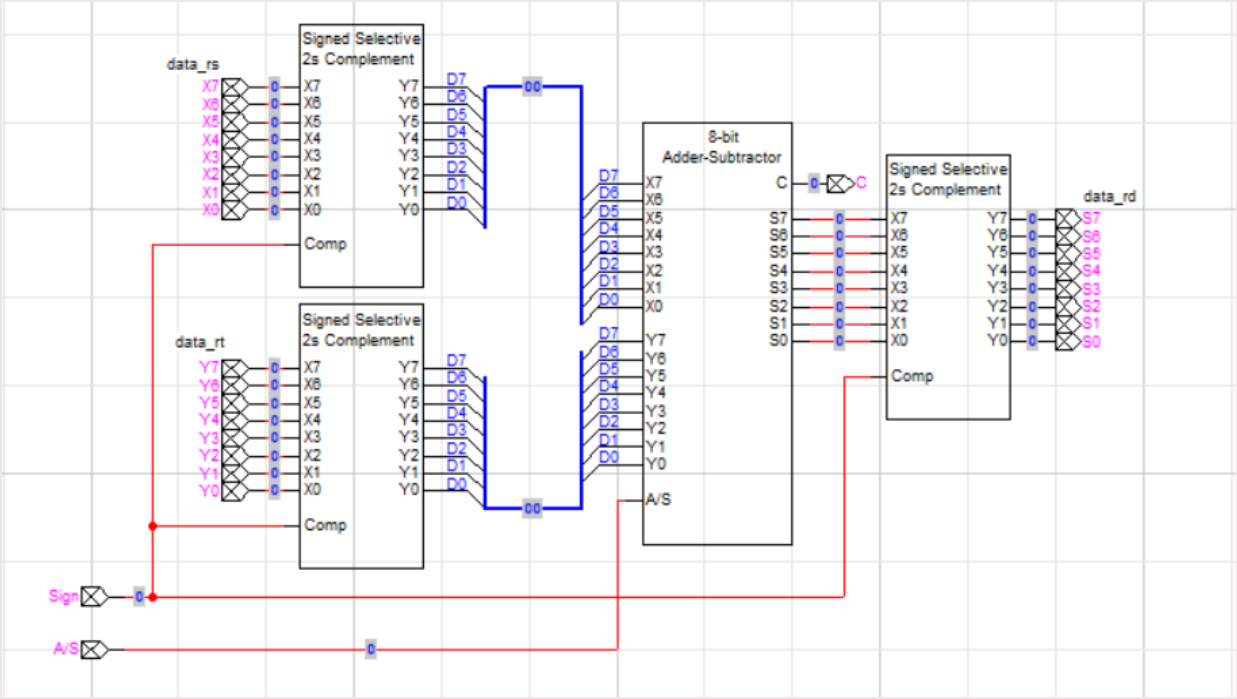
If $opcode$: $\mathbf{1}0\mathbf{0}\mathbf{0}1\mathbf{1}$ or $opcode$: $\mathbf{1}0\mathbf{1}\mathbf{0}1\mathbf{1}$, then $A_1A_0 = 11$



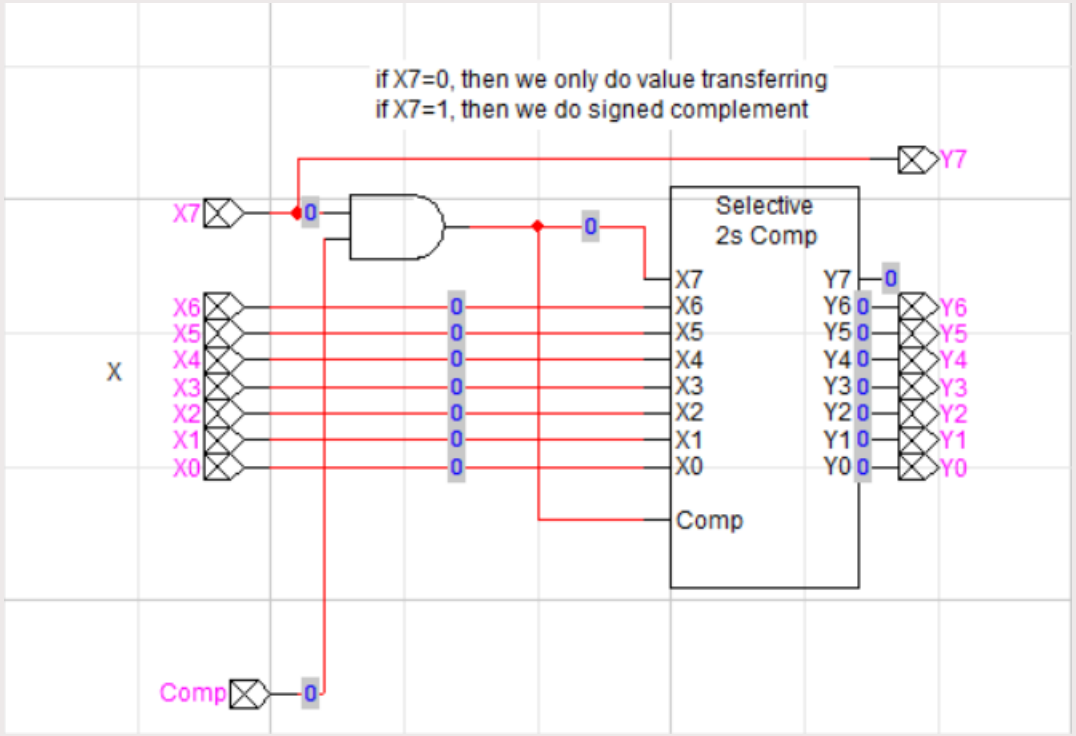
Interpret instruction (opcode) to the multiplexer

03

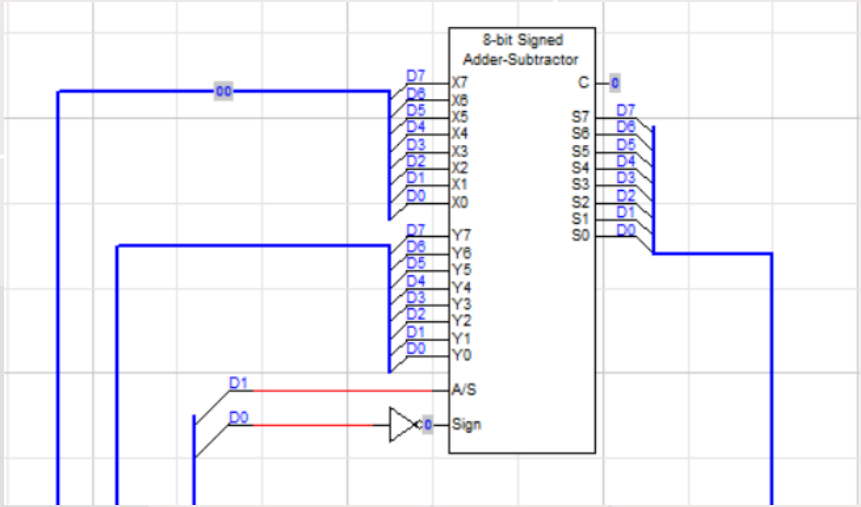
R-type Instruction: 8-bit Signed Adder-Subtractor



Utilizing Signed Selective 2s Complement to help perform signed operations, and 8-bit Adder-Subtractor to Add/Subtract data values



This picture shows how the Signed Selective 2-s Complement works



In the circuit, a NOT gate is used for the Sign input port to match the instruction ($funct$)

- If $opcode$: 000000 and $funct$: 00, then signed addition will be performed
- If $opcode$: 000000 and $funct$: 10, then signed subtraction will be performed
- If $opcode$: 000000 and $funct$: 01, then unsigned addition will be performed
- If $opcode$: 000000 and $funct$: 11, then unsigned subtraction will be performed

04

R-type Instruction: 8- bit AND-OR-XOR-NOR Functional Block

Specification

- Input: 1-bit A, 1-bit B

- Mode: $M_{AND}, M_{OR}, M_{XOR}, M_{NOR}$

  Only one of the modes can be 1, if all modes are 0, then do value transfer of A

- Output: 1-bit C

  For $M_{AND} = 1,\ C = A \cdot B$

  For $M_{OR} = 1,\ C = A + B$

  For $M_{XOR} = 1,\ C = A \oplus B$

  For $M_{NOR} = 1,\ C = \overline{A + B}$

  If $M_{AND} + M_{OR} + M_{NOR} + M_{XOR} = 0,\ C = A$

Formulation

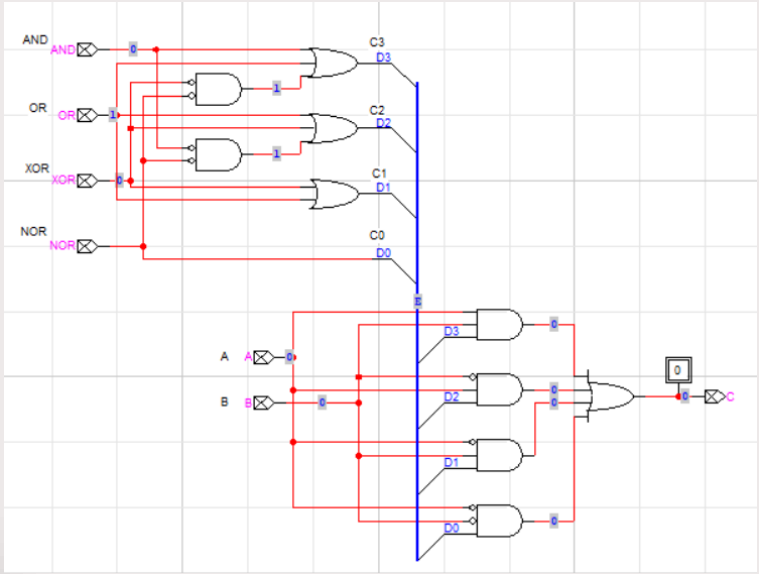
A	$AND = 0$ $OR = 0$ $XOR = 0$ $NOR = 0$	$AND = 1$		$OR = 1$		$XOR = 1$		$NOR = 1$	
		$B = 0$	$B = 1$	$B = 0$	$B = 1$	$B = 0$	$B = 1$	$B = 0$	$B = 1$
0	0	0	0	0	1	0	1	1	0
1	1	0	1	1	1	1	0	0	0

$$C = M_{AND}(A \cdot B) + M_{OR}(A + B) + M_{XOR}(A \oplus B) + M_{NOR}\left(\overline{A + B}\right) + \overline{M_{AND}} \cdot \overline{M_{OR}} \cdot \overline{M_{XOR}} \cdot \overline{M_{NOR}} \cdot A$$

$$C = M_{AND} \cdot m_3 + M_{OR} \cdot \Sigma m(1,2,3) + M_{XOR} \cdot \Sigma m(1,2) + M_{NOR} \cdot m_0 + \overline{M_{AND}} \cdot \overline{M_{OR}} \cdot \overline{M_{XOR}} \cdot \overline{M_{NOR}} \cdot \Sigma m(2,3)$$

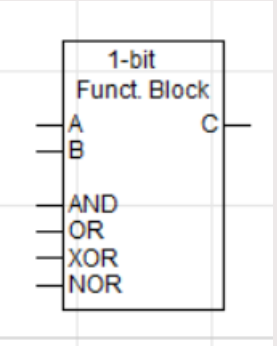
Optimisation

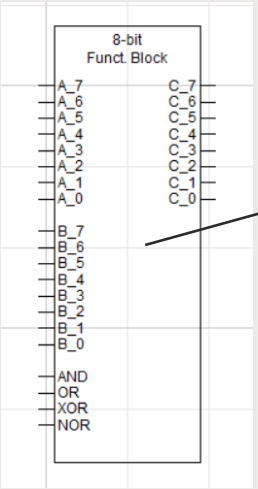
$$C = m_3\left(M_{AND} + M_{OR} + \overline{M_{XOR}} \cdot \overline{M_{NOR}}\right)$$
$$+m_2\left(M_{OR} + M_{XOR} + \overline{M_{AND}} \cdot \overline{M_{NOR}}\right)$$
$$+m_1\left(M_{OR} + M_{XOR}\right)$$
$$+m_0\left(M_{NOR}\right)$$

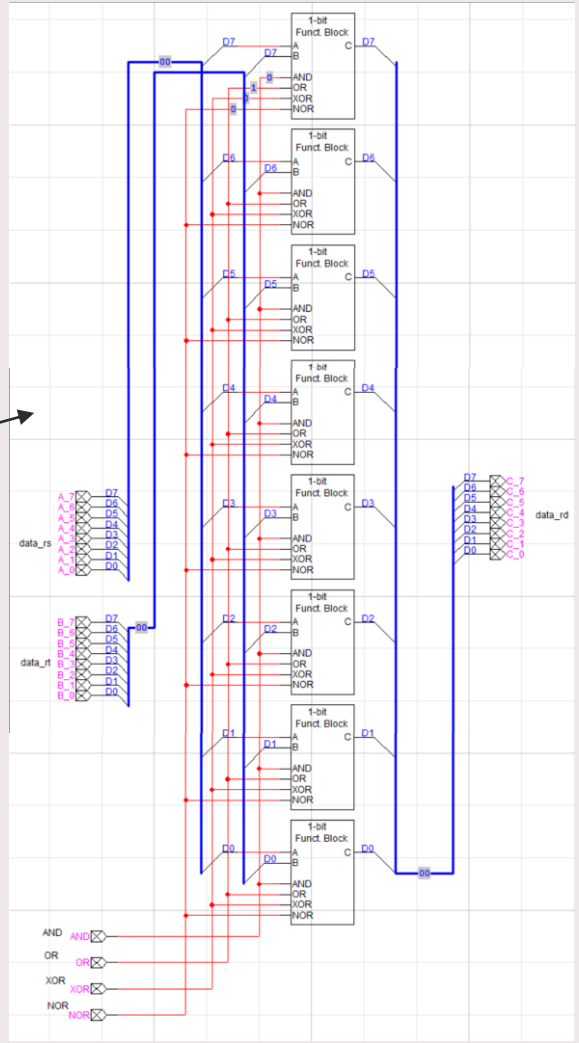


Technology Mapping

This is the 1-bit AND-OR-XOR-NOR Functional Block



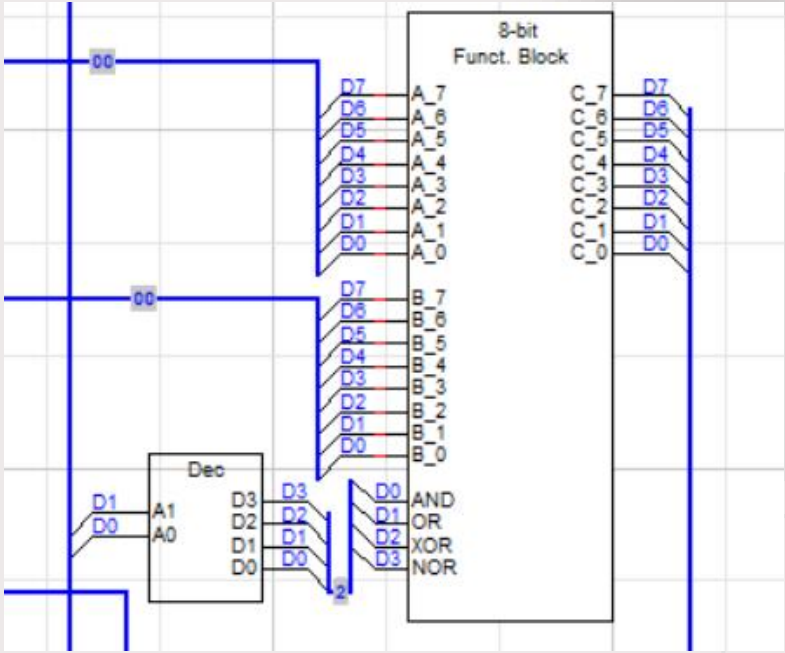




This is the 8-bit AND-OR-XOR-NOR Functional Block

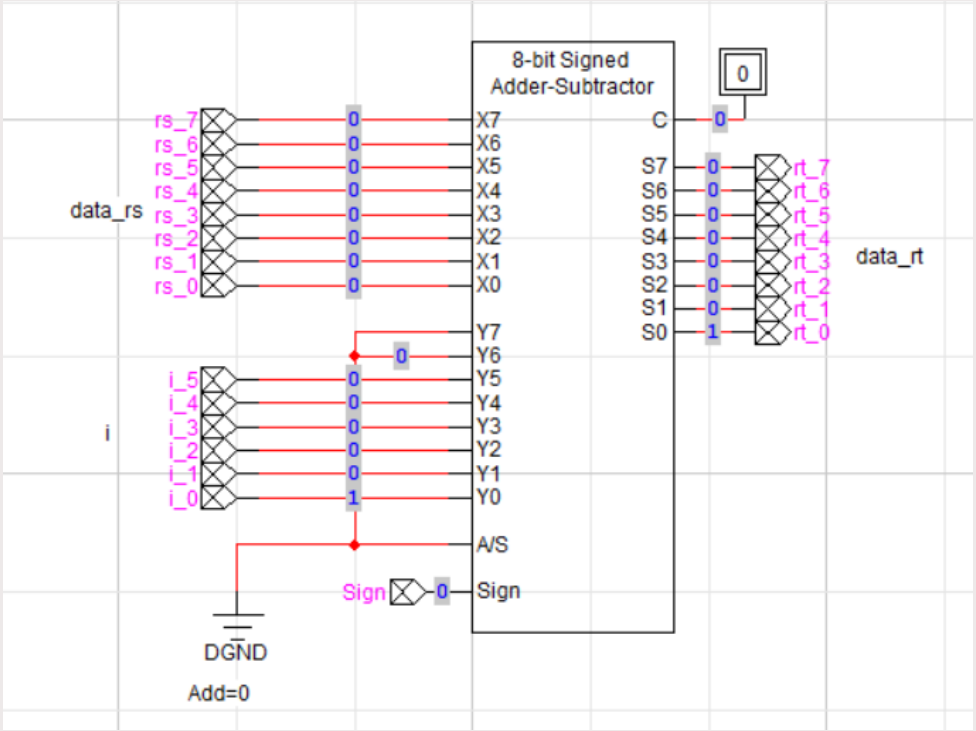
In the circuit, the 2-to-4 Decoder is used to decide the mode (AND/OR/XOR/NOR) according to the instruction ($funct$)

- If $opcode$: 000100 and $funct$: 00, then bitwise AND will be performed
- If $opcode$: 000100 and $funct$: 01, then bitwise OR will be performed
- If $opcode$: 000100 and $funct$: 10, then bitwise XOR will be performed
- If $opcode$: 000100 and $funct$: 11, then bitwise NOR will be performed

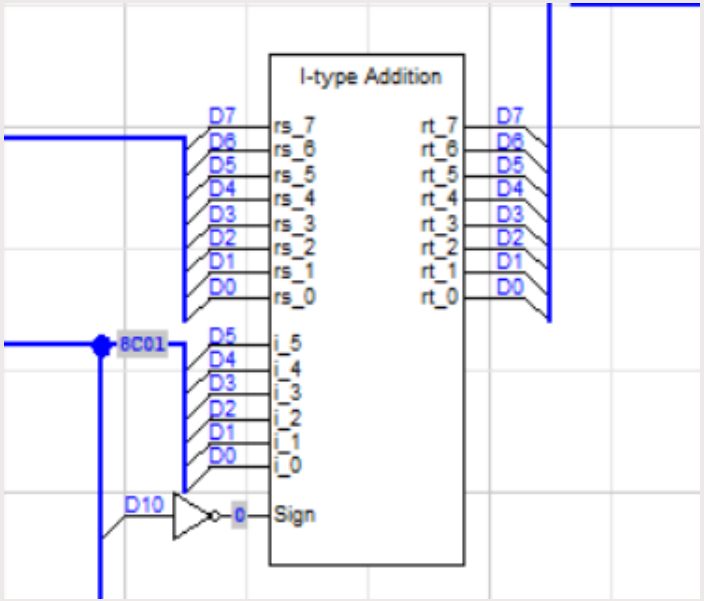


05

I-type Instruction: 8-bit and 6-bit Signed Adder



The $rs + i$ Adder is a modified version of the 8-bit Signed Adder-Subtractor



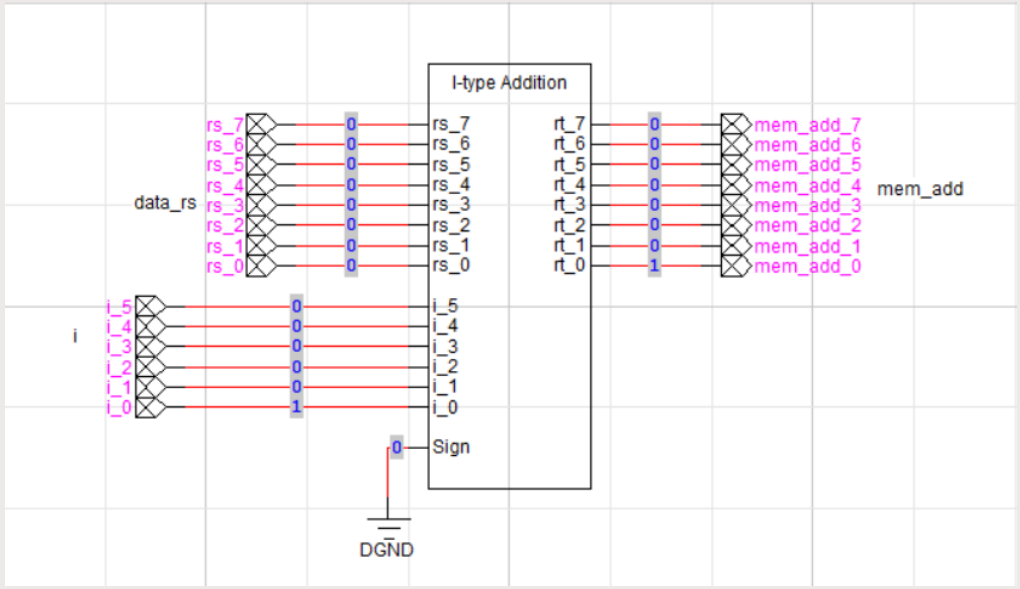
In the circuit, a NOT gate is used for the Sign input port to match the instruction ($opcode$)

- If $opcode$: 001000, then signed $rs + i$ addition will be performed
- If $opcode$: 001001, then unsigned $rs + i$ addition will be performed

06

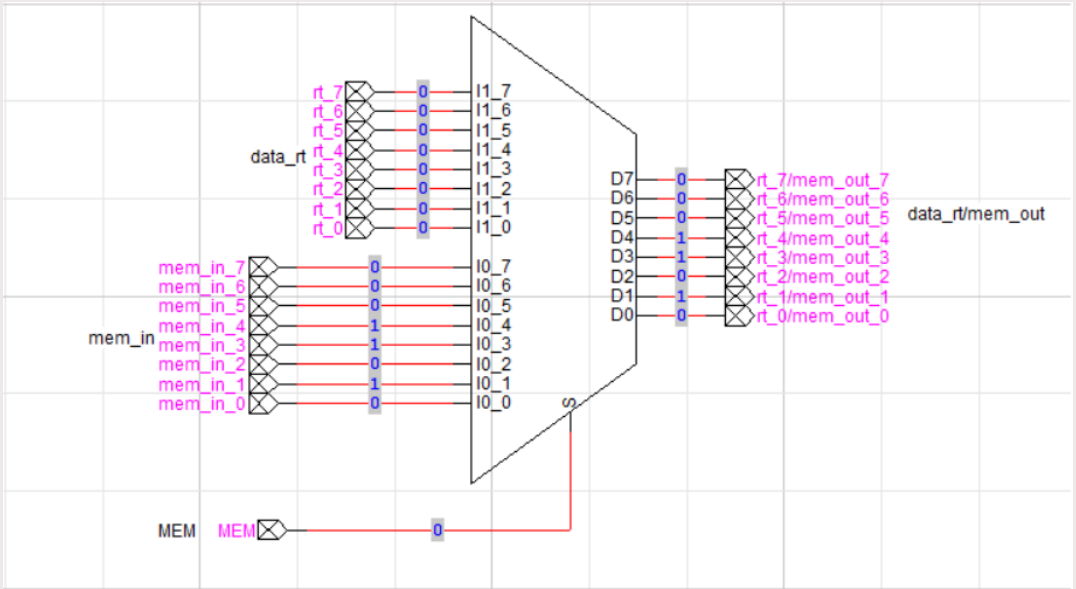
I-type Instruction: Memory Operations Functional Block

Part 1

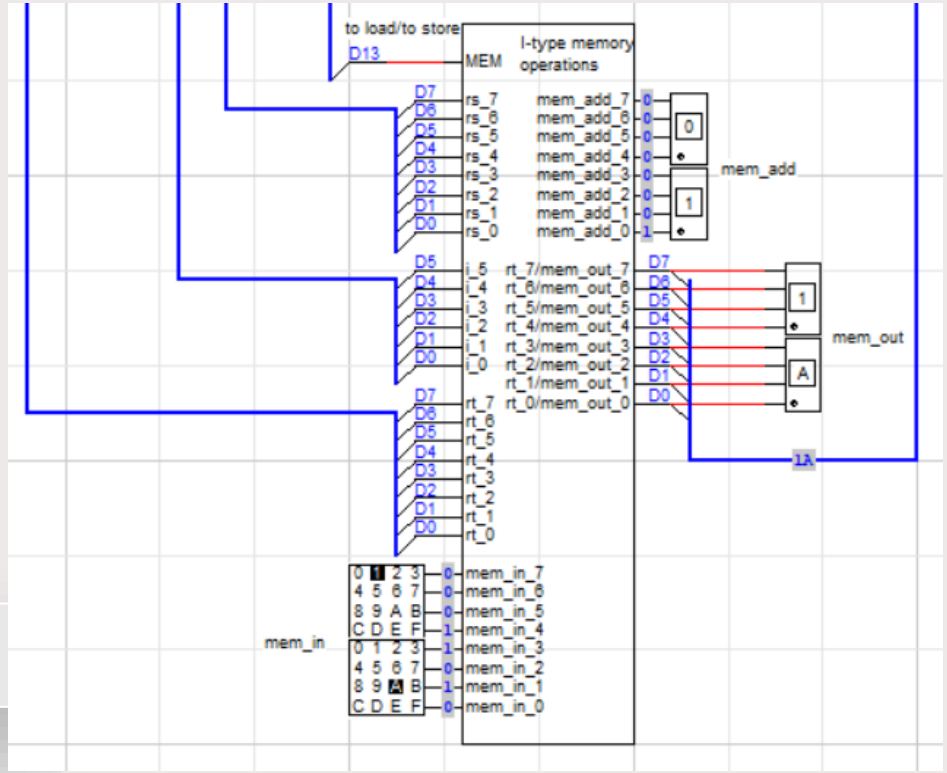


Using the $rs + i$ Adder to specify the address of a memory location (mem_add)

Part 2



Using a 2-channel 8-bit Multiplexer to output either the new data to be retrieved from the memory (rt) or the new data to be stored in memory (mem_out) according to the instruction (opcode)

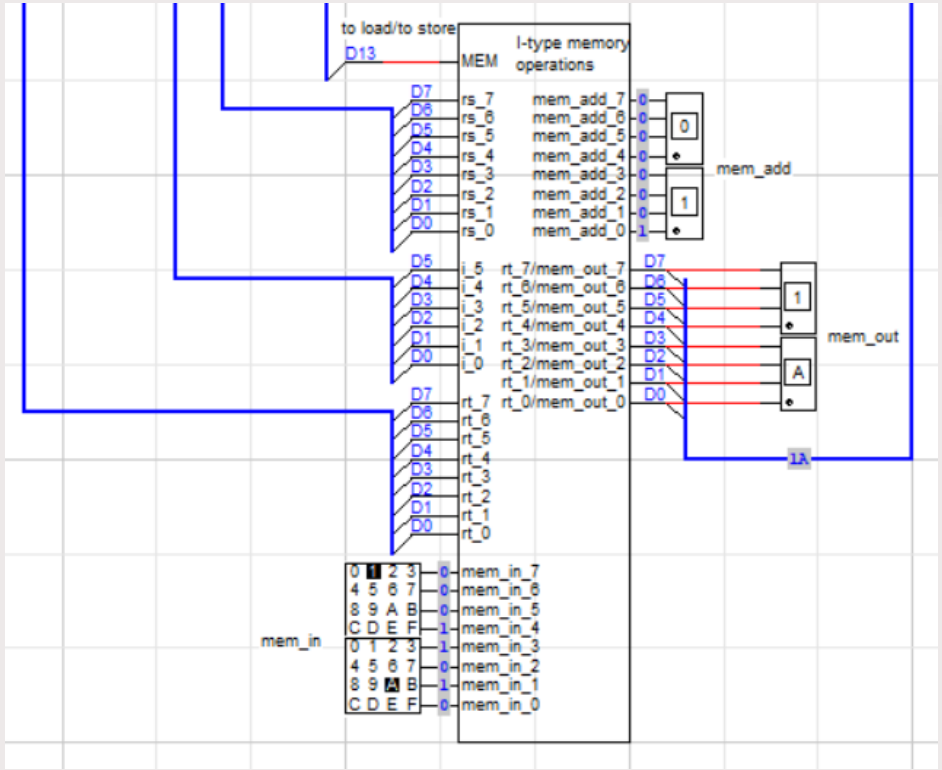


In the circuit;

- One pair of 2x hex keyboard is used to input the new data to be stored in memory
- One pair of 2x hex display is used to specify the address of a memory location
- Another pair of 2x hex display is used to output the new data to be stored in memory

- If $opcode$: $10\mathbf{0}011$, then the new data inputted/retrieved from the specified memory (mem_in) will be stored into rt

- If $opcode$: $10\mathbf{1}011$, then the data value of rt will be the new data to be stored in memory (mem_out), in the specified address



Thank you!

CREDITS: This presentation template was created by **Slidesgo,** and includes icons by **Flaticon,** and infographics & images by **Freepik**