Lab Division Algorithm

Kinner Parikh

October 16, 2022

CMPEN 331 - 001

1 Code

```
`timescale 1ns / 1ps
module src(
    input clk, clrn, start, [31:0] a, [15:0] b,
    output reg [15:0] reg_r, reg [31:0] reg_q, reg [4:0] count, reg busy, re
    );
    reg [15:0] reg_b;

    wire [16:0] sub_out;
    wire [16:0] subtrahend = {1'b0, reg_b};
    wire [16:0] minuend = {reg_r, reg_q[31]};
    sub sub_mod(minuend, subtrahend, sub_out);

    wire [15:0] r = {reg_r[14:0], reg_q[31]};
    wire sub_mux_sel = sub_out[16];
    wire [15:0] sub_mux_out;
    sub_mux sub_mux_mod(r, sub_out[15:0], sub_mux_sel, sub_mux_out);

    wire [31:0] ina = a;
    wire [31:0] q = {reg_q[30:0], !sub_out[16]};
    wire [31:0] q_mux_out;
    q_mux q_mux_mod(ina, q, start, q_mux_out);

    always @(posedge clk)
    begin
        //innebeningin
        if (clrn == 0)
        begin
            reg_b <= 0;
            reg_q <= 0;
            reg_r <= 0;
            count <= 0;
            ready <= 0;
            busy  <= 0;
        end
        else if (start == 1)
        begin
            reg_b <= b;
            reg_q <= q_mux_out;
            reg_r <= 0;
            count <= 0;
            ready <= 0;
            busy  <= 1;
        end
        else if (busy == 1)
        begin
            reg_q <= q_mux_out;
            reg_r <= sub_mux_out;
            count <= count + 1;
            if (count == 31)
            begin
                busy <= 0;
                ready <= 1;
            end
        end
    end
endmodule
```

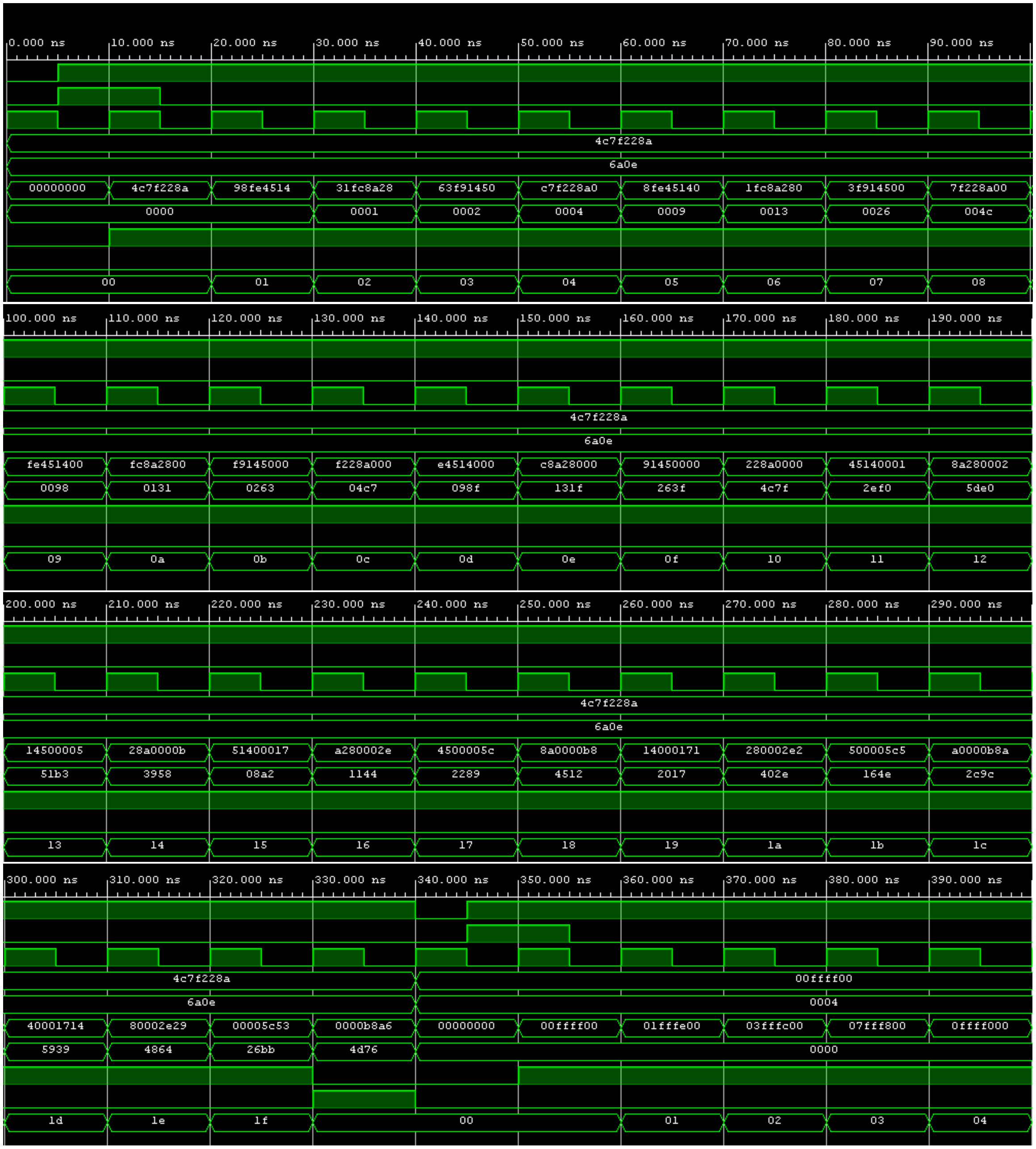
```
‘timescale 1ns / 1ps
module sub(minuend, subtrahend, difference);
    input [16:0] minuend;
    input [16:0] subtrahend;
    output reg [16:0] difference;

    always @(*)
    begin
        difference = minuend - subtrahend;
    end
endmodule
```

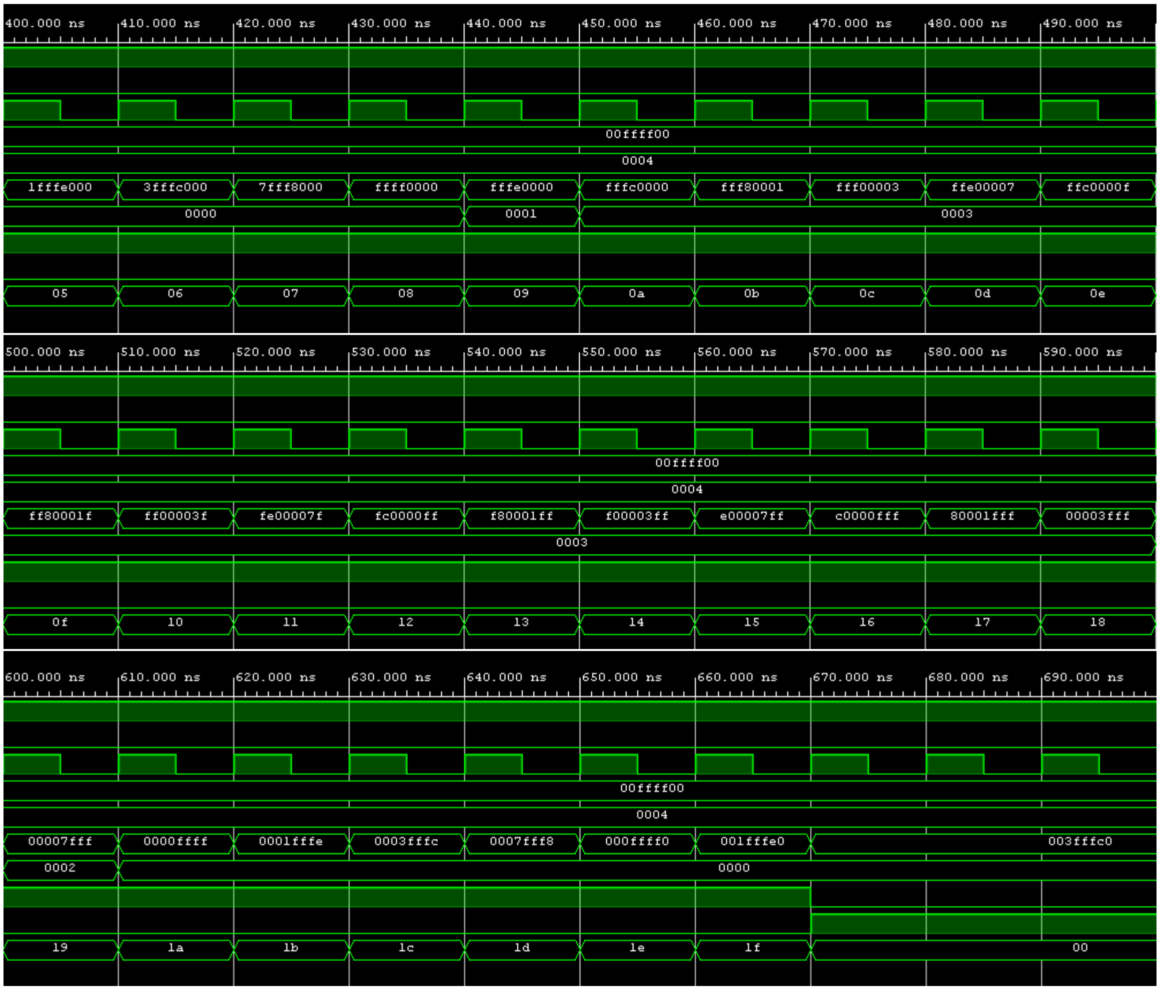
```
‘timescale 1ns / 1ps
module sub_mux(r, sub_in, sel, reg_r);
    input [15:0] r;
    input [15:0] sub_in;
    input sel;
    output reg [15:0] reg_r;

    always @(*)
    begin
        if (sel == 1) // negative case
        begin
            reg_r = r;
        end
        else // positive case
        begin
            reg_r = sub_in[15:0];
        end
    end
endmodule
```

```
‘timescale 1ns / 1ps

module testbench();
    reg clrn, start, clk;
    reg [31:0] a;
    reg [15:0] b;
    wire [31:0] q;
    wire [15:0] r;
    wire busy, ready;
    wire [4:0] count;

    src div(clk, clrn, start, a, b, r, q, count, busy, ready);
    initial
    begin
        clrn = 0;
        start = 0;
        clk = 1;
        a = ’h4c7f228a;
        b = ’h6a0e;
        #5
        start = 1;
        clrn = 1;
        #10
        start = 0;

        #325
        clrn = 0;
        start = 0;
        a = ’hffff00;
        b = ’h4;
        #5
        start = 1;
        clrn = 1;
        #10
        start = 0;
    end
    always
    begin
        #5
        clk = ˜clk;
    end
endmodule
```

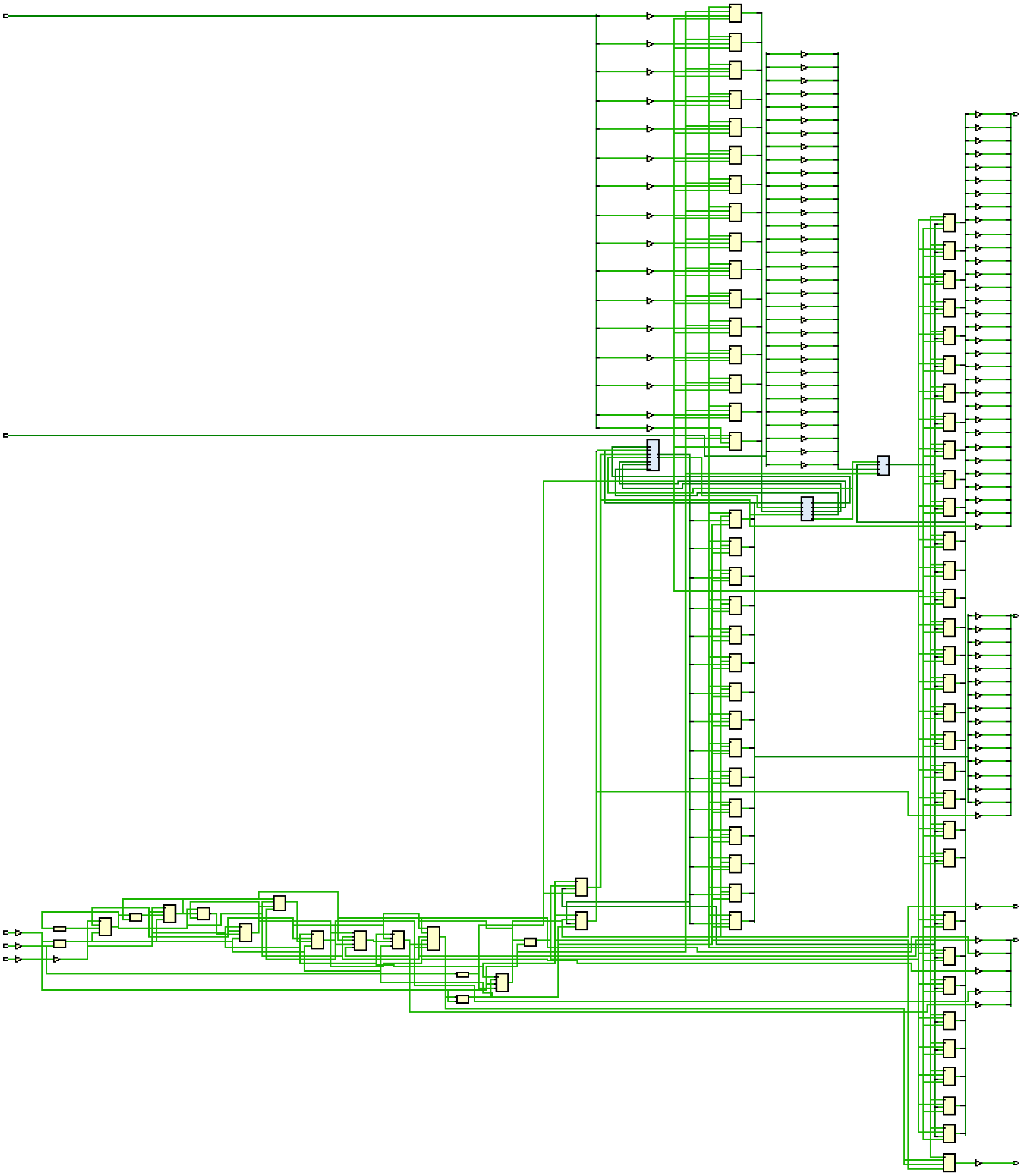
2 Images

Timing Waveform

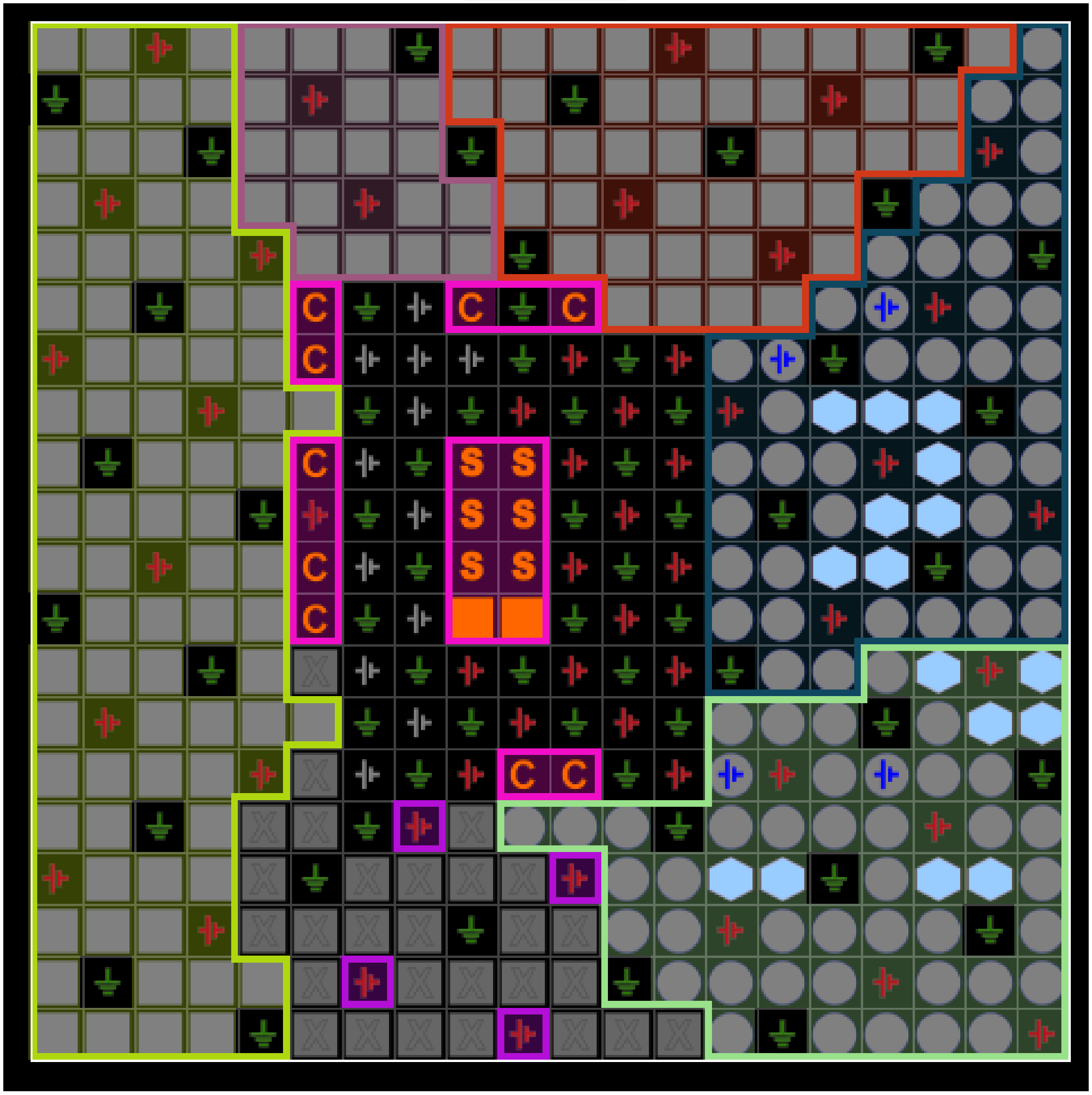



400.000 ns
410.000 ns
420.000 ns
430.000 ns
440.000 ns
450.000 ns
460.000 ns
470.000 ns
480.000 ns
490.000 ns
00ffff00
0004
1fffe000
3fffc000
7fff8000
ffff0000
fffe0000
fffc0000
fff80001
fff00003
ffe00007
ffc0000f
0000
0001
0003
05
06
07
08
09
0a
0b
0c
0d
0e
500.000 ns
510.000 ns
520.000 ns
530.000 ns
540.000 ns
550.000 ns
560.000 ns
570.000 ns
580.000 ns
590.000 ns
00ffff00
0004
ff80001f
ff00003f
fe00007f
fc0000ff
f80001ff
f00003ff
e00007ff
c0000fff
80001fff
00003fff
0003
0f
10
11
12
13
14
15
16
17
18
600.000 ns
610.000 ns
620.000 ns
630.000 ns
640.000 ns
650.000 ns
660.000 ns
670.000 ns
680.000 ns
690.000 ns
00ffff00
0004
00007fff
0000ffff
0001fffe
0003fffc
0007fff8
000ffff0
001fffe0
003fffc0
0002
0000
19
1a
1b
1c
1d
1e
1f
00

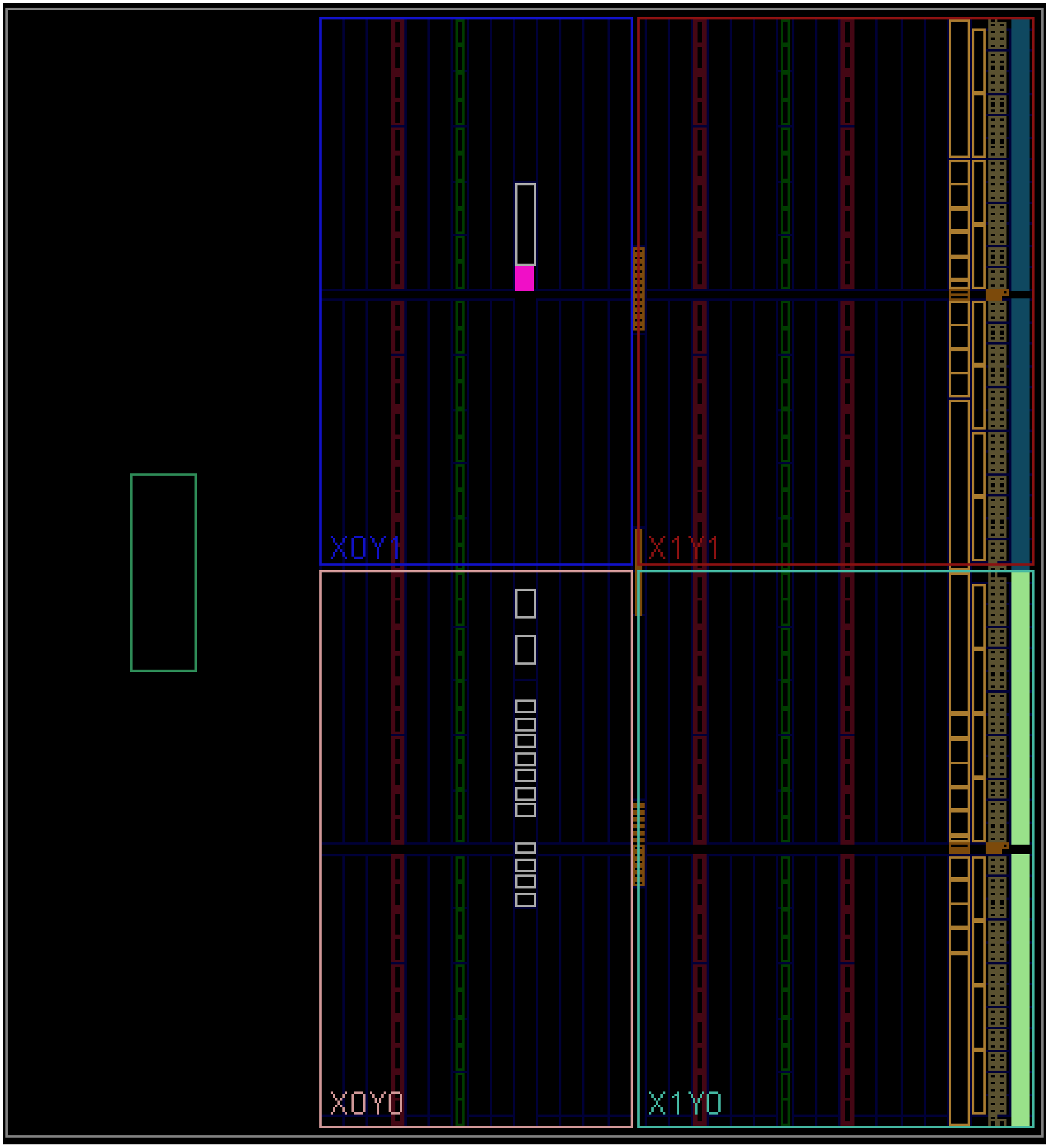
Schematic



I/O Planning



Floor Planning


X0Y1
X1Y1
X0Y0
X1Y0