گزارش کار آزمایشگاه سیستم های دیجیتال2
امیرعلی آرزم جو81019342
فرناز فیض 810198454

در این آزمایش خواسته هایی مطرح شده بود که به شرح زیر هستند:

1.آشنایی با برد

2.طراحی اولیه پردازنده

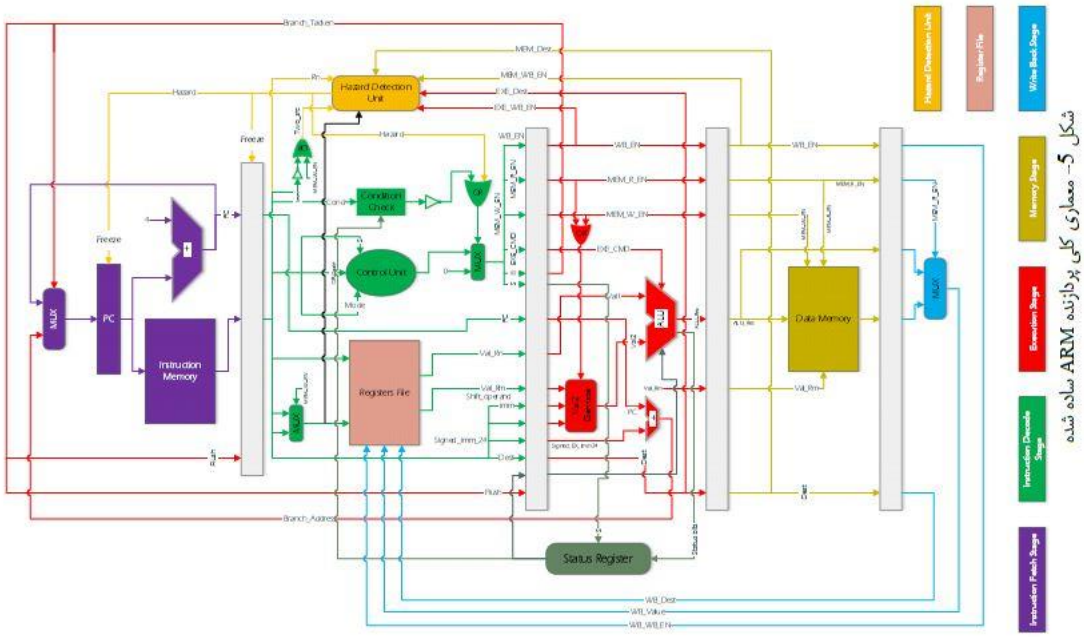
3.تست کردن پایپ لاین

1.در ابتدا یک کد خیلی ساده نوشتیم تا با پروسه سنتز و کار با برد آشنا بشویم:

```
module helloWorld(
    a, b, w
);
    input a, b;
    output w;
    assign w = a || b;
endmodule
```

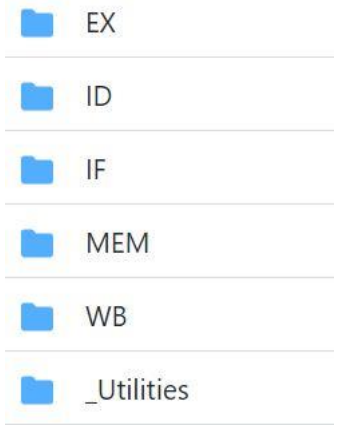
این کد را پس از سنتز pin assignment کردیم و وصل کردیم و با سوییچ ها روشن شدن LED را مشاهده نمودیم.

2 برای طراحی پردازنده ی زیر فایل های رجیستر و 5 بخش رنگی که نشان داده شده است را ساختیم و در مادل سیم کد هارا زدیم.



شکل 5- معماری کلی پردازنده ARM ساده شده

گزارش کار آزمایشگاه سیستم های دیجیتال2
امیرعلی آرزم جو81019342
فرناز فیض 810198454



در جلسه ی اول به روی بخش Instruction Fetch کار کردیم و در بخش Utilities ماژول های پرکاربرد مانند Mux و جمع کننده تعریف کردیم تا بتوانیم در بخش های مختلف پروژه از آن استفاده کنیم.

کد های تعریف شده در تصاویر زیر آورده شده اند

```
1   module EX(
2           CLK,
3           RST,
4           PC_In,
5           PC_Out
6   /*      Signed_EX_imm_24,
7           Branch_Address */
8   );
9
10          input CLK,RST;
11          input [31:0] PC_In;
12          output [31:0] PC_Out;
13
14          assign PC_Out = PC_In;
15
16  /*      input[23:0] Signed_EX_imm_24;
17          output [31:0]Branch_Address; */
18
19  endmodule
```

گزارش کار آزمایشگاه سیستم های دیجیتال2
امیرعلی آرزم جو81019342
فرناز فیض 810198454

```
2   module EX_Reg(
3           CLK,
4           RST,
5           PC_In,
6           PC_Out
7   /*      WB_EN_In,
8           MEM_R_EN_In,
9           MEM_W_EN_In,
10          ALU_Res_In,
11          Val_Rm_In,
12          Dest_In,
13
14          Dest_Out,
15          Val_Rm_Out,
16          ALU_Rs_Out,
17          WB_EN_Out,
18          MEM_R_EN_Out,
19          MEM_W_EN_Out, */
20          );
21          input CLK, RST;
22          input [31:0] PC_In;
23          output [31:0] PC_Out;
24
25          Register #(32) EX_PC_Reg (
26          .clk(CLK),
27          .rst(RST),
28          .ld(1'b1),
29          .regIn(PC_In),
30          .regOut(PC_Out)
31      );
```

```
module IF(clk, rst, freeze, Branch_taken, BranchAddr, PC, Instruction);
    input clk, rst, freeze, Branch_taken;
    input [31:0] BranchAddr;
    output [31:0] PC, Instruction;

    wire [31:0] PC_in, PC_out, PC_plus4;

    assign PC = PC_plus4;

    Mux2 #(32) PC_IN (
        .d0(PC_plus4),
        .d1(BranchAddr),
        .sel(Branch_taken),
        .w(PC_in)
    );

    Register #(32) PC_inst (
        .clk(clk),
        .rst(rst),
        .ld(~freeze),
        .regIn(PC_in),
        .regOut(PC_out)
    );

    PC_Addr PC_Adder_inst (
        .PC(PC_out),
        .PC_plus4(PC_plus4)
    );

    InstructionMem InstMem_inst (
        .Address(PC_out),
        .Instruction(Instruction)
    );

endmodule
```

```
module IF(clk, rst, freeze, Branch_taken, BranchAddr, PC, Instruction);
    input clk, rst, freeze, Branch_taken;
    input [31:0] BranchAddr;
    output [31:0] PC, Instruction;

    wire [31:0] PC_in, PC_out, PC_plus4;

    assign PC = PC_plus4;

    Mux2 #(32) PC_IN (
        .d0(PC_plus4),
        .d1(BranchAddr),
        .sel(Branch_taken),
        .w(PC_in)
    );

    Register #(32) PC_inst (
        .clk(clk),
        .rst(rst),
        .ld(~freeze),
        .regIn(PC_in),
        .regOut(PC_out)
    );

    PC_Addr PC_Adder_inst (
        .PC(PC_out),
        .PC_plus4(PC_plus4)
    );

    InstructionMem InstMem_inst (
        .Address(PC_out),
        .Instruction(Instruction)
    );

endmodule
```

```
module IF_Reg(
        CLK,
        RST,
        freeze,
        PC_In,
        flush,
        InstructionMemory_In,
        PC_Out,
        InstructionMemory_Out
        );

        input CLK,RST,flush,freeze;
        input [31:0] PC_In, InstructionMemory_In;
        output [31:0] PC_Out, InstructionMemory_Out;

        wire reset = RST || flush;
/*      wire clock = CLK && ~freeze;

        always @(posedge clock) begin
                if(reset)
                        {PC_Out, InstructionMemory_Out} <= 0;
                else
                        {PC_Out, InstructionMemory_Out} <= {PC_In, InstructionMemory_In};
        end */

        Register #(32) IF_PC_Reg (
        .clk(CLK),
        .rst(reset),
        .ld(~freeze),
        .regIn(PC_In),
        .regOut(PC_Out)
    );

        Register #(32) IF_InstMem_Reg (
        .clk(CLK),
        .rst(reset),
        .ld(~freeze),
        .regIn(InstructionMemory_In),
        .regOut(InstructionMemory_Out)
    );
```

```
2   module MEM_Reg(
3           CLK,
4           RST,
5           PC_In,
6           PC_Out
7   /*      WB_EN_In,
8           MEM_R_EN_In,
9           ALU_Res_In,
10          Data_In,
11          Dest_In,
12
13          WB_EN_Out,
14          MEM_R_EN_Out,
15          ALU_Res_Out,
16          Data_Out,
17          Dest_Out */
18          );
19          input CLK, RST;
20          input [31:0] PC_In;
21          output [31:0] PC_Out;
22
23          Register #(32) MEM_PC_Reg (
24          .clk(CLK),
25          .rst(RST),
26          .ld(1'b1),
27          .regIn(PC_In),
28          .regOut(PC_Out)
29      );
```

3.میخواستیم روند افزایش سیگنال PC و پایپ لاین را مشاهده بکنیم پس از بخش های دیگر بخش های لازم را نیز کد زدیم.فایل ARM.v را جداگانه ساختیم که قرار است فایل تاپ ماژول ما باشد.

کد تست بنچ نهایی:

گزارش کار آزمایشگاه سیستم های دیجیتال2
امیرعلی آرزم جو81019342
فرناز فیض 810198454

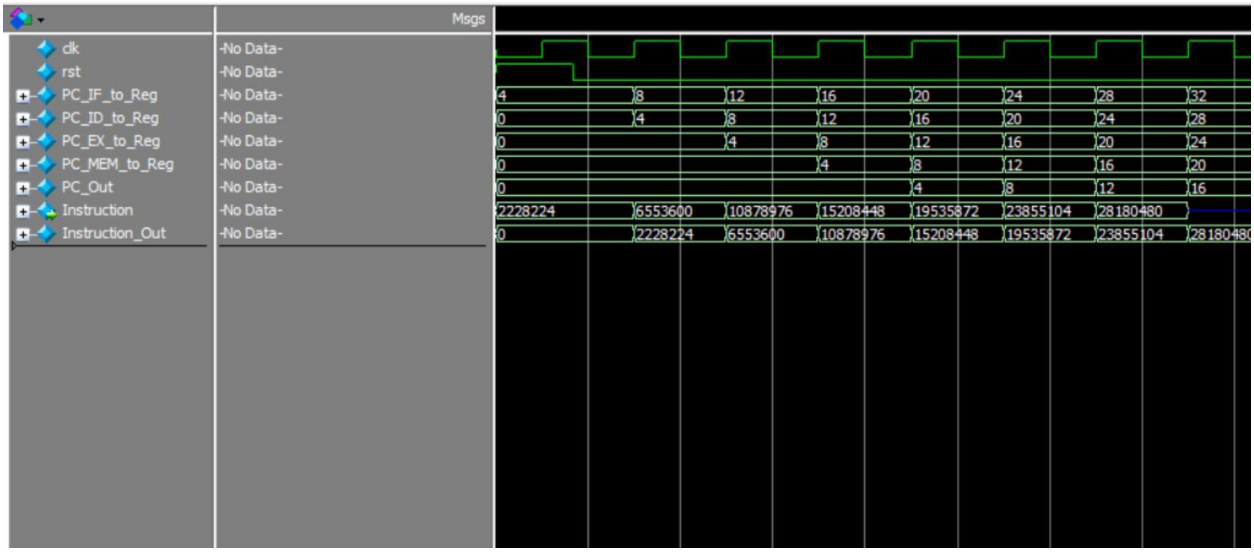
```
`timescale 1ns/1ns
module testbench1();

    reg clk = 0, rst;

    localparam clk_period = 20;
    always #(clk_period/2) clk = ~clk;

    ARM uut (
        .clk(clk),
        .rst(rst)
    );

    initial begin
        rst = 1;
        #17 rst = 0;
        #300 $stop;
    end

endmodule
```

کد تست بنچ برای مشاهده روند تغییرات سیگنال PC:

گزارش کار آزمایشگاه سیستم های دیجیتال2
امیرعلی آرزم جو81019342
فرناز فیض 810198454

```
1   `timescale 1ns/1ns
2   module PC_Branch_Addr_tb();
3       reg [31:0] pc, signed_immed_24;
4       wire [31:0] Branch_Address;
5
6       PC_Branch_Addr uut (
7           .PC(pc),
8           .signed_immed_24(signed_immed_24),
9           .Branch_Address(Branch_Address)
10      );
11
12      initial begin
13          pc = 32'd1000; signed_immed_24 = 0;
14          #5 signed_immed_24 = 32'd500;
15          #100 $stop;
16      end
17
18  endmodule
```

خروجی تست بنچ:



در این شکل موج پایپ لاین و پیشروی سیگنال PC را مشاهده میکنیم که چهار تا چهار تا افزایش میابد.

برای بخش signal tab هم با توجه به دستور العمل داده شده دستور هارا طراحی کردیم تا شکل موج دستور ها را بتوانیم مشاهده کنیم که در پایپ لاین تغییر تک تک سیگنال ها را مشاهده میکنیم.

گزارش کار آزمایشگاه سیستم های دیجیتال2

امیرعلی آرزم جو81019342

فرناز فیض 810198454