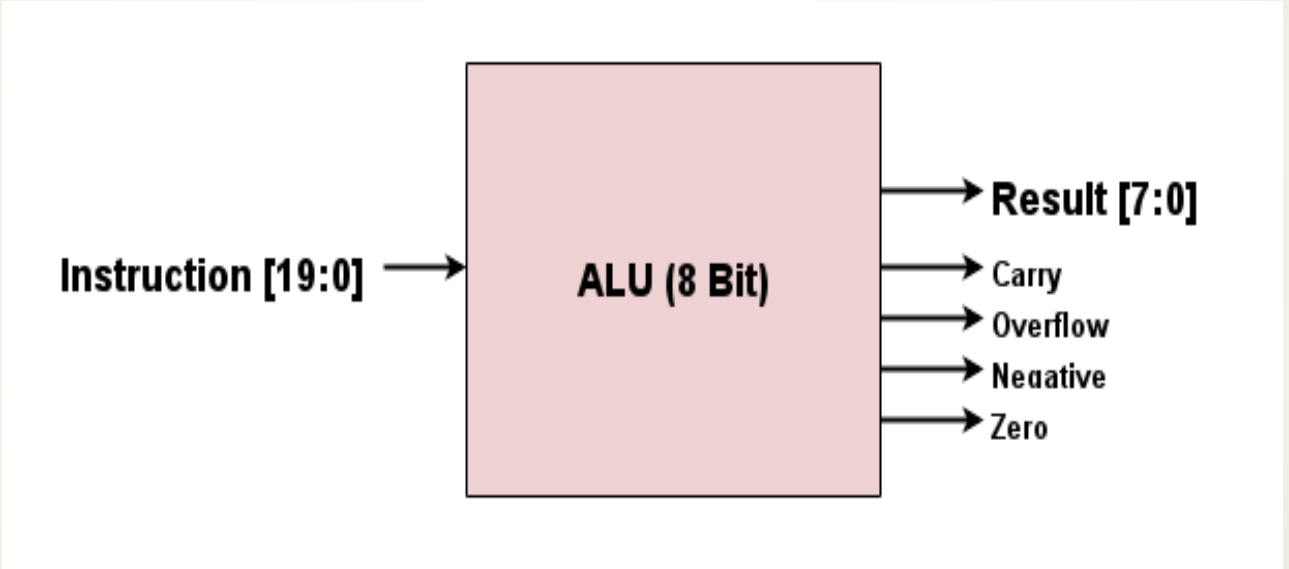
Design and Implementation of 8 Bit ALU and CPU on Xilinx using Verilog HDL



Mir Tafseer Nayeem (mir.nayeem@uleth.edu)

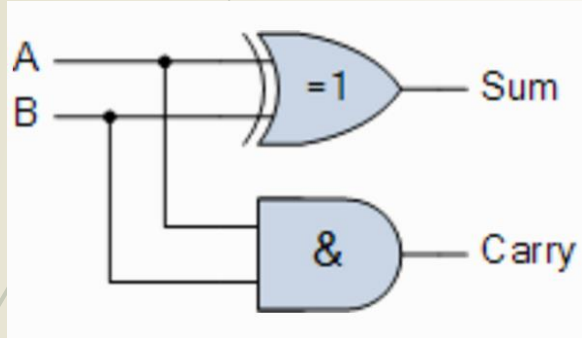
Submitted to : Prof. Hua Li

Department of Mathematics and Computer Science, University of Lethbridge

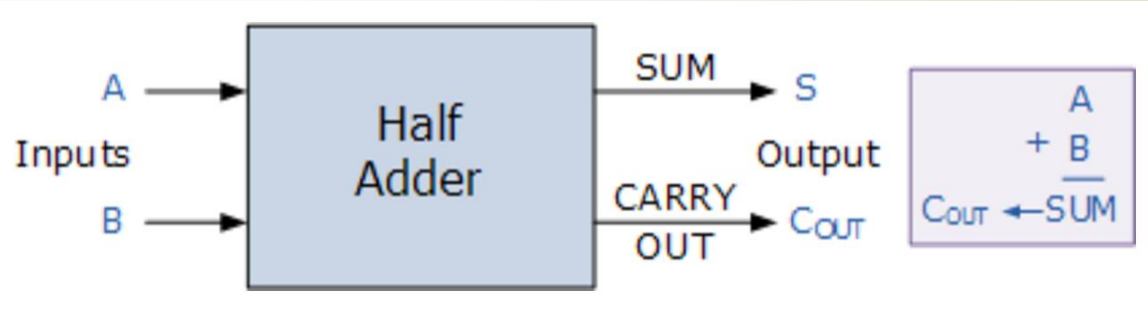
Outline

- Design of Arithmetic Unit
- Design of Logic Unit
- Design of Shift Unit
- Design of Arithmetic Logic Unit (ALU)
- Design of Instruction Set
- High Level Block Design of CPU
- Functional Table of ALU with control unit

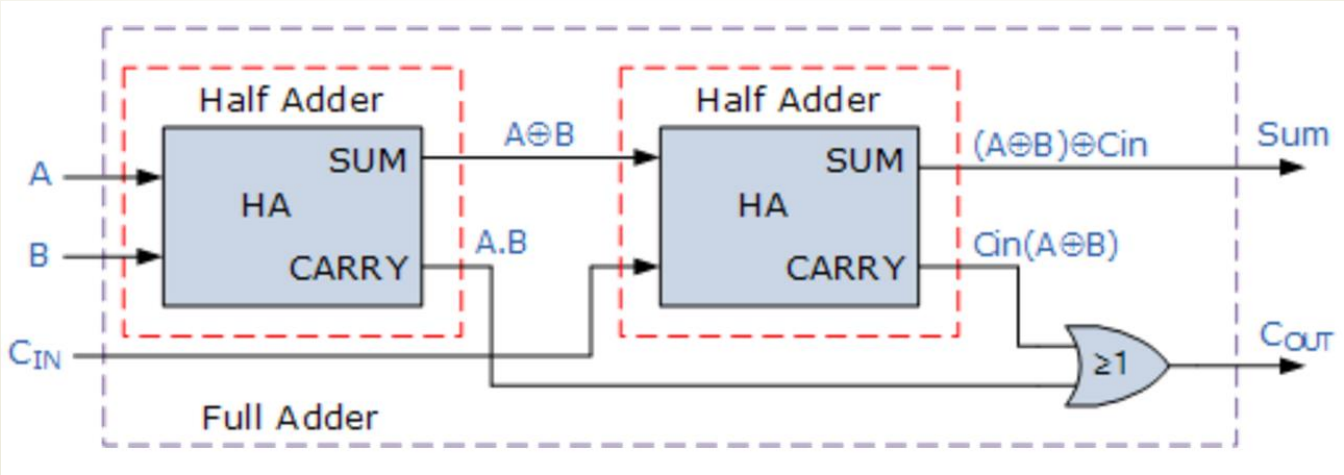
Design of Arithmetic Unit



Half Adder (Logic Diagram)

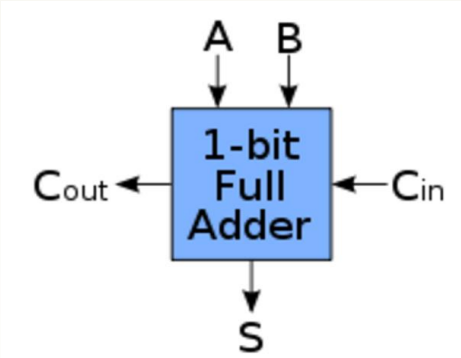


Half Adder (Block Diagram)

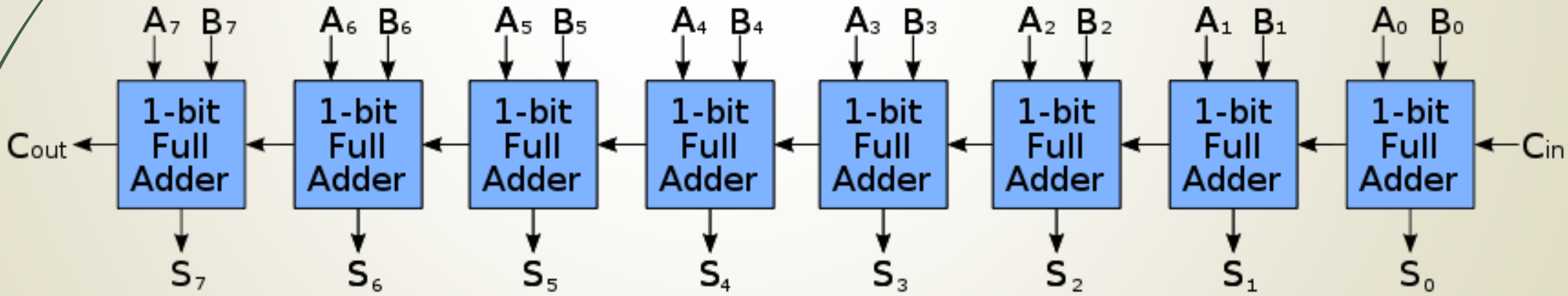


Full Adder using two Half Adder

Design of Arithmetic Unit (Adder)

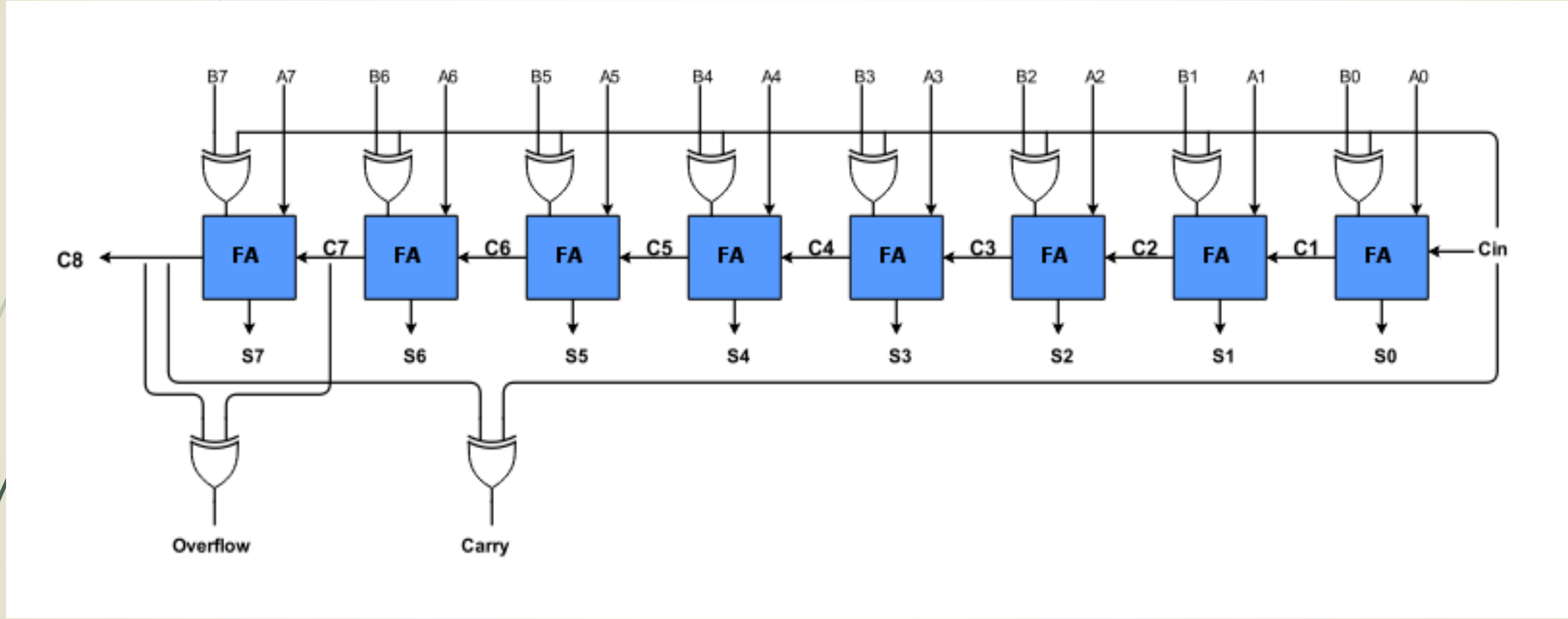


1 Bit Full Adder (Block Diagram)



8 Bit Adder (Block Diagram)

Design of Arithmetic Unit (Add-Sub)

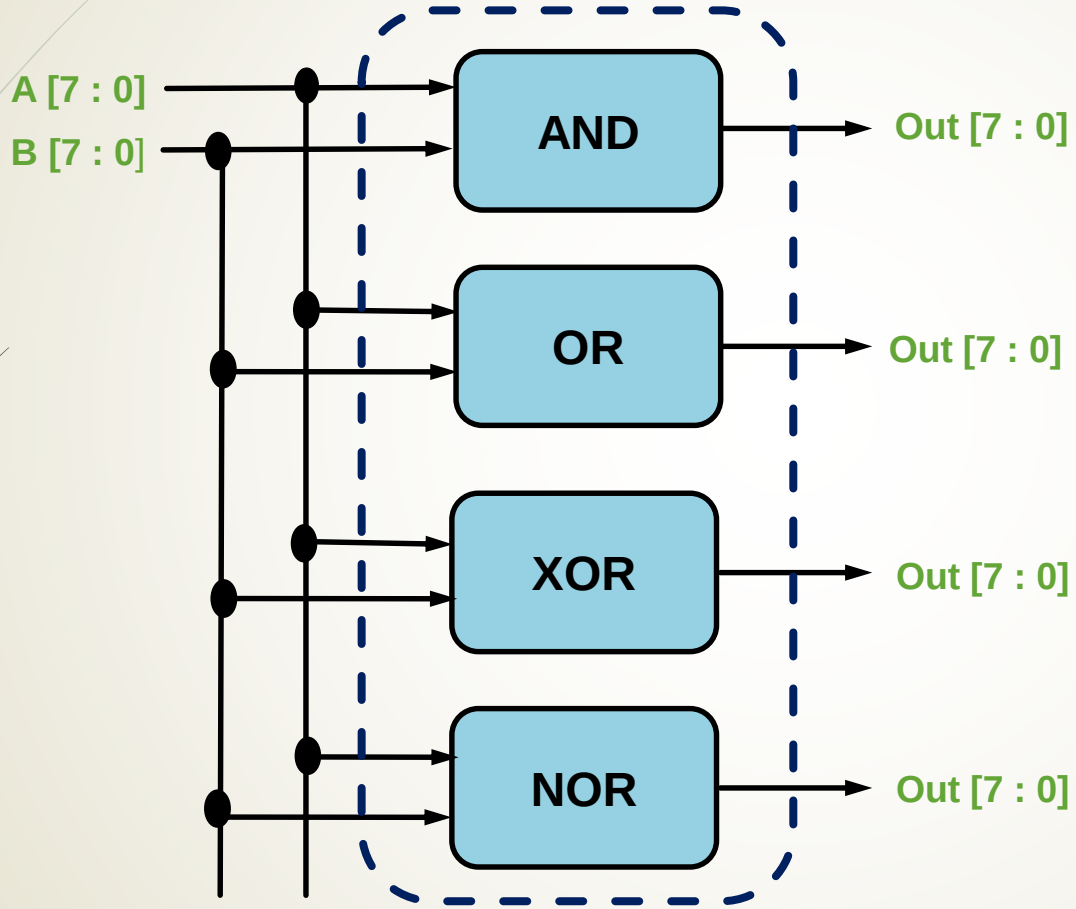


8 Bit Add / Sub (Block Diagram)

ADD / SUB Module

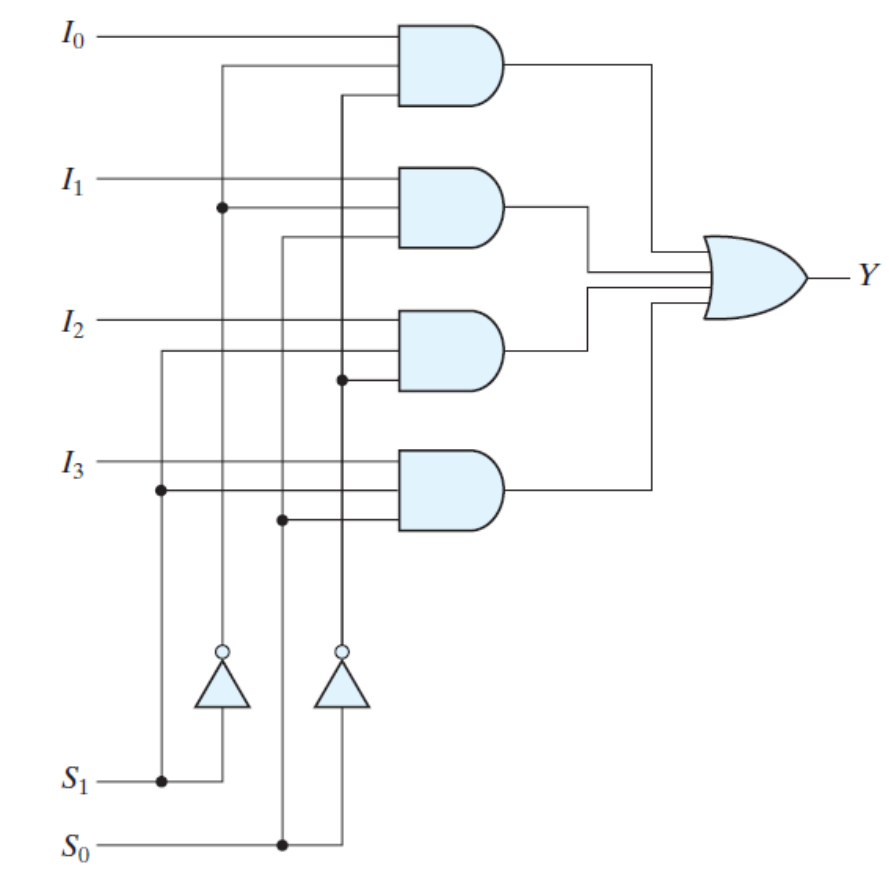
```
module FULLADD_SUB_8Bit(sum, c_out, overflow, a, b, c_in);

    output [7:0] sum;
    output c_out;
    output overflow;

    input [7:0] a, b;
    input c_in;

    wire c1, c2, c3, c4, c5, c6, c7, c8;
    wire [7:0] b1;

    xor(b1[0],b[0],c_in);
    xor(b1[1],b[1],c_in);
    xor(b1[2],b[2],c_in);
    xor(b1[3],b[3],c_in);

    xor(b1[4],b[4],c_in);
    xor(b1[5],b[5],c_in);
    xor(b1[6],b[6],c_in);
    xor(b1[7],b[7],c_in);

    FULLADD2_1Bit FA0(sum[0],c1,a[0],b1[0],c_in);
    FULLADD2_1Bit FA1(sum[1],c2,a[1],b1[1],c1);
    FULLADD2_1Bit FA2(sum[2],c3,a[2],b1[2],c2);
    FULLADD2_1Bit FA3(sum[3],c4,a[3],b1[3],c3);

    FULLADD2_1Bit FA4(sum[4],c5,a[4],b1[4],c4);
    FULLADD2_1Bit FA5(sum[5],c6,a[5],b1[5],c5);
    FULLADD2_1Bit FA6(sum[6],c7,a[6],b1[6],c6);
    FULLADD2_1Bit FA7(sum[7],c8,a[7],b1[7],c7);

    xor(c_out, c8, c_in);  //Carry = c8 for Addition and Carry = not(c8) for subtrtion
    xor(overflow, c8, c7);

endmodule
```

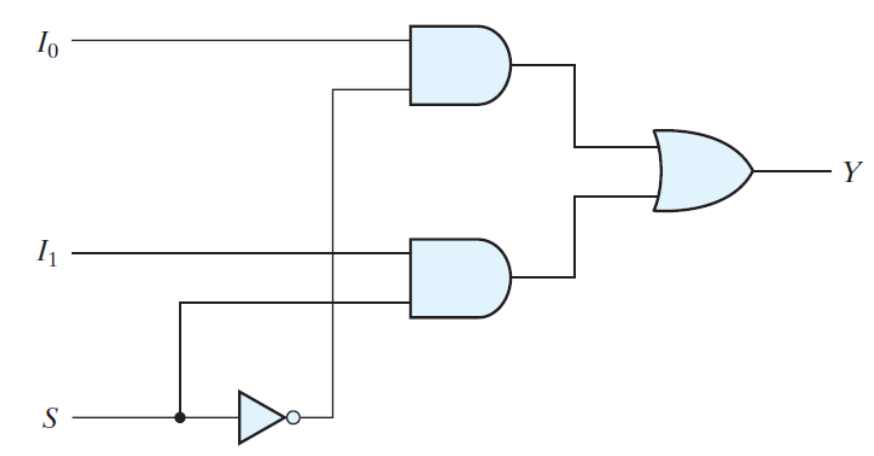
Design of Logic Unit



8 Bit Logic Unit (Block Diagram)

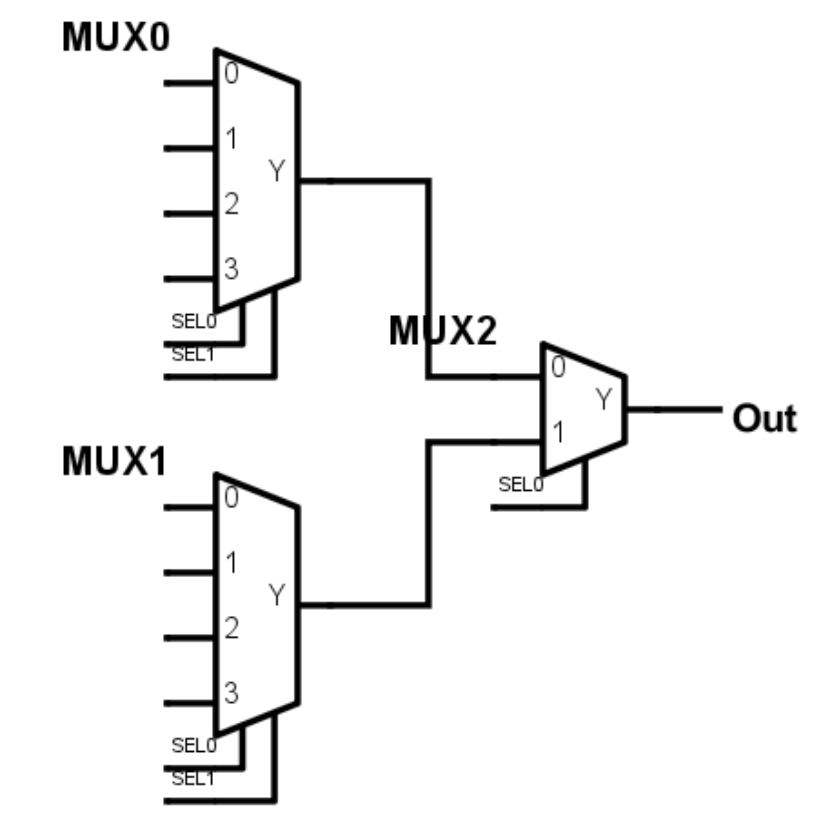
Design of Shift Unit



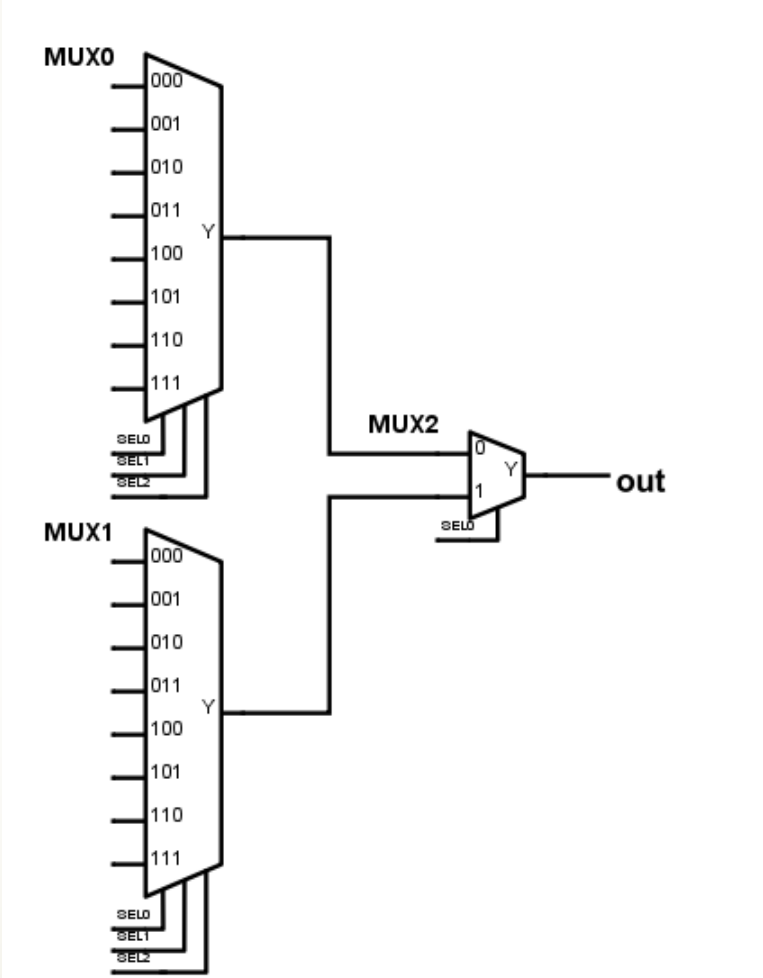


4 to 1 (Left) & 2 to 1 (Right) (Logic Diagram)

Design of Shift Unit (Contd.)

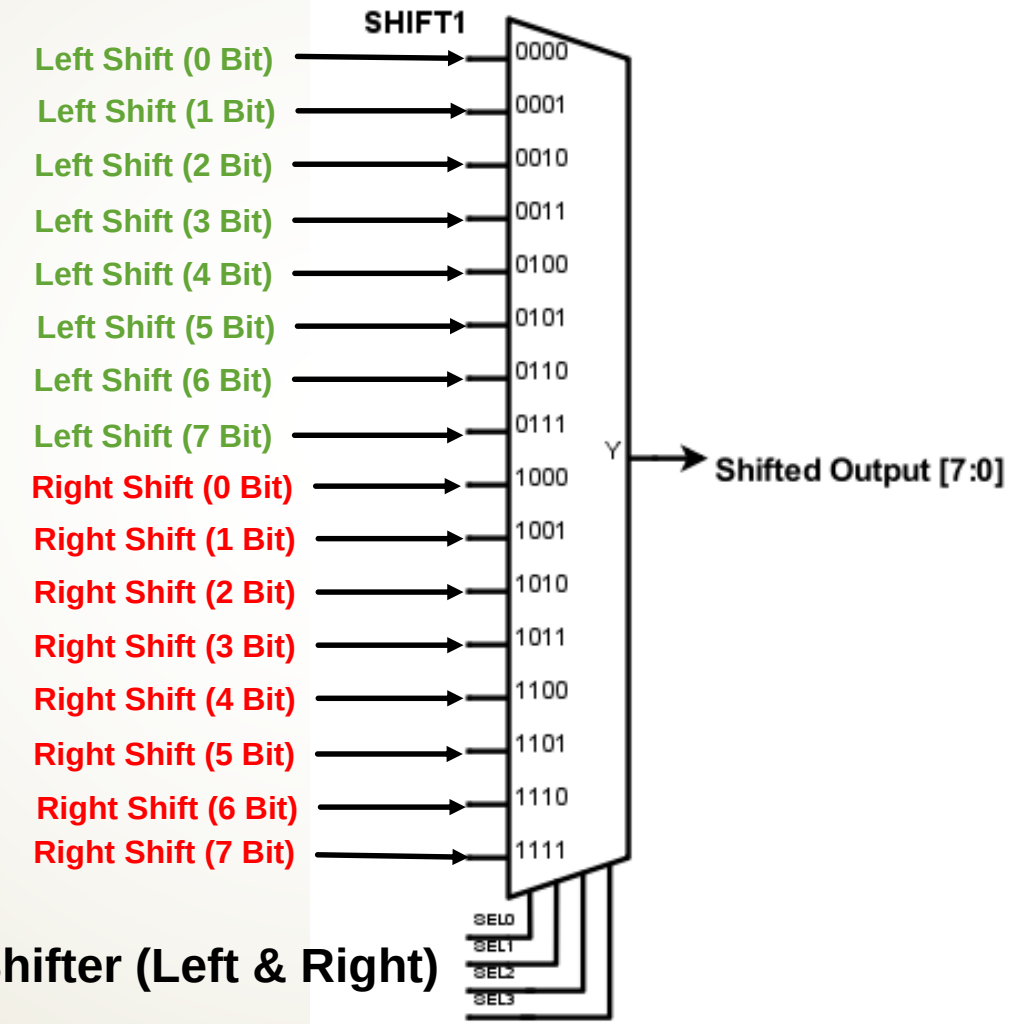


Block Diagram of 8 to 1 MUX



Block Diagram of 16 to 1 MUX

Design of Shift Unit (Contd.)

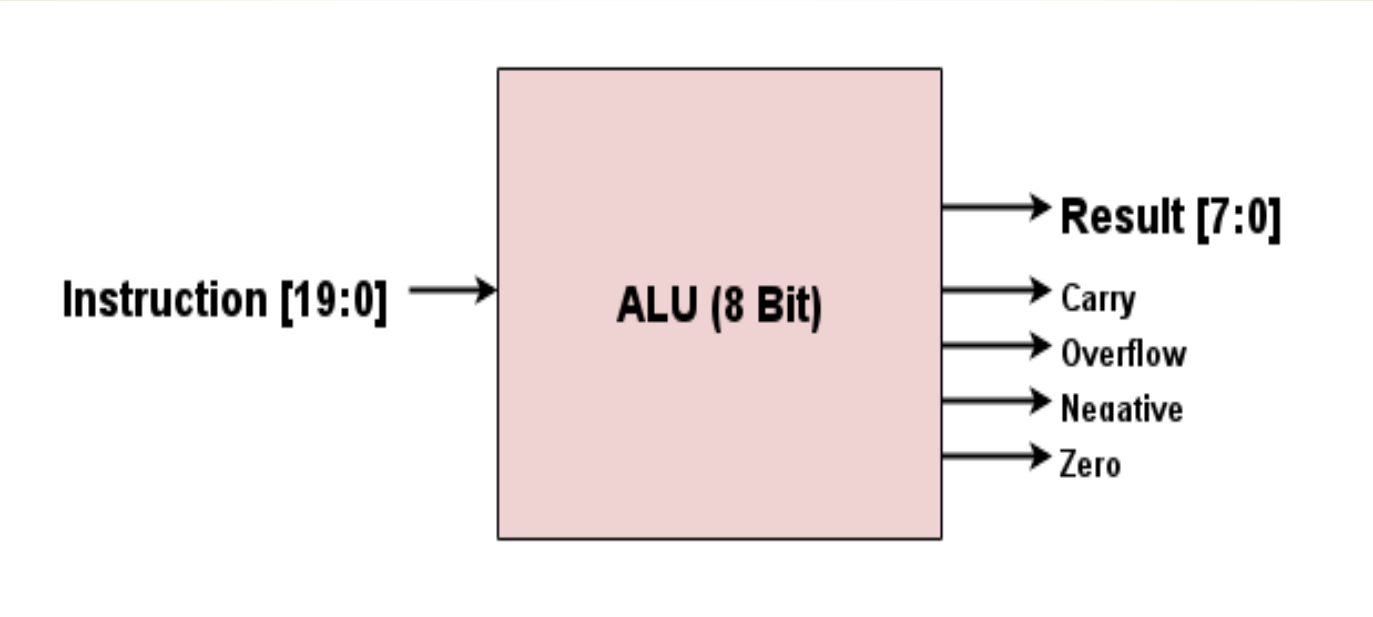


Block Diagram of 8 Bit Shifter (Left & Right)

Shifter Module

```
module SHIFTER_8Bit(out, A, S);

input [7:0] A;
input [3:0] S;

output [7:0] out;

wire [7:0] D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15;

assign D0=A;
assign D1={A[6:0],1'b0};
assign D2={A[5:0],2'b00};
assign D3={A[4:0],3'b000};

assign D4={A[3:0],4'b0000};
assign D5={A[2:0],5'b00000};
assign D6={A[1:0],6'b000000};
assign D7={A[0],7'b0000000};

assign D8=A;
assign D9={1'b0,A[7:1]};
assign D10={2'b00,A[7:2]};
assign D11={3'b000,A[7:3]};

assign D12={4'b0000,A[7:4]};
assign D13={5'b00000,A[7:5]};
assign D14={6'b000000,A[7:6]};
assign D15={7'b0000000,A[7]};

MUX_16to1_8Bit myMUX(out, D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, S);

endmodule
```

Design of Arithmetic Logic Unit (ALU)



Block Diagram of Arithmetic Logic Unit (ALU)

Design of Instruction Set

Op Code = 4 Bit

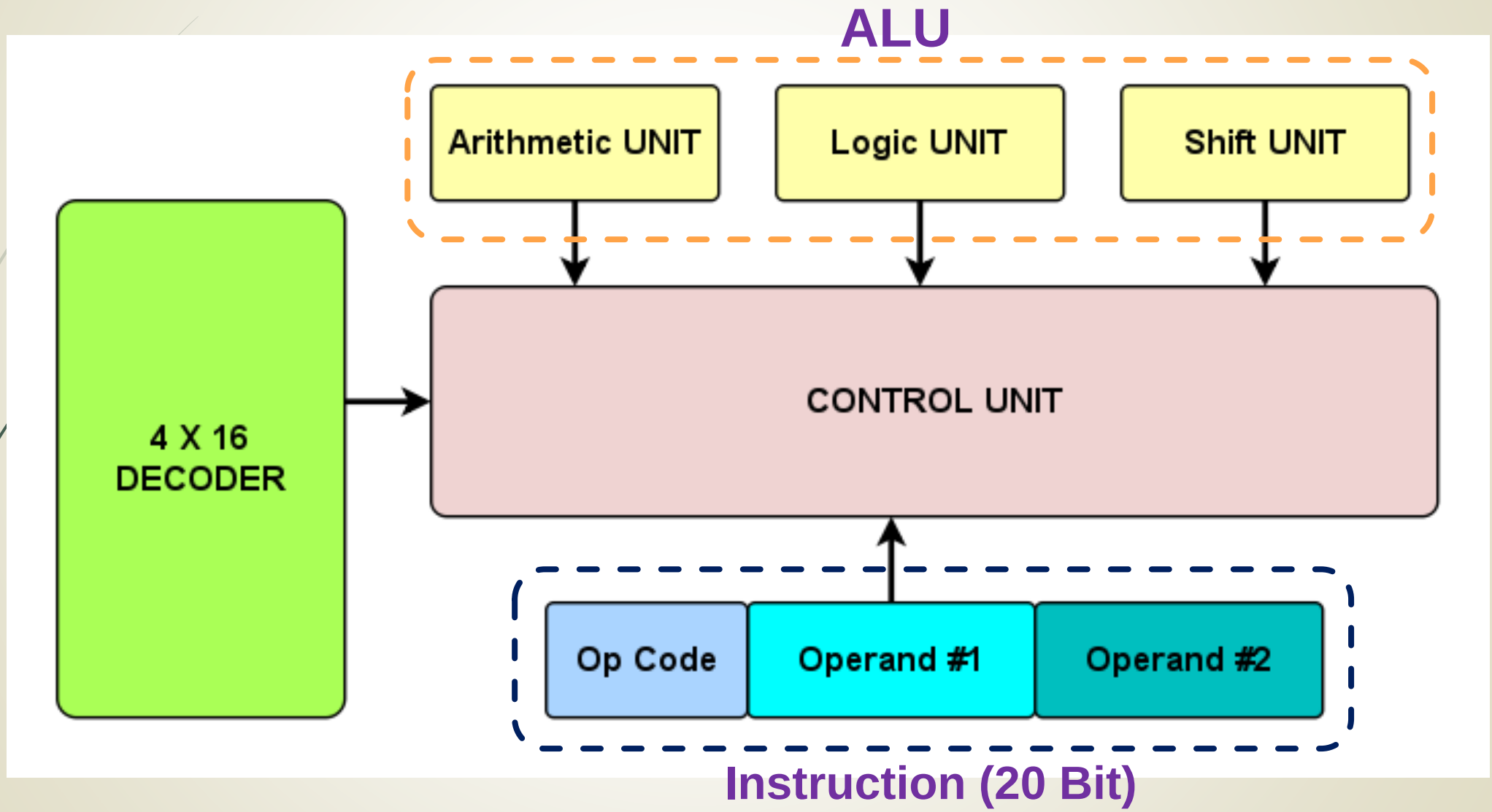
Operand #1 = 8 Bit

Operand #2 = 8 Bit

Instruction Set = 20 Bit

Diagram of 20 Bit Instruction Set

High Level Description of CPU



Adding Instruction(s)

- I have added a XNOR logic operation.
- **Changes:**
  - Add the XNOR operation in the Logic Unit.
  - Add this output of the XNOR operation to one line of the DECODER output.
  - Test the XNOR Operation with specified Op Code.

Functional Table of ALU

OP Code	A	B	Operation	Result	Carry	Overflow	NEG	ZERO
0000	00101000 (40 d)	00010100 (20 d)	**ADD**	00111100 (60 d)	0	0	0	0
0000	11110000 (240 d)	00010100 (20 d)	**ADD**	00000100 (?)	1	0	0	0
0000	10111010 (-70d)	10110000 (-80d)	**ADD**	01101010	1	1	0	0
0001	00011110 (30 d)	00010100 (20 d)	**SUB**	00001010 (10 d)	0	0	0	0
0001	00111100 (60 d)	01010000 (80 d)	**SUB**	11101100 (-20 d)	1	0	1	0
0001	00011110 (30 d)	00011110 (30 d)	**SUB**	00000000	0	0	0	1
0010	00010001	00110011	**AND**	00010001	0	0	0	0
0011	00010001	00110011	**OR**	00110011	0	0	0	0
0100	00010001	00110011	**XOR**	00100010	0	0	0	0
0101	00010001	00110011	**NOR**	11001100	0	0	0	0
0110	00010001	00110011	**XNOR**	11011101	0	0	0	0

Functional Table of ALU (contd..)

OP Code	A	B	Operation	Result	Carry	Overflow	NEG	ZERO
0111	11111111	00000000	**Left (0 Bit)**	11111111	0	0	0	0
0111	11111111	00010000	**Left (1 Bit)**	11111110	0	0	0	0
0111	11111111	10110000	**Right (3 Bit)**	00011111	0	0	0	0
0111	11111111	11110000	**Right (7 Bit)**	00000001	0	0	0	0
Otherwise	00010001	00110011	**NOP**	xxxxxxxx	0	0	0	x

Thank you, any question?