Duke

Progress Report II

Team Members:

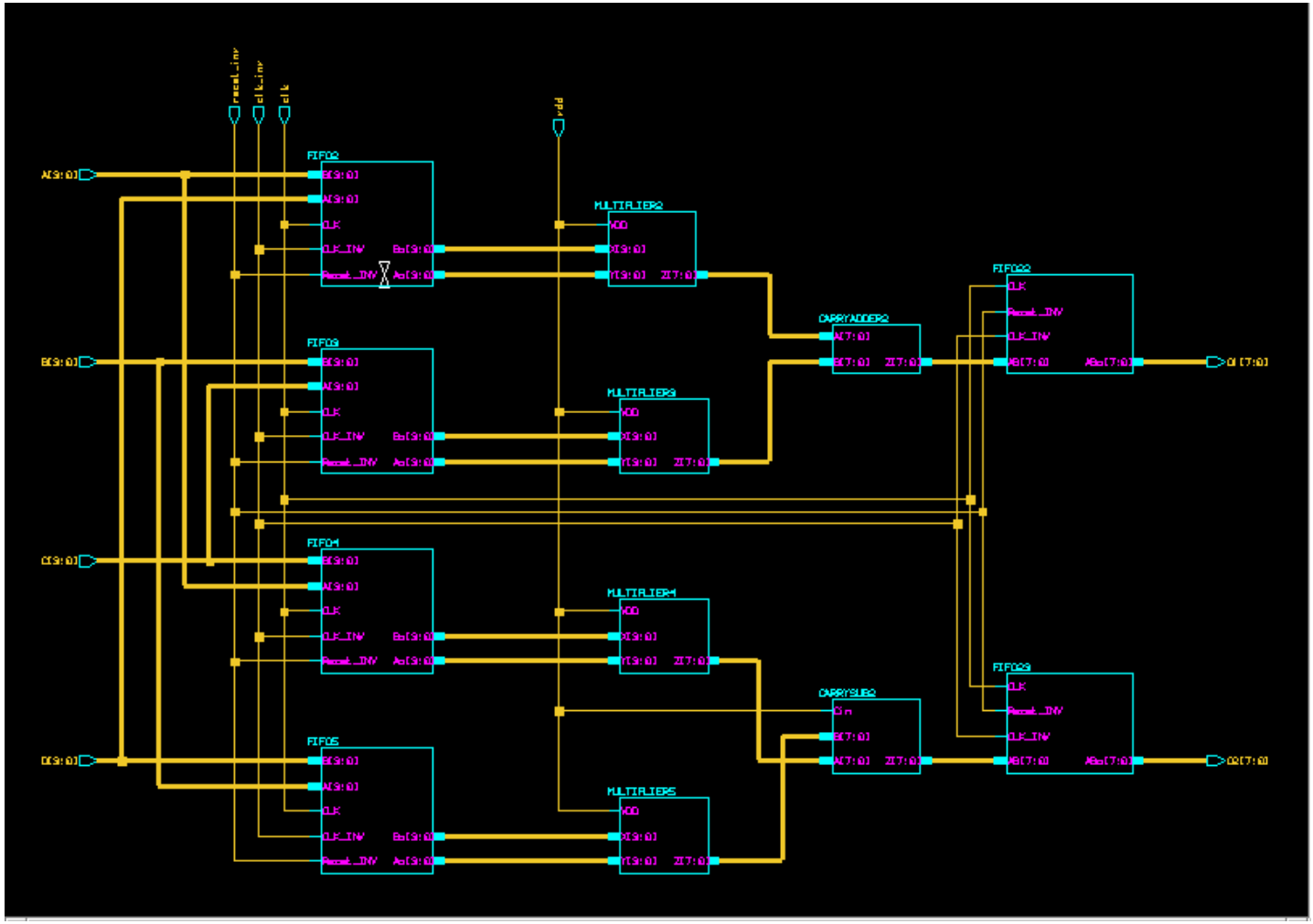
Xuan Bao
Kai Wang
Xiaoyan Yin
Yang Jiang

ECE 261

Project 2008

Index

Overview of Our Present Step

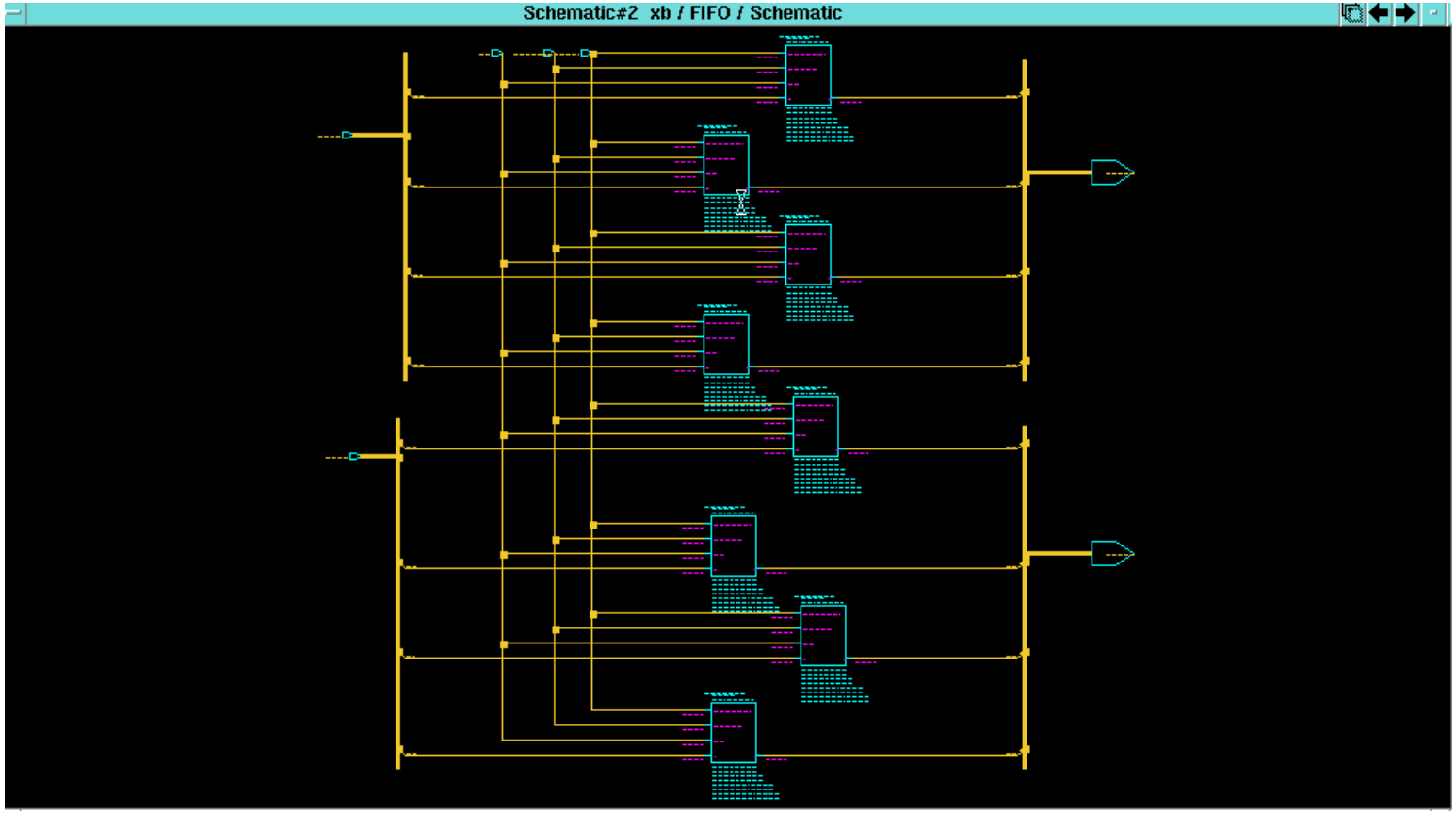


This is the top level schematic of our project.

Every block here has already been fully defined.

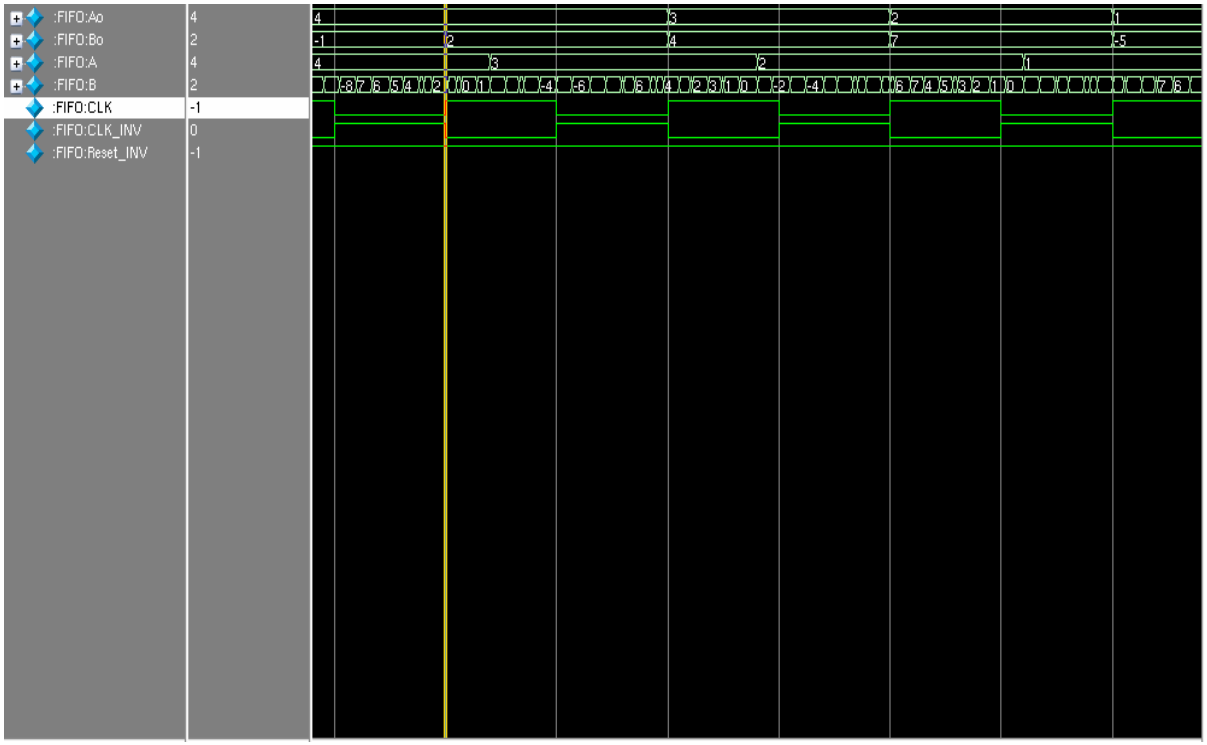
All components and the whole system have correctly passed digital simulation.

Introduction to the FIFO



This is the schematic of one FIFO component. It contains 8 Dflipflops with reset (for further testing) and works as a buffer for both input and output.

Introduction to the FIFO

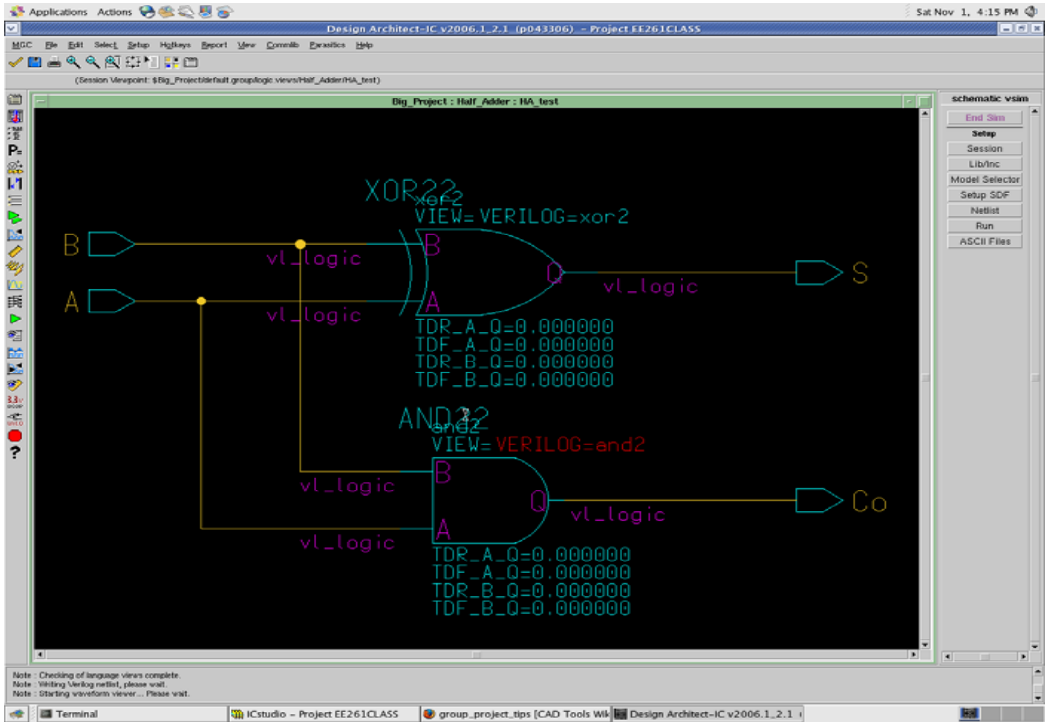



```
`celldefine
`suppress_faults
`enable_portfaults
`timescale 1ns/10ps
module dflipflop (D,CLK,CLK_inv,Reset_inv,Q);
   output Q;
   input D,CLK,CLK_inv,Reset_inv;
   reg Q;

   always @ ( posedge CLK)
 if (~Reset_inv) begin
   Q <= 0;
 end  else begin
   Q <= D;
 end
```

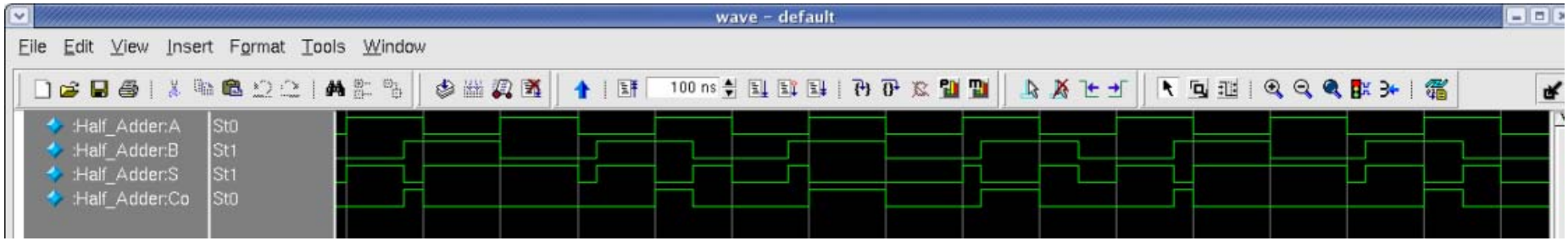
This is the digital simulation result of FIFO component. All possible patterns have been exhaustively tested. To make top level testing easier, we then rewrote the verilog for Dflipflop. The key part of codes is shown above.

Index

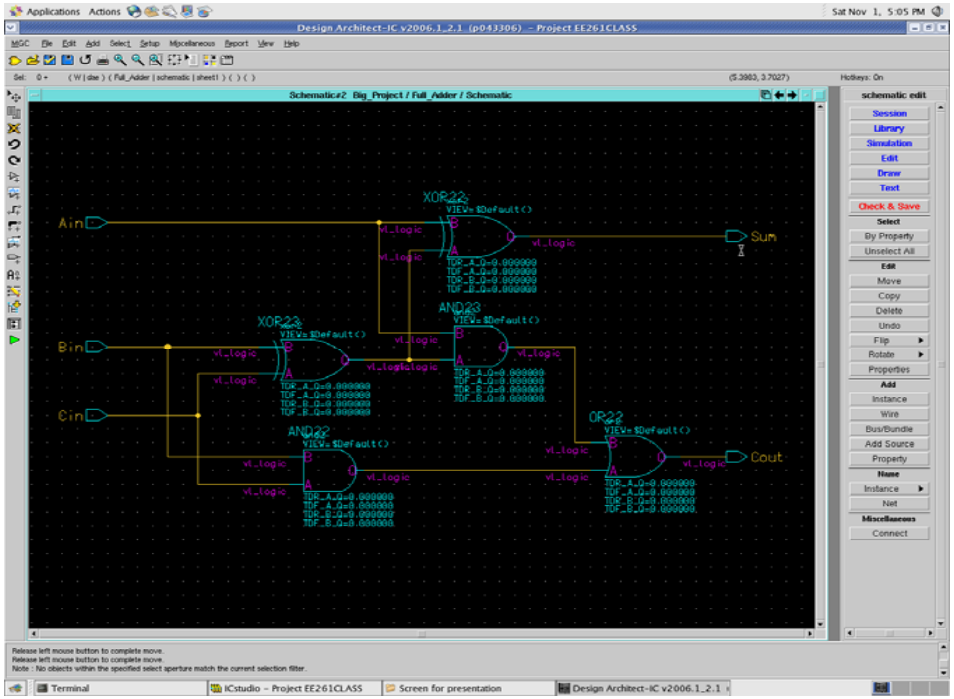
Half Adder



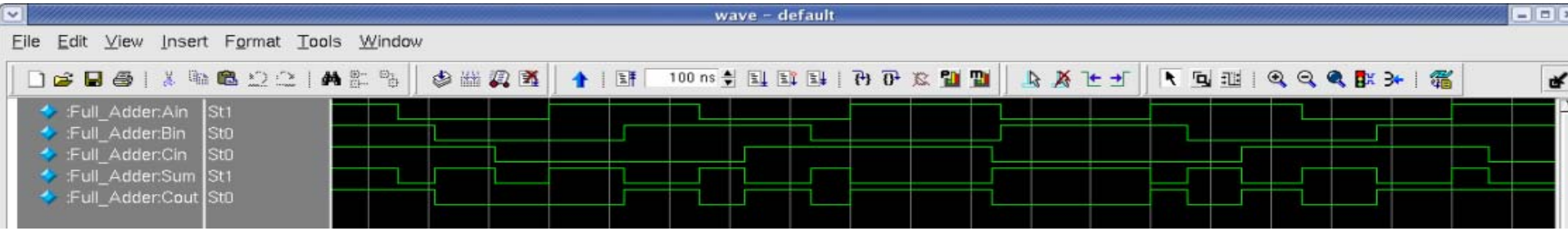
A	B	S	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Full Adder



Input			Output	
Ain	Bin	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

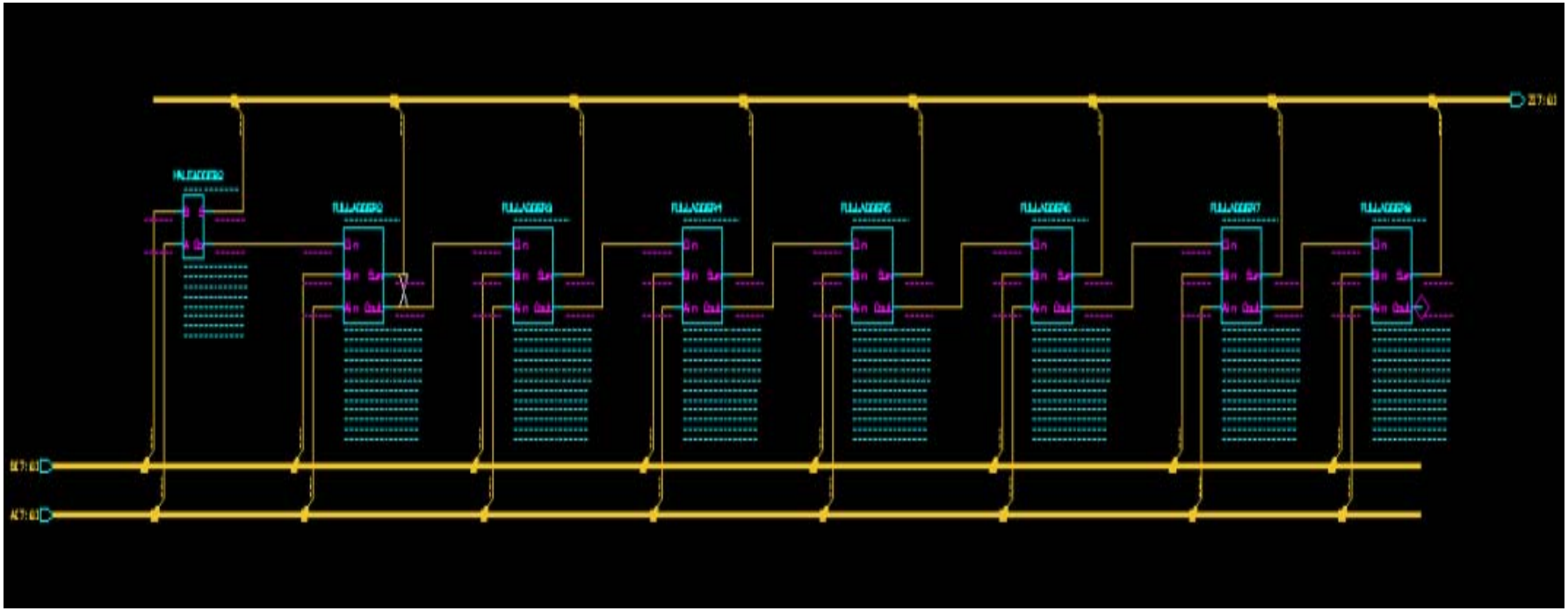


Verilog for Half Adder and Full Adder:

```
`celldefine
`suppress_faults
`enable_portfaults
`timescale 1ns/10ps
module halfadder (Co,S,A,B);
  output S,Co;
  input A,B;

  assign {Co,S}= A+B;

    parameter TDR_A_Co  = 0.000;
    parameter TDF_A_Co  = 0.000;
    parameter TDR_B_Co  = 0.000;
    parameter TDF_B_Co  = 0.000;
    parameter TDR_A_S   = 0.000;
    parameter TDF_A_S   = 0.000;
    parameter TDR_B_S   = 0.000;
    parameter TDF_B_S   = 0.000;

   `ifdef functional
     initial $display("\n *** ATTENTION! Functional Simulation,
   `else
     `ifdef TETRAMAX
   `else
    specify

  `ifdef INCA
     showcancelled S;
     pulsestyle_ondetect S;
  `else
     `ifdef verilog
        $showcancelled;
     $pulsestyle_ondetect;
     `endif
  `endif

       (A => Co)=(TDR_A_Co,TDF_A_Co);
       (B => Co)=(TDR_B_Co,TDF_B_Co);
       (A => S)=(TDR_A_S,TDF_A_S);
       (B => S)=(TDR_B_S,TDF_B_S);


   endspecify
   `endif
   `endif
endmodule
`disable_portfaults
`nosuppress_faults
`endcelldefine
```

```
`celldefine
`suppress_faults
`enable_portfaults
`timescale 1ns/10ps
module fulladder(Cout,Sum,Ain,Bin,Cin);
 input Ain,Bin,Cin;
 output Sum,Cout;

 assign {Cout,Sum}=Ain+Bin+Cin;

    parameter TDR_Ain_Cout  = 0.000;
    parameter TDF_Ain_Cout  = 0.000;
    parameter TDR_Bin_Cout  = 0.000;
    parameter TDF_Bin_Cout  = 0.000;
    parameter TDR_Cin_Cout  = 0.000;
    parameter TDF_Cin_Cout  = 0.000;
    parameter TDR_Ain_Sum   = 0.000;
    parameter TDF_Ain_Sum   = 0.000;
    parameter TDR_Bin_Sum   = 0.000;
    parameter TDF_Bin_Sum   = 0.000;
    parameter TDR_Cin_Sum   = 0.000;
    parameter TDF_Cin_Sum   = 0.000;

`ifdef functional
     initial $display("\n *** ATTENTION! Functional Simulation
   `else
     `ifdef TETRAMAX
   `else
    specify

  `ifdef INCA
     showcancelled Sum;
     pulsestyle_ondetect Sum;
  `else
     `ifdef verilog
        $showcancelled;
     $pulsestyle_ondetect;
     `endif
  `endif

       (Ain => Cout)=(TDR_Ain_Cout,TDF_Ain_Cout);
       (Bin => Cout)=(TDR_Bin_Cout,TDF_Bin_Cout);
       (Cin => Cout)=(TDR_Cin_Cout,TDF_Cin_Cout);
       (Ain => Sum)=(TDR_Ain_Sum,TDF_Ain_Sum);
       (Bin => Sum)=(TDR_Bin_Sum,TDF_Bin_Sum);
       (Cin => Sum)=(TDR_Cin_Sum,TDF_Cin_Sum);

   endspecify
   `endif
   `endif
endmodule
`disable_portfaults
`nosuppress_faults
`endcelldefine
```

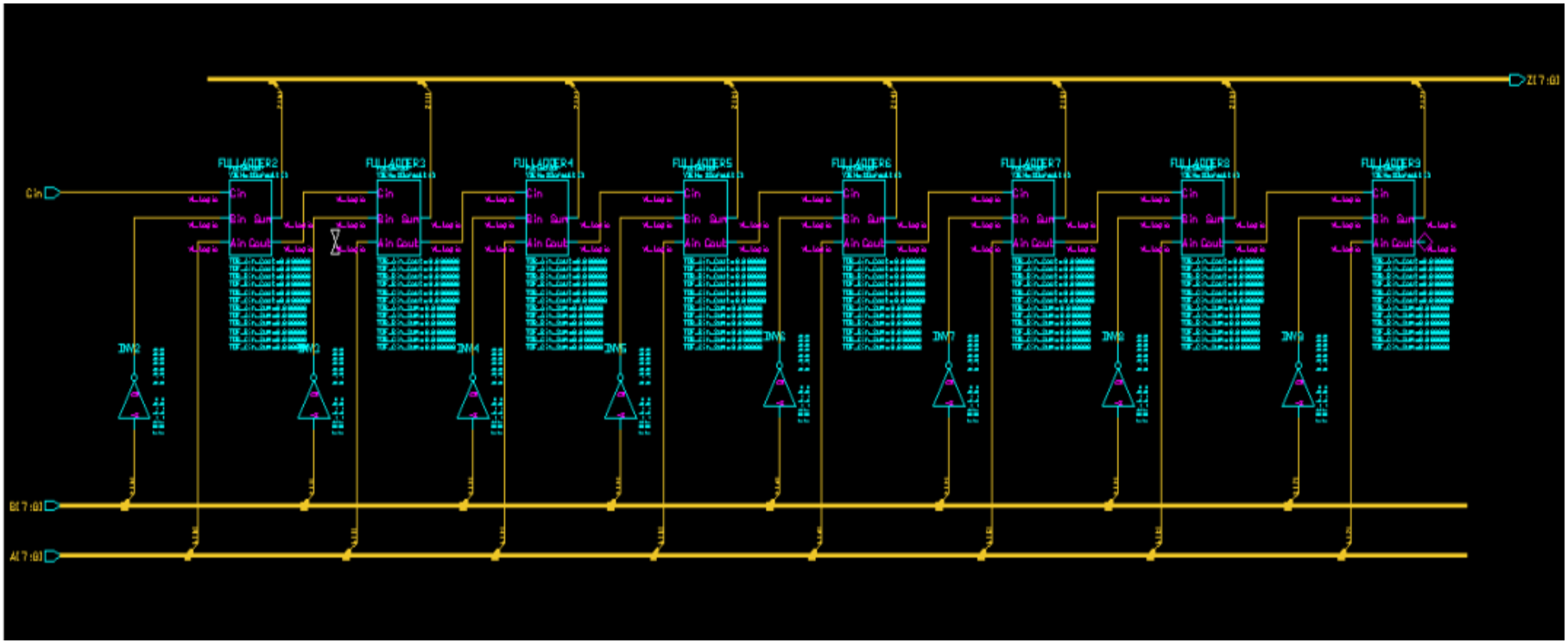
8-bit Ripple Carry Adder



wave - default

:carryadder:A	13	64	-14				18	26	-6			-30	13
:carryadder:B	-29	16	-24	-17	15						-17	-29	
:carryadder:Z	-16	80	-38	-31	1		33	41	9		-23	-59	-16

8-bit Ripple Carry Subtractor



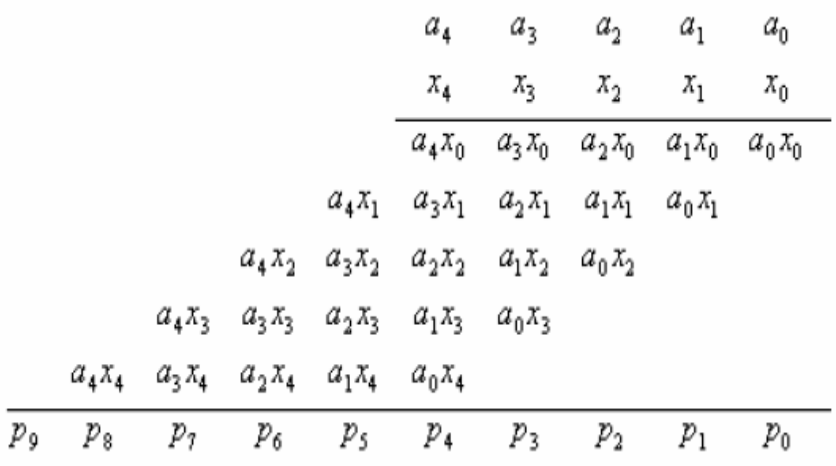
:carrysub:Cin	-1									
:carrysub:A	31	-19		-3		-15		29		31
:carrysub:B	-7	21	-11		-13		-29		-31	25	-7
:carrysub:Z	38	-40	-8	8	10	-2	14	58	60	6	38


Index

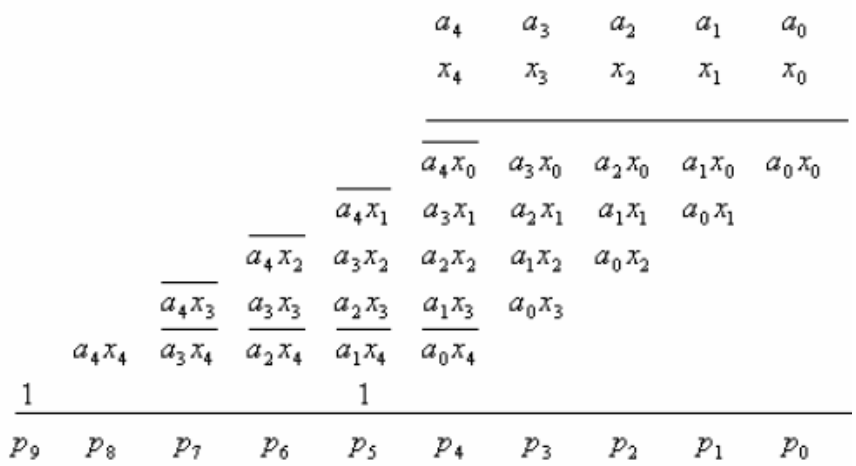
4*4 Bits Signed Multiplier

Quick Overview…

Unsigned Multiplication:



Signed Multiplication:



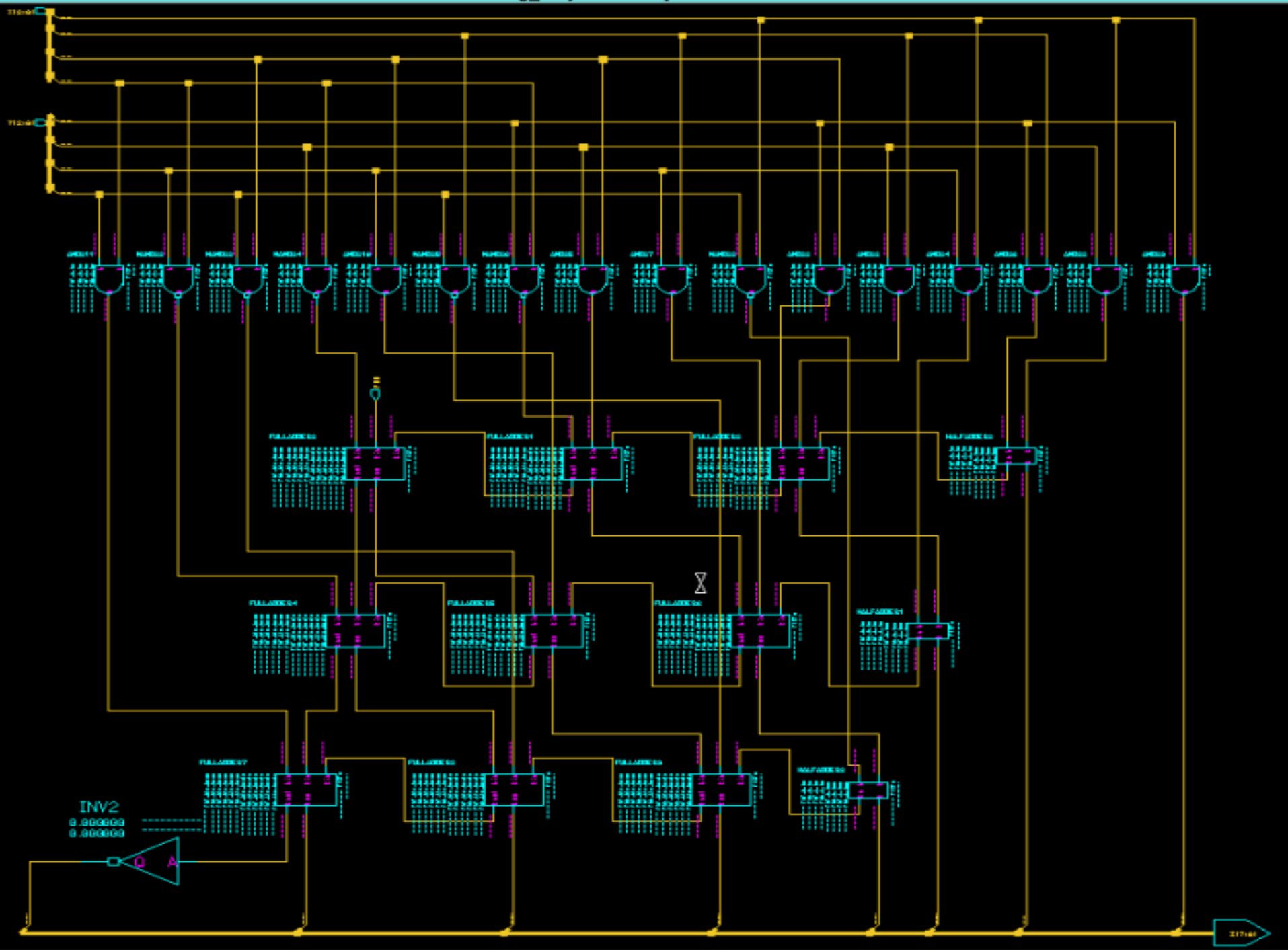
Algorithm: shift and addition

Algorithm: Baugh-Woodley technique

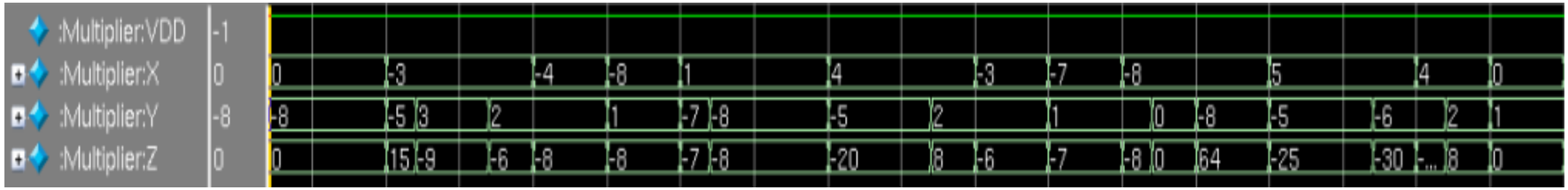
Substitute AND2 gates to NAND2 gates, plus inverter…

Schematics?

Schematic#2 Big_Project / Multiplier / Schematic



- Verification of Multiplier's Results:



Results: 0*8=0

-3*-5=15

-3*3=-9

3*2=-6

-4*2=-8

…

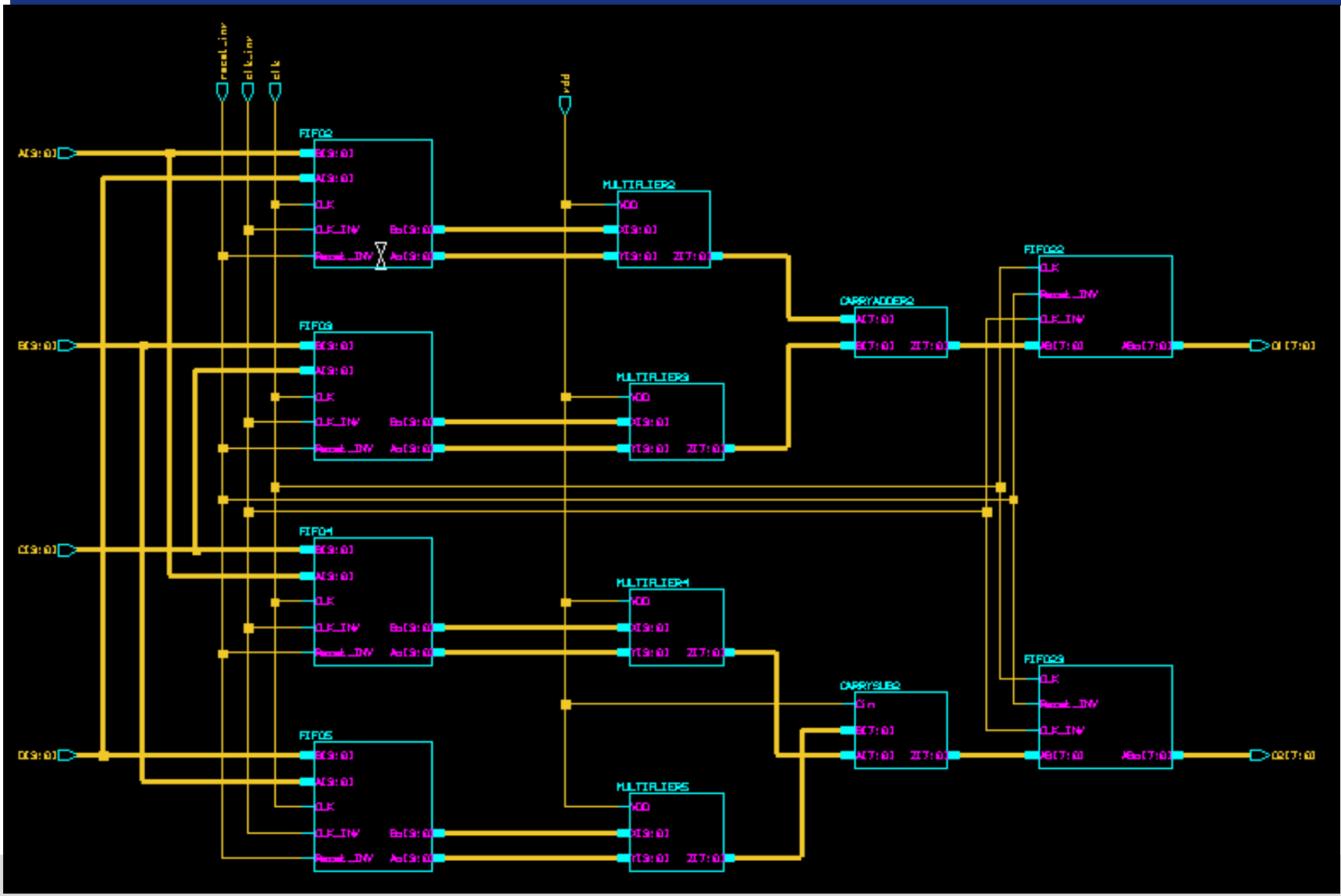
Realizes the multiplication of input X and Y, with the correct outcome Z=XY, which will be sent to Ripple Carry Adder and Subtractor to calculate: AD+BC & AC-BD

Consists of:

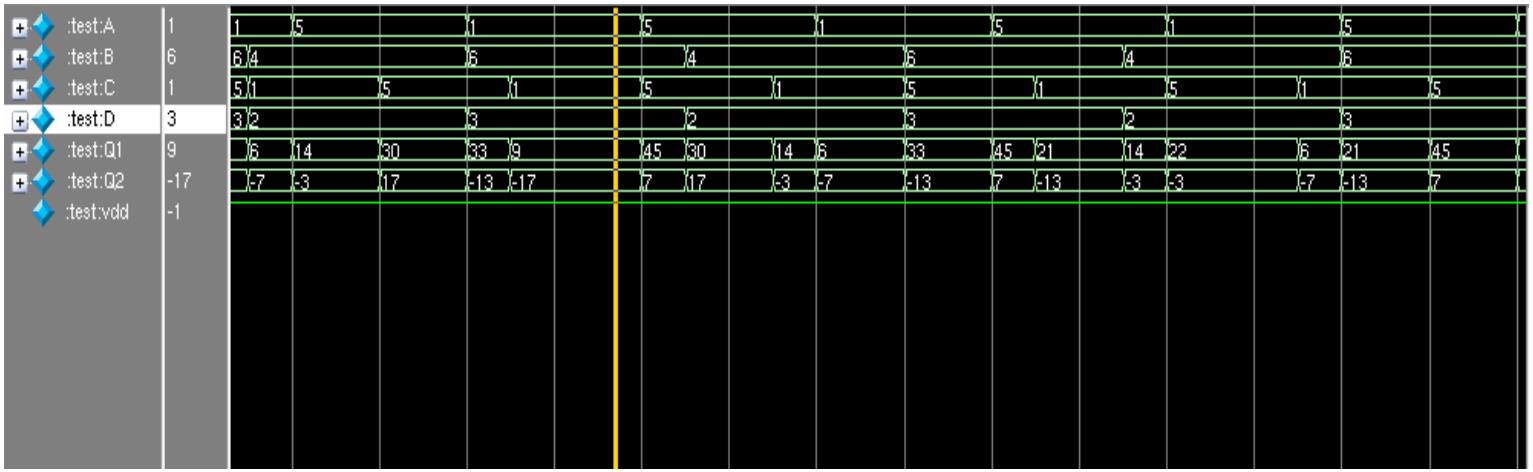
Nand2	× 6
And2	× 10
Full Adders	× 9
Half Adders	× 3
Inv	× 1

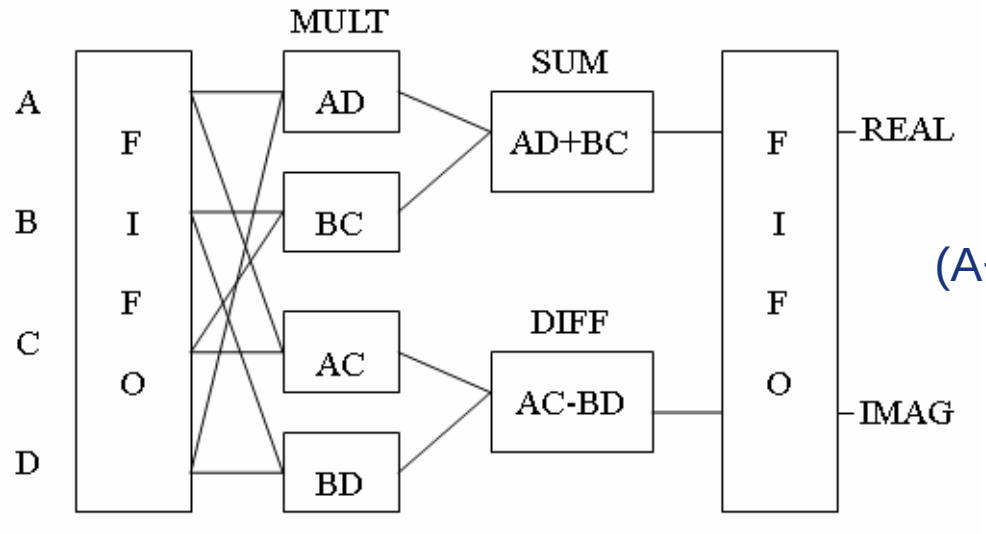
Index

System schematic



System Simulation





$$(A+Bj)(C+Dj)=(AC-BD)+j(BC+AD)$$

Feature estimation

Number of transistors

Components				Total transistors
SUM*1	DIFF*1	MULT*4	FIFO*6	
252	288	434*4	256*6	3812

Area (mm^2)

Components				120%*Total area
SUM*1	DIFF*1	MULT*4	FIFO*6	
0.028	0.033	0.047*4	0.030*6	0.515

Feature estimation

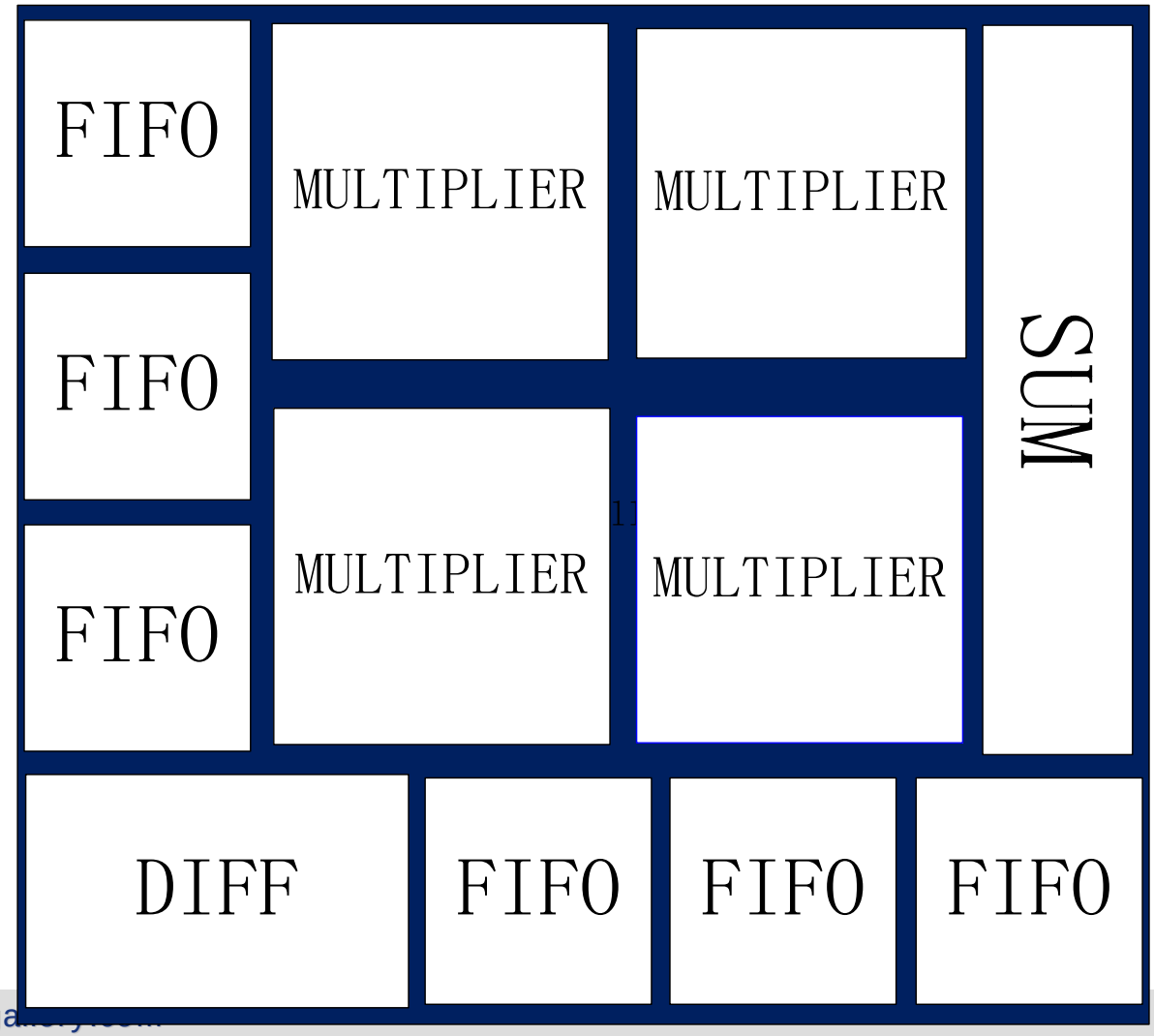
- Power consumption

$$P = \alpha \cdot C \cdot V^2 \cdot f$$

$$= 0.1 \cdot \left[3812 \cdot (12\lambda) \cdot (0.8\mu m / 2\lambda) \cdot (2\, fF / \mu m) \right] \cdot 5^2 \cdot 10^6$$

$$= 0.0915 mW / MHz$$

Floor Plan


FIFO
FIFO
FIFO
MULTIPLIER
MULTIPLIER
MULTIPLIER
MULTIPLIER
SUM
DIFF
FIFO
FIFO
FIFO

To be continued…