Appendix F – MIPS Verilog Code

Design Hierarchy

```
ZoiroSoko.v
 |button.v
 |VGA_Soko.v
 |dataMemory.v
 |instMemory.v
   |textSeg.v
   |interSeg.v
 |srp.v
   |ifStage.v
   |idStage.v
     |regFile.v
   |exStage.v
     |aluUnit.v
     |ksAdder.v
       |ksOpGray.v
       |stage0.v
         |g_p.v
       |stage1.v
         |ksOpBlack.v
       |stage2.v
         |ksOpBlack.v
       |stage3.v
         |ksOpBlack.v
       |stage4.v
         |ksOpBlack.v
       |stage5.v
         |ksOpBlack.v
   |wbStage.v
   |controlUnit.v
   |hazardCtrl.v
   |branchPredict.v
   |coSrp.v
```

Listing 1 mips_def.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : mips_def.v
 * type      :
 * function  : define variables
 * edit      : -
 * author    : iprayudi
 * rev. date : 20081105 - created
 *
 */

`define DELAY        3

`define DONT_CARE_2  2'bxx
`define DONT_CARE_32 32'bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

/* interrupt and exception vector */

`define INT_VECTOR   32'h80000180
`define TEXT_VECTOR  32'h00400000

/* instruction definition */
//-- co-processor inteface
`define OP_RFE       6'b010000
```

```
`define FN_RFE       6'b011000
`define B25_RFE      1'b1

`define OP_MFC0      6'b010000
`define FN_MFC0      6'b000000
`define RS_MFC0      5'b00000

`define OP_MTC0      6'b010000
`define FN_MTC0      6'b000000
`define RS_MTC0      5'b00100

//-- opcode

`define OP_ALU       6'b000000
`define OP_ADD       6'b000000
`define OP_ADDU      6'b000000
`define OP_AND       6'b000000
`define OP_OR        6'b000000
`define OP_SUB       6'b000000
`define OP_SLT       6'b000000
`define OP_SLL       6'b000000
`define OP_SRL       6'b000000

`define OP_J         6'b000010
`define OP_BEQ       6'b000100
`define OP_ADDI      6'b001000
`define OP_ADDIU     6'b001001
`define OP_ANDI      6'b001100
`define OP_ORI       6'b001101
`define OP_LUI       6'b001111
`define OP_LW        6'b100011
`define OP_SW        6'b101011

//-- function code for ADD, SUB, SLT
`define FN_SLL       6'b000000
`define FN_SRL       6'b000010
`define FN_ADD       6'b100000
`define FN_ADDU      6'b100001
`define FN_AND       6'b100100
`define FN_OR        6'b100101
`define FN_SUB       6'b100010
`define FN_SLT       6'b101010

//-- registers
`define R0           5'b00000
`define R1           5'b00001
`define R2           5'b00010
`define R3           5'b00011
`define R4           5'b00100
`define R5           5'b00101
`define R6           5'b00110
`define R7           5'b00111
`define R8           5'b01000
`define R9           5'b01001
`define R10          5'b01010
`define R11          5'b01011
`define R12          5'b01100
`define R13          5'b01101
`define R14          5'b01110
`define R15          5'b01111
`define R16          5'b10000
`define R17          5'b10001
`define R18          5'b10010
`define R19          5'b10011
`define R20          5'b10100
`define R21          5'b10101
`define R22          5'b10110
```

```
`define R23          5'b10111
`define R24          5'b11000
`define R25          5'b11001
`define R26          5'b11010
`define R27          5'b11011
`define R28          5'b11100
`define R29          5'b11101
`define R30          5'b11110
`define R31          5'b11111

//--mc0 register
`define STAT         5'd12
`define CAUSE        5'd13
`define EPC          5'd14

/* alu operation definition */
`define ADD          3'b000
`define SUB          3'b001
`define AND          3'b010
`define OR           3'b011
`define PASS         3'b100
`define SLL          3'b101
`define SRL          3'b111

/* extension operation definition */
`define SIGN         2'b00
`define ZERO         2'b01
`define LUI          2'b10

/* brach prediction status */
`define TAKEN_1      2'd3
`define TAKEN_0      2'd2
`define NOTTAKEN_1   2'd1
`define NOTTAKEN_0   2'd0

/*
 * alhamdulillah
 */
```

Listing 2 aluUnit.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : aluUnit.v
 * type      : rtl
 * function  : arithmetic logic unit
 * edit      : -
 * author    : iprayudi
 * rev. date : 20081015 - created
 *
 */

module aluUnit (
   aluInA    , // input
   aluInB    , // input
   setLessEn , // input
   aluOp     , // input
   shamt     , // input
   aluOut    , // output
   overflow    // output
   );

 /* input ports */
 input wire [31:0] aluInA    ;
```

```
 input wire [31:0] aluInB    ;
 input wire        setLessEn ;
 input wire [2:0]  aluOp     ;
 input wire [4:0]  shamt     ;

 /* output ports */
 output wire [31:0] aluOut   ;
 output wire        overflow ;

 /* internal variables */
 wire [31:0] aluOutBuff   ;

 wire [31:0] aluInBInv    ;
 wire [31:0] aluInB2sComp ;
 wire [31:0] aluInBBuff   ;

 wire [31:0] addResult    ;
 wire [31:0] andResult    ;
 wire [31:0] orResult     ;
 wire [31:0] sllResult    ;
 wire [31:0] srlResult    ;
 wire [2:0]  msb          ;

 /* 2's complement for aluInB */
 assign aluInBInv    = ~aluInB           ;
 assign aluInB2sComp = aluInBInv + 32'd1 ;
 assign aluInBBuff   = (aluOp == `SUB) ? aluInB2sComp : aluInB ;

 /* add operation */
 assign addResult  = aluInA + aluInBBuff ;

 /* and operation */
 assign andResult  = aluInA & aluInBBuff ;

 /* or operation */
 assign orResult   = aluInA | aluInBBuff ;

 /* sll operation */
 assign sllResult  = aluInA << shamt     ;

 /* srl operation */
 assign srlResult  = aluInBBuff >> shamt ;

 /* alu behavior */
 assign aluOutBuff = (aluOp == `ADD) | (aluOp == `SUB) ? addResult :
                     (aluOp == `SLL)                   ? sllResult :
                     (aluOp == `SRL)                   ? srlResult :
                     (aluOp == `AND)                   ? andResult : orResult ;

 /* overflow exception detector */
 assign msb      = {aluInA[31], aluInBBuff[31], addResult[31]};
 assign overflow = ((aluOp == `ADD)&((msb == 3'b001)|(msb == 3'b110)) ) |
                   ((aluOp == `SUB)&((msb == 3'b011)|(msb == 3'b100)) ) ? 1'b1 : 1'b0 ;

 /* output behavior */
 assign aluOut = (setLessEn) ? {{31{1'b0}},aluOutBuff[31]} : aluOutBuff ;

endmodule

/*
 * alhamdulillah
 */
```

Listing 3 regFile.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : regFile.v
-- TYPE      : rtl
-- FUNCTION  : MIPS register File
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/16
-----------------------------------------------*/

module regFile (
   clk       , // input
   rst_n     , // input
   rs        , // input
   rt        , // input
   regFWEn   , // input
   regFWData , // input
   regFWAddr , // input
   regRS     , // output
   regRT       // output
   );

 // input ports
 input wire        clk       ;
 input wire        rst_n     ;

 input wire [4:0]  rs        ;
 input wire [4:0]  rt        ;

 input wire        regFWEn   ;
 input wire [31:0] regFWData ;
 input wire [4:0]  regFWAddr ;

 // output ports
 output wire [31:0] regRS;
 output wire [31:0] regRT;

 // internal variables
 reg [31:0] registerBank [0:31] ;

 // read process
 assign regRS = registerBank[rs] ;
 assign regRT = registerBank[rt] ;

 // write process
 always @ (negedge clk or negedge rst_n)
  if (~rst_n)
   begin
    registerBank[0]  <= #`DELAY 32'd0 ;
    registerBank[1]  <= #`DELAY 32'd0 ;
    registerBank[2]  <= #`DELAY 32'd0 ;
    registerBank[3]  <= #`DELAY 32'd0 ;
    registerBank[4]  <= #`DELAY 32'd0 ;
    registerBank[5]  <= #`DELAY 32'd0 ;
    registerBank[6]  <= #`DELAY 32'd0 ;
    registerBank[7]  <= #`DELAY 32'd0 ;
    registerBank[8]  <= #`DELAY 32'd0 ;
    registerBank[9]  <= #`DELAY 32'd0 ;

    registerBank[10] <= #`DELAY 32'd0 ;
    registerBank[11] <= #`DELAY 32'd0 ;
    registerBank[12] <= #`DELAY 32'd0 ;
    registerBank[13] <= #`DELAY 32'd0 ;
    registerBank[14] <= #`DELAY 32'd0 ;
    registerBank[15] <= #`DELAY 32'd0 ;
```

```
    registerBank[16] <= #`DELAY 32'd0 ;
    registerBank[17] <= #`DELAY 32'd0 ;
    registerBank[18] <= #`DELAY 32'd0 ;
    registerBank[19] <= #`DELAY 32'd0 ;

    registerBank[20] <= #`DELAY 32'd0 ;
    registerBank[21] <= #`DELAY 32'd0 ;
    registerBank[22] <= #`DELAY 32'd0 ;
    registerBank[23] <= #`DELAY 32'd0 ;
    registerBank[24] <= #`DELAY 32'd0 ;
    registerBank[25] <= #`DELAY 32'd0 ;
    registerBank[26] <= #`DELAY 32'd0 ;
    registerBank[27] <= #`DELAY 32'd0 ;
    registerBank[28] <= #`DELAY 32'd0 ;
    registerBank[29] <= #`DELAY 32'd0 ;

    registerBank[30] <= #`DELAY 32'd0 ;
    registerBank[31] <= #`DELAY 32'd0 ;
   end
  else if (regFWEn)
   begin
    registerBank[regFWAddr] <= #`DELAY regFWData;
   end

endmodule

/*
 * alhamdulillah
 */
```

Listing 4 controlUnit.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : controlUnit.v
-- TYPE      : rtl
-- FUNCTION  : control unit
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/15
-----------------------------------------------*/

module controlUnit (
   clk           , // input
   rst_n         , // input

   ovIntReq      , // input

   opCode        , // input
   rs            , // input
   funct         , // input
   shamt         , // input

   // to co-proc
   mc0WEn        , // output

   // to hazardCtrl
   dMemWEnID     , // output

   // to idStage
   takeEn        , // input
   regFDst       , // output
   aluSrc        , // output
   extOp         , // output
```

```
  // to exStage
  aluOpIDEX     , // output
  setLessEnIDEX , // output
  dMemWEnIDEX   , // output
  regFWEnIDEX   , // output
  dMemREnIDEX   , // output
  shamtIDEX     , // output
  ovEnIDEX      , // output
  mc0REnIDEX    , // output

  // to wbStage
  regFWEnEXWB   , // output

  // to brancPredict
  intFinish     , // output
  beq           , // output
  brEn            // output
  );

// input ports
input wire       clk           ;
input wire       rst_n         ;

input wire [5:0] opCode        ;
input wire [4:0] rs            ;
input wire [5:0] funct         ;
input wire [4:0] shamt         ;

input wire       takeEn        ;

input wire       ovIntReq      ;

// output ports
output wire       regFDst      ;
output wire       aluSrc       ;
output wire [1:0] extOp        ;

output wire [2:0] aluOpIDEX    ;
output wire       setLessEnIDEX;
output wire       dMemWEnIDEX  ;
output wire       regFWEnIDEX  ;
output wire       dMemREnIDEX  ;
output wire [4:0] shamtIDEX    ;
output wire       ovEnIDEX     ;
output wire       mc0REnIDEX   ;

output wire       regFWEnEXWB  ;

output wire       dMemWEnID    ;

output wire       beq          ;
output wire       brEn         ;

output wire       mc0WEn       ;

output wire       intFinish    ;

// internal variables
wire       rTypeAdd  ;
wire       rTypeAddu ;
wire       rTypeAnd  ;
wire       rTypeOr   ;
wire       rTypeSub  ;
wire       rTypeSlt  ;
wire       rTypeSll  ;
wire       rTypeSrl  ;
wire       addi      ;
```

```
 wire       addiu     ;
 wire       lw        ;
 wire       sw        ;
 wire       andi      ;
 wire       ori       ;
 wire       lui       ;

 wire       mc0REn    ;

 wire [2:0] aluOp     ;
 wire       setLessEn ;
 wire       ovEn      ;

 wire       dMemWEnBuff     ;
 wire       regFWEnBuff     ;
 wire       regFWEnIDEXBuff ;


 assign rTypeAdd  = (opCode == `OP_ADD ) & (funct == `FN_ADD ) ? 1'b1 : 1'b0 ;
 assign rTypeAddu = (opCode == `OP_ADDU) & (funct == `FN_ADDU) ? 1'b1 : 1'b0 ;
 assign rTypeAnd  = (opCode == `OP_AND ) & (funct == `FN_AND ) ? 1'b1 : 1'b0 ;
 assign rTypeOr   = (opCode == `OP_OR  ) & (funct == `FN_OR  ) ? 1'b1 : 1'b0 ;
 assign rTypeSub  = (opCode == `OP_SUB ) & (funct == `FN_SUB ) ? 1'b1 : 1'b0 ;
 assign rTypeSlt  = (opCode == `OP_SLT ) & (funct == `FN_SLT ) ? 1'b1 : 1'b0 ;
 assign rTypeSll  = (opCode == `OP_SLL ) & ~(shamt == 5'd0 ) & (funct == `FN_SLL ) ?
1'b1 : 1'b0 ;
 assign rTypeSrl  = (opCode == `OP_SRL ) & ~(shamt == 5'd0 ) & (funct == `FN_SRL ) ?
1'b1 : 1'b0 ;

 assign addi      = (opCode == `OP_ADDI ) ? 1'b1 : 1'b0 ;
 assign addiu     = (opCode == `OP_ADDIU) ? 1'b1 : 1'b0 ;
 assign andi      = (opCode == `OP_ANDI ) ? 1'b1 : 1'b0 ;
 assign ori       = (opCode == `OP_ORI  ) ? 1'b1 : 1'b0 ;
 assign lui       = (opCode == `OP_LUI  ) ? 1'b1 : 1'b0 ;
 assign lw        = (opCode == `OP_LW   ) ? 1'b1 : 1'b0 ;
 assign sw        = (opCode == `OP_SW   ) ? 1'b1 : 1'b0 ;

 assign beq       = (opCode == `OP_BEQ ) ? 1'b1   : 1'b0 ;
 assign brEn      = (beq               ) ? takeEn : 1'b0 ;

 assign mc0WEn    = ((opCode == `OP_MTC0) & (rs == `RS_MTC0)) ? 1'b1 : 1'b0 ;
 assign intFinish = (opCode == `OP_RFE) & (funct == `FN_RFE)  ? 1'b1 : 1'b0 ;

 //
 // control signal for ID-stage
 //

 // register file write destination address selector
 assign regFDst   = (opCode == 6'h0) ? 1'd1 : 1'd0 ;  // 1 = rd ; 0 = rt

 // alu input B selector
 assign aluSrc    = (opCode == 6'h0) ? 1'b0 : 1'b1 ;

 // extension control signal
//  assign extOp     = ((ori) | (andi)) ? `ZERO :
//                     (lui           ) ? `LUI  : `SIGN ;

 assign extOp[0] = (ori | andi) ? 1'b1 : 1'b0 ;
 assign extOp[1] = (lui)        ? 1'b1 : 1'b0 ;

 //
 // control signal for EX-stage
 //

 // overflow enable signal
 assign ovEn      = (rTypeAdd | rTypeSub | addi ) ? 1'b1 : 1'b0 ;
```

```
delay1 PIPE_ovEnIDEX (
   .clk   (clk      ), // input
   .rst_n (rst_n    ), // input

   .inp   (ovEn     ), // input
   .outp  (ovEnIDEX )  // output
   );

 // mc0 read enable signal
 assign mc0REn    = ((opCode == `OP_MFC0) & (rs == `RS_MFC0)) ? 1'b1 : 1'b0 ;

delay1 PIPE_mc0REnIDEX (
   .clk   (clk        ), // input
   .rst_n (rst_n      ), // input

   .inp   (mc0REn     ), // input
   .outp  (mc0REnIDEX )  // output
   );

 // shamt
delay5 PIPE_shamtIDEX (
   .clk   (clk       ), // input
   .rst_n (rst_n     ), // input

   .inp   (shamt     ), // input
   .outp  (shamtIDEX )  // output
   );

 // set less than enable signal
 assign setLessEn = (rTypeSlt) ? 1'b1 : 1'b0 ;

delay1 PIPE_setLessEnIDEX (
   .clk   (clk           ), // input
   .rst_n (rst_n         ), // input

   .inp   (setLessEn     ), // input
   .outp  (setLessEnIDEX )  // output
   );

 // alu operation signal
//  assign aluOp = (rTypeSub | rTypeSlt) ? `SUB  :
//                 (rTypeAnd | andi    ) ? `AND  :
//                 (rTypeOr  | ori     ) ? `OR   :
//                 (rTypeSll           ) ? `SLL  :
//                 (rTypeSrlv          ) ? `SRLV :
//                 (lui                ) ? `PASS : `ADD ;

 assign aluOp[0] = (rTypeSub | rTypeSlt | rTypeOr | ori | rTypeSll | rTypeSrl) ? 1'b1 :
1'b0 ;
 assign aluOp[1] = (rTypeAnd | andi | rTypeOr | ori | rTypeSrl)                ? 1'b1 :
1'b0 ;
 assign aluOp[2] = (lui | rTypeSll | rTypeSrl)                                 ? 1'b1 :
1'b0 ;

delay3 PIPE_aluOpIDEX (
   .clk   (clk       ), // input
   .rst_n (rst_n     ), // input

   .inp   (aluOp     ), // input
   .outp  (aluOpIDEX )  // output
   );

 // data memory write enable signal
 assign dMemWEn     = (sw)       ? 1'b1 : 1'b0    ;
 assign dMemWEnBuff = (ovIntReq) ? 1'b0 : dMemWEn ;
 assign dMemWEnID   = dMemWEn ;
```

```
delay1 PIPE_dMemWEnIDEX (
   .clk   (clk         ), // input
   .rst_n (rst_n       ), // input

   .inp   (dMemWEnBuff ), // input
   .outp  (dMemWEnIDEX )  // output
   );

 // data memory read enable signal
 assign dMemREn = (lw) ? 1'b1 : 1'b0 ;

delay1 PIPE_dMemREnIDEX (
   .clk   (clk         ), // input
   .rst_n (rst_n       ), // input

   .inp   (dMemREn     ), // input
   .outp  (dMemREnIDEX )  // output
   );

 //
 // control signal for WB-stage
 //

 // register file write enable signal
 assign regFWEn = (rTypeAdd  |
                   rTypeAddu |
                   rTypeAnd  |
                   rTypeOr   |
                   rTypeSub  |
                   rTypeSlt  |
                   rTypeSll  |
                   rTypeSrl  |
                   lw        |
                   addi      |
                   addiu     |
                   lui       |
                   ori       |
                   andi      |
                   mc0REn    ) ? 1'b1 : 1'b0 ;

 assign regFWEnBuff = (ovIntReq) ? 1'b0 : regFWEn ;

delay1 PIPE_regFWEnIDEX (
   .clk   (clk         ), // input
   .rst_n (rst_n       ), // input

   .inp   (regFWEnBuff ), // input
   .outp  (regFWEnIDEX )  // output
   );

 assign regFWEnIDEXBuff = (ovIntReq) ? 1'b0 : regFWEnIDEX ;

delay1 PIPE_regFWEnEXWB (
   .clk   (clk             ), // input
   .rst_n (rst_n           ), // input

   .inp   (regFWEnIDEXBuff ), // input
   .outp  (regFWEnEXWB     )  // output
   );

endmodule

//
// alhamdulillah
//
```

Listing 5 branchPredict.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : branchPredict.v
-- TYPE      : rtl
-- FUNCTION  : branch prediction
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/11/11
-----------------------------------------------*/

module branchPredict (
   clk       , // input
   rst_n     , // input

   opCode    , // input
   funct     , // input

   // to controlUnit
   intFinish , // input
   beq       , // input
   brEn      , // input

   // to ifStage
   wrong     , // output
   pcSel     , // output
   flush       // output
   );

 // input ports
 input wire       clk       ;
 input wire       rst_n     ;

 input wire [5:0] opCode    ;
 input wire [5:0] funct     ;
 input wire       beq       ;
 input wire       brEn      ;

 input wire       intFinish ;

 // output ports
 output wire [1:0] pcSel     ;
 output wire       flush     ;
 output wire       wrong     ;

 // internal variables
 reg  [1:0] p_predict      ;
 wire [1:0] n_predict      ;

 wire       taken          ;
 wire       nottaken       ;

 wire       start_jump     ;
 wire       start_predict  ;
 wire [1:0] prediction     ;
 wire       nottaken_wrong ;
 wire       taken_wrong    ;

 // state machine
 assign pred_stat = p_predict ;

 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
   begin
    p_predict <= #`DELAY `NOTTAKEN_0;
   end
```

```
  else
   begin
    p_predict <= #`DELAY n_predict;
   end

 function [1:0] N_PRED  ;
  input [1:0] p_predict ;
  input       taken     ;
  input       nottaken  ;

  begin
   case (p_predict)
    `NOTTAKEN_0 : begin
                   if (taken)
                    N_PRED = `NOTTAKEN_1 ;
                   else if (nottaken)
                    N_PRED = `NOTTAKEN_0 ;
                   else
                    N_PRED = `NOTTAKEN_0 ;
                  end
    `NOTTAKEN_1 : begin
                   if (taken)
                    N_PRED = `TAKEN_1    ;
                   else if (nottaken)
                    N_PRED = `NOTTAKEN_0 ;
                   else
                    N_PRED = `NOTTAKEN_1 ;
                  end
    `TAKEN_0    : begin
                   if (taken)
                    N_PRED = `TAKEN_1    ;
                   else if (nottaken)
                    N_PRED = `NOTTAKEN_0 ;
                   else
                    N_PRED = `TAKEN_0    ;
                  end
    `TAKEN_1    : begin
                   if (taken)
                    N_PRED = `TAKEN_1 ;
                   else if (nottaken)
                    N_PRED = `TAKEN_0 ;
                   else
                    N_PRED = `TAKEN_1 ;
                  end
    default     : begin
                    N_PRED = `DONT_CARE_2 ;
                  end
   endcase
  end
 endfunction

 assign n_predict = N_PRED(p_predict, taken, nottaken);

 // flush enable
 assign flush = ((p_predict == `NOTTAKEN_0) & taken    ) |
                ((p_predict == `NOTTAKEN_1) & taken    ) |
                ((p_predict == `TAKEN_0   ) & nottaken ) |
                ((p_predict == `TAKEN_1   ) & nottaken ) |
                (intFinish)                              ? 1'b1 : 1'b0 ;

 // program counter selector
 assign taken          = (brEn)  & (beq) ? 1'b1 : 1'b0 ;
 assign nottaken       = (~brEn) & (beq) ? 1'b1 : 1'b0 ;

 assign prediction     = ((p_predict == `TAKEN_0) | (p_predict == `TAKEN_1)) ? 2'd1 :
2'd0 ;
```

```
 assign nottaken_wrong = (((p_predict == `NOTTAKEN_0) & taken) | ((p_predict ==
`NOTTAKEN_1) & taken)) ? 1'b1 : 1'b0 ;
 assign taken_wrong    = (((p_predict == `TAKEN_0) & nottaken) | ((p_predict ==
`TAKEN_1) & nottaken)) ? 1'b1 : 1'b0 ;
 assign wrong          = (taken_wrong | nottaken_wrong) ? 1'b1 : 1'b0 ;

 assign start_jump     = (opCode == `OP_J)   ? 1'b1 : 1'b0 ;
 assign start_predict  = (opCode == `OP_BEQ) ? 1'b1 : 1'b0 ;

 function [1:0] PC_SELECT    ;
  input       nottaken_wrong ;
  input       taken_wrong    ;
  input       start_predict  ;
  input       start_jump     ;
  input [1:0] prediction     ;

  begin
   if (nottaken_wrong)
    PC_SELECT = 2'd1 ;
   else if (taken_wrong)
    PC_SELECT = 2'd3 ;
   else if (start_predict)
    PC_SELECT = prediction ;
   else if (start_jump)
    PC_SELECT = 2'd2 ;
   else
    PC_SELECT = 2'd0 ;
  end
 endfunction

 assign pcSel = PC_SELECT(nottaken_wrong, taken_wrong, start_predict, start_jump,
prediction);


endmodule

//
// alhamdulillah
//
```

Listing 6 hazardCtrl.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : hazardCtrl.v
-- TYPE      : rtl
-- FUNCTION  : hazard detector
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/18
-----------------------------------------------*/

module hazardCtrl (
   clk            , // input
   rst_n          , // input

   // to controlUnit
   dMemWEnID      , // input

   // to idStage
   rs             , // input
   rt             , // input
   fwdMem         , // output
   fwdA           , // output
   fwdB           , // output
```

```
   // to exStage
   regFWAddrIDEX  , // input
   regFWEnIDEX    , // input
   dMemREnIDEX      // input
   );

 // input ports
 input wire       clk           ;
 input wire       rst_n         ;

 input wire       dMemWEnID     ;

 input wire [4:0] rs            ;
 input wire [4:0] rt            ;
 input wire [4:0] regFWAddrIDEX ;
 input wire       regFWEnIDEX   ;
 input wire       dMemREnIDEX   ;

 // output ports
 output wire      fwdMem        ;
 output wire      fwdA          ;
 output wire      fwdB          ;

 // forwarding control signal
 assign fwdA    = (regFWEnIDEX) & (rs == regFWAddrIDEX) & ~(regFWAddrIDEX == 5'd0) ?
1'd1 : 1'd0 ;
 assign fwdB    = (regFWEnIDEX) & (rt == regFWAddrIDEX) & ~(regFWAddrIDEX == 5'd0) ?
1'd1 : 1'd0 ;
 assign fwdMem  = (rt == regFWAddrIDEX) & (dMemWEnID) & (dMemREnIDEX) ? 1'b1 : 1'b0 ;

endmodule

//
// alhamdulillah
//
```

Listing 7 delay1.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : delay1.v
-- TYPE      : rtl
-- FUNCTION  : 1-bit delay unit
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/11/05
-----------------------------------------------*/

module delay1 (
   clk   , // input
   rst_n , // input

   inp   , // input
   outp    // output
   );

 // input ports
 input wire clk   ; // input
 input wire rst_n ; // input

 input wire inp   ; // input

 // output ports
 output reg outp  ; // input
```

```
 // behavior
 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
   begin
    outp <= #`DELAY 1'd0 ;
   end
  else
   begin
    outp <= #`DELAY inp ;
   end

endmodule

//
// alhamdulillah
//
```

Listing 8 delay3.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : delay3.v
-- TYPE      : rtl
-- FUNCTION  : 3-bit delay unit
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/11/05
-----------------------------------------------*/

module delay3 (
   clk   , // input
   rst_n , // input

   inp   , // input
   outp    // output
   );

 // input ports
 input wire       clk   ; // input
 input wire       rst_n ; // input

 input wire [2:0] inp   ; // input

 // output ports
 output reg [2:0] outp  ; // input

 // behavior
 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
   begin
    outp <= #`DELAY 3'd0 ;
   end
  else
   begin
    outp <= #`DELAY inp ;
   end

endmodule

//
// alhamdulillah
//
```

Listing 9 delay5.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : delay5.v
-- TYPE      : rtl
-- FUNCTION  : 5-bit delay unit
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/11/05
-----------------------------------------------*/

module delay5 (
   clk   , // input
   rst_n , // input

   inp   , // input
   outp    // output
   );

 // input ports
 input wire       clk   ; // input
 input wire       rst_n ; // input

 input wire [4:0] inp   ; // input

 // output ports
 output reg [4:0] outp  ; // input

 // behavior
 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
   begin
    outp <= #`DELAY 5'd0 ;
   end
  else
   begin
    outp <= #`DELAY inp ;
   end

endmodule

//
// alhamdulillah
//
```

Listing 10 delay32.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : delay32.v
-- TYPE      : rtl
-- FUNCTION  : 32-bit delay unit
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/11/05
-----------------------------------------------*/

module delay32 (
   clk   , // input
   rst_n , // input

   inp   , // input
   outp    // output
   );
```

```
 // input ports
 input wire        clk   ; // input
 input wire        rst_n ; // input

 input wire [31:0] inp   ; // input

 // output ports
 output reg [31:0] outp  ; // input

 // behavior
 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
   begin
    outp <= #`DELAY 32'd0 ;
   end
  else
   begin
    outp <= #`DELAY inp ;
   end

endmodule

//
// alhamdulillah
//
```

Listing 11 ifStage.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : ifStage.v
-- TYPE      : rtl
-- FUNCTION  : instruction fetch stage
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/15
-----------------------------------------------*/

module ifStage (
   clk             , // input
   rst_n           , // input

   ovIntReq        , // input
   extIntReq       , // input

   intFinish       , // input
   intRet          , // input

   pcIF            , // output

   // to branchPredict
   wrong           , // input
   pcSel           , // input
   flush           , // input

   // to idStage
   branchTAddrIDIF , // input
   pcPlus4IDIF     , // input
   pcID            , // output
   branchTAddrIFID , // output
   pcPlus4IFID     , // output
   inst            , // output

   // to instMemory
```

```
  instIn          , // input
  instAddr          // output
  );

// input ports
input wire        clk             ;
input wire        rst_n           ;

input wire        flush           ;

input wire        wrong           ;
input wire [1:0]  pcSel           ;
input wire [31:0] instIn          ;

input wire [31:0] branchTAddrIDIF  ;
input wire [31:0] pcPlus4IDIF      ;

input wire        ovIntReq         ;
input wire        extIntReq        ;

input wire        intFinish        ;
input wire [31:0] intRet           ;

// output ports
output wire [31:0] branchTAddrIFID ;
output wire [31:0] pcPlus4IFID     ;
output wire [31:0] inst            ;
output wire [31:0] instAddr        ;

output wire [31:0] pcIF            ;
output wire [31:0] pcID            ;

// internal variables
reg  [31:0] p_pc            ;
wire [31:0] n_pc            ;
wire [31:0] instBuff        ;
wire [31:0] pcPlus4IFIDBuff ;
wire [31:0] pcIFBuff        ;

wire [15:0] imm             ;
wire [31:0] signExtImm      ;

wire [31:0] pcPlus4         ;

wire [31:0] iman_ganteng    ;

wire [25:0] addr            ;
wire [31:0] jumpTAddr       ;
wire [31:0] branchTAddr     ;

// program counter
always @ (posedge clk or negedge rst_n)
 if (~rst_n)
  begin
   p_pc <= #`DELAY `TEXT_VECTOR;
  end
 else
  begin
   p_pc <= #`DELAY n_pc;
  end

function [31:0] NEXT_PC   ;
 input [1:0]  pcSel       ;
 input [31:0] pcPlus4     ;
 input [31:0] branchTAddr ;
 input [31:0] jumpTAddr   ;
 input [31:0] pcPlus4IDIF ;
```

```
  begin
   case (pcSel)
    2'd0    : NEXT_PC = pcPlus4       ;
    2'd1    : NEXT_PC = branchTAddr   ;
    2'd2    : NEXT_PC = jumpTAddr     ;
    2'd3    : NEXT_PC = pcPlus4IDIF   ;
    default : NEXT_PC = `DONT_CARE_32 ;
   endcase
  end
 endfunction

 assign n_pc = (ovIntReq | extIntReq) ? `INT_VECTOR :
               (intFinish)            ? intRet  :
               NEXT_PC(pcSel, pcPlus4, branchTAddr, jumpTAddr, pcPlus4IDIF);

 // pc at IF
 assign pcIF = p_pc ;

 // pc+4 calculation
 assign pcPlus4 = p_pc + 32'd4 ;

 // jump target address calculation
 assign addr      = instIn[25:0]                ;
 assign jumpTAddr = {pcPlus4[31:28], addr, 2'd0} ;

 // branch target address calculation
 assign imm          = instIn[15:0] ;
 assign signExtImm   = {{16{imm[15]}},imm} ;
 assign iman_ganteng = {signExtImm[29:0], 2'd0} + pcPlus4 ;
 assign branchTAddr  = (wrong) ? branchTAddrIDIF : iman_ganteng ;

 // instruction address
 assign instAddr = p_pc ;

 // pcIF Buffer
 assign pcIFBuff        = (flush | ovIntReq) ? 32'd0 : pcIF ;

 // pcPlus4 Buffer
 assign pcPlus4IFIDBuff = (flush | ovIntReq) ? 32'd0 : pcPlus4 ;

 // instruction Buffer
 assign instBuff        = (flush | ovIntReq) ? 32'd0 : instIn  ;

 // pipeline register to idStage
delay32 PIPE_pcID (
   .clk   (clk             ), // input
   .rst_n (rst_n           ), // input

   .inp   (pcIFBuff        ), // input
   .outp  (pcID            )  // output
   );

delay32 PIPE_branchTAddrIFID (
   .clk   (clk             ), // input
   .rst_n (rst_n           ), // input

   .inp   (branchTAddr     ), // input
   .outp  (branchTAddrIFID )  // output
   );

delay32 PIPE_pcPlus4IFID (
   .clk   (clk             ), // input
   .rst_n (rst_n           ), // input

   .inp   (pcPlus4IFIDBuff ), // input
   .outp  (pcPlus4IFID     )  // output
```

```
   );

delay32 PIPE_inst (
   .clk   (clk      ), // input
   .rst_n (rst_n    ), // input

   .inp   (instBuff ), // input
   .outp  (inst     )  // output
   );

endmodule

//
// alhamdulillah
//
```

Listing 12 idStage.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : idStage.v
-- TYPE      : rtl
-- FUNCTION  : instruction decode stage
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/15
-----------------------------------------------*/

module idStage (
   clk             , // input
   rst_n           , // input

   // instruction decode
   opCode          , // output
   funct           , // output
   shamt           , // output
   rs              , // output
   rt              , // output
   rd              , // output

   // for forwarding
   fwdAluOutEXID   , // input

   // to co-proc
   mc0WData        , // output

   // to hazardCtrl
   fwdA            , // input
   fwdB            , // input
   fwdMem          , // input

   // to controlUnit
   regFDst         , // input
   aluSrc          , // input
   extOp           , // input
   takeEn          , // output

   // to ifStage
   pcID            , // input
   branchTAddrIFID , // input
   pcPlus4IFID     , // input
   inst            , // input
   branchTAddrIDIF , // output
   pcPlus4IDIF     , // output
```

```
  // to exStage
  pcEX            , // output
  pcPlus4IDEX     , // output
  regFWAddrIDEX   , // output
  aluInA          , // output
  aluInB          , // output
  dMemWDataIDEX   , // output

  // to wbStage
  regFWEnWBID     , // input
  regFWAddrWBID   , // input
  regFWData         // input
  );

// input ports
input wire        clk             ;
input wire        rst_n           ;

input wire [31:0] branchTAddrIFID ;
input wire [31:0] pcPlus4IFID     ;
input wire [31:0] inst            ;

input wire        regFWEnWBID     ;
input wire [4:0]  regFWAddrWBID   ;
input wire [31:0] regFWData       ;

input wire        fwdA            ;
input wire        fwdB            ;
input wire [31:0] fwdAluOutEXID   ;

input wire        regFDst         ;
input wire        aluSrc          ;
input wire [1:0]  extOp           ;

input wire        fwdMem          ;

input wire [31:0] pcID            ;

// output ports
output wire [31:0] pcPlus4IDIF    ;
output wire [31:0] branchTAddrIDIF ;

output wire [31:0] aluInA         ;
output wire [31:0] aluInB         ;
output wire [4:0]  regFWAddrIDEX  ;
output wire [31:0] dMemWDataIDEX  ;
output wire [31:0] pcPlus4IDEX    ;
output wire [31:0] pcEX           ;

output wire        takeEn         ;

output wire [5:0]  opCode         ;
output wire [5:0]  funct          ;
output wire [4:0]  shamt          ;
output wire [4:0]  rs             ;
output wire [4:0]  rt             ;
output wire [4:0]  rd             ;

output wire [31:0] mc0WData       ;

// internal variables
wire [15:0] imm               ;
wire [31:0] regRS             ;
wire [31:0] regRT             ;
wire [31:0] regRSBuff         ;
wire [31:0] regRTBuff         ;
```

```
 reg  [31:0] signExtImm        ;

 wire [4:0]  regFWAddrIDEXBuff ;
 wire [31:0] aluInBBuff        ;
 wire [31:0] dMemWDataIDEXBuff ;

 assign opCode = inst[31:26] ;
 assign rs     = inst[25:21] ;
 assign rt     = inst[20:16] ;
 assign rd     = inst[15:11] ;
 assign shamt  = inst[10:6]  ;
 assign funct  = inst[5:0]   ;

 assign imm    = inst[15:0]  ;

 // extension
 always @ (extOp or imm)
  case (extOp)
   `SIGN   : signExtImm = {{16{imm[15]}},imm} ;
   `ZERO   : signExtImm = {{16{1'b0}},imm}    ;
   `LUI    : signExtImm = {imm,{16{1'b0}}}    ;
   default : signExtImm = {{16{imm[15]}},imm} ;
  endcase

 // pcPlus4
 assign pcPlus4IDIF  = pcPlus4IFID ;

 // take branch control signal
 assign takeEn       = (regRSBuff == regRTBuff) ? 1'b1 : 1'b0 ;

 // branch target address
 assign branchTAddrIDIF = branchTAddrIFID ;

 // aluInA
 assign regRSBuff  = (fwdA)   ? fwdAluOutEXID : regRS     ;

 // aluInB
 assign regRTBuff  = (fwdB)   ? fwdAluOutEXID : regRT     ;
 assign aluInBBuff = (aluSrc) ? signExtImm    : regRTBuff ;

 // dMemory writee data
 assign dMemWDataIDEXBuff = (fwdMem) ? fwdAluOutEXID : regRTBuff ;

 // mc0 write data
 assign mc0WData = regRTBuff ;

 // register file write address
 assign regFWAddrIDEXBuff = (regFDst == 1'd1) ? rd : rt ;

 // register file
regFile regFile (
   .clk       (clk           ), // input
   .rst_n     (rst_n         ), // input
   .rs        (rs            ), // input
   .rt        (rt            ), // input
   .regFWEn   (regFWEnWBID   ), // input
   .regFWData (regFWData     ), // input
   .regFWAddr (regFWAddrWBID ), // input
   .regRS     (regRS         ), // output
   .regRT     (regRT         )  // output
   );

 // pipeline register to exStages
delay32 PIPE_pcEX (
   .clk   (clk             ), // input
   .rst_n (rst_n           ), // input
```

```
   .inp   (pcID            ), // input
   .outp  (pcEX            )  // output
   );

delay32 PIPE_pcPlus4IDEX (
   .clk   (clk             ), // input
   .rst_n (rst_n           ), // input

   .inp   (pcPlus4IFID     ), // input
   .outp  (pcPlus4IDEX     )  // output
   );

delay32 PIPE_aluInA (
   .clk   (clk       ), // input
   .rst_n (rst_n     ), // input

   .inp   (regRSBuff ), // input
   .outp  (aluInA    )  // output
   );

delay32 PIPE_aluInB (
   .clk   (clk        ), // input
   .rst_n (rst_n      ), // input

   .inp   (aluInBBuff ), // input
   .outp  (aluInB     )  // output
   );

delay32 PIPE_dMemWDataIDEX (
   .clk   (clk               ), // input
   .rst_n (rst_n             ), // input

   .inp   (dMemWDataIDEXBuff ), // input
   .outp  (dMemWDataIDEX     )  // output
   );

delay5 PIPE_regFWAddrIDEX (
   .clk   (clk               ), // input
   .rst_n (rst_n             ), // input

   .inp   (regFWAddrIDEXBuff ), // input
   .outp  (regFWAddrIDEX     )  // output
   );

endmodule

//
// alhamdulillah
//
```

Listing 13 exStage.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : exStage.v
-- TYPE      : rtl
-- FUNCTION  : execute stage
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/15
-----------------------------------------------*/

module exStage (
   clk            , // input
   rst_n          , // input
```

```
  overflow       , // output

  // for forwading
  fwdAluOutEXID  , // output

  // to controlUnit
  aluOpIDEX      , // input
  setLessEnIDEX  , // input
  dMemWEnIDEX    , // input
  dMemREnIDEX    , // input
  shamtIDEX      , // input
  ovEnIDEX       , // input
  mc0REnIDEX     , // input

  // to co-proc
  mc0RData       , // input

  // to idStage
  regFWAddrIDEX  , // input
  aluInA         , // input
  aluInB         , // input
  dMemWDataIDEX  , // input

  // to dMemory
  dMemRData      , // input
  dMemWEn        , // output
  dMemWData      , // output
  dMemWRAddr     , // output

  // to wbStage
  aluOut         , // output
  regFWAddrEXWB    // output
  );

// input ports
input wire        clk            ;
input wire        rst_n          ;

input wire [4:0]  regFWAddrIDEX  ;
input wire        setLessEnIDEX  ;
input wire [2:0]  aluOpIDEX      ;
input wire        dMemWEnIDEX    ;
input wire        dMemREnIDEX    ;
input wire [4:0]  shamtIDEX      ;
input wire        ovEnIDEX       ;
input wire        mc0REnIDEX     ;

input wire [31:0] aluInA         ;
input wire [31:0] aluInB         ;
input wire [31:0] dMemWDataIDEX  ;

input wire [31:0] dMemRData      ;

input wire [31:0] mc0RData       ;

// output ports
output wire [31:0] fwdAluOutEXID ;

output wire [4:0]  regFWAddrEXWB ;
output wire [31:0] aluOut        ;

output wire        dMemWEn       ;
output wire [31:0] dMemWData     ;
output wire [6:0]  dMemWRAddr    ;

output wire        overflow      ;
```

```
 // internal variables
 wire [31:0] aluOutBuff     ;
 wire [31:0] dMemWRAddrBuff ;
 wire [31:0] regFWDataBuff  ;
 reg  [31:0] dMemRDataBuff  ;

 // alu Unit
aluUnit aluUnit (
   .aluInA    (aluInA         ), // input
   .aluInB    (aluInB         ), // input
   .setLessEn (setLessEnIDEX  ), // input
   .aluOp     (aluOpIDEX      ), // input
   .shamt     (shamtIDEX      ), // input
   .aluOut    (aluOutBuff     ), // output
   .overflow  (overflowBuff   )  // output
   );

 // overflow
 assign overflow = (ovEnIDEX) ? overflowBuff : 1'b0 ;

 // data memory control and data signal
 assign dMemWEn   = dMemWEnIDEX   ;
 assign dMemWData = dMemWDataIDEX ;

 // data memory address calculation
 assign dMemWRAddr = dMemWRAddrBuff[8:2] ;

ksAdder ksAdder (
   .x32 (aluInA         ), // input
   .y32 (aluInB         ), // input
   .cin  (1'b0           ), // input
   .s32 (dMemWRAddrBuff )  // output
   );

 // register file write data selector
 assign regFWDataBuff = (dMemREnIDEX) ? dMemRData :
                        (mc0REnIDEX)  ? mc0RData  : aluOutBuff ;

 // data for forwarding
 assign fwdAluOutEXID = regFWDataBuff ;

 // pipeline register to wbStage
delay5 PIPE_regFWAddrEXWB (
   .clk   (clk           ), // input
   .rst_n (rst_n         ), // input

   .inp   (regFWAddrIDEX ), // input
   .outp  (regFWAddrEXWB )  // output
   );

delay32 PIPE_aluOut (
   .clk   (clk           ), // input
   .rst_n (rst_n         ), // input

   .inp   (regFWDataBuff ), // input
   .outp  (aluOut        )  // output
   );

endmodule

//
// alhamdulillah
//
```

Listing 14 wbStage.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : wbStage.v
-- TYPE      : rtl
-- FUNCTION  : write back stage
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/15
-----------------------------------------------*/

module wbStage (

   // to controlUnit
   regFWEnEXWB    , // input

   // to exStage
   aluOut         , // input
   regFWAddrEXWB  , // input

   // to idStage
   regFWEnWBID    , // output
   regFWAddrWBID  , // output
   regFWData        // output
   );

 // input ports
 input wire [31:0] aluOut         ;
 input wire [4:0]  regFWAddrEXWB  ;

 input wire        regFWEnEXWB    ;

 // output ports
 output wire        regFWEnWBID   ;
 output wire [4:0]  regFWAddrWBID ;
 output wire [31:0] regFWData     ;

 // output behavior
 assign regFWEnWBID   = regFWEnEXWB   ;
 assign regFWAddrWBID = regFWAddrEXWB ;
 assign regFWData     = aluOut        ;

endmodule

//
// alhamdulillah
//
```

Listing 15 coSrp.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : coSrp.v
-- TYPE      : rtl
-- FUNCTION  : co-processor
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/12/18
-----------------------------------------------*/

module coSrp (
   clk       , // input
   rst_n     , // input
```

```
  // external interrupt ports
  ext_n_0   , // input
  ext_n_1   , // input
  ext_n_2   , // input
  ext_n_3   , // input

  // to srp
  mc0WEn    , // input
  mc0WAddr  , // input
  mc0WData  , // input

  mc0RAddr  , // input
  mc0RData  , // output

  pcIF      , // input
  pcEX      , // input

  overflow  , // input
  ovIntReq  , // output
  extIntReq , // output
  intRet      // output
  );

// input ports
input wire        clk       ;
input wire        rst_n     ;

input wire        ext_n_0   ;
input wire        ext_n_1   ;
input wire        ext_n_2   ;
input wire        ext_n_3   ;

input wire        mc0WEn    ;
input wire [4:0]  mc0WAddr  ;
input wire [31:0] mc0WData  ;

input wire [4:0]  mc0RAddr  ;

input wire [31:0] pcIF      ;
input wire [31:0] pcEX      ;

input wire        overflow  ;

// output
output wire        ovIntReq  ;
output wire        extIntReq ;
output wire [31:0] intRet    ;
output wire [31:0] mc0RData  ;

// internal variables
wire        statusWEn       ;
wire [31:0] statusWData     ;
wire        causeWEn        ;
wire [31:0] causeWData      ;
wire        epcWEn          ;
wire [31:0] epcWData        ;

wire [31:0] mc0RDataBuff    ;
wire        ovInt           ;

reg  [31:0] status          ; // 12
reg  [31:0] cause           ; // 13
reg  [31:0] epc             ; // 14

// mc0 read data
assign mc0RDataBuff = (mc0RAddr == `EPC)   ? epc    :
                      (mc0RAddr == `CAUSE) ? cause  :
```

```
                        (mc0RAddr == `STAT)  ? status : 32'd0 ;

delay32 PIPE_mc0RData (
   .clk   (clk          ), // input
   .rst_n (rst_n        ), // input

   .inp   (mc0RDataBuff ), // input
   .outp  (mc0RData     )  // output
   );

 // interrupt enable
 assign extIntEn0 = (status[8]  & ~(status[1]) & (status[0])) ? 1'b1 : 1'b0 ;
 assign extIntEn1 = (status[9]  & ~(status[1]) & (status[0])) ? 1'b1 : 1'b0 ;
 assign extIntEn2 = (status[10] & ~(status[1]) & (status[0])) ? 1'b1 : 1'b0 ;
 assign extIntEn3 = (status[11] & ~(status[1]) & (status[0])) ? 1'b1 : 1'b0 ;
 assign ovIntEn   = (status[12] & ~(status[1]) & (status[0])) ? 1'b1 : 1'b0 ;

 // interrupt signal
 assign extInt0   = (~ext_n_0 &  extIntEn0) ? 1'b1 : 1'b0 ;
 assign extInt1   = (~ext_n_1 &  extIntEn1) ? 1'b1 : 1'b0 ;
 assign extInt2   = (~ext_n_2 &  extIntEn2) ? 1'b1 : 1'b0 ;
 assign extInt3   = (~ext_n_3 &  extIntEn3) ? 1'b1 : 1'b0 ;
 assign ovInt     = (overflow &  ovIntEn)   ? 1'b1 : 1'b0 ;

 // interrupt request signal
 assign ovIntReq  = (ovInt) ? 1'b1 : 1'b0 ;
 assign extIntReq = (extInt0|
                     extInt1|
                     extInt2|
                     extInt3) ? 1'b1 : 1'b0 ;

 // interrupt return address
 assign intRet   = epc ;

 //
 // STATUS register inteface
 //

 // -----------------------------------------
 // interrupt mask = status[15:8]
 // bit | type
 // ----+------
 // 8   | external interrupt 0
 // 9   | external interrupt 1
 // 10  | external interrupt 2
 // 11  | external interrupt 3
 // 12  | overflow interrupt
 // 13  | n/a
 // 14  | n/a
 // 15  | n/a
 // -----------------------------------------

 // register status write enable
 assign statusWEn = (ovIntReq |
                     extIntReq|
                    (mc0WEn & (mc0WAddr == `STAT))) ? 1'b1 : 1'b0 ;

 // register status write data
 assign statusWData = (ovIntReq | extIntReq)         ? ({status[31:2],2'b10}) :
                      (mc0WEn & (mc0WAddr == `STAT)) ? mc0WData         : 32'd0 ; //
written by instruction

 // register status
 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
   status <= #`DELAY 32'd0 ; // status
  else if (statusWEn)
```

```
   status <= #`DELAY statusWData ; // status

 //
 // CAUSE register inteface
 //

 // -----------------------------------------
 // exception code = status[6:2]
 // bit | type
 // ----+------
 // 1   | external interrupt 0
 // 2   | external interrupt 1
 // 3   | external interrupt 2
 // 4   | external interrupt 4
 // 12  | overflow interrupt
 // -----------------------------------------

 // register cause write enable
 assign causeWEn = (ovInt   |
                    extInt0 |
                    extInt1 |
                    extInt2 |
                    extInt3 |
                   (mc0WEn & (mc0WAddr == `CAUSE)))? 1'b1 : 1'b0 ;

 // register cause write data
 assign causeWData = (extInt0)                        ? (32'h00000004 | cause) : //
exception code = 5'b00001
                     (extInt1)                        ? (32'h00000008 | cause) : //
exception code = 5'b00010
                     (extInt2)                        ? (32'h0000000c | cause) : //
exception code = 5'b00011
                     (extInt3)                        ? (32'h00000010 | cause) : //
exception code = 5'b00100
                     (ovInt)                          ? (32'h00000030 | cause) : //
exception code = 5'b01100
                     (mc0WEn & (mc0WAddr == `CAUSE)) ? mc0WData               : 32'b0 ;
// written by instruction

 // register cause
 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
   cause <= #`DELAY 32'd0 ; // cause
  else if (causeWEn)
   cause <= #`DELAY causeWData ; // cause

 //
 // EPC register inteface
 //

 // register epc write enable
 assign epcWEn = (ovInt   |
                  extInt0 |
                  extInt1 |
                  extInt2 |
                  extInt3 |
                 (mc0WEn & (mc0WAddr == `EPC))) ? 1'b1 : 1'b0 ;

 // register epc write data
 assign epcWData  = (ovInt)                                  ? pcEX     :
                    (extInt0 | extInt1 | extInt2 | extInt3) ? pcIF     :
                    (mc0WEn & (mc0WAddr == `EPC) )          ? mc0WData : 32'd0 ; //
written by instruction

 // register epc
 always @ (posedge clk or negedge rst_n)
  if (~rst_n)
```

```
   epc <= #`DELAY 32'd0 ; // epc
  else if (epcWEn)
   epc <= #`DELAY epcWData ; // epc

endmodule

//
// alhamdulillah
//
```

Listing 16 srp.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : srp.v
-- TYPE      : rtl
-- FUNCTION  : simple risc processor top level
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/16
-----------------------------------------------*/

module srp (
   Clock       , // input
   Reset       , // input

   // external interrupt ports
   ext_n_0   , // input
   ext_n_1   , // input
   ext_n_2   , // input
   ext_n_3   , // input

   // to instruction memory
   Inst        , // input
   Iadd        , // output

   // to data memory
   Dadd        , // input
   WE          , // output
   Wtdata      , // output
   Rddata        // output
   );

 // input ports
 input wire        Clock       ;
 input wire        Reset       ;

 input wire [31:0] Inst        ;
 input wire [31:0] Rddata      ;

 input wire        ext_n_0     ;
 input wire        ext_n_1     ;
 input wire        ext_n_2     ;
 input wire        ext_n_3     ;

 // output ports
 output wire [31:0] Iadd       ;
 output wire        WE         ;
 output wire [6:0]  Dadd       ;
 output wire [31:0] Wtdata     ;

 // internal variables

 wire        ovIntReq          ;
 wire        extIntReq         ;
```

```
wire        overflow          ;
wire [31:0] intRet            ;

wire        mc0WEn            ;
wire [4:0]  mc0WAddr          ;
wire [31:0] mc0WData          ;
wire [31:0] mc0RData          ;
wire [4:0]  mc0RAddr          ;

wire [31:0] pcIF              ;
wire [31:0] pcID              ;
wire [31:0] pcEX              ;

wire [31:0] instIn            ;
wire [31:0] dMemRData         ;

wire [31:0] instAddr          ;
wire        dMemWEn           ;
wire [6:0]  dMemWRAddr        ;
wire [31:0] dMemWData         ;

wire [1:0]  pcSel             ;
wire [31:0] branchTAddrIFID   ;
wire [31:0] branchTAddrIDIF   ;
wire [31:0] inst              ;

wire [31:0] pcPlus4IFID       ;
wire [31:0] pcPlus4IDIF       ;
wire [31:0] pcPlus4IDEX       ;

wire [4:0]  regFWAddrIDEX     ;
wire [2:0]  aluOpIDEX         ;
wire [31:0] aluInA            ;
wire [31:0] aluInB            ;
wire [31:0] dMemWDataIDEX     ;

wire [31:0] aluOut            ;

wire [4:0]  regFWAddrEXWB     ;

wire [4:0]  regFWAddrWBID     ;
wire [31:0] regFWData         ;

wire [31:0] fwdAluOutEXID     ;

wire [5:0]  opCode            ;
wire [5:0]  funct             ;
wire [4:0]  shamt             ;
wire [4:0]  rs                ;
wire [4:0]  rt                ;
wire [4:0]  rd                ;

wire [1:0]  extOp             ;
wire [4:0]  shamtIDEX         ;

// I/O port assignments
assign clk       = Clock      ;
assign rst_n     = Reset      ;

assign instIn    = Inst       ;
assign dMemRData = Rddata     ;

assign Iadd      = instAddr   ;
assign WE        = dMemWEn    ;
assign Dadd      = dMemWRAddr ;
assign Wtdata    = dMemWData  ;
```

```
 assign pcPlus4ID = pcPlus4IFID ;
 assign pcPlus4EX = pcPlus4IDEX ;

 assign mc0WAddr  = rd          ;
 assign mc0RAddr  = rd          ;

ifStage ifStage (
   .clk             (clk             ), // input
   .rst_n           (rst_n           ), // input

   .ovIntReq        (ovIntReq        ), // input
   .extIntReq       (extIntReq       ), // input

   .intFinish       (intFinish       ), // input
   .intRet          (intRet          ), // input

   .pcIF            (pcIF            ), // output

   // to branchPredict
   .wrong           (wrong           ), // input
   .pcSel           (pcSel           ), // input
   .flush           (flush           ), // input

   // to idStage
   .branchTAddrIDIF (branchTAddrIDIF ), // input
   .pcPlus4IDIF     (pcPlus4IDIF     ), // input
   .pcID            (pcID            ), // output
   .branchTAddrIFID (branchTAddrIFID ), // output
   .pcPlus4IFID     (pcPlus4IFID     ), // output
   .inst            (inst            ), // output

   // to instMemory
   .instIn          (instIn          ), // input
   .instAddr        (instAddr        )  // output
   );

idStage idStage (
   .clk             (clk             ), // input
   .rst_n           (rst_n           ), // input

   // instruction decode
   .opCode          (opCode          ), // output
   .funct           (funct           ), // output
   .shamt           (shamt           ), // output
   .rs              (rs              ), // output
   .rt              (rt              ), // output
   .rd              (rd              ), // output

   // for forwarding
   .fwdAluOutEXID   (fwdAluOutEXID   ), // input

   // to co-proc
   .mc0WData        (mc0WData        ), // output

   // to hazardCtrl
   .fwdA            (fwdA            ), // input
   .fwdB            (fwdB            ), // input
   .fwdMem          (fwdMem          ), // input

   // to controlUnit
   .regFDst         (regFDst         ), // input
   .aluSrc          (aluSrc          ), // input
   .extOp           (extOp           ), // input
   .takeEn          (takeEn          ), // output

   // to ifStage
   .pcID            (pcID            ), // input
```

```
    .branchTAddrIFID (branchTAddrIFID ), // input
    .pcPlus4IFID     (pcPlus4IFID     ), // input
    .inst            (inst            ), // input
    .pcPlus4IDIF     (pcPlus4IDIF     ), // output
    .branchTAddrIDIF (branchTAddrIDIF ), // output

    // to exStage
    .pcEX            (pcEX            ), // output
    .pcPlus4IDEX     (pcPlus4IDEX     ), // output
    .regFWAddrIDEX   (regFWAddrIDEX   ), // output
    .aluInA          (aluInA          ), // output
    .aluInB          (aluInB          ), // output
    .dMemWDataIDEX   (dMemWDataIDEX   ), // output

    // to wbStage
    .regFWEnWBID     (regFWEnWBID     ), // input
    .regFWAddrWBID   (regFWAddrWBID   ), // input
    .regFWData       (regFWData       )  // input
    );

exStage exStage (
    .clk             (clk           ), // input
    .rst_n           (rst_n         ), // input

    .overflow        (overflow      ), // output

    // for forwading
    .fwdAluOutEXID   (fwdAluOutEXID ), // output

    // to controlUnit
    .aluOpIDEX       (aluOpIDEX     ), // input
    .setLessEnIDEX   (setLessEnIDEX ), // input
    .dMemWEnIDEX     (dMemWEnIDEX   ), // input
    .dMemREnIDEX     (dMemREnIDEX   ), // input
    .shamtIDEX       (shamtIDEX     ), // input
    .ovEnIDEX        (ovEnIDEX      ), // output
    .mc0REnIDEX      (mc0REnIDEX    ), // input

    // to co-proc
    .mc0RData        (mc0RData      ), // input

    // to idStage
    .regFWAddrIDEX   (regFWAddrIDEX ), // input
    .aluInA          (aluInA        ), // input
    .aluInB          (aluInB        ), // input
    .dMemWDataIDEX   (dMemWDataIDEX ), // input

    // to dMemory
    .dMemRData       (dMemRData     ), // input
    .dMemWEn         (dMemWEn       ), // output
    .dMemWData       (dMemWData     ), // output
    .dMemWRAddr      (dMemWRAddr    ), // output

    // to wbStage
    .aluOut          (aluOut        ), // output
    .regFWAddrEXWB   (regFWAddrEXWB )  // output
    );

wbStage wbStage (

    // to controlUnit
    .regFWEnEXWB     (regFWEnEXWB   ), // input

    // to exStage
    .aluOut          (aluOut        ), // input
    .regFWAddrEXWB   (regFWAddrEXWB ), // input
```

```
   // to idStage
   .regFWEnWBID     (regFWEnWBID   ), // output
   .regFWAddrWBID   (regFWAddrWBID ), // output
   .regFWData       (regFWData     )  // output
   );

controlUnit controlUnit (
   .clk             (clk           ), // input
   .rst_n           (rst_n         ), // input

   .ovIntReq        (ovIntReq      ), // input

   .opCode          (opCode        ), // input
   .rs              (rs            ), // input
   .funct           (funct         ), // input
   .shamt           (shamt         ), // input

   // to co-proc
   .mc0WEn          (mc0WEn        ), // output

   // to hazardCtrl
   .dMemWEnID       (dMemWEnID     ), // output

   // to idStage
   .takeEn          (takeEn        ), // input
   .regFDst         (regFDst       ), // output
   .aluSrc          (aluSrc        ), // output
   .extOp           (extOp         ), // output

   // to exStage
   .aluOpIDEX       (aluOpIDEX     ), // output
   .setLessEnIDEX   (setLessEnIDEX ), // output
   .dMemWEnIDEX     (dMemWEnIDEX   ), // output
   .regFWEnIDEX     (regFWEnIDEX   ), // output
   .dMemREnIDEX     (dMemREnIDEX   ), // output
   .shamtIDEX       (shamtIDEX     ), // output
   .ovEnIDEX        (ovEnIDEX      ), // output
   .mc0REnIDEX      (mc0REnIDEX    ), // output

   // to wbStage
   .regFWEnEXWB     (regFWEnEXWB   ), // output

   // to brancPredict
   .intFinish       (intFinish     ), // output
   .beq             (beq           ), // output
   .brEn            (brEn          )  // output
   );

branchPredict branchPredict (
   .clk             (clk           ), // input
   .rst_n           (rst_n         ), // input

   .opCode          (instIn[31:26] ), // input
   .funct           (instIn[5:0]   ), // input

   // to controlUnit
   .intFinish       (intFinish     ), // input
   .beq             (beq           ), // input
   .brEn            (brEn          ), // input

   // to ifStage
   .wrong           (wrong         ), // output
   .pcSel           (pcSel         ), // output
   .flush           (flush         )  // output
   );

hazardCtrl hazardCtrl (
```

```
   .clk             (clk           ), // input
   .rst_n           (rst_n         ), // input

   // to controlUnit
   .dMemWEnID       (dMemWEnID     ), // input

   // to idStage
   .rs              (rs            ), // input
   .rt              (rt            ), // input
   .fwdMem          (fwdMem        ), // output
   .fwdA            (fwdA          ), // output
   .fwdB            (fwdB          ), // output

   // to exStage
   .regFWAddrIDEX   (regFWAddrIDEX ), // input
   .regFWEnIDEX     (regFWEnIDEX   ), // input
   .dMemREnIDEX     (dMemREnIDEX   )  // input
   );

coSrp coSrp (
   .clk             (clk           ), // input
   .rst_n           (rst_n         ), // input

   // external interrupt ports
   .ext_n_0         (ext_n_0       ), // input
   .ext_n_1         (ext_n_1       ), // input
   .ext_n_2         (ext_n_2       ), // input
   .ext_n_3         (ext_n_3       ), // input

   // to srp
   .mc0WEn          (mc0WEn        ), // input
   .mc0WAddr        (mc0WAddr      ), // input
   .mc0WData        (mc0WData      ), // input

   .mc0RAddr        (mc0RAddr      ), // input
   .mc0RData        (mc0RData      ), // output

   .pcIF            (pcIF          ), // input
   .pcEX            (pcEX          ), // input

   .overflow        (overflow      ), // input
   .ovIntReq        (ovIntReq      ), // output
   .extIntReq       (extIntReq     ), // output
   .intRet          (intRet        )  // output
   );

endmodule

//
// alhamdulillah
//
```

Listing 17 g_p.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : G&P.v
 * type      : rtl
 * function  : carry generator
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */
```

```
 module g_p (
             xi,
             yi,
             gi,
             pi
                         );

// input port declaration
input wire xi;
input wire yi;

// output port declaration
output wire gi;
output wire pi;

assign gi = xi&yi;
assign pi = xi^yi;


endmodule

/*
 * alhamdulillah
 */
```

Listing 18 ksOpBlack.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : ksOpBlack.v
 * type      : rtl
 * function  : kagge stone operator
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

 module ksOpBlack (
             pi,
             pk,
             gi,
             gk,
             gik,
             pik
                         );

// input port declaration

input wire pi;
input wire pk;
input wire gi;
input wire gk;
// output port declaration
output wire gik;
output wire pik;

assign gik = gi|(pi&gk);
assign pik = pi&pk;


endmodule
```

```
/*
 * alhamdulillah
 */
```

Listing 19 ksOpGray.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : ksOpGray.v
 * type      : rtl
 * function  : kagge stone operator
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

module ksOpGray (
                  Gc,
                  P,
                  Gg,
                  G
);

input  wire Gc ;
input  wire P  ;
input  wire Gg ;

output wire G  ;

assign Gtemp = Gc&P     ;
assign G     = Gtemp|Gg ;

endmodule

/*
 * alhamdulillah
 */
```

Listing 20 stage0.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : stage0.v
 * type      : rtl
 * function  : stage to generate and propagate earliest carrier
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

 module stage0 (
             x0 ,x16,y0 ,y16,p0_0  ,p16_16,g0_0  ,g16_16,
             x1 ,x17,y1 ,y17,p1_1  ,p17_17,g1_1  ,g17_17,
             x2 ,x18,y2 ,y18,p2_2  ,p18_18,g2_2  ,g18_18,
             x3 ,x19,y3 ,y19,p3_3  ,p19_19,g3_3  ,g19_19,
             x4 ,x20,y4 ,y20,p4_4  ,p20_20,g4_4  ,g20_20,
             x5 ,x21,y5 ,y21,p5_5  ,p21_21,g5_5  ,g21_21,
             x6 ,x22,y6 ,y22,p6_6  ,p22_22,g6_6  ,g22_22,
             x7 ,x23,y7 ,y23,p7_7  ,p23_23,g7_7  ,g23_23,
             x8 ,x24,y8 ,y24,p8_8  ,p24_24,g8_8  ,g24_24,
```

```
             x9 ,x25,y9 ,y25,p9_9  ,p25_25,g9_9  ,g25_25,
             x10,x26,y10,y26,p10_10,p26_26,g10_10,g26_26,
             x11,x27,y11,y27,p11_11,p27_27,g11_11,g27_27,
             x12,x28,y12,y28,p12_12,p28_28,g12_12,g28_28,
             x13,x29,y13,y29,p13_13,p29_29,g13_13,g29_29,
             x14,x30,y14,y30,p14_14,p30_30,g14_14,g30_30,
             x15,x31,y15,y31,p15_15,p31_31,g15_15,g31_31 );

// input port declaration
input wire  x0 ,  x16, y0 , y16;
input wire  x1 ,  x17, y1 , y17;
input wire  x2 ,  x18, y2 , y18;
input wire  x3 ,  x19, y3 , y19;
input wire  x4 ,  x20, y4 , y20;
input wire  x5 ,  x21, y5 , y21;
input wire  x6 ,  x22, y6 , y22;
input wire  x7 ,  x23, y7 , y23;
input wire  x8 ,  x24, y8 , y24;
input wire  x9 ,  x25, y9 , y25;
input wire  x10,  x26, y10, y26;
input wire  x11,  x27, y11, y27;
input wire  x12,  x28, y12, y28;
input wire  x13,  x29, y13, y29;
input wire  x14,  x30, y14, y30;
input wire  x15,  x31, y15, y31;

// output port declaration
output wire  p0_0  ,p16_16,g0_0  ,g16_16;
output wire  p1_1  ,p17_17,g1_1  ,g17_17;
output wire  p2_2  ,p18_18,g2_2  ,g18_18;
output wire  p3_3  ,p19_19,g3_3  ,g19_19;
output wire  p4_4  ,p20_20,g4_4  ,g20_20;
output wire  p5_5  ,p21_21,g5_5  ,g21_21;
output wire  p6_6  ,p22_22,g6_6  ,g22_22;
output wire  p7_7  ,p23_23,g7_7  ,g23_23;
output wire  p8_8  ,p24_24,g8_8  ,g24_24;
output wire  p9_9  ,p25_25,g9_9  ,g25_25;
output wire  p10_10,p26_26,g10_10,g26_26;
output wire  p11_11,p27_27,g11_11,g27_27;
output wire  p12_12,p28_28,g12_12,g28_28;
output wire  p13_13,p29_29,g13_13,g29_29;
output wire  p14_14,p30_30,g14_14,g30_30;
output wire  p15_15,p31_31,g15_15,g31_31;


g_p g_p0 (
          .xi (x0),
          .yi (y0),
          .gi (g0_0),
          .pi (p0_0)
          );

g_p g_p1 (
          .xi (x1),
          .yi (y1),
          .gi (g1_1),
          .pi (p1_1)
          );

g_p g_p2 (
          .xi (x2),
          .yi (y2),
          .gi (g2_2),
          .pi (p2_2)
          );

g_p g_p3 (
```

```
          .xi (x3),
          .yi (y3),
          .gi (g3_3),
          .pi (p3_3)
          );

g_p g_p4 (
          .xi (x4),
          .yi (y4),
          .gi (g4_4),
          .pi (p4_4)
          );

g_p g_p5 (
          .xi (x5),
          .yi (y5),
          .gi (g5_5),
          .pi (p5_5)
          );

g_p g_p6 (
          .xi (x6),
          .yi (y6),
          .gi (g6_6),
          .pi (p6_6)
          );

g_p g_p7 (
          .xi (x7),
          .yi (y7),
          .gi (g7_7),
          .pi (p7_7)
          );

g_p g_p8 (
          .xi (x8),
          .yi (y8),
          .gi (g8_8),
          .pi (p8_8)
          );

g_p g_p9 (
          .xi (x9),
          .yi (y9),
          .gi (g9_9),
          .pi (p9_9)
          );

g_p g_p10 (
          .xi (x10),
          .yi (y10),
          .gi (g10_10),
          .pi (p10_10)
          );

g_p g_p11 (
          .xi (x11),
          .yi (y11),
          .gi (g11_11),
          .pi (p11_11)
          );

g_p g_p12 (
          .xi (x12),
          .yi (y12),
          .gi (g12_12),
          .pi (p12_12)
```

```
          );

g_p g_p13 (
          .xi (x13),
          .yi (y13),
          .gi (g13_13),
          .pi (p13_13)
          );

g_p g_p14 (
          .xi (x14),
          .yi (y14),
          .gi (g14_14),
          .pi (p14_14)
          );

g_p g_p15 (
          .xi (x15),
          .yi (y15),
          .gi (g15_15),
          .pi (p15_15)
          );

g_p g_p16 (
          .xi (x16),
          .yi (y16),
          .gi (g16_16),
          .pi (p16_16)
          );

g_p g_p17 (
          .xi (x17),
          .yi (y17),
          .gi (g17_17),
          .pi (p17_17)
          );

g_p g_p18 (
          .xi (x18),
          .yi (y18),
          .gi (g18_18),
          .pi (p18_18)
          );

g_p g_p19 (
          .xi (x19),
          .yi (y19),
          .gi (g19_19),
          .pi (p19_19)
          );

g_p g_p20 (
          .xi (x20),
          .yi (y20),
          .gi (g20_20),
          .pi (p20_20)
          );

g_p g_p21 (
          .xi (x21),
          .yi (y21),
          .gi (g21_21),
          .pi (p21_21)
          );

g_p g_p22 (
          .xi (x22),
```

```
          .yi (y22),
          .gi (g22_22),
          .pi (p22_22)
          );

g_p g_p23 (
          .xi (x23),
          .yi (y23),
          .gi (g23_23),
          .pi (p23_23)
          );

g_p g_p24 (
          .xi (x24),
          .yi (y24),
          .gi (g24_24),
          .pi (p24_24)
          );

g_p g_p25 (
          .xi (x25),
          .yi (y25),
          .gi (g25_25),
          .pi (p25_25)
          );

g_p g_p26 (
          .xi (x26),
          .yi (y26),
          .gi (g26_26),
          .pi (p26_26)
          );

g_p g_p27 (
          .xi (x27),
          .yi (y27),
          .gi (g27_27),
          .pi (p27_27)
          );

g_p g_p28 (
          .xi (x28),
          .yi (y28),
          .gi (g28_28),
          .pi (p28_28)
          );

g_p g_p29 (
          .xi (x29),
          .yi (y29),
          .gi (g29_29),
          .pi (p29_29)
          );

g_p g_p30 (
          .xi (x30),
          .yi (y30),
          .gi (g30_30),
          .pi (p30_30)
          );

g_p g_p31 (
          .xi (x31),
          .yi (y31),
          .gi (g31_31),
          .pi (p31_31)
          );
```

```
endmodule

/*
 * alhamdulillah
 */
```

Listing 21 stage1.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : stage1.v
 * tgpe      : rtl
 * function  : stage to generate and propagate earliest carrier
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

module stage1 (
             pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16,p0_0  ,p16_16,g0_0  ,g16_16,
             pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17,p1_1  ,p17_17,g1_1  ,g17_17,
             pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18,p2_2  ,p18_18,g2_2  ,g18_18,
             pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19,p3_3  ,p19_19,g3_3  ,g19_19,
             pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20,p4_4  ,p20_20,g4_4  ,g20_20,
             pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21,p5_5  ,p21_21,g5_5  ,g21_21,
             pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22,p6_6  ,p22_22,g6_6  ,g22_22,
             pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23,p7_7  ,p23_23,g7_7  ,g23_23,
             pi8 ,pi24,pk8 ,pk24,gi8 ,gi24,gk8 ,gk24,p8_8  ,p24_24,g8_8  ,g24_24,
             pi9 ,pi25,pk9 ,pk25,gi9 ,gi25,gk9 ,gk25,p9_9  ,p25_25,g9_9  ,g25_25,
             pi10,pi26,pk10,pk26,gi10,gi26,gk10,gk26,p10_10,p26_26,g10_10,g26_26,
             pi11,pi27,pk11,pk27,gi11,gi27,gk11,gk27,p11_11,p27_27,g11_11,g27_27,
             pi12,pi28,pk12,pk28,gi12,gi28,gk12,gk28,p12_12,p28_28,g12_12,g28_28,
             pi13,pi29,pk13,pk29,gi13,gi29,gk13,gk29,p13_13,p29_29,g13_13,g29_29,
             pi14,pi30,pk14,pk30,gi14,gi30,gk14,gk30,p14_14,p30_30,g14_14,g30_30,
             pi15,     pk15,     gi15,     gk15,     p15_15,       g15_15);

// input port declaration
input wire  pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16;
input wire  pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17;
input wire  pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18;
input wire  pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19;
input wire  pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20;
input wire  pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21;
input wire  pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22;
input wire  pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23;
input wire  pi8 ,pi24,pk8 ,pk24,gi8 ,gi24,gk8 ,gk24;
input wire  pi9 ,pi25,pk9 ,pk25,gi9 ,gi25,gk9 ,gk25;
input wire  pi10,pi26,pk10,pk26,gi10,gi26,gk10,gk26;
input wire  pi11,pi27,pk11,pk27,gi11,gi27,gk11,gk27;
input wire  pi12,pi28,pk12,pk28,gi12,gi28,gk12,gk28;
input wire  pi13,pi29,pk13,pk29,gi13,gi29,gk13,gk29;
input wire  pi14,pi30,pk14,pk30,gi14,gi30,gk14,gk30;
input wire  pi15,     pk15,     gi15,     gk15;

// output port declaration
output wire  p0_0  ,p16_16,g0_0  ,g16_16;
output wire  p1_1  ,p17_17,g1_1  ,g17_17;
output wire  p2_2  ,p18_18,g2_2  ,g18_18;
output wire  p3_3  ,p19_19,g3_3  ,g19_19;
output wire  p4_4  ,p20_20,g4_4  ,g20_20;
output wire  p5_5  ,p21_21,g5_5  ,g21_21;
output wire  p6_6  ,p22_22,g6_6  ,g22_22;
```

```
output wire  p7_7  ,p23_23,g7_7  ,g23_23;
output wire  p8_8  ,p24_24,g8_8  ,g24_24;
output wire  p9_9  ,p25_25,g9_9  ,g25_25;
output wire  p10_10,p26_26,g10_10,g26_26;
output wire  p11_11,p27_27,g11_11,g27_27;
output wire  p12_12,p28_28,g12_12,g28_28;
output wire  p13_13,p29_29,g13_13,g29_29;
output wire  p14_14,p30_30,g14_14,g30_30;
output wire  p15_15,       g15_15;


ksOpBlack ksOpBlack0 (
          .pi  (pi0),
          .pk  (pk0),
          .gi  (gi0),
          .gk  (gk0),
          .gik (g0_0),
          .pik (p0_0)
          );

ksOpBlack ksOpBlack1 (
          .pi  (pi1),
          .pk  (pk1),
          .gi  (gi1),
          .gk  (gk1),
          .gik (g1_1),
          .pik (p1_1)
          );

ksOpBlack ksOpBlack2 (
          .pi  (pi2),
          .pk  (pk2),
          .gi  (gi2),
          .gk  (gk2),
          .gik (g2_2),
          .pik (p2_2)
          );

ksOpBlack ksOpBlack3 (
          .pi  (pi3),
          .pk  (pk3),
          .gi  (gi3),
          .gk  (gk3),
          .gik (g3_3),
          .pik (p3_3)
          );

ksOpBlack ksOpBlack4 (
          .pi  (pi4),
          .pk  (pk4),
          .gi  (gi4),
          .gk  (gk4),
          .gik (g4_4),
          .pik (p4_4)
          );

ksOpBlack ksOpBlack5 (
          .pi  (pi5),
          .pk  (pk5),
          .gi  (gi5),
          .gk  (gk5),
          .gik (g5_5),
          .pik (p5_5)
          );

ksOpBlack ksOpBlack6 (
          .pi  (pi6),
```

```
          .pk  (pk6),
          .gi  (gi6),
          .gk  (gk6),
          .gik (g6_6),
          .pik (p6_6)
          );

ksOpBlack ksOpBlack7 (
          .pi  (pi7),
          .pk  (pk7),
          .gi  (gi7),
          .gk  (gk7),
          .gik (g7_7),
          .pik (p7_7)
          );

ksOpBlack ksOpBlack8 (
          .pi  (pi8),
          .pk  (pk8),
          .gi  (gi8),
          .gk  (gk8),
          .gik (g8_8),
          .pik (p8_8)
          );

ksOpBlack ksOpBlack9 (
          .pi  (pi9),
          .pk  (pk9),
          .gi  (gi9),
          .gk  (gk9),
          .gik (g9_9),
          .pik (p9_9)
          );

ksOpBlack ksOpBlack10 (
          .pi  (pi10),
          .pk  (pk10),
          .gi  (gi10),
          .gk  (gk10),
          .gik (g10_10),
          .pik (p10_10)
          );

ksOpBlack ksOpBlack11 (
          .pi  (pi11),
          .pk  (pk11),
          .gi  (gi11),
          .gk  (gk11),
          .gik (g11_11),
          .pik (p11_11)
          );

ksOpBlack ksOpBlack12 (
          .pi  (pi12),
          .pk  (pk12),
          .gi  (gi12),
          .gk  (gk12),
          .gik (g12_12),
          .pik (p12_12)
          );

ksOpBlack ksOpBlack13 (
          .pi  (pi13),
          .pk  (pk13),
          .gi  (gi13),
          .gk  (gk13),
          .gik (g13_13),
```

```
          .pik (p13_13)
          );

ksOpBlack ksOpBlack14 (
          .pi  (pi14),
          .pk  (pk14),
          .gi  (gi14),
          .gk  (gk14),
          .gik (g14_14),
          .pik (p14_14)
          );

ksOpBlack ksOpBlack15 (
          .pi  (pi15),
          .pk  (pk15),
          .gi  (gi15),
          .gk  (gk15),
          .gik (g15_15),
          .pik (p15_15)
          );

ksOpBlack ksOpBlack16 (
          .pi  (pi16),
          .pk  (pk16),
          .gi  (gi16),
          .gk  (gk16),
          .gik (g16_16),
          .pik (p16_16)
          );

ksOpBlack ksOpBlack17 (
          .pi  (pi17),
          .pk  (pk17),
          .gi  (gi17),
          .gk  (gk17),
          .gik (g17_17),
          .pik (p17_17)
          );

ksOpBlack ksOpBlack18 (
          .pi  (pi18),
          .pk  (pk18),
          .gi  (gi18),
          .gk  (gk18),
          .gik (g18_18),
          .pik (p18_18)
          );

ksOpBlack ksOpBlack19 (
          .pi  (pi19),
          .pk  (pk19),
          .gi  (gi19),
          .gk  (gk19),
          .gik (g19_19),
          .pik (p19_19)
          );

ksOpBlack ksOpBlack20 (
          .pi  (pi20),
          .pk  (pk20),
          .gi  (gi20),
          .gk  (gk20),
          .gik (g20_20),
          .pik (p20_20)
          );

ksOpBlack ksOpBlack21 (
```

```
          .pi  (pi21),
          .pk  (pk21),
          .gi  (gi21),
          .gk  (gk21),
          .gik (g21_21),
          .pik (p21_21)
          );

ksOpBlack ksOpBlack22 (
          .pi  (pi22),
          .pk  (pk22),
          .gi  (gi22),
          .gk  (gk22),
          .gik (g22_22),
          .pik (p22_22)
          );

ksOpBlack ksOpBlack23 (
          .pi  (pi23),
          .pk  (pk23),
          .gi  (gi23),
          .gk  (gk23),
          .gik (g23_23),
          .pik (p23_23)
          );

ksOpBlack ksOpBlack24 (
          .pi  (pi24),
          .pk  (pk24),
          .gi  (gi24),
          .gk  (gk24),
          .gik (g24_24),
          .pik (p24_24)
          );

ksOpBlack ksOpBlack25 (
          .pi  (pi25),
          .pk  (pk25),
          .gi  (gi25),
          .gk  (gk25),
          .gik (g25_25),
          .pik (p25_25)
          );

ksOpBlack ksOpBlack26 (
          .pi  (pi26),
          .pk  (pk26),
          .gi  (gi26),
          .gk  (gk26),
          .gik (g26_26),
          .pik (p26_26)
          );

ksOpBlack ksOpBlack27 (
          .pi  (pi27),
          .pk  (pk27),
          .gi  (gi27),
          .gk  (gk27),
          .gik (g27_27),
          .pik (p27_27)
          );

ksOpBlack ksOpBlack28 (
          .pi  (pi28),
          .pk  (pk28),
          .gi  (gi28),
          .gk  (gk28),
```

```
          .gik (g28_28),
          .pik (p28_28)
          );

ksOpBlack ksOpBlack29 (
          .pi  (pi29),
          .pk  (pk29),
          .gi  (gi29),
          .gk  (gk29),
          .gik (g29_29),
          .pik (p29_29)
          );

ksOpBlack ksOpBlack30 (
          .pi  (pi30),
          .pk  (pk30),
          .gi  (gi30),
          .gk  (gk30),
          .gik (g30_30),
          .pik (p30_30)
          );


endmodule

/*
 * alhamdulillah
 */
```

Listing 22 stage2.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : stage2.v
 * tgpe      : rtl
 * function  : stage to generate and propagate earliest carrier
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

 module stage2 (
             pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16,p0_0  ,p16_16,g0_0  ,g16_16,
             pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17,p1_1  ,p17_17,g1_1  ,g17_17,
             pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18,p2_2  ,p18_18,g2_2  ,g18_18,
             pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19,p3_3  ,p19_19,g3_3  ,g19_19,
             pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20,p4_4  ,p20_20,g4_4  ,g20_20,
             pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21,p5_5  ,p21_21,g5_5  ,g21_21,
             pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22,p6_6  ,p22_22,g6_6  ,g22_22,
             pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23,p7_7  ,p23_23,g7_7  ,g23_23,
             pi8 ,pi24,pk8 ,pk24,gi8 ,gi24,gk8 ,gk24,p8_8  ,p24_24,g8_8  ,g24_24,
             pi9 ,pi25,pk9 ,pk25,gi9 ,gi25,gk9 ,gk25,p9_9  ,p25_25,g9_9  ,g25_25,
             pi10,pi26,pk10,pk26,gi10,gi26,gk10,gk26,p10_10,p26_26,g10_10,g26_26,
             pi11,pi27,pk11,pk27,gi11,gi27,gk11,gk27,p11_11,p27_27,g11_11,g27_27,
             pi12,pi28,pk12,pk28,gi12,gi28,gk12,gk28,p12_12,p28_28,g12_12,g28_28,
             pi13,pi29,pk13,pk29,gi13,gi29,gk13,gk29,p13_13,p29_29,g13_13,g29_29,
             pi14,     pk14,     gi14,     gk14,     p14_14,       g14_14,
             pi15,     pk15,     gi15,     gk15,     p15_15,       g15_15);

// input port declaration
input wire  pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16;
input wire  pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17;
input wire  pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18;
```

```
input wire  pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19;
input wire  pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20;
input wire  pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21;
input wire  pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22;
input wire  pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23;
input wire  pi8 ,pi24,pk8 ,pk24,gi8 ,gi24,gk8 ,gk24;
input wire  pi9 ,pi25,pk9 ,pk25,gi9 ,gi25,gk9 ,gk25;
input wire  pi10,pi26,pk10,pk26,gi10,gi26,gk10,gk26;
input wire  pi11,pi27,pk11,pk27,gi11,gi27,gk11,gk27;
input wire  pi12,pi28,pk12,pk28,gi12,gi28,gk12,gk28;
input wire  pi13,pi29,pk13,pk29,gi13,gi29,gk13,gk29;
input wire  pi14,     pk14,     gi14,     gk14;
input wire  pi15,     pk15,     gi15,     gk15;

// output port declaration
output wire  p0_0  ,p16_16,g0_0  ,g16_16;
output wire  p1_1  ,p17_17,g1_1  ,g17_17;
output wire  p2_2  ,p18_18,g2_2  ,g18_18;
output wire  p3_3  ,p19_19,g3_3  ,g19_19;
output wire  p4_4  ,p20_20,g4_4  ,g20_20;
output wire  p5_5  ,p21_21,g5_5  ,g21_21;
output wire  p6_6  ,p22_22,g6_6  ,g22_22;
output wire  p7_7  ,p23_23,g7_7  ,g23_23;
output wire  p8_8  ,p24_24,g8_8  ,g24_24;
output wire  p9_9  ,p25_25,g9_9  ,g25_25;
output wire  p10_10,p26_26,g10_10,g26_26;
output wire  p11_11,p27_27,g11_11,g27_27;
output wire  p12_12,p28_28,g12_12,g28_28;
output wire  p13_13,p29_29,g13_13,g29_29;
output wire  p14_14,       g14_14;
output wire  p15_15,       g15_15;


ksOpBlack ksOpBlack0 (
          .pi  (pi0),
          .pk  (pk0),
          .gi  (gi0),
          .gk  (gk0),
          .gik (g0_0),
          .pik (p0_0)
          );

ksOpBlack ksOpBlack1 (
          .pi  (pi1),
          .pk  (pk1),
          .gi  (gi1),
          .gk  (gk1),
          .gik (g1_1),
          .pik (p1_1)
          );

ksOpBlack ksOpBlack2 (
          .pi  (pi2),
          .pk  (pk2),
          .gi  (gi2),
          .gk  (gk2),
          .gik (g2_2),
          .pik (p2_2)
          );

ksOpBlack ksOpBlack3 (
          .pi  (pi3),
          .pk  (pk3),
          .gi  (gi3),
          .gk  (gk3),
          .gik (g3_3),
          .pik (p3_3)
```

```
          );

ksOpBlack ksOpBlack4 (
          .pi  (pi4),
          .pk  (pk4),
          .gi  (gi4),
          .gk  (gk4),
          .gik (g4_4),
          .pik (p4_4)
          );

ksOpBlack ksOpBlack5 (
          .pi  (pi5),
          .pk  (pk5),
          .gi  (gi5),
          .gk  (gk5),
          .gik (g5_5),
          .pik (p5_5)
          );

ksOpBlack ksOpBlack6 (
          .pi  (pi6),
          .pk  (pk6),
          .gi  (gi6),
          .gk  (gk6),
          .gik (g6_6),
          .pik (p6_6)
          );

ksOpBlack ksOpBlack7 (
          .pi  (pi7),
          .pk  (pk7),
          .gi  (gi7),
          .gk  (gk7),
          .gik (g7_7),
          .pik (p7_7)
          );

ksOpBlack ksOpBlack8 (
          .pi  (pi8),
          .pk  (pk8),
          .gi  (gi8),
          .gk  (gk8),
          .gik (g8_8),
          .pik (p8_8)
          );

ksOpBlack ksOpBlack9 (
          .pi  (pi9),
          .pk  (pk9),
          .gi  (gi9),
          .gk  (gk9),
          .gik (g9_9),
          .pik (p9_9)
          );

ksOpBlack ksOpBlack10 (
          .pi  (pi10),
          .pk  (pk10),
          .gi  (gi10),
          .gk  (gk10),
          .gik (g10_10),
          .pik (p10_10)
          );

ksOpBlack ksOpBlack11 (
          .pi  (pi11),
```

```
          .pk  (pk11),
          .gi  (gi11),
          .gk  (gk11),
          .gik (g11_11),
          .pik (p11_11)
          );

ksOpBlack ksOpBlack12 (
          .pi  (pi12),
          .pk  (pk12),
          .gi  (gi12),
          .gk  (gk12),
          .gik (g12_12),
          .pik (p12_12)
          );

ksOpBlack ksOpBlack13 (
          .pi  (pi13),
          .pk  (pk13),
          .gi  (gi13),
          .gk  (gk13),
          .gik (g13_13),
          .pik (p13_13)
          );

ksOpBlack ksOpBlack14 (
          .pi  (pi14),
          .pk  (pk14),
          .gi  (gi14),
          .gk  (gk14),
          .gik (g14_14),
          .pik (p14_14)
          );

ksOpBlack ksOpBlack15 (
          .pi  (pi15),
          .pk  (pk15),
          .gi  (gi15),
          .gk  (gk15),
          .gik (g15_15),
          .pik (p15_15)
          );

ksOpBlack ksOpBlack16 (
          .pi  (pi16),
          .pk  (pk16),
          .gi  (gi16),
          .gk  (gk16),
          .gik (g16_16),
          .pik (p16_16)
          );

ksOpBlack ksOpBlack17 (
          .pi  (pi17),
          .pk  (pk17),
          .gi  (gi17),
          .gk  (gk17),
          .gik (g17_17),
          .pik (p17_17)
          );

ksOpBlack ksOpBlack18 (
          .pi  (pi18),
          .pk  (pk18),
          .gi  (gi18),
          .gk  (gk18),
          .gik (g18_18),
```

```
          .pik (p18_18)
          );

ksOpBlack ksOpBlack19 (
          .pi  (pi19),
          .pk  (pk19),
          .gi  (gi19),
          .gk  (gk19),
          .gik (g19_19),
          .pik (p19_19)
          );

ksOpBlack ksOpBlack20 (
          .pi  (pi20),
          .pk  (pk20),
          .gi  (gi20),
          .gk  (gk20),
          .gik (g20_20),
          .pik (p20_20)
          );

ksOpBlack ksOpBlack21 (
          .pi  (pi21),
          .pk  (pk21),
          .gi  (gi21),
          .gk  (gk21),
          .gik (g21_21),
          .pik (p21_21)
          );

ksOpBlack ksOpBlack22 (
          .pi  (pi22),
          .pk  (pk22),
          .gi  (gi22),
          .gk  (gk22),
          .gik (g22_22),
          .pik (p22_22)
          );

ksOpBlack ksOpBlack23 (
          .pi  (pi23),
          .pk  (pk23),
          .gi  (gi23),
          .gk  (gk23),
          .gik (g23_23),
          .pik (p23_23)
          );

ksOpBlack ksOpBlack24 (
          .pi  (pi24),
          .pk  (pk24),
          .gi  (gi24),
          .gk  (gk24),
          .gik (g24_24),
          .pik (p24_24)
          );

ksOpBlack ksOpBlack25 (
          .pi  (pi25),
          .pk  (pk25),
          .gi  (gi25),
          .gk  (gk25),
          .gik (g25_25),
          .pik (p25_25)
          );

ksOpBlack ksOpBlack26 (
```

```
          .pi  (pi26),
          .pk  (pk26),
          .gi  (gi26),
          .gk  (gk26),
          .gik (g26_26),
          .pik (p26_26)
          );

ksOpBlack ksOpBlack27 (
          .pi  (pi27),
          .pk  (pk27),
          .gi  (gi27),
          .gk  (gk27),
          .gik (g27_27),
          .pik (p27_27)
          );

ksOpBlack ksOpBlack28 (
          .pi  (pi28),
          .pk  (pk28),
          .gi  (gi28),
          .gk  (gk28),
          .gik (g28_28),
          .pik (p28_28)
          );

ksOpBlack ksOpBlack29 (
          .pi  (pi29),
          .pk  (pk29),
          .gi  (gi29),
          .gk  (gk29),
          .gik (g29_29),
          .pik (p29_29)
          );




endmodule

/*
 * alhamdulillah
 */
```

Listing 23 stage3.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : stage3.v
 * tgpe      : rtl
 * function  : stage to generate and propagate earliest carrier
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

 module stage3 (
             pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16,p0_0  ,p16_16,g0_0  ,g16_16,
             pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17,p1_1  ,p17_17,g1_1  ,g17_17,
             pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18,p2_2  ,p18_18,g2_2  ,g18_18,
             pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19,p3_3  ,p19_19,g3_3  ,g19_19,
             pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20,p4_4  ,p20_20,g4_4  ,g20_20,
             pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21,p5_5  ,p21_21,g5_5  ,g21_21,
```

```
            pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22,p6_6  ,p22_22,g6_6  ,g22_22,
            pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23,p7_7  ,p23_23,g7_7  ,g23_23,
            pi8 ,pi24,pk8 ,pk24,gi8 ,gi24,gk8 ,gk24,p8_8  ,p24_24,g8_8  ,g24_24,
            pi9 ,pi25,pk9 ,pk25,gi9 ,gi25,gk9 ,gk25,p9_9  ,p25_25,g9_9  ,g25_25,
            pi10,pi26,pk10,pk26,gi10,gi26,gk10,gk26,p10_10,p26_26,g10_10,g26_26,
            pi11,pi27,pk11,pk27,gi11,gi27,gk11,gk27,p11_11,p27_27,g11_11,g27_27,
            pi12,     pk12,     gi12,     gk12,     p12_12,       g12_12,
            pi13,     pk13,     gi13,     gk13,     p13_13,       g13_13,
            pi14,     pk14,     gi14,     gk14,     p14_14,       g14_14,
            pi15,     pk15,     gi15,     gk15,     p15_15,       g15_15);

// input port declaration
input wire  pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16;
input wire  pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17;
input wire  pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18;
input wire  pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19;
input wire  pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20;
input wire  pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21;
input wire  pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22;
input wire  pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23;
input wire  pi8 ,pi24,pk8 ,pk24,gi8 ,gi24,gk8 ,gk24;
input wire  pi9 ,pi25,pk9 ,pk25,gi9 ,gi25,gk9 ,gk25;
input wire  pi10,pi26,pk10,pk26,gi10,gi26,gk10,gk26;
input wire  pi11,pi27,pk11,pk27,gi11,gi27,gk11,gk27;
input wire  pi12,     pk12,     gi12,     gk12;
input wire  pi13,     pk13,     gi13,     gk13;
input wire  pi14,     pk14,     gi14,     gk14;
input wire  pi15,     pk15,     gi15,     gk15;

// output port declaration
output wire  p0_0  ,p16_16,g0_0  ,g16_16;
output wire  p1_1  ,p17_17,g1_1  ,g17_17;
output wire  p2_2  ,p18_18,g2_2  ,g18_18;
output wire  p3_3  ,p19_19,g3_3  ,g19_19;
output wire  p4_4  ,p20_20,g4_4  ,g20_20;
output wire  p5_5  ,p21_21,g5_5  ,g21_21;
output wire  p6_6  ,p22_22,g6_6  ,g22_22;
output wire  p7_7  ,p23_23,g7_7  ,g23_23;
output wire  p8_8  ,p24_24,g8_8  ,g24_24;
output wire  p9_9  ,p25_25,g9_9  ,g25_25;
output wire  p10_10,p26_26,g10_10,g26_26;
output wire  p11_11,p27_27,g11_11,g27_27;
output wire  p12_12,       g12_12;
output wire  p13_13,       g13_13;
output wire  p14_14,       g14_14;
output wire  p15_15,       g15_15;


ksOpBlack ksOpBlack0 (
          .pi  (pi0),
          .pk  (pk0),
          .gi  (gi0),
          .gk  (gk0),
          .gik (g0_0),
          .pik (p0_0)
          );

ksOpBlack ksOpBlack1 (
          .pi (pi1),
          .pk (pk1),
          .gi (gi1),
          .gk (gk1),
          .gik (g1_1),
          .pik (p1_1)
          );

ksOpBlack ksOpBlack2 (
```

```
          .pi  (pi2),
          .pk  (pk2),
          .gi  (gi2),
          .gk  (gk2),
          .gik (g2_2),
          .pik (p2_2)
          );

ksOpBlack ksOpBlack3 (
          .pi  (pi3),
          .pk  (pk3),
          .gi  (gi3),
          .gk  (gk3),
          .gik (g3_3),
          .pik (p3_3)
          );

ksOpBlack ksOpBlack4 (
          .pi  (pi4),
          .pk  (pk4),
          .gi  (gi4),
          .gk  (gk4),
          .gik (g4_4),
          .pik (p4_4)
          );

ksOpBlack ksOpBlack5 (
          .pi  (pi5),
          .pk  (pk5),
          .gi  (gi5),
          .gk  (gk5),
          .gik (g5_5),
          .pik (p5_5)
          );

ksOpBlack ksOpBlack6 (
          .pi  (pi6),
          .pk  (pk6),
          .gi  (gi6),
          .gk  (gk6),
          .gik (g6_6),
          .pik (p6_6)
          );

ksOpBlack ksOpBlack7 (
          .pi  (pi7),
          .pk  (pk7),
          .gi  (gi7),
          .gk  (gk7),
          .gik (g7_7),
          .pik (p7_7)
          );

ksOpBlack ksOpBlack8 (
          .pi  (pi8),
          .pk  (pk8),
          .gi  (gi8),
          .gk  (gk8),
          .gik (g8_8),
          .pik (p8_8)
          );

ksOpBlack ksOpBlack9 (
          .pi  (pi9),
          .pk  (pk9),
          .gi  (gi9),
          .gk  (gk9),
```

```
          .gik (g9_9),
          .pik (p9_9)
          );

ksOpBlack ksOpBlack10 (
          .pi  (pi10),
          .pk  (pk10),
          .gi  (gi10),
          .gk  (gk10),
          .gik (g10_10),
          .pik (p10_10)
          );

ksOpBlack ksOpBlack11 (
          .pi  (pi11),
          .pk  (pk11),
          .gi  (gi11),
          .gk  (gk11),
          .gik (g11_11),
          .pik (p11_11)
          );

ksOpBlack ksOpBlack12 (
          .pi  (pi12),
          .pk  (pk12),
          .gi  (gi12),
          .gk  (gk12),
          .gik (g12_12),
          .pik (p12_12)
          );

ksOpBlack ksOpBlack13 (
          .pi  (pi13),
          .pk  (pk13),
          .gi  (gi13),
          .gk  (gk13),
          .gik (g13_13),
          .pik (p13_13)
          );

ksOpBlack ksOpBlack14 (
          .pi  (pi14),
          .pk  (pk14),
          .gi  (gi14),
          .gk  (gk14),
          .gik (g14_14),
          .pik (p14_14)
          );

ksOpBlack ksOpBlack15 (
          .pi  (pi15),
          .pk  (pk15),
          .gi  (gi15),
          .gk  (gk15),
          .gik (g15_15),
          .pik (p15_15)
          );

ksOpBlack ksOpBlack16 (
          .pi  (pi16),
          .pk  (pk16),
          .gi  (gi16),
          .gk  (gk16),
          .gik (g16_16),
          .pik (p16_16)
          );
```

```
ksOpBlack ksOpBlack17 (
          .pi  (pi17),
          .pk  (pk17),
          .gi  (gi17),
          .gk  (gk17),
          .gik (g17_17),
          .pik (p17_17)
          );

ksOpBlack ksOpBlack18 (
          .pi  (pi18),
          .pk  (pk18),
          .gi  (gi18),
          .gk  (gk18),
          .gik (g18_18),
          .pik (p18_18)
          );

ksOpBlack ksOpBlack19 (
          .pi  (pi19),
          .pk  (pk19),
          .gi  (gi19),
          .gk  (gk19),
          .gik (g19_19),
          .pik (p19_19)
          );

ksOpBlack ksOpBlack20 (
          .pi  (pi20),
          .pk  (pk20),
          .gi  (gi20),
          .gk  (gk20),
          .gik (g20_20),
          .pik (p20_20)
          );

ksOpBlack ksOpBlack21 (
          .pi  (pi21),
          .pk  (pk21),
          .gi  (gi21),
          .gk  (gk21),
          .gik (g21_21),
          .pik (p21_21)
          );

ksOpBlack ksOpBlack22 (
          .pi  (pi22),
          .pk  (pk22),
          .gi  (gi22),
          .gk  (gk22),
          .gik (g22_22),
          .pik (p22_22)
          );

ksOpBlack ksOpBlack23 (
          .pi  (pi23),
          .pk  (pk23),
          .gi  (gi23),
          .gk  (gk23),
          .gik (g23_23),
          .pik (p23_23)
          );

ksOpBlack ksOpBlack24 (
          .pi  (pi24),
          .pk  (pk24),
          .gi  (gi24),
```

```
          .gk  (gk24),
          .gik (g24_24),
          .pik (p24_24)
          );

ksOpBlack ksOpBlack25 (
          .pi  (pi25),
          .pk  (pk25),
          .gi  (gi25),
          .gk  (gk25),
          .gik (g25_25),
          .pik (p25_25)
          );

ksOpBlack ksOpBlack26 (
          .pi  (pi26),
          .pk  (pk26),
          .gi  (gi26),
          .gk  (gk26),
          .gik (g26_26),
          .pik (p26_26)
          );

ksOpBlack ksOpBlack27 (
          .pi  (pi27),
          .pk  (pk27),
          .gi  (gi27),
          .gk  (gk27),
          .gik (g27_27),
          .pik (p27_27)
          );



endmodule

/*
 * alhamdulillah
 */
```

Listing 24 stage4.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : stage4.v
 * tgpe      : rtl
 * function  : stage to generate and propagate earliest carrier
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

 module stage4 (
             pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16,p0_0  ,p16_16,g0_0  ,g16_16,
             pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17,p1_1  ,p17_17,g1_1  ,g17_17,
             pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18,p2_2  ,p18_18,g2_2  ,g18_18,
             pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19,p3_3  ,p19_19,g3_3  ,g19_19,
             pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20,p4_4  ,p20_20,g4_4  ,g20_20,
             pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21,p5_5  ,p21_21,g5_5  ,g21_21,
             pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22,p6_6  ,p22_22,g6_6  ,g22_22,
             pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23,p7_7  ,p23_23,g7_7  ,g23_23,
             pi8 ,     pk8 ,     gi8 ,     gk8 ,     p8_8  ,       g8_8  ,
             pi9 ,     pk9 ,     gi9 ,     gk9 ,     p9_9  ,       g9_9  ,
```

```
            pi10,     pk10,     gi10,     gk10,     p10_10,       g10_10,
            pi11,     pk11,     gi11,     gk11,     p11_11,       g11_11,
            pi12,     pk12,     gi12,     gk12,     p12_12,       g12_12,
            pi13,     pk13,     gi13,     gk13,     p13_13,       g13_13,
            pi14,     pk14,     gi14,     gk14,     p14_14,       g14_14,
            pi15,     pk15,     gi15,     gk15,     p15_15,       g15_15);

// input port declaration
input wire  pi0 ,pi16,pk0 ,pk16,gi0 ,gi16,gk0 ,gk16;
input wire  pi1 ,pi17,pk1 ,pk17,gi1 ,gi17,gk1 ,gk17;
input wire  pi2 ,pi18,pk2 ,pk18,gi2 ,gi18,gk2 ,gk18;
input wire  pi3 ,pi19,pk3 ,pk19,gi3 ,gi19,gk3 ,gk19;
input wire  pi4 ,pi20,pk4 ,pk20,gi4 ,gi20,gk4 ,gk20;
input wire  pi5 ,pi21,pk5 ,pk21,gi5 ,gi21,gk5 ,gk21;
input wire  pi6 ,pi22,pk6 ,pk22,gi6 ,gi22,gk6 ,gk22;
input wire  pi7 ,pi23,pk7 ,pk23,gi7 ,gi23,gk7 ,gk23;
input wire  pi8 ,     pk8 ,     gi8 ,     gk8 ;
input wire  pi9 ,     pk9 ,     gi9 ,     gk9 ;
input wire  pi10,     pk10,     gi10,     gk10;
input wire  pi11,     pk11,     gi11,     gk11;
input wire  pi12,     pk12,     gi12,     gk12;
input wire  pi13,     pk13,     gi13,     gk13;
input wire  pi14,     pk14,     gi14,     gk14;
input wire  pi15,     pk15,     gi15,     gk15;

// output port declaration
output wire  p0_0  ,p16_16,g0_0  ,g16_16;
output wire  p1_1  ,p17_17,g1_1  ,g17_17;
output wire  p2_2  ,p18_18,g2_2  ,g18_18;
output wire  p3_3  ,p19_19,g3_3  ,g19_19;
output wire  p4_4  ,p20_20,g4_4  ,g20_20;
output wire  p5_5  ,p21_21,g5_5  ,g21_21;
output wire  p6_6  ,p22_22,g6_6  ,g22_22;
output wire  p7_7  ,p23_23,g7_7  ,g23_23;
output wire  p8_8  ,       g8_8  ;
output wire  p9_9  ,       g9_9  ;
output wire  p10_10,       g10_10;
output wire  p11_11,       g11_11;
output wire  p12_12,       g12_12;
output wire  p13_13,       g13_13;
output wire  p14_14,       g14_14;
output wire  p15_15,       g15_15;


ksOpBlack ksOpBlack0 (
          .pi  (pi0),
          .pk  (pk0),
          .gi  (gi0),
          .gk  (gk0),
          .gik (g0_0),
          .pik (p0_0)
          );

ksOpBlack ksOpBlack1 (
          .pi (pi1),
          .pk (pk1),
          .gi (gi1),
          .gk (gk1),
          .gik (g1_1),
          .pik (p1_1)
          );

ksOpBlack ksOpBlack2 (
          .pi  (pi2),
          .pk  (pk2),
          .gi  (gi2),
          .gk  (gk2),
```

```
          .gik (g2_2),
          .pik (p2_2)
          );

ksOpBlack ksOpBlack3 (
          .pi  (pi3),
          .pk  (pk3),
          .gi  (gi3),
          .gk  (gk3),
          .gik (g3_3),
          .pik (p3_3)
          );

ksOpBlack ksOpBlack4 (
          .pi  (pi4),
          .pk  (pk4),
          .gi  (gi4),
          .gk  (gk4),
          .gik (g4_4),
          .pik (p4_4)
          );

ksOpBlack ksOpBlack5 (
          .pi  (pi5),
          .pk  (pk5),
          .gi  (gi5),
          .gk  (gk5),
          .gik (g5_5),
          .pik (p5_5)
          );

ksOpBlack ksOpBlack6 (
          .pi  (pi6),
          .pk  (pk6),
          .gi  (gi6),
          .gk  (gk6),
          .gik (g6_6),
          .pik (p6_6)
          );

ksOpBlack ksOpBlack7 (
          .pi  (pi7),
          .pk  (pk7),
          .gi  (gi7),
          .gk  (gk7),
          .gik (g7_7),
          .pik (p7_7)
          );

ksOpBlack ksOpBlack8 (
          .pi  (pi8),
          .pk  (pk8),
          .gi  (gi8),
          .gk  (gk8),
          .gik (g8_8),
          .pik (p8_8)
          );

ksOpBlack ksOpBlack9 (
          .pi  (pi9),
          .pk  (pk9),
          .gi  (gi9),
          .gk  (gk9),
          .gik (g9_9),
          .pik (p9_9)
          );
```

```
ksOpBlack ksOpBlack10 (
          .pi  (pi10),
          .pk  (pk10),
          .gi  (gi10),
          .gk  (gk10),
          .gik (g10_10),
          .pik (p10_10)
          );

ksOpBlack ksOpBlack11 (
          .pi  (pi11),
          .pk  (pk11),
          .gi  (gi11),
          .gk  (gk11),
          .gik (g11_11),
          .pik (p11_11)
          );

ksOpBlack ksOpBlack12 (
          .pi  (pi12),
          .pk  (pk12),
          .gi  (gi12),
          .gk  (gk12),
          .gik (g12_12),
          .pik (p12_12)
          );

ksOpBlack ksOpBlack13 (
          .pi  (pi13),
          .pk  (pk13),
          .gi  (gi13),
          .gk  (gk13),
          .gik (g13_13),
          .pik (p13_13)
          );

ksOpBlack ksOpBlack14 (
          .pi  (pi14),
          .pk  (pk14),
          .gi  (gi14),
          .gk  (gk14),
          .gik (g14_14),
          .pik (p14_14)
          );

ksOpBlack ksOpBlack15 (
          .pi  (pi15),
          .pk  (pk15),
          .gi  (gi15),
          .gk  (gk15),
          .gik (g15_15),
          .pik (p15_15)
          );

ksOpBlack ksOpBlack16 (
          .pi  (pi16),
          .pk  (pk16),
          .gi  (gi16),
          .gk  (gk16),
          .gik (g16_16),
          .pik (p16_16)
          );

ksOpBlack ksOpBlack17 (
          .pi  (pi17),
          .pk  (pk17),
          .gi  (gi17),
```

```
          .gk  (gk17),
          .gik (g17_17),
          .pik (p17_17)
          );

ksOpBlack ksOpBlack18 (
          .pi  (pi18),
          .pk  (pk18),
          .gi  (gi18),
          .gk  (gk18),
          .gik (g18_18),
          .pik (p18_18)
          );

ksOpBlack ksOpBlack19 (
          .pi  (pi19),
          .pk  (pk19),
          .gi  (gi19),
          .gk  (gk19),
          .gik (g19_19),
          .pik (p19_19)
          );

ksOpBlack ksOpBlack20 (
          .pi  (pi20),
          .pk  (pk20),
          .gi  (gi20),
          .gk  (gk20),
          .gik (g20_20),
          .pik (p20_20)
          );

ksOpBlack ksOpBlack21 (
          .pi  (pi21),
          .pk  (pk21),
          .gi  (gi21),
          .gk  (gk21),
          .gik (g21_21),
          .pik (p21_21)
          );

ksOpBlack ksOpBlack22 (
          .pi  (pi22),
          .pk  (pk22),
          .gi  (gi22),
          .gk  (gk22),
          .gik (g22_22),
          .pik (p22_22)
          );

ksOpBlack ksOpBlack23 (
          .pi  (pi23),
          .pk  (pk23),
          .gi  (gi23),
          .gk  (gk23),
          .gik (g23_23),
          .pik (p23_23)
          );

endmodule

/*
 * alhamdulillah
 */
```

Listing 25 stage5.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : stage5.v
 * tgpe      : rtl
 * function  : stage to generate and propagate earliest carrier
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

 module stage5 (
             pi0 ,     pk0 ,     gi0 ,     gk0 ,     p0_0  ,       g0_0  ,
             pi1 ,     pk1 ,     gi1 ,     gk1 ,     p1_1  ,       g1_1  ,
             pi2 ,     pk2 ,     gi2 ,     gk2 ,     p2_2  ,       g2_2  ,
             pi3 ,     pk3 ,     gi3 ,     gk3 ,     p3_3  ,       g3_3  ,
             pi4 ,     pk4 ,     gi4 ,     gk4 ,     p4_4  ,       g4_4  ,
             pi5 ,     pk5 ,     gi5 ,     gk5 ,     p5_5  ,       g5_5  ,
             pi6 ,     pk6 ,     gi6 ,     gk6 ,     p6_6  ,       g6_6  ,
             pi7 ,     pk7 ,     gi7 ,     gk7 ,     p7_7  ,       g7_7  ,
             pi8 ,     pk8 ,     gi8 ,     gk8 ,     p8_8  ,       g8_8  ,
             pi9 ,     pk9 ,     gi9 ,     gk9 ,     p9_9  ,       g9_9  ,
             pi10,     pk10,     gi10,     gk10,     p10_10,       g10_10,
             pi11,     pk11,     gi11,     gk11,     p11_11,       g11_11,
             pi12,     pk12,     gi12,     gk12,     p12_12,       g12_12,
             pi13,     pk13,     gi13,     gk13,     p13_13,       g13_13,
             pi14,     pk14,     gi14,     gk14,     p14_14,       g14_14,
             pi15,     pk15,     gi15,     gk15,     p15_15,       g15_15 );

// input port declaration
input wire  pi0 ,     pk0 ,     gi0 ,     gk0 ;
input wire  pi1 ,     pk1 ,     gi1 ,     gk1 ;
input wire  pi2 ,     pk2 ,     gi2 ,     gk2 ;
input wire  pi3 ,     pk3 ,     gi3 ,     gk3 ;
input wire  pi4 ,     pk4 ,     gi4 ,     gk4 ;
input wire  pi5 ,     pk5 ,     gi5 ,     gk5 ;
input wire  pi6 ,     pk6 ,     gi6 ,     gk6 ;
input wire  pi7 ,     pk7 ,     gi7 ,     gk7 ;
input wire  pi8 ,     pk8 ,     gi8 ,     gk8 ;
input wire  pi9 ,     pk9 ,     gi9 ,     gk9 ;
input wire  pi10,     pk10,     gi10,     gk10;
input wire  pi11,     pk11,     gi11,     gk11;
input wire  pi12,     pk12,     gi12,     gk12;
input wire  pi13,     pk13,     gi13,     gk13;
input wire  pi14,     pk14,     gi14,     gk14;
input wire  pi15,     pk15,     gi15,     gk15;

// output port declaration
output wire  p0_0  ,       g0_0  ;
output wire  p1_1  ,       g1_1  ;
output wire  p2_2  ,       g2_2  ;
output wire  p3_3  ,       g3_3  ;
output wire  p4_4  ,       g4_4  ;
output wire  p5_5  ,       g5_5  ;
output wire  p6_6  ,       g6_6  ;
output wire  p7_7  ,       g7_7  ;
output wire  p8_8  ,       g8_8  ;
output wire  p9_9  ,       g9_9  ;
output wire  p10_10,       g10_10;
output wire  p11_11,       g11_11;
output wire  p12_12,       g12_12;
```

```
output wire  p13_13,       g13_13;
output wire  p14_14,       g14_14;
output wire  p15_15,       g15_15;


ksOpBlack ksOpBlack0 (
          .pi  (pi0),
          .pk  (pk0),
          .gi  (gi0),
          .gk  (gk0),
          .gik (g0_0),
          .pik (p0_0)
          );

ksOpBlack ksOpBlack1 (
          .pi (pi1),
          .pk (pk1),
          .gi (gi1),
          .gk (gk1),
          .gik (g1_1),
          .pik (p1_1)
          );

ksOpBlack ksOpBlack2 (
          .pi  (pi2),
          .pk  (pk2),
          .gi  (gi2),
          .gk  (gk2),
          .gik (g2_2),
          .pik (p2_2)
          );

ksOpBlack ksOpBlack3 (
          .pi  (pi3),
          .pk  (pk3),
          .gi  (gi3),
          .gk  (gk3),
          .gik (g3_3),
          .pik (p3_3)
          );

ksOpBlack ksOpBlack4 (
          .pi  (pi4),
          .pk  (pk4),
          .gi  (gi4),
          .gk  (gk4),
          .gik (g4_4),
          .pik (p4_4)
          );

ksOpBlack ksOpBlack5 (
          .pi  (pi5),
          .pk  (pk5),
          .gi  (gi5),
          .gk  (gk5),
          .gik (g5_5),
          .pik (p5_5)
          );

ksOpBlack ksOpBlack6 (
          .pi  (pi6),
          .pk  (pk6),
          .gi  (gi6),
          .gk  (gk6),
          .gik (g6_6),
          .pik (p6_6)
          );
```

```
ksOpBlack ksOpBlack7 (
          .pi  (pi7),
          .pk  (pk7),
          .gi  (gi7),
          .gk  (gk7),
          .gik (g7_7),
          .pik (p7_7)
          );

ksOpBlack ksOpBlack8 (
          .pi  (pi8),
          .pk  (pk8),
          .gi  (gi8),
          .gk  (gk8),
          .gik (g8_8),
          .pik (p8_8)
          );

ksOpBlack ksOpBlack9 (
          .pi  (pi9),
          .pk  (pk9),
          .gi  (gi9),
          .gk  (gk9),
          .gik (g9_9),
          .pik (p9_9)
          );

ksOpBlack ksOpBlack10 (
          .pi  (pi10),
          .pk  (pk10),
          .gi  (gi10),
          .gk  (gk10),
          .gik (g10_10),
          .pik (p10_10)
          );

ksOpBlack ksOpBlack11 (
          .pi  (pi11),
          .pk  (pk11),
          .gi  (gi11),
          .gk  (gk11),
          .gik (g11_11),
          .pik (p11_11)
          );

ksOpBlack ksOpBlack12 (
          .pi  (pi12),
          .pk  (pk12),
          .gi  (gi12),
          .gk  (gk12),
          .gik (g12_12),
          .pik (p12_12)
          );

ksOpBlack ksOpBlack13 (
          .pi  (pi13),
          .pk  (pk13),
          .gi  (gi13),
          .gk  (gk13),
          .gik (g13_13),
          .pik (p13_13)
          );

ksOpBlack ksOpBlack14 (
          .pi  (pi14),
          .pk  (pk14),
```

```
          .gi  (gi14),
          .gk  (gk14),
          .gik (g14_14),
          .pik (p14_14)
          );

ksOpBlack ksOpBlack15 (
          .pi  (pi15),
          .pk  (pk15),
          .gi  (gi15),
          .gk  (gk15),
          .gik (g15_15),
          .pik (p15_15)
          );

endmodule

/*
 * alhamdulillah
 */
```

Listing 26 ksAdder.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : ksAdder.v
 * type      : rtl
 * function  : fast adder logic unit for 32 bit
 * edit      : -
 * author    : afirdaus
 * rev. date : 20081013 - created
 *
 */

module ksAdder (
   x32,
   y32,
   cin,
   s32
   );

// input port declaration
input [31:0] x32;
input [31:0] y32;
input        cin;

wire  [31:0] x32;
wire  [31:0] y32;
wire         cin;

// output port declaration
output [31:0] s32;
wire   [31:0] s32;


wire  x0 ,  x16, y0 , y16;
wire  x1 ,  x17, y1 , y17;
wire  x2 ,  x18, y2 , y18;
wire  x3 ,  x19, y3 , y19;
wire  x4 ,  x20, y4 , y20;
wire  x5 ,  x21, y5 , y21;
```

```
wire  x6 ,  x22, y6 , y22;
wire  x7 ,  x23, y7 , y23;
wire  x8 ,  x24, y8 , y24;
wire  x9 ,  x25, y9 , y25;
wire  x10,  x26, y10, y26;
wire  x11,  x27, y11, y27;
wire  x12,  x28, y12, y28;
wire  x13,  x29, y13, y29;
wire  x14,  x30, y14, y30;
wire  x15,  x31, y15, y31;


wire  s0 ,s16;
wire  s1 ,s17;
wire  s2 ,s18;
wire  s3 ,s19;
wire  s4 ,s20;
wire  s5 ,s21;
wire  s6 ,s22;
wire  s7 ,s23;
wire  s8 ,s24;
wire  s9 ,s25;
wire  s10,s26;
wire  s11,s27;
wire  s12,s28;
wire  s13,s29;
wire  s14,s30;
wire  s15,s31;

wire   p0_0  ,p16_16,g0_0  ,g16_16, p1_0  ,p17_16,g1_0  ,g17_16, p2_0  ,p18_15,g2_0
,g18_15, p4_0  ,p20_13,g4_0  ,g20_13,  p8_0 ,p24_9 ,g8_0 ,g24_9 , p16_0,g16_0;
wire   p1_1  ,p17_17,g1_1  ,g17_17, p2_1  ,p18_17,g2_1  ,g18_17, p3_0  ,p19_16,g3_0
,g19_16, p5_0  ,p21_14,g5_0  ,g21_14,  p9_0 ,p25_10,g9_0 ,g25_10, p17_0,g17_0;
wire   p2_2  ,p18_18,g2_2  ,g18_18, p3_2  ,p19_18,g3_2  ,g19_18, p4_1  ,p20_17,g4_1
,g20_17, p6_0  ,p22_15,g6_0  ,g22_15,  p10_0,p26_11,g10_0,g26_11, p18_0,g18_0;
wire   p3_3  ,p19_19,g3_3  ,g19_19, p4_3  ,p20_19,g4_3  ,g20_19, p5_2  ,p21_18,g5_2
,g21_18, p7_0  ,p23_16,g7_0  ,g23_16,  p11_0,p27_12,g11_0,g27_12, p19_0,g19_0;
wire   p4_4  ,p20_20,g4_4  ,g20_20, p5_4  ,p21_20,g5_4  ,g21_20, p6_3  ,p22_19,g6_3
,g22_19, p8_1  ,p24_17,g8_1  ,g24_17,  p12_0,p28_13,g12_0,g28_13, p20_0,g20_0;
wire   p5_5  ,p21_21,g5_5  ,g21_21, p6_5  ,p22_21,g6_5  ,g22_21, p7_4  ,p23_20,g7_4
,g23_20, p9_2  ,p25_18,g9_2  ,g25_18,  p13_0,p29_14,g13_0,g29_14, p21_0,g21_0;
wire   p6_6  ,p22_22,g6_6  ,g22_22, p7_6  ,p23_22,g7_6  ,g23_22, p8_5  ,p24_21,g8_5
,g24_21, p10_3 ,p26_19,g10_3 ,g26_19,  p14_0,p30_15,g14_0,g30_15, p22_0,g22_0;
wire   p7_7  ,p23_23,g7_7  ,g23_23, p8_7  ,p24_23,g8_7  ,g24_23, p9_6  ,p25_22,g9_6
,g25_22, p11_4 ,p27_20,g11_4 ,g27_20,  p15_0,p31_16,g15_0,g31_16, p23_0,g23_0;
wire   p8_8  ,p24_24,g8_8  ,g24_24, p9_8  ,p25_24,g9_8  ,g25_24, p10_7 ,p26_23,g10_7
,g26_23, p12_5 ,p28_21,g12_5 ,g28_21,  p16_1,       g16_1,        p24_0,g24_0;
wire   p9_9  ,p25_25,g9_9  ,g25_25, p10_9 ,p26_25,g10_9 ,g26_25, p11_8 ,p27_24,g11_8
,g27_24, p13_6 ,p29_22,g13_6 ,g29_22,  p17_2,       g17_2,        p25_0,g25_0;
wire   p10_10,p26_26,g10_10,g26_26, p11_10,p27_26,g11_10,g27_26, p12_9 ,p28_25,g12_9
,g28_25, p14_7 ,p30_23,g14_7 ,g30_23,  p18_3,       g18_3,        p26_0,g26_0;
wire   p11_11,p27_27,g11_11,g27_27, p12_11,p28_27,g12_11,g28_27,
p13_10,p29_26,g13_10,g29_26, p15_8 ,p31_24,g15_8 ,g31_24,  p19_4,       g19_4,
p27_0,g27_0;
wire   p12_12,p28_28,g12_12,g28_28, p13_12,p29_28,g13_12,g29_28,
p14_11,p30_27,g14_11,g30_27, p16_9 ,       g16_9 ,       p20_5,       g20_5,
p28_0,g28_0;
wire   p13_13,p29_29,g13_13,g29_29, p14_13,p30_29,g14_13,g30_29,
p15_12,p31_28,g15_12,g31_28, p17_10,       g17_10,       p21_6,       g21_6,
p29_0,g29_0;
wire   p14_14,p30_30,g14_14,g30_30, p15_14,p31_30,g15_14,g31_30, p16_13,       g16_13,
p18_11,       g18_11,       p22_7,       g22_7,        p30_0,g30_0;
wire   p15_15,p31_31,g15_15,g31_31, p16_15,       g16_15,       p17_14,       g17_14,
p19_12,       g19_12,       p23_8,       g23_8,        p31_0,g31_0;


assign x0  = x32 [0 ];
```

```
assign x1  = x32 [1 ];
assign x2  = x32 [2 ];
assign x3  = x32 [3 ];
assign x4  = x32 [4 ];
assign x5  = x32 [5 ];
assign x6  = x32 [6 ];
assign x7  = x32 [7 ];
assign x8  = x32 [8 ];
assign x9  = x32 [9 ];
assign x10 = x32 [10];
assign x11 = x32 [11];
assign x12 = x32 [12];
assign x13 = x32 [13];
assign x14 = x32 [14];
assign x15 = x32 [15];
assign x16 = x32 [16];
assign x17 = x32 [17];
assign x18 = x32 [18];
assign x19 = x32 [19];
assign x20 = x32 [20];
assign x21 = x32 [21];
assign x22 = x32 [22];
assign x23 = x32 [23];
assign x24 = x32 [24];
assign x25 = x32 [25];
assign x26 = x32 [26];
assign x27 = x32 [27];
assign x28 = x32 [28];
assign x29 = x32 [29];
assign x30 = x32 [30];
assign x31 = x32 [31];

assign y0  = y32 [0 ];
assign y1  = y32 [1 ];
assign y2  = y32 [2 ];
assign y3  = y32 [3 ];
assign y4  = y32 [4 ];
assign y5  = y32 [5 ];
assign y6  = y32 [6 ];
assign y7  = y32 [7 ];
assign y8  = y32 [8 ];
assign y9  = y32 [9 ];
assign y10 = y32 [10];
assign y11 = y32 [11];
assign y12 = y32 [12];
assign y13 = y32 [13];
assign y14 = y32 [14];
assign y15 = y32 [15];
assign y16 = y32 [16];
assign y17 = y32 [17];
assign y18 = y32 [18];
assign y19 = y32 [19];
assign y20 = y32 [20];
assign y21 = y32 [21];
assign y22 = y32 [22];
assign y23 = y32 [23];
assign y24 = y32 [24];
assign y25 = y32 [25];
assign y26 = y32 [26];
assign y27 = y32 [27];
assign y28 = y32 [28];
assign y29 = y32 [29];
assign y30 = y32 [30];
assign y31 = y32 [31];
```

```
// to stage 0

stage0 stage0 (
                .x0 (x0 ), .x16 (x16),.y0  (y0 ),.y16 (y16),.p0_0   (p0_0  ),.p16_16
(p16_16),.g0_0   (g0_0  ),.g16_16 (g16_16),
                .x1 (x1 ), .x17 (x17),.y1  (y1 ),.y17 (y17),.p1_1   (p1_1  ),.p17_17
(p17_17),.g1_1   (g1_1  ),.g17_17 (g17_17),
                .x2 (x2 ), .x18 (x18),.y2  (y2 ),.y18 (y18),.p2_2   (p2_2  ),.p18_18
(p18_18),.g2_2   (g2_2  ),.g18_18 (g18_18),
                .x3 (x3 ), .x19 (x19),.y3  (y3 ),.y19 (y19),.p3_3   (p3_3  ),.p19_19
(p19_19),.g3_3   (g3_3  ),.g19_19 (g19_19),
                .x4 (x4 ), .x20 (x20),.y4  (y4 ),.y20 (y20),.p4_4   (p4_4  ),.p20_20
(p20_20),.g4_4   (g4_4  ),.g20_20 (g20_20),
                .x5 (x5 ), .x21 (x21),.y5  (y5 ),.y21 (y21),.p5_5   (p5_5  ),.p21_21
(p21_21),.g5_5   (g5_5  ),.g21_21 (g21_21),
                .x6 (x6 ), .x22 (x22),.y6  (y6 ),.y22 (y22),.p6_6   (p6_6  ),.p22_22
(p22_22),.g6_6   (g6_6  ),.g22_22 (g22_22),
                .x7 (x7 ), .x23 (x23),.y7  (y7 ),.y23 (y23),.p7_7   (p7_7  ),.p23_23
(p23_23),.g7_7   (g7_7  ),.g23_23 (g23_23),
                .x8 (x8 ), .x24 (x24),.y8  (y8 ),.y24 (y24),.p8_8   (p8_8  ),.p24_24
(p24_24),.g8_8   (g8_8  ),.g24_24 (g24_24),
                .x9 (x9 ), .x25 (x25),.y9  (y9 ),.y25 (y25),.p9_9   (p9_9  ),.p25_25
(p25_25),.g9_9   (g9_9  ),.g25_25 (g25_25),
                .x10(x10), .x26 (x26),.y10 (y10),.y26 (y26),.p10_10 (p10_10),.p26_26
(p26_26),.g10_10 (g10_10),.g26_26 (g26_26),
                .x11(x11), .x27 (x27),.y11 (y11),.y27 (y27),.p11_11 (p11_11),.p27_27
(p27_27),.g11_11 (g11_11),.g27_27 (g27_27),
                .x12(x12), .x28 (x28),.y12 (y12),.y28 (y28),.p12_12 (p12_12),.p28_28
(p28_28),.g12_12 (g12_12),.g28_28 (g28_28),
                .x13(x13), .x29 (x29),.y13 (y13),.y29 (y29),.p13_13 (p13_13),.p29_29
(p29_29),.g13_13 (g13_13),.g29_29 (g29_29),
                .x14(x14), .x30 (x30),.y14 (y14),.y30 (y30),.p14_14 (p14_14),.p30_30
(p30_30),.g14_14 (g14_14),.g30_30 (g30_30),
                .x15(x15), .x31 (x31),.y15 (y15),.y31 (y31),.p15_15 (p15_15),.p31_31
(p31_31),.g15_15 (g15_15),.g31_31 (g31_31));

stage1 stage1 (
               .pi0 (p1_1  ),.pi16(p17_17),.pk0 (p0_0  ),.pk16(p16_16),.gi0 (g1_1
),.gi16(g17_17),.gk0 (g0_0  ),.gk16(g16_16),.p0_0  (p1_0  ),.p16_16(p17_16),.g0_0
(g1_0  ),.g16_16(g17_16),
               .pi1 (p2_2  ),.pi17(p18_18),.pk1 (p1_1  ),.pk17(p17_17),.gi1 (g2_2
),.gi17(g18_18),.gk1 (g1_1  ),.gk17(g17_17),.p1_1  (p2_1  ),.p17_17(p18_17),.g1_1
(g2_1  ),.g17_17(g18_17),
               .pi2 (p3_3  ),.pi18(p19_19),.pk2 (p2_2  ),.pk18(p18_18),.gi2 (g3_3
),.gi18(g19_19),.gk2 (g2_2  ),.gk18(g18_18),.p2_2  (p3_2  ),.p18_18(p19_18),.g2_2
(g3_2  ),.g18_18(g19_18),
               .pi3 (p4_4  ),.pi19(p20_20),.pk3 (p3_3  ),.pk19(p19_19),.gi3 (g4_4
),.gi19(g20_20),.gk3 (g3_3  ),.gk19(g19_19),.p3_3  (p4_3  ),.p19_19(p20_19),.g3_3
(g4_3  ),.g19_19(g20_19),
               .pi4 (p5_5  ),.pi20(p21_21),.pk4 (p4_4  ),.pk20(p20_20),.gi4 (g5_5
),.gi20(g21_21),.gk4 (g4_4  ),.gk20(g20_20),.p4_4  (p5_4  ),.p20_20(p21_20),.g4_4
(g5_4  ),.g20_20(g21_20),
               .pi5 (p6_6  ),.pi21(p22_22),.pk5 (p5_5  ),.pk21(p21_21),.gi5 (g6_6
),.gi21(g22_22),.gk5 (g5_5  ),.gk21(g21_21),.p5_5  (p6_5  ),.p21_21(p22_21),.g5_5
(g6_5  ),.g21_21(g22_21),
               .pi6 (p7_7  ),.pi22(p23_23),.pk6 (p6_6  ),.pk22(p22_22),.gi6 (g7_7
),.gi22(g23_23),.gk6 (g6_6  ),.gk22(g22_22),.p6_6  (p7_6  ),.p22_22(p23_22),.g6_6
(g7_6  ),.g22_22(g23_22),
               .pi7 (p8_8  ),.pi23(p24_24),.pk7 (p7_7  ),.pk23(p23_23),.gi7 (g8_8
```

```
),.gi23(g24_24),.gk7 (g7_7  ),.gk23(g23_23),.p7_7  (p8_7  ),.p23_23(p24_23),.g7_7
(g8_7  ),.g23_23(g24_23),
              .pi8 (p9_9  ),.pi24(p25_25),.pk8 (p8_8  ),.pk24(p24_24),.gi8 (g9_9
),.gi24(g25_25),.gk8 (g8_8  ),.gk24(g24_24),.p8_8  (p9_8  ),.p24_24(p25_24),.g8_8
(g9_8  ),.g24_24(g25_24),
              .pi9 (p10_10),.pi25(p26_26),.pk9 (p9_9  ),.pk25(p25_25),.gi9
(g10_10),.gi25(g26_26),.gk9 (g9_9  ),.gk25(g25_25),.p9_9  (p10_9
),.p25_25(p26_25),.g9_9  (g10_9 ),.g25_25(g26_25),

.pi10(p11_11),.pi26(p27_27),.pk10(p10_10),.pk26(p26_26),.gi10(g11_11),.gi26(g27_27),.gk
10(g10_10),.gk26(g26_26),.p10_10(p11_10),.p26_26(p27_26),.g10_10(g11_10),.g26_26(g27_26
),

.pi11(p12_12),.pi27(p28_28),.pk11(p11_11),.pk27(p27_27),.gi11(g12_12),.gi27(g28_28),.gk
11(g11_11),.gk27(g27_27),.p11_11(p12_11),.p27_27(p28_27),.g11_11(g12_11),.g27_27(g28_27
),

.pi12(p13_13),.pi28(p29_29),.pk12(p12_12),.pk28(p28_28),.gi12(g13_13),.gi28(g29_29),.gk
12(g12_12),.gk28(g28_28),.p12_12(p13_12),.p28_28(p29_28),.g12_12(g13_12),.g28_28(g29_28
),

.pi13(p14_14),.pi29(p30_30),.pk13(p13_13),.pk29(p29_29),.gi13(g14_14),.gi29(g30_30),.gk
13(g13_13),.gk29(g29_29),.p13_13(p14_13),.p29_29(p30_29),.g13_13(g14_13),.g29_29(g30_29
),

.pi14(p15_15),.pi30(p31_31),.pk14(p14_14),.pk30(p30_30),.gi14(g15_15),.gi30(g31_31),.gk
14(g14_14),.gk30(g30_30),.p14_14(p15_14),.p30_30(p31_30),.g14_14(g15_14),.g30_30(g31_30
),
              .pi15(p16_16),              .pk15(p15_15),              .gi15(g16_16),
.gk15(g15_15),              .p15_15(p16_15),                .g15_15(g16_15) );

stage2 stage2 (
              .pi0 (p2_1  ),.pi16(p18_17),.pk0 (p0_0  ),.pk16(p16_15),.gi0 (g2_1
),.gi16(g18_17),.gk0 (g0_0  ),.gk16(g16_15),.p0_0  (p2_0  ),.p16_16(p18_15),.g0_0
(g2_0  ),.g16_16(g18_15),
              .pi1 (p3_2  ),.pi17(p19_18),.pk1 (p1_0  ),.pk17(p17_16),.gi1 (g3_2
),.gi17(g19_18),.gk1 (g1_0  ),.gk17(g17_16),.p1_1  (p3_0  ),.p17_17(p19_16),.g1_1
(g3_0  ),.g17_17(g19_16),
              .pi2 (p4_3  ),.pi18(p20_19),.pk2 (p2_1  ),.pk18(p18_17),.gi2 (g4_3
),.gi18(g20_19),.gk2 (g2_1  ),.gk18(g18_17),.p2_2  (p4_1  ),.p18_18(p20_17),.g2_2
(g4_1  ),.g18_18(g20_17),
              .pi3 (p5_4  ),.pi19(p21_20),.pk3 (p3_2  ),.pk19(p19_18),.gi3 (g5_4
),.gi19(g21_20),.gk3 (g3_2  ),.gk19(g19_18),.p3_3  (p5_2  ),.p19_19(p21_18),.g3_3
(g5_2  ),.g19_19(g21_18),
              .pi4 (p6_5  ),.pi20(p22_21),.pk4 (p4_3  ),.pk20(p20_19),.gi4 (g6_5
),.gi20(g22_21),.gk4 (g4_3  ),.gk20(g20_19),.p4_4  (p6_3  ),.p20_20(p22_19),.g4_4
(g6_3  ),.g20_20(g22_19),
              .pi5 (p7_6  ),.pi21(p23_22),.pk5 (p5_4  ),.pk21(p21_20),.gi5 (g7_6
),.gi21(g23_22),.gk5 (g5_4  ),.gk21(g21_20),.p5_5  (p7_4  ),.p21_21(p23_20),.g5_5
(g7_4  ),.g21_21(g23_20),
              .pi6 (p8_7  ),.pi22(p24_23),.pk6 (p6_5  ),.pk22(p22_21),.gi6 (g8_7
),.gi22(g24_23),.gk6 (g6_5  ),.gk22(g22_21),.p6_6  (p8_5  ),.p22_22(p24_21),.g6_6
(g8_5  ),.g22_22(g24_21),
              .pi7 (p9_8  ),.pi23(p25_24),.pk7 (p7_6  ),.pk23(p23_22),.gi7 (g9_8
),.gi23(g25_24),.gk7 (g7_6  ),.gk23(g23_22),.p7_7  (p9_6  ),.p23_23(p25_22),.g7_7
(g9_6  ),.g23_23(g25_22),
              .pi8 (p10_9 ),.pi24(p26_25),.pk8 (p8_7  ),.pk24(p24_23),.gi8 (g10_9
),.gi24(g26_25),.gk8 (g8_7  ),.gk24(g24_23),.p8_8  (p10_7 ),.p24_24(p26_23),.g8_8
(g10_7 ),.g24_24(g26_23),
              .pi9 (p11_10),.pi25(p27_26),.pk9 (p9_8  ),.pk25(p25_24),.gi9
(g11_10),.gi25(g27_26),.gk9 (g9_8  ),.gk25(g25_24),.p9_9  (p11_8
),.p25_25(p27_24),.g9_9  (g11_8 ),.g25_25(g27_24),
              .pi10(p12_11),.pi26(p28_27),.pk10(p10_9
),.pk26(p26_25),.gi10(g12_11),.gi26(g28_27),.gk10(g10_9 ),.gk26(g26_25),.p10_10(p12_9
),.p26_26(p28_25),.g10_10(g12_9 ),.g26_26(g28_25),

.pi11(p13_12),.pi27(p29_28),.pk11(p11_10),.pk27(p27_26),.gi11(g13_12),.gi27(g29_28),.gk
```

```
11(g11_10),.gk27(g27_26),.p11_11(p13_10),.p27_27(p29_26),.g11_11(g13_10),.g27_27(g29_26
),

.pi12(p14_13),.pi28(p30_29),.pk12(p12_11),.pk28(p28_27),.gi12(g14_13),.gi28(g30_29),.gk
12(g12_11),.gk28(g28_27),.p12_12(p14_11),.p28_28(p30_27),.g12_12(g14_11),.g28_28(g30_27
),

.pi13(p15_14),.pi29(p31_30),.pk13(p13_12),.pk29(p29_28),.gi13(g15_14),.gi29(g31_30),.gk
13(g13_12),.gk29(g29_28),.p13_13(p15_12),.p29_29(p31_28),.g13_13(g15_12),.g29_29(g31_28
),
              .pi14(p16_15),              .pk14(p14_13),              .gi14(g16_15),
.gk14(g14_13),              .p14_14(p16_13),                .g14_14(g16_13),
              .pi15(p17_16),              .pk15(p15_14),              .gi15(g17_16),
.gk15(g15_14),              .p15_15(p17_14),                .g15_15(g17_14) );

stage3 stage3 (
              .pi0 (p4_1  ),.pi16(p20_17),.pk0 (p0_0  ),.pk16(p16_13),.gi0 (g4_1
),.gi16(g20_17),.gk0 (g0_0  ),.gk16(g16_13),.p0_0  (p4_0  ),.p16_16(p20_13),.g0_0
(g4_0  ),.g16_16(g20_13),
              .pi1 (p5_2  ),.pi17(p21_18),.pk1 (p1_0  ),.pk17(p17_14),.gi1 (g5_2
),.gi17(g21_18),.gk1 (g1_0  ),.gk17(g17_14),.p1_1  (p5_0  ),.p17_17(p21_14),.g1_1
(g5_0  ),.g17_17(g21_14),
              .pi2 (p6_3  ),.pi18(p22_19),.pk2 (p2_0  ),.pk18(p18_15),.gi2 (g6_3
),.gi18(g22_19),.gk2 (g2_0  ),.gk18(g18_15),.p2_2  (p6_0  ),.p18_18(p22_15),.g2_2
(g6_0  ),.g18_18(g22_15),
              .pi3 (p7_4  ),.pi19(p23_20),.pk3 (p3_0  ),.pk19(p19_16),.gi3 (g7_4
),.gi19(g23_20),.gk3 (g3_0  ),.gk19(g19_16),.p3_3  (p7_0  ),.p19_19(p23_16),.g3_3
(g7_0  ),.g19_19(g23_16),
              .pi4 (p8_5  ),.pi20(p24_21),.pk4 (p4_1  ),.pk20(p20_17),.gi4 (g8_5
),.gi20(g24_21),.gk4 (g4_1  ),.gk20(g20_17),.p4_4  (p8_1  ),.p20_20(p24_17),.g4_4
(g8_1  ),.g20_20(g24_17),
              .pi5 (p9_6  ),.pi21(p25_22),.pk5 (p5_2  ),.pk21(p21_18),.gi5 (g9_6
),.gi21(g25_22),.gk5 (g5_2  ),.gk21(g21_18),.p5_5  (p9_2  ),.p21_21(p25_18),.g5_5
(g9_2  ),.g21_21(g25_18),
              .pi6 (p10_7 ),.pi22(p26_23),.pk6 (p6_3  ),.pk22(p22_19),.gi6 (g10_7
),.gi22(g26_23),.gk6 (g6_3  ),.gk22(g22_19),.p6_6  (p10_3 ),.p22_22(p26_19),.g6_6
(g10_3 ),.g22_22(g26_19),
              .pi7 (p11_8 ),.pi23(p27_24),.pk7 (p7_4  ),.pk23(p23_20),.gi7 (g11_8
),.gi23(g27_24),.gk7 (g7_4  ),.gk23(g23_20),.p7_7  (p11_4 ),.p23_23(p27_20),.g7_7
(g11_4 ),.g23_23(g27_20),
              .pi8 (p12_9 ),.pi24(p28_25),.pk8 (p8_5  ),.pk24(p24_21),.gi8 (g12_9
),.gi24(g28_25),.gk8 (g8_5  ),.gk24(g24_21),.p8_8  (p12_5 ),.p24_24(p28_21),.g8_8
(g12_5 ),.g24_24(g28_21),
              .pi9 (p13_10),.pi25(p29_26),.pk9 (p9_6  ),.pk25(p25_22),.gi9
(g13_10),.gi25(g29_26),.gk9 (g9_6  ),.gk25(g25_22),.p9_9  (p13_6
),.p25_25(p29_22),.g9_9  (g13_6 ),.g25_25(g29_22),
              .pi10(p14_11),.pi26(p30_27),.pk10(p10_7
),.pk26(p26_23),.gi10(g14_11),.gi26(g30_27),.gk10(g10_7 ),.gk26(g26_23),.p10_10(p14_7
),.p26_26(p30_23),.g10_10(g14_7 ),.g26_26(g30_23),
              .pi11(p15_12),.pi27(p31_28),.pk11(p11_8
),.pk27(p27_24),.gi11(g15_12),.gi27(g31_28),.gk11(g11_8 ),.gk27(g27_24),.p11_11(p15_8
),.p27_27(p31_24),.g11_11(g15_8 ),.g27_27(g31_24),
              .pi12(p16_13),              .pk12(p12_9 ),              .gi12(g16_13),
.gk12(g12_9 ),              .p12_12(p16_9 ),                .g12_12(g16_9 ),
              .pi13(p17_14),              .pk13(p13_10),              .gi13(g17_14),
.gk13(g13_10),              .p13_13(p17_10),                .g13_13(g17_10),
              .pi14(p18_15),              .pk14(p14_11),              .gi14(g18_15),
.gk14(g14_11),              .p14_14(p18_11),                .g14_14(g18_11),
              .pi15(p19_16),              .pk15(p15_12),              .gi15(g19_16),
.gk15(g15_12),              .p15_15(p19_12),                .g15_15(g19_12) );

stage4 stage4 (
              .pi0 (p8_1  ),.pi16(p24_17),.pk0 (p0_0 ),.pk16(p16_9 ),.gi0 (g8_1
),.gi16(g24_17),.gk0 (g0_0 ),.gk16(g16_9 ),.p0_0  (p8_0 ),.p16_16(p24_9 ),.g0_0  (g8_0
),.g16_16(g24_9 ),
              .pi1 (p9_2  ),.pi17(p25_18),.pk1 (p1_0 ),.pk17(p17_10),.gi1 (g9_2
),.gi17(g25_18),.gk1 (g1_0 ),.gk17(g17_10),.p1_1  (p9_0 ),.p17_17(p25_10),.g1_1  (g9_0
```

```
),.g17_17(g25_10),
              .pi2 (p10_3 ),.pi18(p26_19),.pk2 (p2_0 ),.pk18(p18_11),.gi2 (g10_3
),.gi18(g26_19),.gk2 (g2_0 ),.gk18(g18_11),.p2_2  (p10_0),.p18_18(p26_11),.g2_2
(g10_0),.g18_18(g26_11),
              .pi3 (p11_4 ),.pi19(p27_20),.pk3 (p3_0 ),.pk19(p19_12),.gi3 (g11_4
),.gi19(g27_20),.gk3 (g3_0 ),.gk19(g19_12),.p3_3  (p11_0),.p19_19(p27_12),.g3_3
(g11_0),.g19_19(g27_12),
              .pi4 (p12_5 ),.pi20(p28_21),.pk4 (p4_0 ),.pk20(p20_13),.gi4 (g12_5
),.gi20(g28_21),.gk4 (g4_0 ),.gk20(g20_13),.p4_4  (p12_0),.p20_20(p28_13),.g4_4
(g12_0),.g20_20(g28_13),
              .pi5 (p13_6 ),.pi21(p29_22),.pk5 (p5_0 ),.pk21(p21_14),.gi5 (g13_6
),.gi21(g29_22),.gk5 (g5_0 ),.gk21(g21_14),.p5_5  (p13_0),.p21_21(p29_14),.g5_5
(g13_0),.g21_21(g29_14),
              .pi6 (p14_7 ),.pi22(p30_23),.pk6 (p6_0 ),.pk22(p22_15),.gi6 (g14_7
),.gi22(g30_23),.gk6 (g6_0 ),.gk22(g22_15),.p6_6  (p14_0),.p22_22(p30_15),.g6_6
(g14_0),.g22_22(g30_15),
              .pi7 (p15_8 ),.pi23(p31_24),.pk7 (p7_0 ),.pk23(p23_16),.gi7 (g15_8
),.gi23(g31_24),.gk7 (g7_0 ),.gk23(g23_16),.p7_7  (p15_0),.p23_23(p31_16),.g7_7
(g15_0),.g23_23(g31_16),
              .pi8 (p16_9 ),              .pk8 (p8_1 ),              .gi8 (g16_9 ),
.gk8 (g8_1 ),              .p8_8  (p16_1),              .g8_8  (g16_1),
              .pi9 (p17_10),              .pk9 (p9_2 ),              .gi9 (g17_10),
.gk9 (g9_2 ),              .p9_9  (p17_2),              .g9_9  (g17_2),
              .pi10(p18_11),              .pk10(p10_3),              .gi10(g18_11),
.gk10(g10_3),              .p10_10(p18_3),              .g10_10(g18_3),
              .pi11(p19_12),              .pk11(p11_4),              .gi11(g19_12),
.gk11(g11_4),              .p11_11(p19_4),              .g11_11(g19_4),
              .pi12(p20_13),              .pk12(p12_5),              .gi12(g20_13),
.gk12(g12_5),              .p12_12(p20_5),              .g12_12(g20_5),
              .pi13(p21_14),              .pk13(p13_6),              .gi13(g21_14),
.gk13(g13_6),              .p13_13(p21_6),              .g13_13(g21_6),
              .pi14(p22_15),              .pk14(p14_7),              .gi14(g22_15),
.gk14(g14_7),              .p14_14(p22_7),              .g14_14(g22_7),
              .pi15(p23_16),              .pk15(p15_8),              .gi15(g23_16),
.gk15(g15_8),              .p15_15(p23_8),              .g15_15(g23_8) );

stage5 stage5 (
              .pi0 (p16_1 ),              .pk0 (p0_0 ),              .gi0 (g16_1 ),
.gk0 (g0_0 ),              .p0_0  (p16_0),              .g0_0  (g16_0),
              .pi1 (p17_2 ),              .pk1 (p1_0 ),              .gi1 (g17_2 ),
.gk1 (g1_0 ),              .p1_1  (p17_0),              .g1_1  (g17_0),
              .pi2 (p18_3 ),              .pk2 (p2_0 ),              .gi2 (g18_3 ),
.gk2 (g2_0 ),              .p2_2  (p18_0),              .g2_2  (g18_0),
              .pi3 (p19_4 ),              .pk3 (p3_0 ),              .gi3 (g19_4 ),
.gk3 (g3_0 ),              .p3_3  (p19_0),              .g3_3  (g19_0),
              .pi4 (p20_5 ),              .pk4 (p4_0 ),              .gi4 (g20_5 ),
.gk4 (g4_0 ),              .p4_4  (p20_0),              .g4_4  (g20_0),
              .pi5 (p21_6 ),              .pk5 (p5_0 ),              .gi5 (g21_6 ),
.gk5 (g5_0 ),              .p5_5  (p21_0),              .g5_5  (g21_0),
              .pi6 (p22_7 ),              .pk6 (p6_0 ),              .gi6 (g22_7 ),
.gk6 (g6_0 ),              .p6_6  (p22_0),              .g6_6  (g22_0),
              .pi7 (p23_8 ),              .pk7 (p7_0 ),              .gi7 (g23_8 ),
.gk7 (g7_0 ),              .p7_7  (p23_0),              .g7_7  (g23_0),
              .pi8 (p24_9 ),              .pk8 (p8_0 ),              .gi8 (g24_9 ),
.gk8 (g8_0 ),              .p8_8  (p24_0),              .g8_8  (g24_0),
              .pi9 (p25_10),              .pk9 (p9_0 ),              .gi9 (g25_10),
.gk9 (g9_0 ),              .p9_9  (p25_0),              .g9_9  (g25_0),
              .pi10(p26_11),              .pk10(p10_0),              .gi10(g26_11),
.gk10(g10_0),              .p10_10(p26_0),              .g10_10(g26_0),
              .pi11(p27_12),              .pk11(p11_0),              .gi11(g27_12),
.gk11(g11_0),              .p11_11(p27_0),              .g11_11(g27_0),
              .pi12(p28_13),              .pk12(p12_0),              .gi12(g28_13),
.gk12(g12_0),              .p12_12(p28_0),              .g12_12(g28_0),
              .pi13(p29_14),              .pk13(p13_0),              .gi13(g29_14),
.gk13(g13_0),              .p13_13(p29_0),              .g13_13(g29_0),
              .pi14(p30_15),              .pk14(p14_0),              .gi14(g30_15),
.gk14(g14_0),              .p14_14(p30_0),              .g14_14(g30_0),
```

```
              .pi15(p31_16),              .pk15(p15_0),              .gi15(g31_16),
.gk15(g15_0),              .p15_15(p31_0),                .g15_15(g31_0) );
// stage 1
ksOpGray ksOpGray1 (
         .Gc (cin),
         .P  (p0_0),
         .Gg (g0_0),
         .G  (G0) );
//stage 2
ksOpGray ksOpGray2 (
         .Gc (cin),
         .P  (p1_0),
         .Gg (g1_0),
         .G  (G1) );

ksOpGray ksOpGray3 (
         .Gc (G0),
         .P  (p2_0),
         .Gg (g2_0),
         .G  (G2) );
//stage 3
ksOpGray ksOpGray4 (
         .Gc (cin),
         .P  (p3_0),
         .Gg (g3_0),
         .G  (G3) );

ksOpGray ksOpGray5 (
         .Gc (G0),
         .P  (p4_0),
         .Gg (g4_0),
         .G  (G4) );

ksOpGray ksOpGray6 (
         .Gc (G1),
         .P  (p5_0),
         .Gg (g5_0),
         .G  (G5) );

ksOpGray ksOpGray7 (
         .Gc (G2),
         .P  (p6_0),
         .Gg (g6_0),
         .G  (G6) );

// stage 4
ksOpGray ksOpGray8 (
         .Gc (cin),
         .P  (p7_0),
         .Gg (g7_0),
         .G  (G7) );

ksOpGray ksOpGray9 (
         .Gc (G0),
         .P  (p8_0),
         .Gg (g8_0),
         .G  (G8) );

ksOpGray ksOpGray10 (
         .Gc (G1),
         .P  (p9_0),
         .Gg (g9_0),
         .G  (G9) );

ksOpGray ksOpGray11 (
         .Gc (G2),
         .P  (p10_0),
```

```
          .Gg (g10_0),
          .G  (G10) );

ksOpGray ksOpGray12 (
          .Gc (G3),
          .P  (p11_0),
          .Gg (g11_0),
          .G  (G11) );

ksOpGray ksOpGray13 (
          .Gc (G4),
          .P  (p12_0),
          .Gg (g12_0),
          .G  (G12) );

ksOpGray ksOpGray14 (
          .Gc (G5),
          .P  (p13_0),
          .Gg (g13_0),
          .G  (G13) );

ksOpGray ksOpGray15 (
          .Gc (G6),
          .P  (p14_0),
          .Gg (g14_0),
          .G  (G14) );

// stage 5
ksOpGray ksOpGray16 (
          .Gc (cin),
          .P  (p15_0),
          .Gg (g15_0),
          .G  (G15) );

ksOpGray ksOpGray17 (
          .Gc (G0),
          .P  (p16_0),
          .Gg (g16_0),
          .G  (G16) );

ksOpGray ksOpGray18 (
          .Gc (G1),
          .P  (p17_0),
          .Gg (g17_0),
          .G  (G17) );

ksOpGray ksOpGray19 (
          .Gc (G2),
          .P  (p18_0),
          .Gg (g18_0),
          .G  (G18) );

ksOpGray ksOpGray20 (
          .Gc (G3),
          .P  (p19_0),
          .Gg (g19_0),
          .G  (G19) );

ksOpGray ksOpGray21 (
          .Gc (G4),
          .P  (p20_0),
          .Gg (g20_0),
          .G  (G20) );

ksOpGray ksOpGray22 (
          .Gc (G5),
          .P  (p21_0),
```

```
          .Gg (g21_0),
          .G  (G21) );

ksOpGray ksOpGray23 (
          .Gc (G6),
          .P  (p22_0),
          .Gg (g22_0),
          .G  (G22) );

ksOpGray ksOpGray24 (
          .Gc (G7),
          .P  (p23_0),
          .Gg (g23_0),
          .G  (G23) );

ksOpGray ksOpGray25 (
          .Gc (G8),
          .P  (p24_0),
          .Gg (g24_0),
          .G  (G24) );

ksOpGray ksOpGray26 (
          .Gc (G9),
          .P  (p25_0),
          .Gg (g25_0),
          .G  (G25) );

ksOpGray ksOpGray27 (
          .Gc (G10),
          .P  (p26_0),
          .Gg (g26_0),
          .G  (G26) );

ksOpGray ksOpGray28 (
          .Gc (G11),
          .P  (p27_0),
          .Gg (g27_0),
          .G  (G27) );

ksOpGray ksOpGray29 (
          .Gc (G12),
          .P  (p28_0),
          .Gg (g28_0),
          .G  (G28) );

ksOpGray ksOpGray30 (
          .Gc (G13),
          .P  (p29_0),
          .Gg (g29_0),
          .G  (G29) );

ksOpGray ksOpGray31 (
          .Gc (G14),
          .P  (p30_0),
          .Gg (g30_0),
          .G  (G30) );


assign s0  =cin^p0_0  ;
assign s1  =G0^p1_1  ;
assign s2  =G1^p2_2  ;
assign s3  =G2^p3_3  ;
assign s4  =G3^p4_4  ;
assign s5  =G4^p5_5  ;
assign s6  =G5^p6_6  ;
assign s7  =G6^p7_7  ;
assign s8  =G7^p8_8  ;
```

```
assign s9  =G8^p9_9  ;
assign s10 =G9^p10_10;
assign s11 =G10^p11_11;
assign s12 =G11^p12_12;
assign s13 =G12^p13_13;
assign s14 =G13^p14_14;
assign s15 =G14^p15_15;
assign s16 =G15^p16_16;
assign s17 =G16^p17_17;
assign s18 =G17^p18_18;
assign s19 =G18^p19_19;
assign s20 =G19^p20_20;
assign s21 =G20^p21_21;
assign s22 =G21^p22_22;
assign s23 =G22^p23_23;
assign s24 =G23^p24_24;
assign s25 =G24^p25_25;
assign s26 =G25^p26_26;
assign s27 =G26^p27_27;
assign s28 =G27^p28_28;
assign s29 =G28^p29_29;
assign s30 =G29^p30_30;
assign s31 =G30^p31_31;

assign s32 =
{s31,s30,s29,s28,s27,s26,s25,s24,s23,s22,s21,s20,s19,s18,s17,s16,s15,s14,s13,s12,s11,s1
0,s9,s8,s7,s6,s5,s4,s3,s2,s1,s0};
endmodule

/*
 * alhamdulillah
 */
```

Listing 27 textSeg.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : textSeg.v
-- TYPE      : rtl
-- FUNCTION  : ROM (instruction memory)
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/16
-----------------------------------------------*/

// starting address : 0x0040_0000

module textSeg (
   instAddr , // input
   instIn     // output
   );

 // input ports
 input wire [6:0]  instAddr ;

 // output ports
 output reg [31:0] instIn  ;

 // internal variables
 reg [31:0] ROM [0:107];

 // initialize RAM
```

```
 initial
  begin

    // SOKO
    ROM[0]  = 32'h20081f11;
    ROM[1]  = 32'h40886000;
    ROM[2]  = 32'h20080070;
    ROM[3]  = 32'h20110001;
    ROM[4]  = 32'h20120002;
    ROM[5]  = 32'h20130003;
    ROM[6]  = 32'h20140004;
    ROM[7]  = 32'h20150005;
    ROM[8]  = 32'h20160006;
    ROM[9]  = 32'h2010000b;
    ROM[10] = 32'h0810000a;

  end

 // read operation
 always @ (instAddr)
  begin
    instIn <= #10 ROM[instAddr];
  end

endmodule

//
// alhamdulillah
//
```

Listing 28 interSeg.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : instMemory.v
-- TYPE      : rtl
-- FUNCTION  : ROM (instruction memory)
-- edit      : -
-- Author    : iprayudi
-- Rev,Date  : 08/10/16
-----------------------------------------------*/

// starting address : 0x8000_0180

module interSeg (
   instAddr , // input
   instIn     // output
   );

 // input ports
 input wire [13:0]  instAddr ;

 // output ports
 output reg [31:0] instIn  ;

 // internal variables
 reg [31:0] ROM [0:255];

 // initialize RAM
 initial
  begin

    // SOKO
    ROM[0+96]   = 32'h0020d820;
    ROM[1+96]   = 32'h401a6800;
```

```
    ROM[2+96]   = 32'h001a2082;
    ROM[3+96]   = 32'h3084000f;
    ROM[4+96]   = 32'h0204b82a;
    ROM[5+96]   = 32'h12f10076;
    ROM[6+96]   = 32'h00804820;
    ROM[7+96]   = 32'h3c011001;
    ROM[8+96]   = 32'h00280821;
    ROM[9+96]   = 32'h8c390000;
    ROM[10+96]  = 32'h11310003;
    ROM[11+96]  = 32'h1132000e;
    ROM[12+96]  = 32'h11330019;
    ROM[13+96]  = 32'h11340024;
    ROM[14+96]  = 32'h210affd0;
    ROM[15+96]  = 32'h3c011001;
    ROM[16+96]  = 32'h002a0821;
    ROM[17+96]  = 32'h8c2b0000;
    ROM[18+96]  = 32'h11720002;
    ROM[19+96]  = 32'h11730001;
    ROM[20+96]  = 32'h10000029;
    ROM[21+96]  = 32'h210cffa0;
    ROM[22+96]  = 32'h3c011001;
    ROM[23+96]  = 32'h002c0821;
    ROM[24+96]  = 32'h8c2d0000;
    ROM[25+96]  = 32'h10000024;
    ROM[26+96]  = 32'h210a0030;
    ROM[27+96]  = 32'h3c011001;
    ROM[28+96]  = 32'h002a0821;
    ROM[29+96]  = 32'h8c2b0000;
    ROM[30+96]  = 32'h11720002;
    ROM[31+96]  = 32'h11730001;
    ROM[32+96]  = 32'h1000001d;
    ROM[33+96]  = 32'h210c0060;
    ROM[34+96]  = 32'h3c011001;
    ROM[35+96]  = 32'h002c0821;
    ROM[36+96]  = 32'h8c2d0000;
    ROM[37+96]  = 32'h10000018;
    ROM[38+96]  = 32'h210afffc;
    ROM[39+96]  = 32'h3c011001;
    ROM[40+96]  = 32'h002a0821;
    ROM[41+96]  = 32'h8c2b0000;
    ROM[42+96]  = 32'h11720002;
    ROM[43+96]  = 32'h11730001;
    ROM[44+96]  = 32'h10000011;
    ROM[45+96]  = 32'h210cfff8;
    ROM[46+96]  = 32'h3c011001;
    ROM[47+96]  = 32'h002c0821;
    ROM[48+96]  = 32'h8c2d0000;
    ROM[49+96]  = 32'h1000000c;
    ROM[50+96]  = 32'h210a0004;
    ROM[51+96]  = 32'h3c011001;
    ROM[52+96]  = 32'h002a0821;
    ROM[53+96]  = 32'h8c2b0000;
    ROM[54+96]  = 32'h11720002;
    ROM[55+96]  = 32'h11730001;
    ROM[56+96]  = 32'h10000005;
    ROM[57+96]  = 32'h210c0008;
    ROM[58+96]  = 32'h3c011001;
    ROM[59+96]  = 32'h002c0821;
    ROM[60+96]  = 32'h8c2d0000;
    ROM[61+96]  = 32'h10000000;
    ROM[62+96]  = 32'h11600004;
    ROM[63+96]  = 32'h1171000b;
    ROM[64+96]  = 32'h11720012;
    ROM[65+96]  = 32'h11730011;
    ROM[66+96]  = 32'h10000037;
    ROM[67+96]  = 32'h3c011001;
    ROM[68+96]  = 32'h002a0821;
```

```
    ROM[69+96]  = 32'hac360000;
    ROM[70+96]  = 32'h1335002f;
    ROM[71+96]  = 32'h3c011001;
    ROM[72+96]  = 32'h00280821;
    ROM[73+96]  = 32'hac200000;
    ROM[74+96]  = 32'h1000002e;
    ROM[75+96]  = 32'h3c011001;
    ROM[76+96]  = 32'h002a0821;
    ROM[77+96]  = 32'hac350000;
    ROM[78+96]  = 32'h13350027;
    ROM[79+96]  = 32'h3c011001;
    ROM[80+96]  = 32'h00280821;
    ROM[81+96]  = 32'hac200000;
    ROM[82+96]  = 32'h10000026;
    ROM[83+96]  = 32'h11a00002;
    ROM[84+96]  = 32'h11b10011;
    ROM[85+96]  = 32'h10000024;
    ROM[86+96]  = 32'h3c011001;
    ROM[87+96]  = 32'h002c0821;
    ROM[88+96]  = 32'hac320000;
    ROM[89+96]  = 32'h11730004;
    ROM[90+96]  = 32'h3c011001;
    ROM[91+96]  = 32'h002a0821;
    ROM[92+96]  = 32'hac360000;
    ROM[93+96]  = 32'h10000003;
    ROM[94+96]  = 32'h3c011001;
    ROM[95+96]  = 32'h002a0821;
    ROM[96+96]  = 32'hac350000;
    ROM[97+96]  = 32'h13350014;
    ROM[98+96]  = 32'h3c011001;
    ROM[99+96]  = 32'h00280821;
    ROM[100+96] = 32'hac200000;
    ROM[101+96] = 32'h10000013;
    ROM[102+96] = 32'h3c011001;
    ROM[103+96] = 32'h002c0821;
    ROM[104+96] = 32'hac330000;
    ROM[105+96] = 32'h11730004;
    ROM[106+96] = 32'h3c011001;
    ROM[107+96] = 32'h002a0821;
    ROM[108+96] = 32'hac360000;
    ROM[109+96] = 32'h10000003;
    ROM[110+96] = 32'h3c011001;
    ROM[111+96] = 32'h002a0821;
    ROM[112+96] = 32'hac350000;
    ROM[113+96] = 32'h13350004;
    ROM[114+96] = 32'h3c011001;
    ROM[115+96] = 32'h00280821;
    ROM[116+96] = 32'hac200000;
    ROM[117+96] = 32'h10000003;
    ROM[118+96] = 32'h3c011001;
    ROM[119+96] = 32'h00280821;
    ROM[120+96] = 32'hac310000;
    ROM[121+96] = 32'h000a4021;
    ROM[122+96] = 32'h401a7000;
    ROM[123+96] = 32'h10000003;
    ROM[124+96] = 32'h401a7000;
    ROM[125+96] = 32'h275a0004;
    ROM[126+96] = 32'h409a7000;
    ROM[127+96] = 32'h40806800;
    ROM[128+96] = 32'h401a6000;
    ROM[129+96] = 32'h201a1f11;
    ROM[130+96] = 32'h409a6000;
    ROM[131+96] = 32'h03600820;
    ROM[132+96] = 32'h42000018;

  end
```

```
 // read operation
 always @ (instAddr)
  begin
   instIn <= #10 ROM[instAddr];
  end

endmodule

//
// alhamdulillah
//
```

Listing 29 instMemory.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : instMemory.v
 * type      : rtl
 * function  : RAM (data memory)
 * edit      : -
 * author    : iprayudi
 * rev. date : 20090102 - created
 */

module instMemory (
   instAddr , // input
   instIn     // output
   );

 // input ports
 input wire [31:0] instAddr ;

 // output ports
 output wire [31:0] instIn  ;

 /* internal variables */
 wire        ce_a         ;
 wire        ce_b         ;

 wire [31:0] instIn_text ;
 wire [31:0] instIn_inter ;

 // chip enable
 assign ce_a = (instAddr[31:16] == 16'h0040) ? 1'b1 : 1'b0 ;
 assign ce_b = (instAddr[31:16] == 16'h8000) ? 1'b1 : 1'b0 ;

 // output behavior
 assign instIn = (ce_a) ? instIn_text  :
                 (ce_b) ? instIn_inter : 32'd0;

textSeg textSeg (
   .instAddr (instAddr[8:2] ), // input
   .instIn   (instIn_text   )  // output
   );

interSeg interSeg (
   .instAddr (instAddr[15:2] ), // input
   .instIn   (instIn_inter  )  // output
   );

endmodule

/*
 * alhamdulillah
```

```
 */
```

Listing 30 dataMemory.v

```
/*
 * bismillahirrahmanirrahim
 * ------------------------
 * filename  : dataMemory.v
 * type      : rtl
 * function  : RAM (data memory)
 * edit      : -
 * author    : iprayudi - rhw
 * rev. date : 20081016 - created
 *             20081226 - data entered
 */

module dataMemory (
   clk         , // input
   dMemWEn     , // input
   dMemWData   , // input
   dMemWRAddr0 , // input
   dMemRData0  , // output

   dMemWRAddr1 , // input
   dMemRData1    // output
   );

 /* input ports */
 input wire        clk        ;
 input wire        dMemWEn    ;
 input wire [31:0] dMemWData  ;
 input wire [6:0]  dMemWRAddr0 ;
 input wire [6:0]  dMemWRAddr1 ;

 /* output ports */
 output reg [31:0] dMemRData0  ;
 output reg [31:0] dMemRData1  ;

 /* internal variables */
 reg [31:0] RAM [0:107];

 /* initialize RAM */
 initial
  begin

  // bubble sort
//    RAM[0]  = 32'd0;
//    RAM[1]  = 32'd1;
//    RAM[2]  = 32'd2;
//    RAM[3]  = 32'd3;
//    RAM[4]  = 32'd4;
//    RAM[5]  = 32'd5;
//    RAM[6]  = 32'd6;
//    RAM[7]  = 32'd7;
//    RAM[32] = 32'h10010000;
//    RAM[33] = 32'd32;
//    RAM[34] = 32'd4;

  // soko
      RAM[0]  =32'h04;
      RAM[1]  =32'h04;
      RAM[2]  =32'h04;
      RAM[3]  =32'h04;
      RAM[4]  =32'h04;
      RAM[5]  =32'h04;
```

```
        RAM[6]  =32'h04;
        RAM[7]  =32'h04;
        RAM[8]  =32'h04;
        RAM[9]  =32'h04;
        RAM[10] =32'h07;
        RAM[11] =32'h07;
        RAM[12] =32'h04;
        RAM[13] =32'h00;
        RAM[14] =32'h00;
        RAM[15] =32'h00;
        RAM[16] =32'h00;
        RAM[17] =32'h00;
        RAM[18] =32'h00;
        RAM[19] =32'h00;
        RAM[20] =32'h00;
        RAM[21] =32'h04;
        RAM[22] =32'h07;
        RAM[23] =32'h07;
        RAM[24] =32'h04;
        RAM[25] =32'h00;
        RAM[26] =32'h00;
        RAM[27] =32'h02;
        RAM[28] =32'h06; // initial character
        RAM[29] =32'h02;
        RAM[30] =32'h00;
        RAM[31] =32'h02;
        RAM[32] =32'h00;
        RAM[33] =32'h04;
        RAM[34] =32'h07;
        RAM[35] =32'h07;
        RAM[36] =32'h04;
        RAM[37] =32'h04;
        RAM[38] =32'h00;
        RAM[39] =32'h04;
        RAM[40] =32'h04;
        RAM[41] =32'h04;
        RAM[42] =32'h04;
        RAM[43] =32'h04;
        RAM[44] =32'h00;
        RAM[45] =32'h04;
        RAM[46] =32'h07;
        RAM[47] =32'h07;
        RAM[48] =32'h04;
        RAM[49] =32'h00;
        RAM[50] =32'h00;
        RAM[51] =32'h00;
        RAM[52] =32'h01;
        RAM[53] =32'h00;
        RAM[54] =32'h01;
        RAM[55] =32'h00;
        RAM[56] =32'h01;
        RAM[57] =32'h04;
        RAM[58] =32'h07;
        RAM[59] =32'h07;
        RAM[60] =32'h04;
        RAM[61] =32'h00;
        RAM[62] =32'h00;
        RAM[63] =32'h00;
        RAM[64] =32'h00;
        RAM[65] =32'h00;
        RAM[66] =32'h00;
        RAM[67] =32'h00;
        RAM[68] =32'h00;
        RAM[69] =32'h04;
        RAM[70] =32'h07;
        RAM[71] =32'h07;
        RAM[72] =32'h04;
```

```
      RAM[73] =32'h04;
      RAM[74] =32'h04;
      RAM[75] =32'h04;
      RAM[76] =32'h04;
      RAM[77] =32'h04;
      RAM[78] =32'h04;
      RAM[79] =32'h04;
      RAM[80] =32'h04;
      RAM[81] =32'h04;
      RAM[82] =32'h07;
      RAM[83] =32'h07;
      RAM[84] =32'h07;
      RAM[85] =32'h07;
      RAM[86] =32'h07;
      RAM[87] =32'h07;
      RAM[88] =32'h07;
      RAM[89] =32'h07;
      RAM[90] =32'h07;
      RAM[91] =32'h07;
      RAM[92] =32'h07;
      RAM[93] =32'h07;
      RAM[94] =32'h07;
      RAM[95] =32'h07;
      RAM[96] =32'h07;
      RAM[97] =32'h07;
      RAM[98] =32'h07;
      RAM[99] =32'h07;
      RAM[100]=32'h07;
      RAM[101]=32'h07;
      RAM[102]=32'h07;
      RAM[103]=32'h07;
      RAM[104]=32'h07;
      RAM[105]=32'h07;
      RAM[106]=32'h07;
      RAM[107]=32'h07;
  end

 /* write operation */
 always @ (posedge clk)
  begin
   if (dMemWEn)
    begin
     RAM[dMemWRAddr0] = dMemWData;
    end
  end

 /* read operation */
 always @ (dMemWRAddr0)
  begin
   dMemRData0 <= #10 RAM[dMemWRAddr0];
  end

 /* read operation port 1*/
 always @ (dMemWRAddr1)
  begin
   dMemRData1 <= #10 RAM[dMemWRAddr1];
  end

endmodule

/*
 * alhamdulillah
 */
```

Listing 31 ZoiroSoko.v

```
/*-----------------------------------------------
-- bismillahirrahmanirrahim
-- FILE NAME : ZoiroSoko.v
-- TYPE      : rtl
-- FUNCTION  : chip top level
-- edit      : -
-- Author    : randyhw
-- Rev,Date  : 09/01/06
-----------------------------------------------*/

module ZoiroSoko (
   clock       , // input
   rst_n       , // input

   // external interrupt port
   ext_n_0     , // input
   ext_n_1     , // input
   ext_n_2     , // input
   ext_n_3     , // input

   // to vga interface
   vga_Red     , // output
   vga_Blue    , // output
   vga_Green   , // output
   vga_HSync   , // output
   vga_VSync   , // output
   video_blank , // output
   video_clock   // output
   );

 //
 // input ports
 //
 input wire        clock       ;
 input wire        rst_n       ;

 input wire        ext_n_0     ;
 input wire        ext_n_1     ;
 input wire        ext_n_2     ;
 input wire        ext_n_3     ;

 //
 // output ports
 //

 // to vga interface
 output wire [3:0] vga_Red     ;
 output wire [3:0] vga_Green   ;
 output wire [3:0] vga_Blue    ;
 output wire         vga_HSync   ;
 output wire       vga_VSync   ;
 output wire       video_blank ;
 output wire       video_clock ;

 //
 // internal variables
 //

 wire        op_ext0     ;
 wire        op_ext1     ;
 wire      op_ext2     ;
 wire      op_ext3     ;

 wire [6:0]  dMemWRAddr1 ;
 wire [31:0] dMemRData1  ;
```

```
 wire [31:0] dMemRData0  ;

 wire        dMemWEn     ;
 wire [31:0] dMemWData   ;
 wire [6:0]  dMemWRAddr0 ;

 wire [31:0] pcIF        ;
 wire [31:0] pcEX        ;

 wire [31:0] intRet      ;

 wire [4:0]  mc0WAddr    ;
 wire [31:0] mc0WData    ;

 wire [4:0]  mc0RAddr    ;
 wire [31:0] mc0RData    ;

 wire [31:0] instIn      ;
 wire [31:0] instAddr    ;

 //
 // port map
 //

 // push button buffer
button button0 (
   .clk (clock   ) ,
   .in  (ext_n_0 ) ,
   .out (op_ext0 )
   );

button button1 (
   .clk (clock   ) ,
   .in  (ext_n_1 ) ,
   .out (op_ext1 )
   );

button button2 (
   .clk (clock   ) ,
   .in  (ext_n_2 ) ,
   .out (op_ext2 )
   );

button button3 (
   .clk (clock   ) ,
   .in  (ext_n_3 )  ,
   .out (op_ext3 )
   );


 // VGA Driver Soko
VGA_Soko vga_soko (
   .Clock_48Mhz   (clock       ) ,
   .VGA_Red         (vga_Red     ) ,
   .VGA_Green       (vga_Green   ) ,
   .VGA_Blue          (vga_Blue    ) ,
   .VGA_Hsync       (vga_HSync   ) ,
   .VGA_Vsync       (vga_VSync   ) ,
   .Video_blank_out (video_blank ) ,
   .Video_clock_out (video_clock ) ,
   .dMemRData1          (dMemRData1  ) ,
   .dMemWRAddr1     (dMemWRAddr1 )
   );

 // SRP
srp srp (
```

```
   .Clock          (clock          ), // input
   .Reset          (rst_n          ), // input

   // external interrupt ports
   .ext_n_0        (op_ext0        ), // input
   .ext_n_1        (op_ext1        ), // input
   .ext_n_2        (op_ext2        ), // input
   .ext_n_3        (op_ext3        ), // input

   // to instruction memory
   .Inst           (instIn         ), // input
   .Iadd           (instAddr       ), // output

   // to data memory
   .Dadd           (dMemWRAddr0    ), // input
   .WE             (dMemWEn        ), // output
   .Wtdata         (dMemWData      ), // output
   .Rddata         (dMemRData0     )  // output
   );

 // Instruction Memory
instMemory instMemory (
   .instAddr       (instAddr       ), // input
   .instIn         (instIn         )  // output
   );

 // Data Memory
dataMemory dataMemory (
   .clk            (clock          ), // input
   .dMemWEn        (dMemWEn        ), // input
   .dMemWData      (dMemWData      ), // input
   .dMemWRAddr0    (dMemWRAddr0    ), // input
   .dMemWRAddr1    (dMemWRAddr1    ), // input
   .dMemRData0     (dMemRData0     ), // output
   .dMemRData1     (dMemRData1     )  // output
   );

endmodule


//
// alhamdulillah
//
```