PRETTY GATE MACHINE

SYSTEM DESIGN - ALU

COMPLETED: 11/16/18

Table of Contents:

Parts List: primitives and wiring

- Logic Unit
  - wire [7:0] andwire;
  - wire [7:0] orwire;
  - wire [7:0] notawire;
  - wire [7:0] notbwire;
  - wire [7:0] xorwire;
  - reg nullin;
- STtoMUX
  - wire [3:0] notS;
  - wire [15:0] andOUT;
  - not(notS[0], s[0]);
  - not(notS[1], s[1]);
  - not(notS[2], s[2]);
  - not(notS[3], s[3]);
- And_unit
  - and a(result[0], A[0], B[0]);
  - and b(result[1], A[1], B[1]);
  - and c(result[2], A[2], B[2]);
  - and c(result[3], A[3], B[3]);
  - and d(result[4], A[4], B[4]);
  - and e(result[5], A[5], B[5]);
  - and f(result[6], A[6], B[6]);
  - and g(result[7], A[7], B[7]);
- Or_unit
  - or a(result[0], A[0], B[0]);
  - or b(result[1], A[1], B[1]);
  - or c(result[2], A[2], B[2]);
  - or d(result[3], A[3], B[3]);
  - or e(result[4], A[4], B[4]);
  - or f(result[5], A[5], B[5]);
  - or g(result[6], A[6], B[6]);
  - or h(result[7], A[7], B[7]);
- notA_unit
  - not(result[0], A[0]);
  - not(result[1], A[1]);
  - not(result[2], A[2]);
  - not(result[3], A[3]);
  - not(result[4], A[4]);
  - not(result[5], A[5]);
  - not(result[6], A[6]);
  - not(result[7], A[7]);
- notB_unit
  - not(result[0], B[0]);
  - not(result[1], B[1]);
  - not(result[2], B[2]);

    - not(result[3], B[3]);
    - not(result[4], B[4]);
    - not(result[5], B[5]);
    - not(result[6], B[6]);
    - not(result[7], B[7]);
- Xor_unit
    - xor(result[0], A[0], B[0]);
    - xor(result[1], A[1], B[1]);
    - xor(result[2], A[2], B[2]);
    - xor(result[3], A[3], B[3]);
    - xor(result[4], A[4], B[4]);
    - xor(result[5], A[5], B[5]);
    - xor(result[6], A[6], B[6]);
    - xor(result[7], A[7], B[7]);
- ABrightshifter
    - wire notsh;
    - not(notsh, sh);
    - wire [6:0] a1out;
    - wire [6:0] a2out;
    - wire [6:0] orout;
    - and(a1out[6], q[7],sh); //a1
    - and(a2out[6], A[6], notsh);
    - or(orout[6], a1out[6], a2out[6]);
    - and(a1out[5], q[6],sh);
    - and(a2out[5], A[5], notsh);
    - or(orout[5], a1out[5], a2out[5]);
    - and(a1out[4], q[5],sh);
    - and(a2out[4], A[4], notsh);
    - or(orout[4], a1out[4], a2out[4]);
    - and(a1out[3], q[4],sh);
    - and(a2out[3], A[3], notsh);
    - or(orout[3], a1out[3], a2out[3]);
    - and(a1out[2], q[3],sh);
    - and(a2out[2], A[2], notsh);
    - or(orout[2], a1out[2], a2out[2]);
    - and(a1out[1], q[2], sh);
    - and(a2out[1],A[1], notsh);
    - or(orout[1], a1out[1], a2out[1]);
    - and(a1out[0], q[1], sh);
    - and(a2out[0],A[0], notsh);
    - or(orout[0], a1out[0], a2out[0]);
- ABleftshifter
    - wire notsh;
    - not(notsh, sh);
    - wire [6:0] a1out;
    - wire [6:0] a2out;
    - wire [6:0] orout;

```
    and(a2out[6], A[7], notsh);
    and(a1out[6], q[6],sh ); //a1
    or(orout[6], a1out[6], a2out[6]);
    and(a2out[5], A[6], notsh);
    and(a1out[5], q[5],sh ); //a1
    or(orout[5], a1out[5], a2out[5]);
    and(a2out[4], A[5], notsh);
    and(a1out[4], q[3],sh ); //a1
    or(orout[4], a1out[4], a2out[4]);
    and(a2out[3], A[4], notsh);
    and(a1out[3], q[3],sh ); //a1
    or(orout[3], a1out[3], a2out[3]);
    and(a2out[2], A[3], notsh);
    and(a1out[2], q[2],sh ); //a1
    or(orout[2], a1out[2], a2out[2]);
    and(a2out[1], A[2], notsh);
    and(a1out[1], q[1],sh ); //a1
    or(orout[1], a1out[1], a2out[1]);
    and(a2out[0], A[1], notsh);
    and(a1out[0], q[0],sh ); //a1
    or(orout[0], a1out[0], a2out[0]);
```

- DFF

```
    wire   Cn;
    wire   Cnn;
    wire   DQ;
    wire   DQn;
    not(Cn, C);
    not(Cnn, Cn);
```

- D_latch

```
    wire   Dnotout;
    wire   Dandout;
    wire   Dandout2;
    not(Dnotout, D);
    and(Dandout, G, D);
    and(Dandout2, G, Dnotout);
    nor(Qn, Dandout, Q);
    nor(Q, Dandout2, Qn)
```

- Sr_latch_gated

```
    wire   S1;
    wire   R1;
    and(S1, G, S);
    and(R1, G, R);
    nor(Qn, S1, Q);
    nor(Q, R1, Qn);
```

- ALU

```
    wire [7:0] eightbasuOUT;
    wire [7:0] AleftsOUT;
```

    - wire [7:0] ArightsOUT;
    - wire [7:0] AleftsOUTn;
    - wire [7:0] ArightsOUTn;
    -
    - wire [7:0] BleftsOUT;
    - wire [7:0] BrightsOUT;
    - wire [7:0] BleftsOUTn;
    - wire [7:0] BrightsOUTn;
    - wire [7:0] aluLUOUT;
    - reg nullin;
- Add_half
    - xor G1(sum, a, b);
    - and G2(c_out, a, b);
- Add_full
    - wire w1, w2, w3;
    - or (c_out, w1, w3);
- eightB_asu
    - wire [7:0] w;
    - wire [6:0] c;
    - xor x0(w[0], b[0], M);
    - xor x1(w[1], b[1], M);
    - xor x2(w[2], b[2], M);
    - xor x3(w[3], b[3], M);
    - xor x4(w[4], b[4], M);
    - xor x5(w[5], b[5], M);
    - xor x6(w[6], b[6], M);
    - xor x7(w[7], b[7], M);
- Testbench
    - wire [7:0] result;
    - wire carry ; // error code
    - reg [7:0] a;
    - reg [7:0] b;
    - reg M;
    - reg [3:0] selector;
    - reg clk;
    - reg sh;

Input List:

- module logic_unit(output [7:0] result, input [7:0]A, input [7:0] B, input [3:0]selector);
- module STtoMUX
  - input [3:0] s,
  - input oneMUXinput,
  - input twoMUXinput,
  - input threeMUXinput,
  - input fourMUXinput,
  - input fiveMUXinput,
  - input sixMUXinput,
  - input sevenMUXinput,
  - input eightMUXinput,
  - input nineMUXinput,
  - input tenMUXinput,
  - input eleMUXinput,
  - input twelMUXinput,
  - input thirtMUXinput,
  - input fourteenMUXinput,
  - input fifteenMUXinput
  - input sixteenMUXinput
- module and_unit
  - input[7:0] A
  - input [7:0] B
- module or_unit
  - input[7:0] A
  - input [7:0] B
- module notA_unit
  - input[7:0] A
- module notB_unit
  - input [7:0] B
- module xor_unit
  - input[7:0] A
  - input [7:0] B
- module rightshifter
  - input sh
  - input clk
  - input [7:0]A
- module leftshifter
  - input sh
  - input clk
  - input [7:0]A
- module dff
  - input C
  - input D

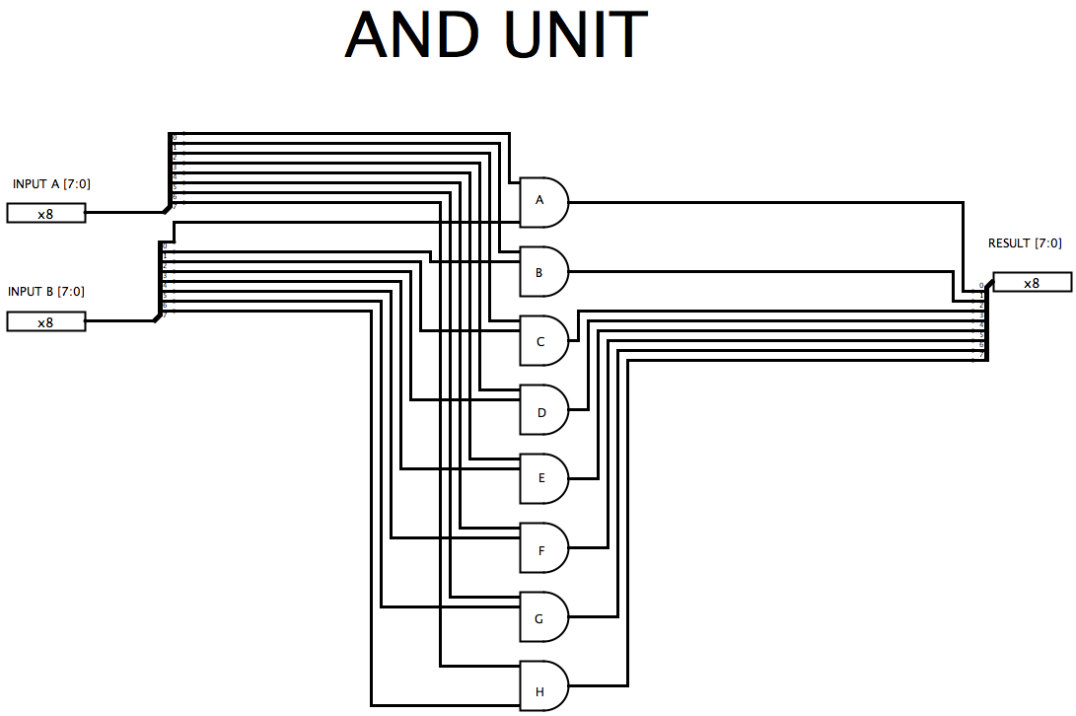
- module d_latch
  - input G
  - input D
- module sr_latch_gated
  - input G
  - input S
  - input R
- module ALU
  - input [7:0] a
  - input[7:0] b
  - input [3:0] opcode
  - input m i
  - nput sh
  - input clk
- module Add_half
  - input a, b
- module Add_full
  - input a, b, c_in
- module eightB_asu
  - input [7:0] a
  - input [7:0] b
  - input M

Output List:

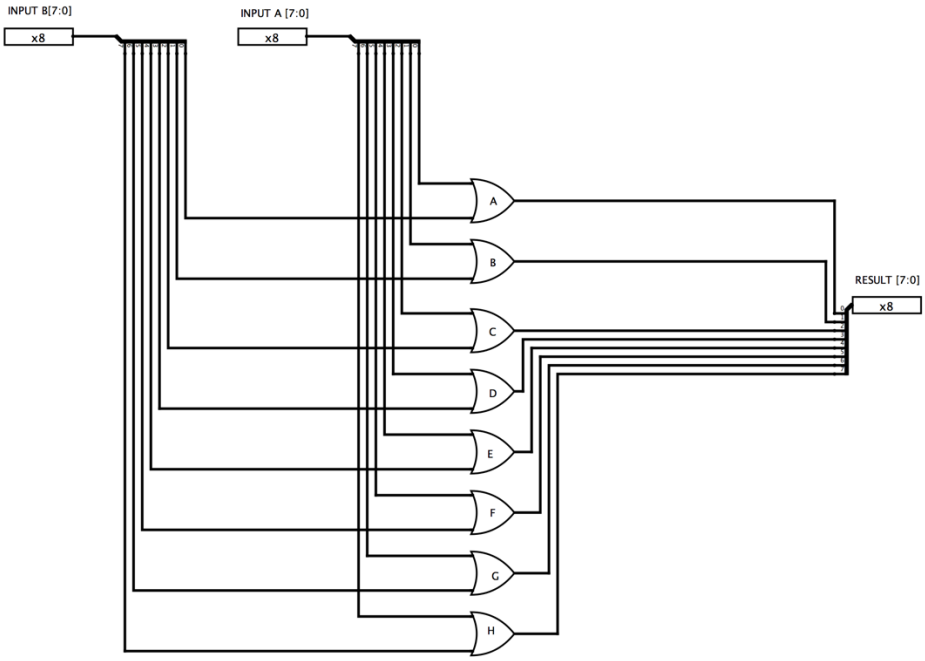
- module logic_unit
  - output [7:0] result
- module STtoMUX
  - output result,
- module and_unit
  - output [7:0] result
- module or_unit
  - output [7:0] result
- module notA_unit
  - output [7:0] result
- module notB_unit
  - output [7:0] result
- module xor_unit
  - output [7:0] result
- module rightshifter
  - output [7:0] q
  - output [7:0] qn
- module leftshifter
  - output [7:0] q
  - output [7:0] qn
- module dff
  - output Q
  - output Qn
- module d_latch
  - output Q
  - output Qn
- module sr_latch_gated
  - output Q
  - output Qn
- module ALU
  - output carry
- module Add_half
  - output c_out
  - output sum
- module Add_full
  - output c_out
  - output sum
- module eightB_asu
  - output C
  - output [7:0] sum

Interface List:

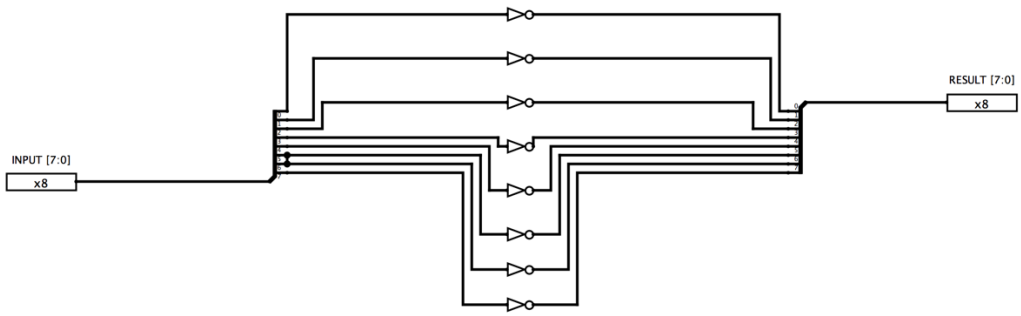
- module and_unit



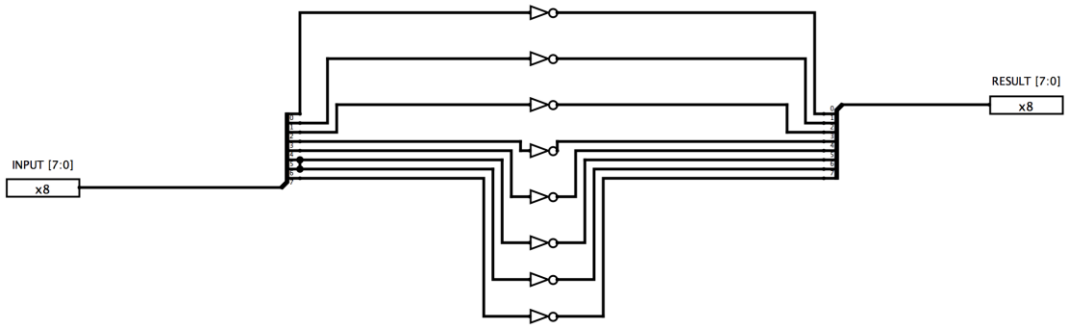
- module or_unit



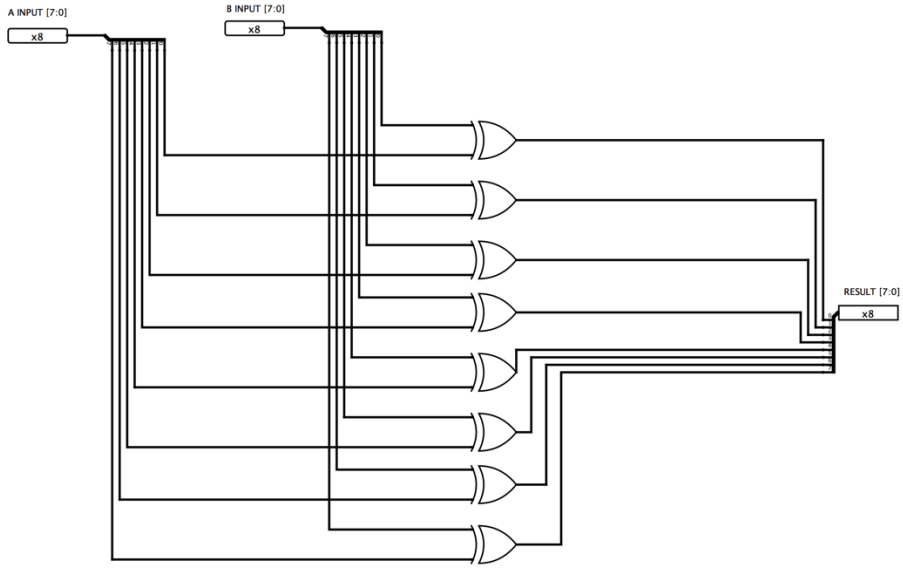
- module notA_unit



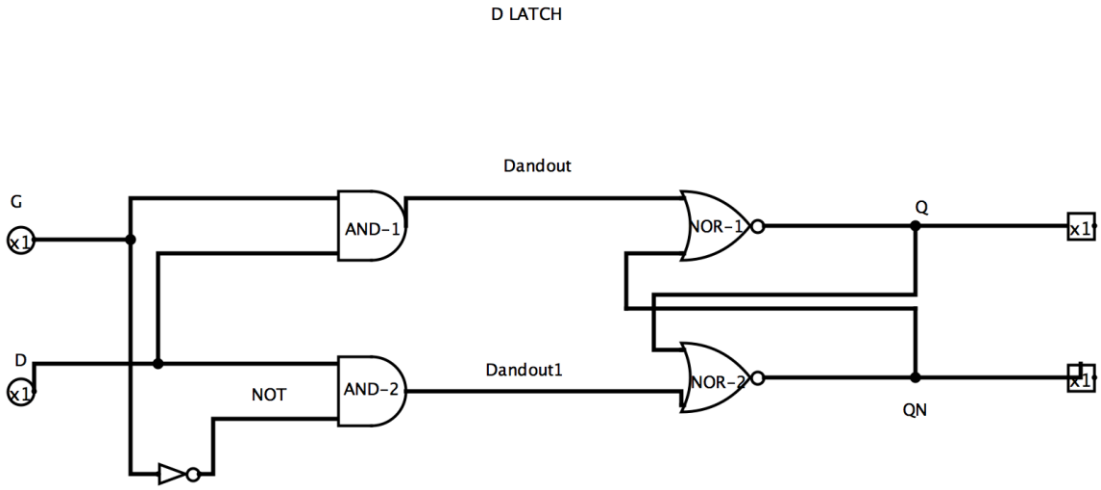
- module notB_unit



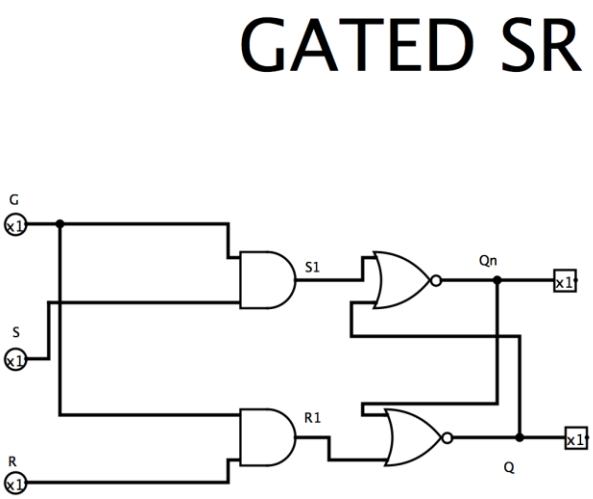
- module xor_unit



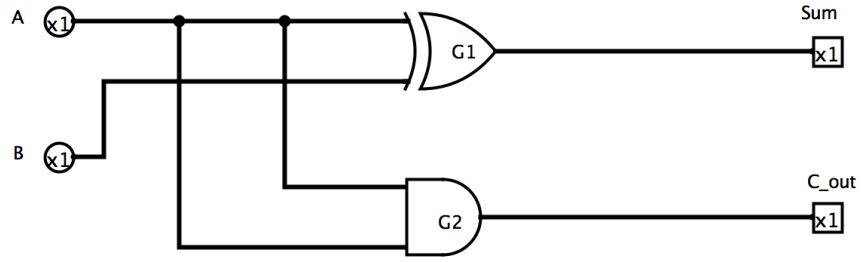
- module d_latch



- module sr_latch_gated

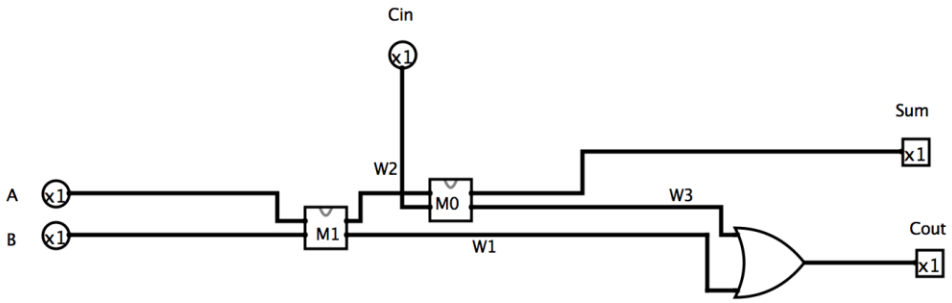


- module Add_half



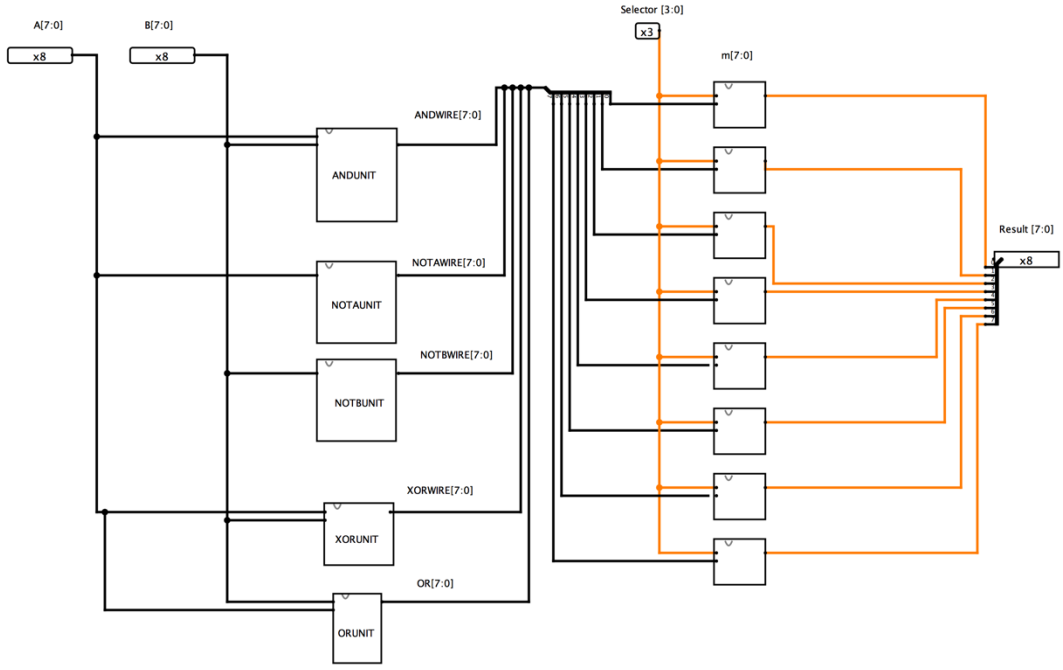
- `module Add_full`

FULL ADDER

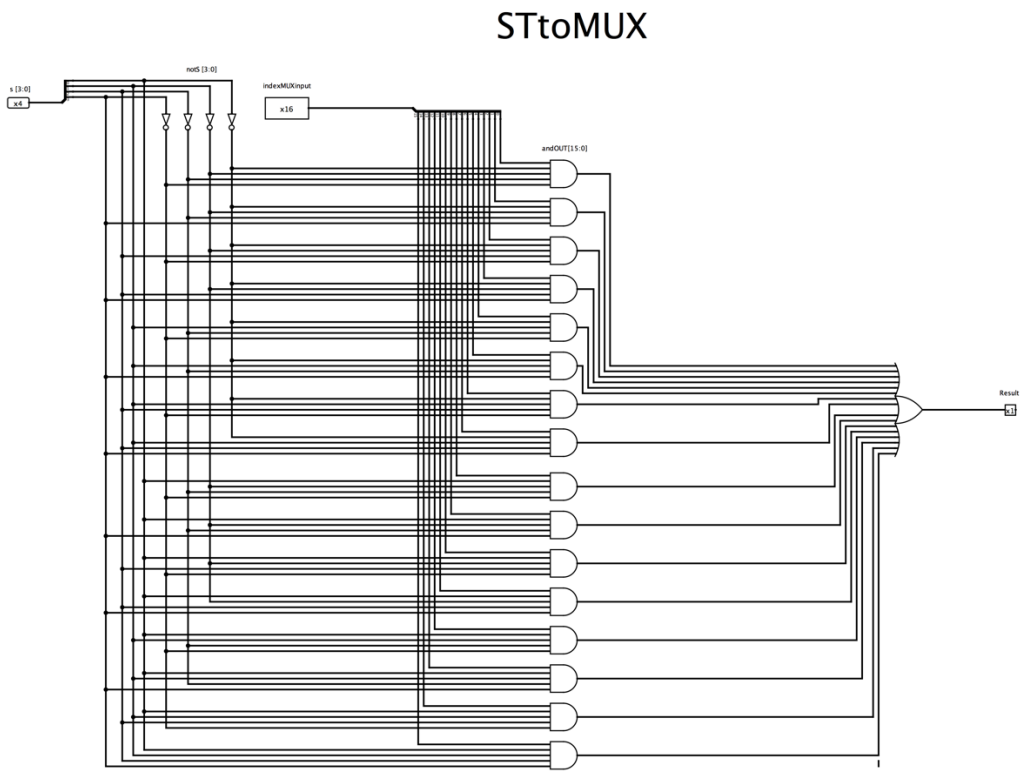


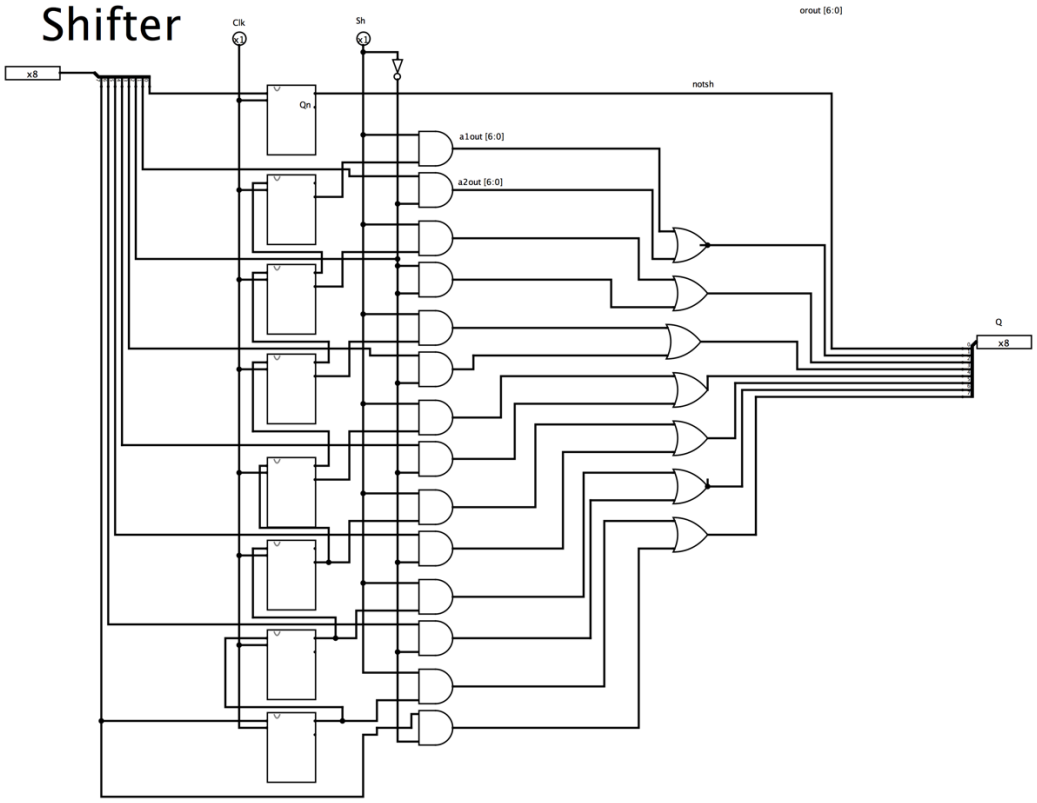
Module List:

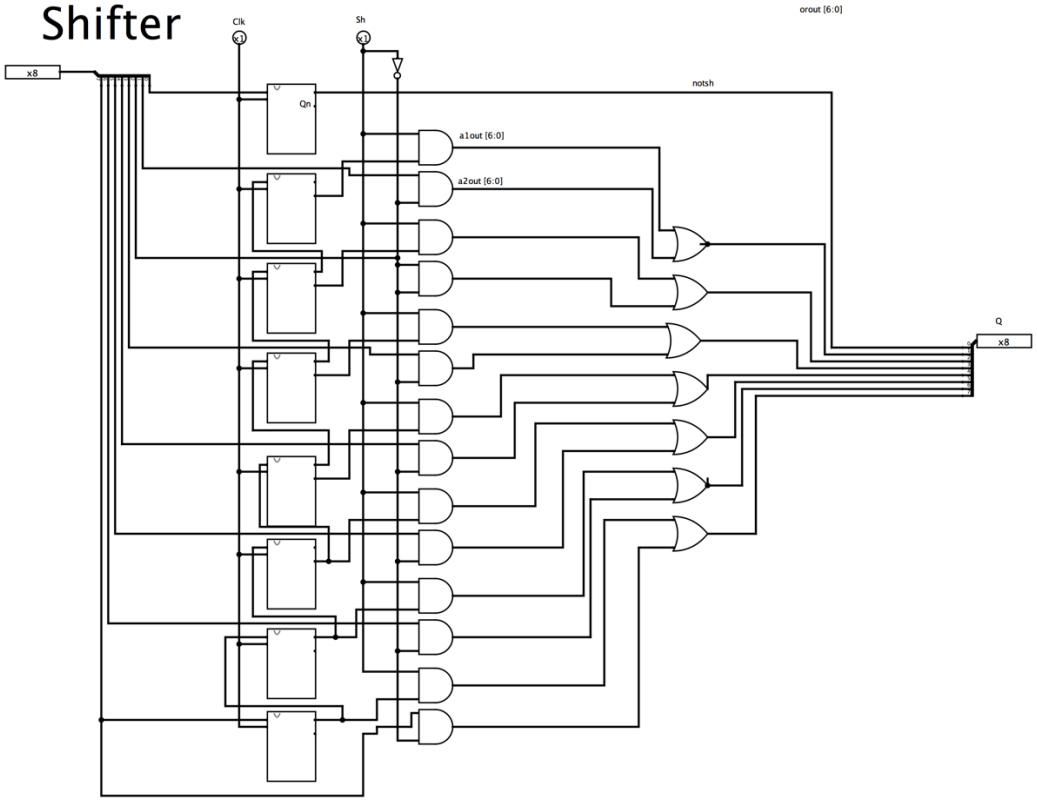
- module logic_unit



- module STtoMUX

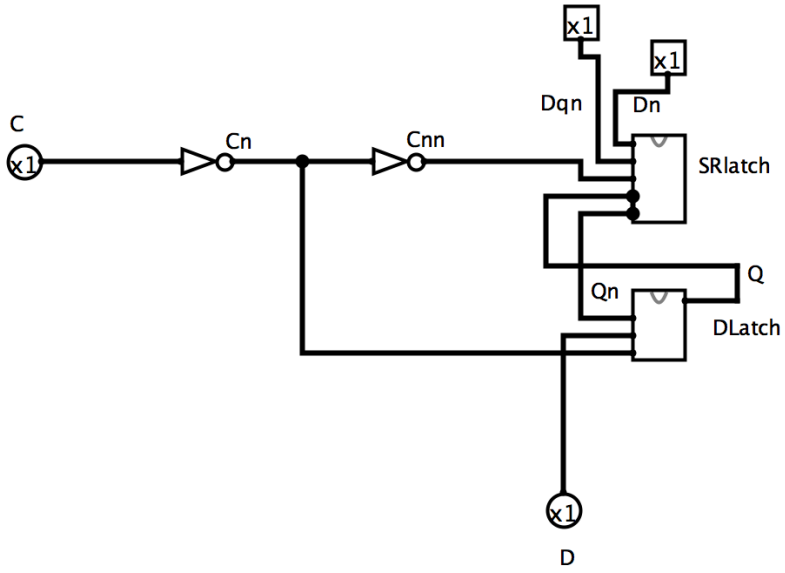


- module rightshifter
  - 

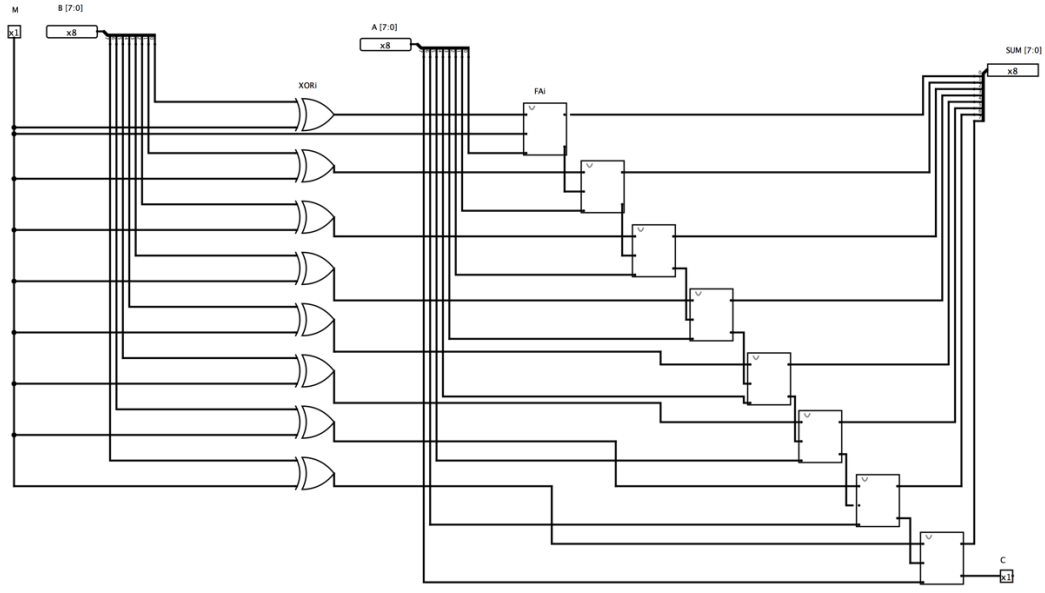
- module leftshifter
  - 

- module dff

DFF

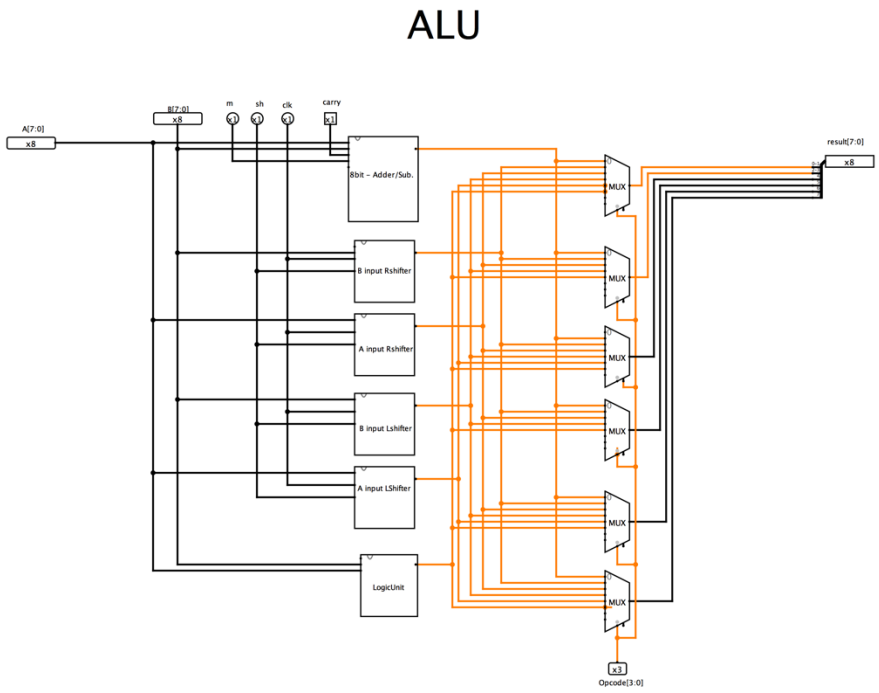


- module eightB_asu



8 BIT ADDER SUBTRACTOR

- module ALU

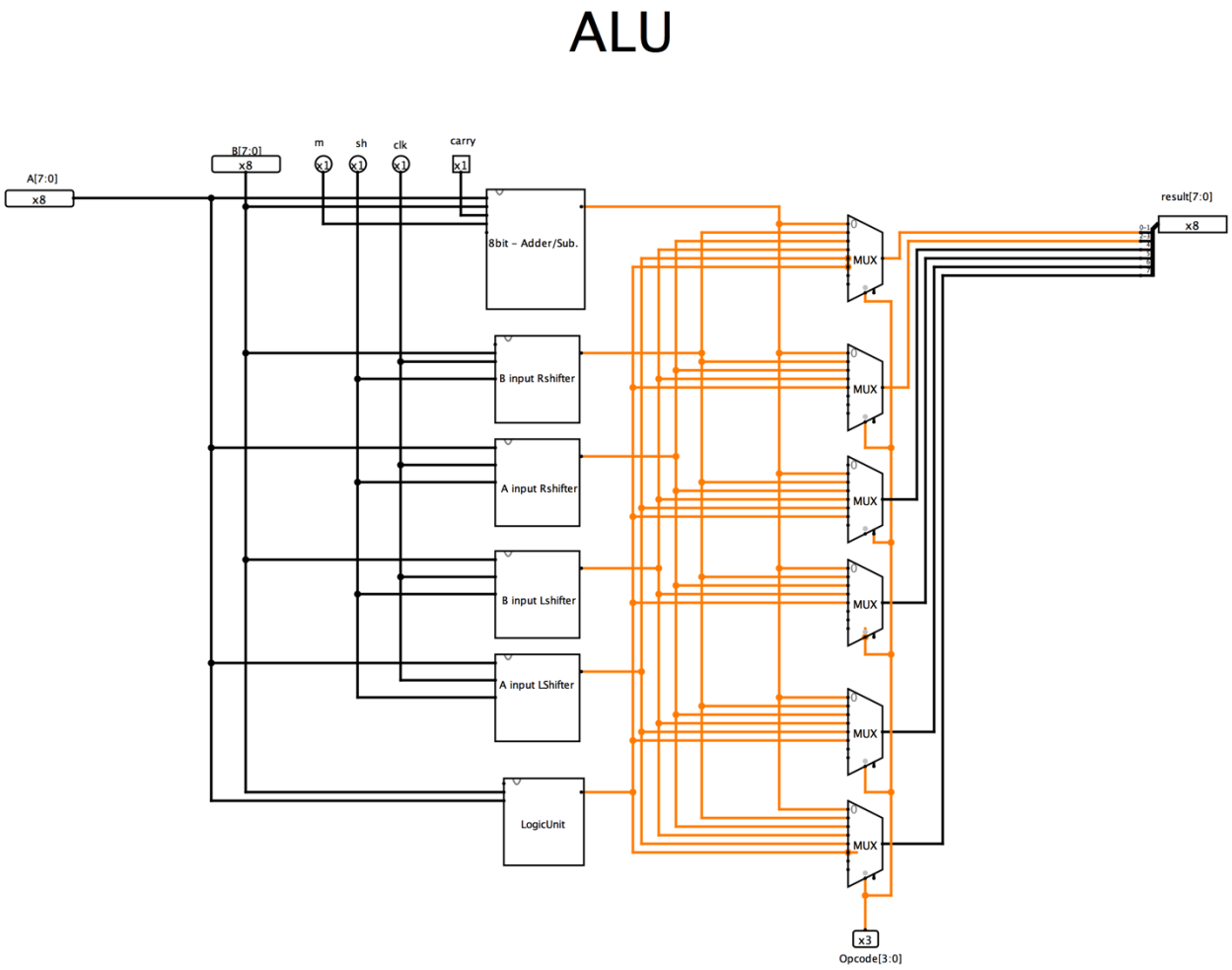


  -
- module testbench

Mode List(states):

BINARY	OPERATION CODES	ERROR CODES	
0000	AND (A ^ B)	NULL	0
0001	OR (A V B)	OVERFLOW	1
0010	~A		2
0011	~B		3
0100	A XOR B	NULL	4
0101	A + OR - B	NULL	5
0110	SHIFT LEFT A	NULL	6
0111	SHIFT RIGHT A	NULL	7
1000	SHIFT LEFT B	NULL	8
1001	SHIFT RIGHT B	NULL	9
BINARY	CLEAR	M	
00	SET	ADD	0
01	RESET	SUBTRACT	1

Circuit Diagram:



State Machines: