Single cycle cpu 的程式碼

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:       23:46:27 12/02/2017
// Design Name:
// Module Name:       Simple_Single_CPU
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module Simple_Single_CPU(
          clk_i,
              rst_n
           );

//I/O port
           input              clk_i;
           input              rst_n;
//Internal Signles
           //PC path data line
           wire   [32-1:0]    pc_out_o;
           wire   [32-1:0]    instr_add_4_o;
           wire   [32-1:0]    Shifter_o;
           wire   [32-1:0]    instr_next_o;
           wire   [32-1:0]    instr_branch_o;
           wire   [32-1:0]    instr_jump_o;
           wire   [32-1:0]jump_address_o;

           wire   [31:0] PCadd;
           //PC path CTRL
           wire   Branch_o;
```

```
        wire    sel_branch_o;
        wire    jump_o;
        //DATA path data line
        wire    [32-1:0]    instr_o;
        wire    [32-1:0]    RSdata_o;
        wire    [32-1:0]    RTdata_o;
        wire    [5-1:0]     RDaddr_o;
        wire    [32-1:0]    SE_o;
        wire    [32-1:0]    ALU_src2_o;
        wire    [32-1:0]    ALU_result_o;
        wire    [32-1:0]MEM_data_o;
        wire    [32-1:0]    WriteBack_o;
        //data path CTRL
        wire    [4-1:0]     ALU_ctrl_o;
        wire    [3-1:0]     ALU_op_o;
        wire    ALUSrc_o;
        wire    RegDst_o;
        wire    RegWrite_o;

        wire    ALU_zero_o;

        wire    ALU_cout_o;
        wire    ALU_overflow_o;
        wire    MemToReg_o;
        wire    MemRead_o;
        wire    MemWrite_o;



//Greate componentes

//PC path
ProgramCounter PC(
        .clk_i(clk_i),
          .rst_n (rst_n),
          .pc_in_i(instr_next_o),
          .pc_out_o(pc_out_o)
        );

Adder Adder1(
        .src1_i(pc_out_o),
          .src2_i(32'd4),
```

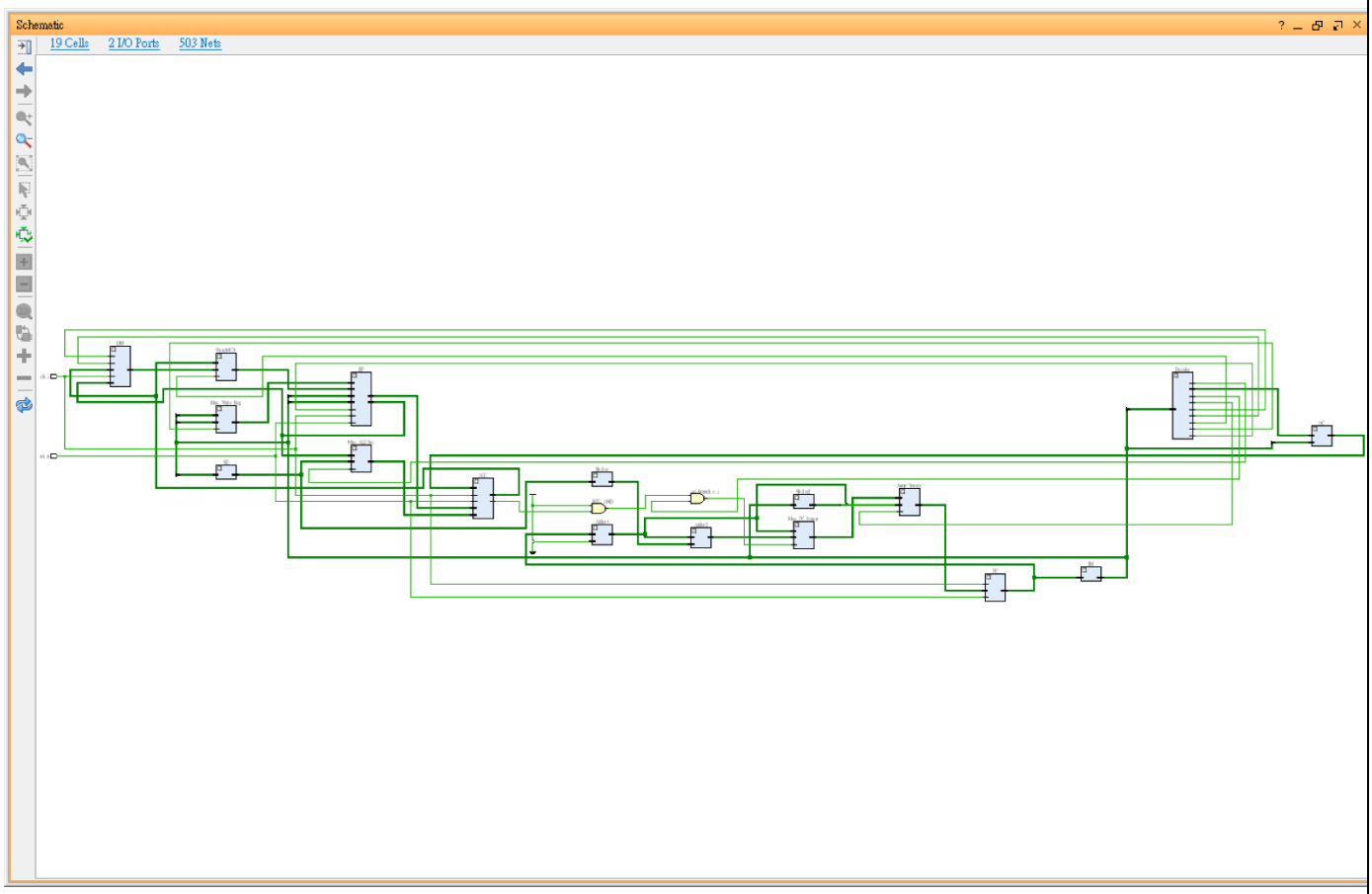
```
            .sum_o(instr_add_4_o)
        );

Adder Adder2(
        .src1_i(instr_add_4_o),
         .src2_i(Shifter_o),
         .sum_o(instr_branch_o)
        );

Shift_Left_Two_32 Shifter(
        .data_i(SE_o),
        .data_o(Shifter_o)
        );

Shift_Left_Two_32 Shifter2(         //JUMP    調整
        .data_i(instr_o),
        .data_o(jump_address_o)
        );

and(sel_branch_o ,ALU_zero_o, Branch_o);


MUX_2to1 #(.size(32)) Mux_PC_Source(        //IF branch
        .data0_i(instr_add_4_o),
        .data1_i(instr_branch_o),
        .select_i(sel_branch_o),
        .data_o(instr_jump_o)
        );

MUX_2to1 #(.size(32)) Jump_Source(                  //IF jump
                .data0_i(instr_jump_o),
                .data1_i({instr_add_4_o[31:28],jump_address_o[27:0]}),
                .select_i(jump_o),
                .data_o(instr_next_o)
                );
        //Data path



Instr_Memory IM(
        .pc_addr_i(pc_out_o),
         .instr_o(instr_o)
```

```
        );


Reg_File RF(
        .clk_i(clk_i),                  //
          .rst_n(rst_n),                //
        .RSaddr_i(instr_o[25:21]),      //5
        .RTaddr_i(instr_o[20:16]),      //5
        .RDaddr_i(RDaddr_o),            //5
        .RDdata_i(WriteBack_o),         //32
        .RegWrite_i(RegWrite_o),        //1
        .RSdata_o(RSdata_o),            //32
        .RTdata_o(RTdata_o)             //32
        );

Decoder Decoder(
        .instr_op_i(instr_o[31:26]),//6b
        .Branch_o(Branch_o),            //1
        .MemtoReg_o(MemToReg_o),        //1
        .MemRead_o(MemRead_o),          //1
        .MemWrite_o(MemWrite_o),//1
        .Jump_o(jump_o),                //1
        .ALUSrc_o(ALUSrc_o),            //1
        .RegWrite_o(RegWrite_o),        //1
        .RegDst_o(RegDst_o),            //1
        .ALU_op_o(ALU_op_o)                 //3b
        );

ALU_Ctrl AC(
        .funct_i(instr_o[5:0]),         //6
           .ALUOp_i(ALU_op_o),          //3
        .ALUCtrl_o(ALU_ctrl_o)          //4
        );

Sign_Extend SE(
        .data_i(instr_o[15:0]),         //16b
        .data_o(SE_o)                   //32b
        );



MUX_2to1 #(.size(5)) Mux_Write_Reg( //selete RT RD
```

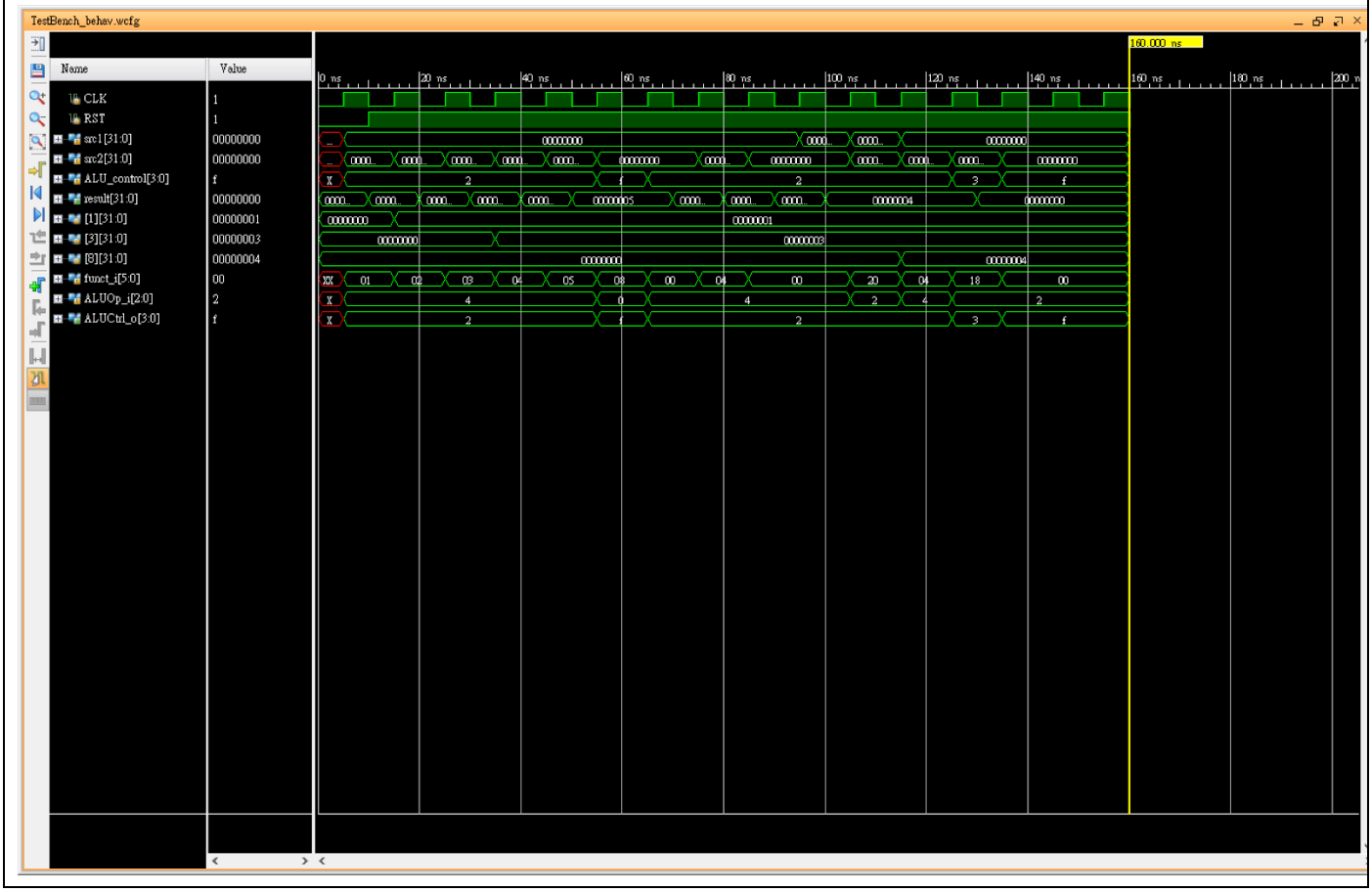
```
        .data0_i(instr_o[20:16]),//32
        .data1_i(instr_o[15:11]),//32
        .select_i(RegDst_o),        //1
        .data_o(RDaddr_o)           //32
        );

MUX_2to1 #(.size(32)) Mux_ALUSrc(   //selete RT imm
        .data0_i(RTdata_o),         //32
        .data1_i(SE_o),                 //32
        .select_i(ALUSrc_o),        //1
        .data_o(ALU_src2_o)             //32
        );

MUX_2to1 #(.size(32)) WriteMUX(     //selete result memory
        .data0_i(ALU_result_o),     //
        .data1_i(MEM_data_o),       //
        .select_i(MemToReg_o),      //
        .data_o(WriteBack_o)        //
        );

Data_Memory DM(
        .clk_i(clk_i),              //
        .addr_i(ALU_result_o),      //
        .data_i(RTdata_o),          //
        .MemRead_i(MemRead_o),      //
        .MemWrite_i(MemWrite_o),    //
        .data_o(MEM_data_o)         //
);

alu ALU(
        .rst(rst_n),                //某一次的測試用 初始化
        .clk(clk_i),                //調整用，沒用
         .src1(RSdata_o),           //
         .src2(ALU_src2_o),         //
         .ALU_control(ALU_ctrl_o),  //
         .result(ALU_result_o),     //
         .zero(ALU_zero_o),         //
         .cout(ALU_cout_o),         //
         .overflow(ALU_ovflow_o)    //
        );
```

```
endmodule
```

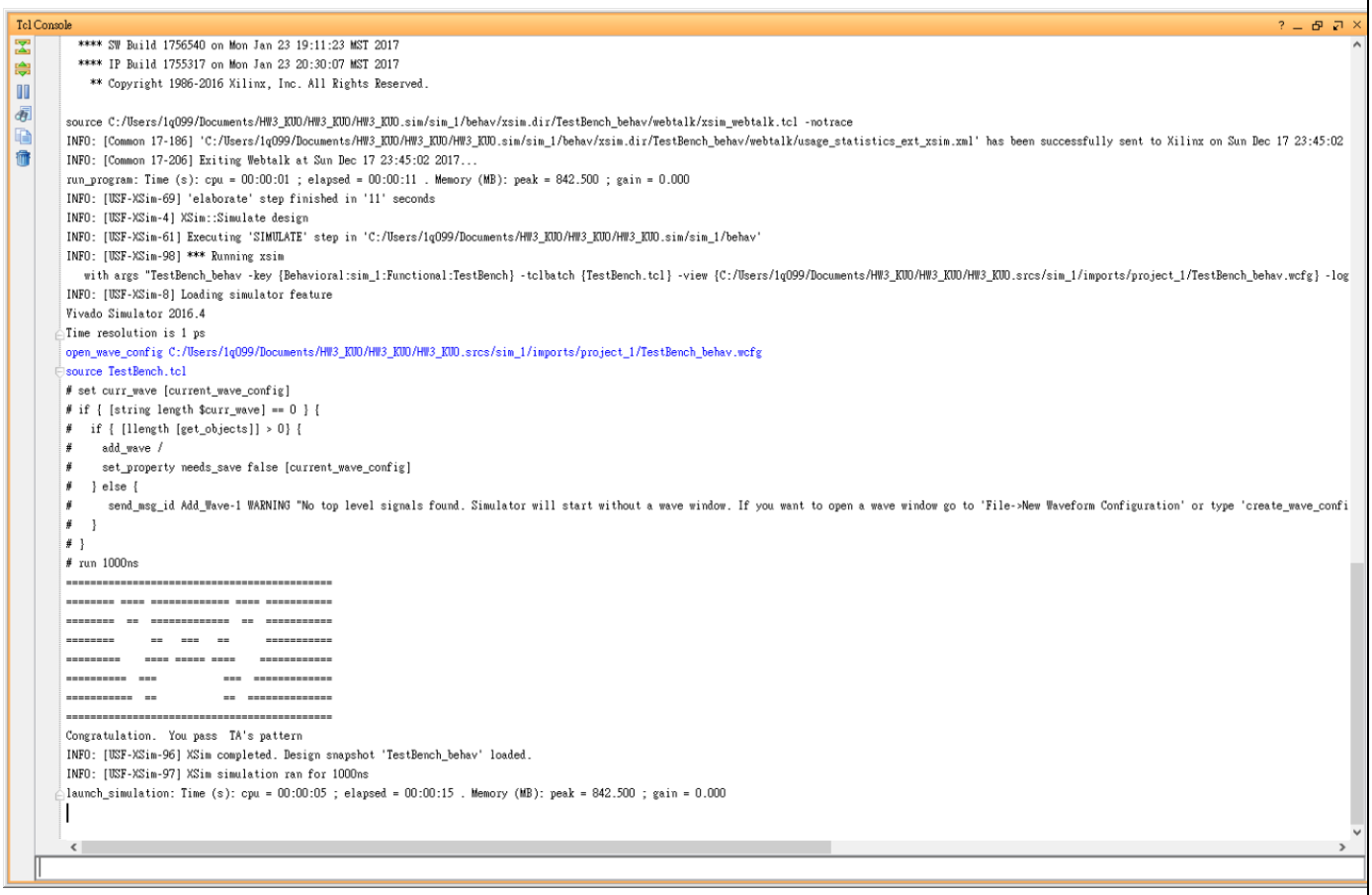
RTL Analysis



Schematic 圖



模擬結果



暫存器資料

Register==========

r0=	0,	r1=	1,	r2=	0,	r3=	3,
r4=	4,	r5=	5,	r6=	1,	r7=	2,
r8=	4,	r9=	2,	r10=	0,	r11=	0,
r12=	0,	r13=	0,	r14=	0,	r15=	0,
r16=	0,	r17=	0,	r18=	0,	r19=	0,
r20=	0,	r21=	0,	r22=	0,	r23=	0,
r24=	0,	r25=	0,	r26=	0,	r27=	0,
r28=	0,	r29=	128,	r30=	0,	r31=	0,

Memory==========

m0=	1,	m1=	2,	m2=	0,	m3=	0,
m4=	0,	m5=	0,	m6=	0,	m7=	0,
m8=	0,	m9=	0,	m10=	0,	m11=	0,
m12=	0,	m13=	0,	m14=	0,	m15=	0,
m16=	0,	m17=	0,	m18=	0,	m19=	0,
m20=	0,	m21=	0,	m22=	0,	m23=	0,
m24=	0,	m25=	0,	m26=	0,	m27=	0,
m28=	0,	m29=	0,	m30=	0,	m31=	0,