pc\_unit haz\_d\_unit

imem

|  |  |
| --- | --- |
| IF/ID | |
| Input port | Output port |
| instr[15:0] | instr\_IF/ID[15:0] |
| wrt\_IF/ID |
| pc\_add2 [15:0] | pc\_add2\_IF/ID [15:0] |

|  |  |
| --- | --- |
| instr\_decoder | |
| Input port | Output port |
| instr\_IF/ID[15:0] | w1\_reg [2:0] |
| reg\_en |
| read\_reg1 [2:0] |
| read\_reg2[2:0] |
| b\_sel |
| mem\_en |
| mem\_wr |
| write\_src[4:0] |
| ext\_16 [15:0] |
| alu\_sign |
| alu\_inv |
| pc\_jump\_B\_sel |
|  |
|  |

{in\_dest\_ctrl, alu\_s, alu\_i, sign\_extender }

rf\_bypass,(connected to signals from WB)

|  |  |
| --- | --- |
| for\_unit\_2 | |
| Input port | Output port |
| read\_reg1 [2:0] (rs) | rs\_bypass[15:0] |
| rs[15:0] |
| w1\_reg\_EX\_MEM [2:0] |
| reg\_en\_EX\_MEM |
| writedata\_EX\_MEM[15:0] |
| w1\_reg\_MEM\_WB[2:0] |
| reg\_en\_MEM\_WB |
| writedata\_MEM\_WB[15:0] |

|  |  |
| --- | --- |
| haz\_d\_2 | |
| Input port | Output port |
| w1\_reg\_ID/EX[2:0] | pause\_pc |
| reg\_en\_ID/EX | mem\_en\_f |
| read\_reg1[2:0] (rs) | mem\_wr\_f |
|  | reg\_en\_f |
| mem\_en\_ID/EX | stall\_count (0->1cylce, 1->1cylce) |
| mem\_wr\_ID/EX |  |
| mem\_en |  |
| mem\_wr |  |
| reg\_en |  |
| w1\_reg\_EX\_MEM[2:0] |  |
| reg\_en\_EX\_MEM |  |
| mem\_en\_EX\_MEM |  |
| mem\_wr\_EX\_MEM |  |

pc\_jump\_B=(pc\_jump\_B\_sel)? pc\_add2\_IF/ID:rs\_bypass

pc\_jump\_out=pc\_jump\_B+ ext\_16\_ID/EX

|  |  |
| --- | --- |
| pc\_sel\_decoder | |
| Input Port | Output Port |
| instr\_IF/ID[15:0] | pc\_sel |
| rs [15:0] |

|  |  |
| --- | --- |
| ID/EX | |
| Input port | Output port |
| w1\_reg[2:0] | w1\_reg \_ID/EX[2:0] |
| reg\_en | reg\_en\_ID/EX |
| b\_sel | b\_sel\_ID/EX |
| mem\_en | mem\_en\_ID/EX |
| mem\_wr | mem\_wr\_ID/EX |
| write\_src[4:0] | write\_src\_ID/EX[4:0] |
| ext\_16[15:0] | ext\_16\_ID/EX [15:0] |
| alu\_sign | alu\_sign\_ID/EX |
| alu\_inv | alu\_inv\_ID/EX |
| rs [15:0] | rs\_ID/EX [15:0] |
| r2 [15:0] | r2\_ID/EX[15:0] |
| pc\_add2\_IF/ID [15:0] | pc\_add2\_ID/EX[15:0] |
| read\_reg1[2:0] | read\_reg1\_ID\_EX[2:0] |
| read\_reg2[2:0] | read\_reg2\_ID\_EX[2:0] |

alu\_bmux, , ALU, lt\_lte,

for\_unit, wrt\_ctrl(with some changes, output port writedata\_EX for all possible data source except for dmem, wrt\_ifdmem(determine if writedata==dmem\_out))

|  |  |
| --- | --- |
| EX/MEM | |
| Input Port | Output Port |
| w1\_reg \_ID/EX[2:0] | w1\_reg \_EX/MEM[2:0] |
| reg\_en\_ID/EX | reg\_en\_EX/MEM |
| mem\_en\_ID/EX | mem\_en\_EX/MEM |
| mem\_wr\_ID/EX | mem\_wr\_EX/MEM |
| writedata\_EX [15:0] | writedata\_EX/MEM[15:0] |
| alu\_out [15:0] | alu\_out\_EX/MEM [15:0] |
| r2\_ID/EX[15:0] | r2\_EX/MEM[15:0] |
| wrt\_ifdmem | wrt\_ifdmem\_EX/MEM |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

dmem, 2-1 mux (determine use writedata\_EX/MEM or dmem\_out)

|  |  |
| --- | --- |
| MEM/WB | |
| Input Port | Output Port |
| w1\_reg \_EX/MEM[2:0] | w1\_reg \_MEM/WB[2:0] |
| reg\_en\_EX/MEM | reg\_en\_MEM/WB |
| writedata [15:0] | writedata\_MEM/WB [15:0] |

Forward unit (alu at EX)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| w1\_reg\_MEM/WB== read\_reg1  reg\_en\_MEM/WB  !reg\_en\_EX/MEM  w1\_reg\_EX/MEM!=read\_reg1 | w1\_reg\_MEM/WB==read\_reg2  reg\_en\_MEM/WB  !reg\_en\_EX/MEM  w1\_reg\_EX/MEM!=read\_reg1 | w1\_reg\_EX/MEM= =read\_reg1  reg\_en\_EX/MEM | w1\_reg\_EX/MEM= =read\_reg2  reg\_en\_EX/MEM | w1\_reg\_MEM/WB==read\_reg2  reg\_en\_MEM/WB  mem\_en\_EX/MEM  mem\_wr\_EX/MEM  (ld followed by st/stu) |
| alu\_rs=writedata\_MEM/WB | alu\_r2=writedata\_MEM/WB | alu\_rs=writedata\_EX/MEM | alu\_r2=writedata\_EX/MEM | dmem.datain(writedata\_MEM/WB) |

Hazad detection unit (for EX)

|  |
| --- |
| mem\_en\_ID/EX && (~mem\_wr\_ID/EX) &&(w1\_reg\_ID/EX==read\_reg1 || w1\_reg\_ID/EX==read\_reg2) |
| reg\_en =0 |
| mem\_en=0 |
| mem\_wr=0 |
| halt\_pc (if halt\_pc==1, then pc\_unit changes pc\_add2\_B to 0) |
| wrt\_IF/ID=0 |
|  |
|  |

Forward\_unit 2( for pc)

|  |  |
| --- | --- |
| w1\_reg\_MEM/WB== read\_reg1  reg\_en\_MEM/WB  !reg\_en\_ID/EX  w1\_reg\_ID/EX!=read\_reg1 | w1\_reg\_EX/MEM==read\_reg1  reg\_en\_EX/MEM |
| pc\_jump.B=writedata\_MEM/WB | pc\_jump.B=writedata\_EX/MEM |

Hazard detection unit (for pc)

|  |
| --- |
| w1\_reg\_ID/EX==read\_reg1 &&mem\_en\_ID/EX && (~mem\_wr\_ID/EX) ->stall 2 cycles  w1\_reg\_ID/EX==read\_reg1 && ~mem\_en\_ID/EX &&(~mem\_en\_ID/EX) ->stall 1 cycle |
|  |
|  |
|  |