Lab 3 - Datapath

Kinner Parikh

November 8, 2022

CMPEN 331 - 001

1 Code

```
`timescale 1ns / 1ps
module DataPath(
    input clk,
    output wire [31:0] pc, dinstOut,
    output wire ewreg, em2reg, ewmem, ealuimm,
    output wire [3:0] ealuc,
    output wire [4:0] edestReg,
    output wire [31:0] eqa, eqb, eimm32
);
    wire [31:0] nextPc;
    PC pc_dp(nextPc, clk, pc);

    PCAdder pcadder_dp(pc, nextPc);

    wire [31:0] instOut;
    InstructionMemory im_dp(pc, instOut);

    IFIDPipelineReg ifidreg_dp(instOut, clk, dinstOut);

    wire [5:0] op = dinstOut[31:26];
    wire [5:0] func = dinstOut[5:0];
    wire wreg, m2reg, wmem, aluimm, regrt;
    wire [3:0] aluc;
    ControlUnit cu_dp(op, func, wreg, m2reg, wmem, aluimm, regrt, aluc);

    wire [4:0] rs = dinstOut[25:21];
    wire [4:0] rt = dinstOut[20:16];
    wire [4:0] rd = dinstOut[15:11];
    wire [4:0] destReg;

    RegrtMux regrtmux_dp(rt, rd, regrt, destReg);
    RegFile rf_dp(rs, rt, qa, qb);

    wire [15:0] imm = dinstOut[15:0];
    wire [31:0] imm32;
    ImmExtender immex_dp(imm, imm32);

    IDEXEPipelineReg idexereg_dp(wreg, m2reg, wmem, aluimm, clk, aluc, destR
endmodule
```

```
‘timescale 1ns / 1ps

module PC(
    input [31:0] nextPc,
    input clk,
    output reg [31:0] pc
);
    initial pc = 32’d100;

    always @(posedge clk)
    begin
        pc = nextPc;
    end
endmodule

module InstructionMemory(
    input [31:0] pc,
    output reg [31:0] instOut
);
    reg [31:0] memory [63:0];

    initial begin
        memory[25] <= {6’b100011, 5’b00001, 5’b00010, 16’h0000};
        memory[26] <= {6’b100011, 5’b00001, 5’b00011, 16’h0004};

        /*
        memory[0] <= hA00000AA;
        memory[1] <= h10000011;
        memory[2] <= h20000022;
        memory[3] <= h30000033;
        memory[4] <= h40000044;
        memory[5] <= h50000055;
        memory[6] <= h60000066;
        memory[7] <= h70000077;
        memory[8] <= h80000088;
        memory[9] <= h90000099;
        */
    end

    always @(*)
        instOut <= memory[pc[7:2]];
endmodule
```

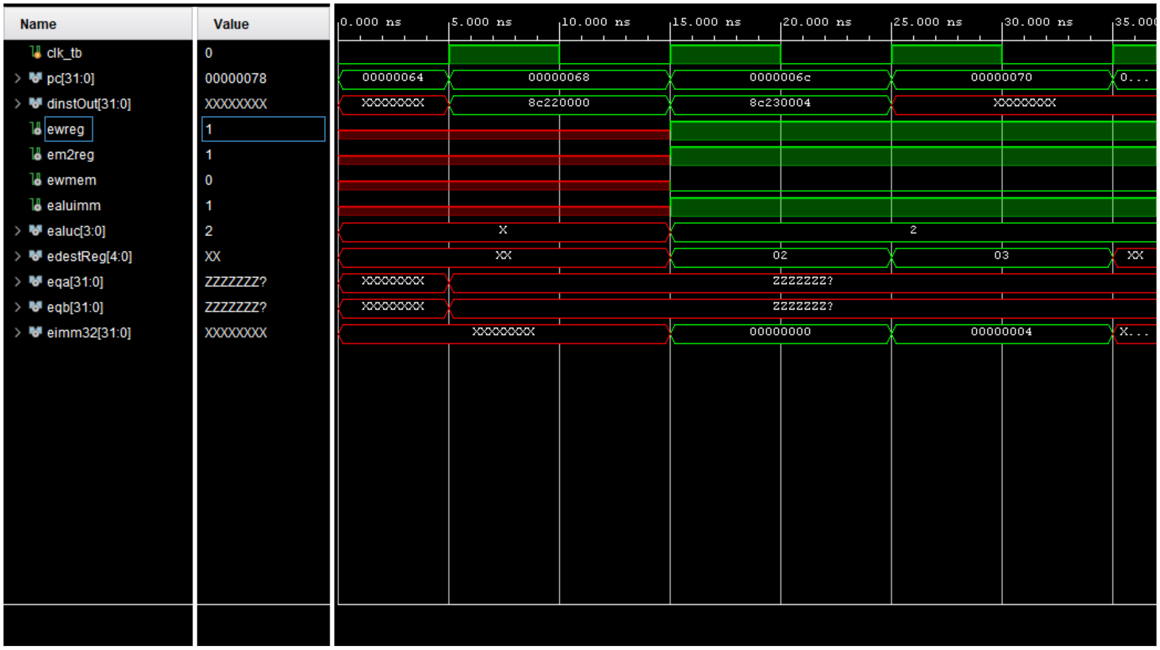
```
module PCAdder(
    input [31:0] pc,
    output reg [31:0] nextPc
);
    always @(*)
        nextPc <= pc + 4;
endmodule

module IFIDPipelineReg(
    input [31:0]instOut,
    input clk,
    output reg [31:0] dinstOut
);
    always @(posedge clk)
        dinstOut <= instOut;
endmodule

module ControlUnit(
    input [5:0] op, func,
    output reg wreg, m2reg, wmem, aluimm, regrt,
    output reg [3:0] aluc
);
    always @(*)
    begin
        case(op)
            'b000000: //r-type
             begin
                wreg   = 'b1;
                m2reg  = 'b0;
                wmem   = 'b0;
                aluimm = 'b0;
                regrt  = 'b0;
                case(func)
                    'b100000: //add
                    begin
                        aluc = 'b0010;
                    end
                    'b100010: //sub
                    begin
                        aluc = 'b0110;
                    end
                endcase
             end
             'b100011: //lw
             begin
                wreg   = 'b1;
                m2reg  = 'b1;
                wmem   = 'b0;
                aluc   = 'b0010;
                aluimm = 'b1;
                regrt  = 'b1;
             end
         endcase
    end
endmodule

module RegrtMux(
    input [4:0] rt, rd,
    input regrt,
    output reg [4:0] destReg
);
    always @(*)
    begin
        if (regrt == 0)
            destReg = rd;
        else
            destReg = rt;
       end
endmodule


module RegFile(
    input [4:0] rs, rt,
    output reg [31:0] qa, qb
);
    reg [31:0] registers[0:31];
    always @(*)
    begin
        qa = registers[rs];
        qb = registers[rt];
    end
endmodule
```

```
module ImmExtender(
    input [15:0] imm,
    output reg [31:0] imm32
);
    always @(*)
        imm32 = {{16{imm[15]}}, imm};
endmodule

module IDEXEPipelineReg(
    input wreg, m2reg, wmem, aluimm, clk,
    input [3:0] aluc,
    input [4:0] destReg,
    input [31:0] qa, qb, imm32,

    output reg ewreg, em2reg, ewmem, ealuimm,
    output reg [3:0] ealuc,
    output reg [4:0] edestReg,
    output reg [31:0] eqa, eqb, eimm32
);
    always @(posedge clk)
    begin
        ewreg    = wreg;
        em2reg   = m2reg;
        ewmem    = wmem;
        ealuc    = aluc;
        ealuimm  = aluimm;
        edestReg = destReg;
        eqa      = qa;
        eqb      = qb;
        eimm32   = imm32;
    end
endmodule
```

```
‘timescale 1ns / 1ps

module testbench();
    reg clk_tb;
    initial clk_tb = 1’b0;

    wire [31:0] pc, dinstOut;
    wire ewreg, em2reg, ewmem, ealuimm;
    wire [3:0] ealuc;
    wire [4:0] edestReg;
    wire [31:0] eqa, eqb, eimm32;

    DataPath dp_tb(clk_tb, pc, dinstOut, ewreg, em2reg, ewmem, ealuimm, ealu

    always
    begin
        #5
        if (clk_tb == 0)
            clk_tb = 1;
        else
            clk_tb = 0;
    end
endmodule
```

2 Images

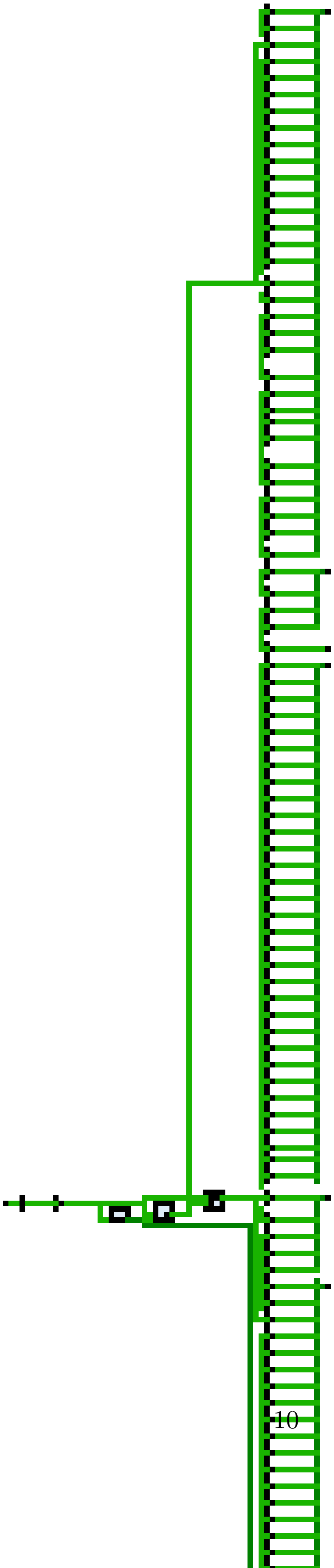
Timing Waveform



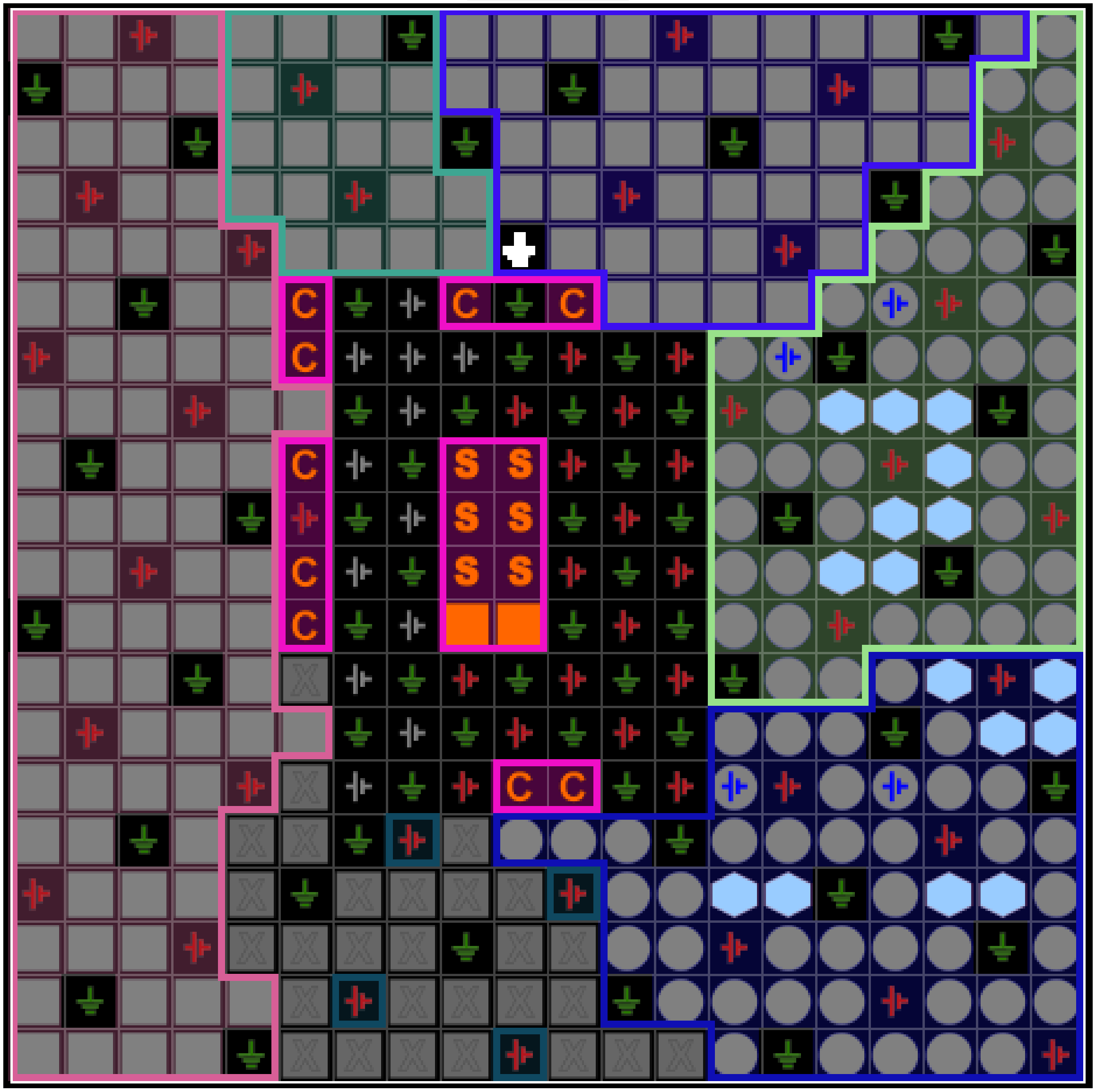
Name	Value
clk_tb	0
pc[31:0]	00000078
dinstOut[31:0]	XXXXXXXX
ewreg	1
em2reg	1
ewmem	0
ealuimm	1
ealuc[3:0]	2
edestReg[4:0]	XX
eqa[31:0]	ZZZZZZZ?
eqb[31:0]	ZZZZZZZ?
eimm32[31:0]	XXXXXXXX

0.000 ns
5.000 ns
10.000 ns
15.000 ns
20.000 ns
25.000 ns
30.000 ns
35.000
00000064
00000068
0000006c
00000070
0...
XXXXXXXX
8c220000
8c230004
XXXXXXXX
X
2
XX
02
03
XX
XXXXXXXX
ZZZZZZZ?
XXXXXXXX
ZZZZZZZ?
XXXXXXXX
00000000
00000004
X...

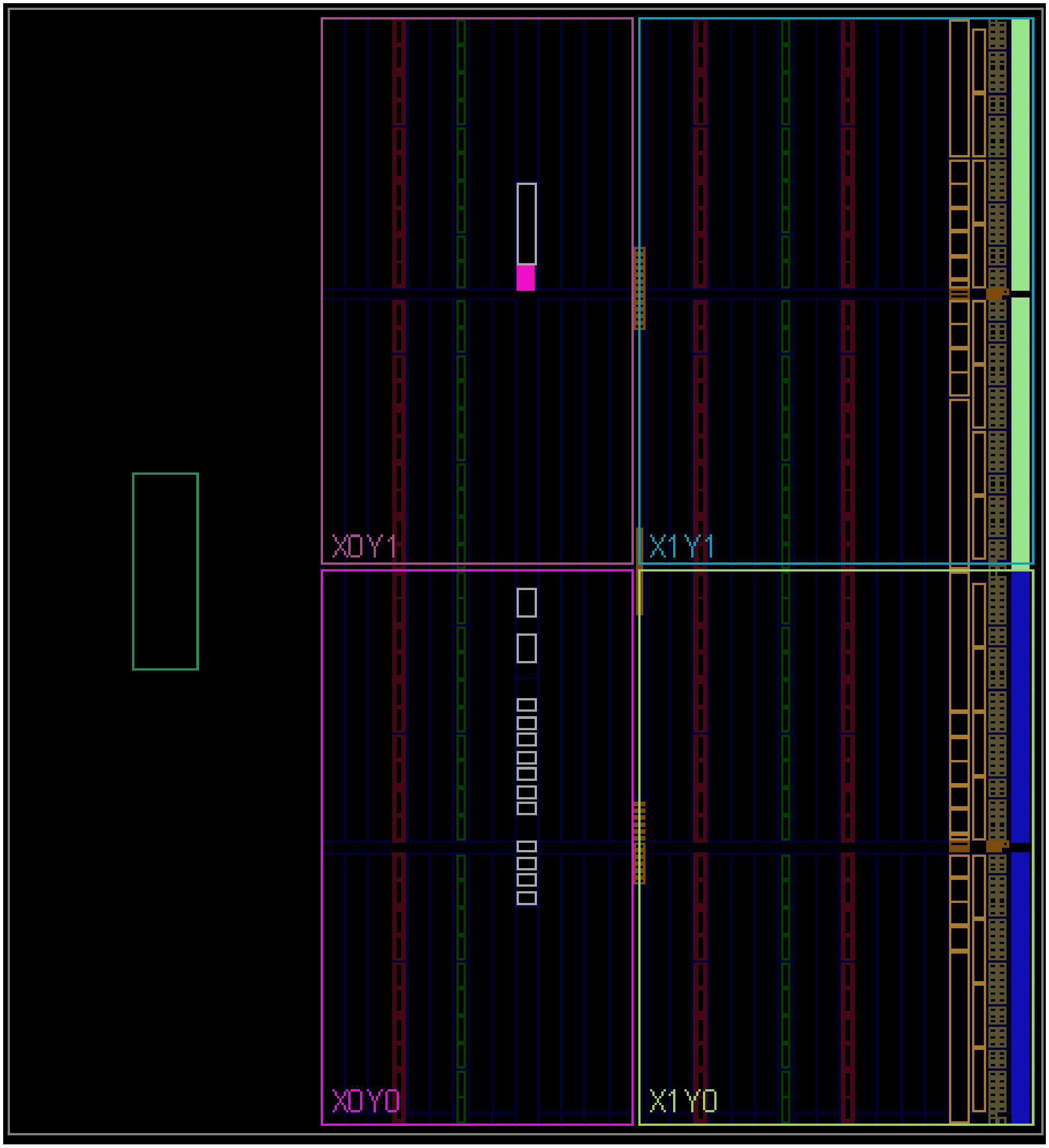
Schematic



I/O Planning



Floor Planning


X0Y1
X1Y1
X0Y0
X1Y0