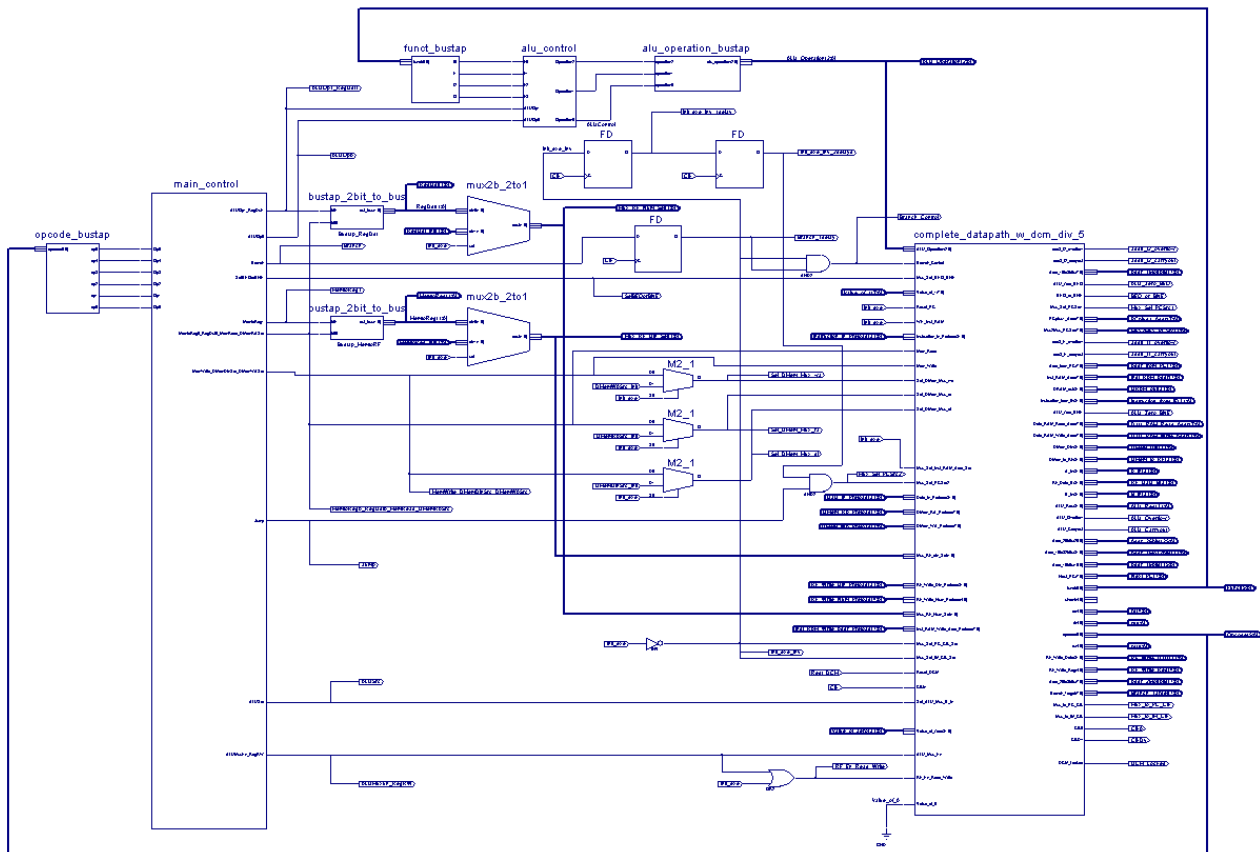


APPENDIX D

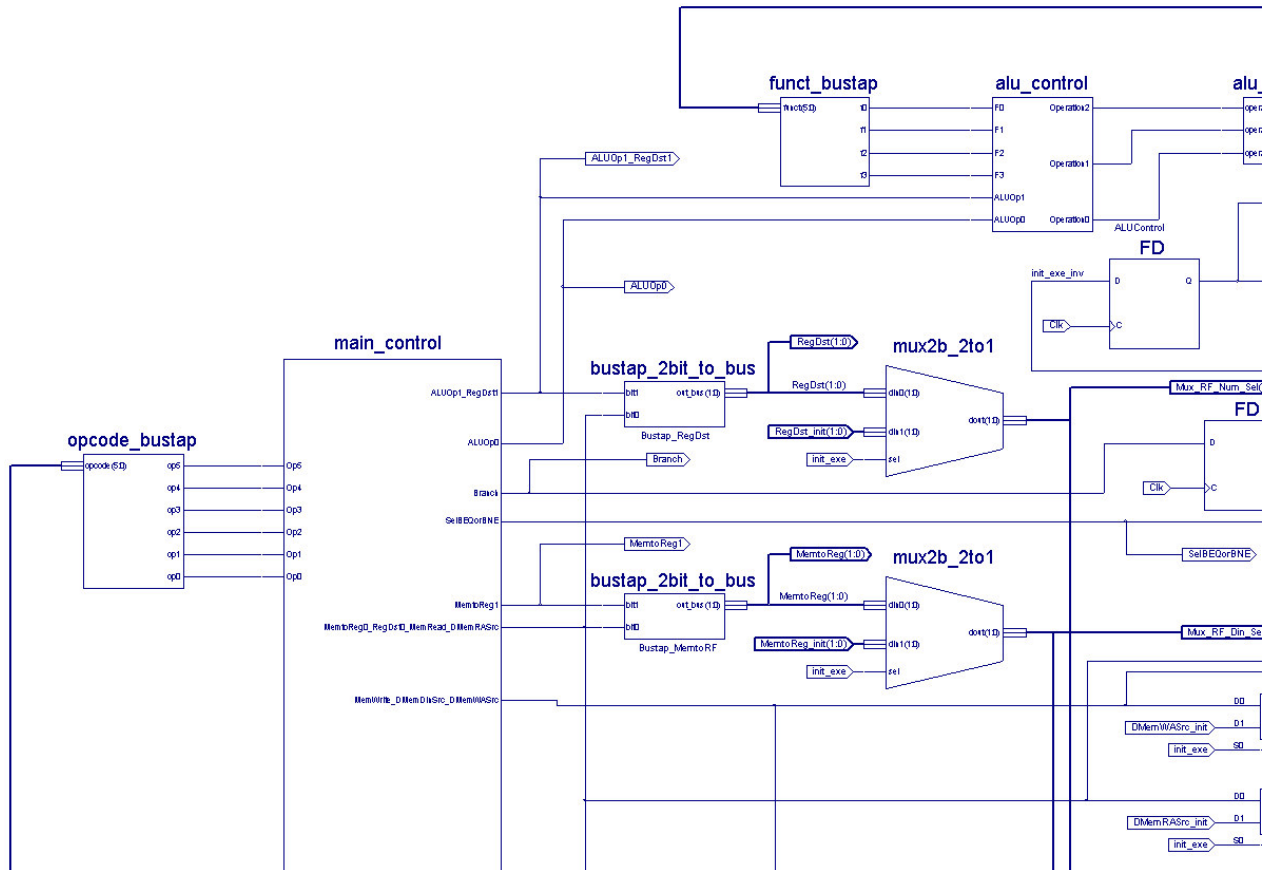
SUPPLEMENTARY MATERIAL FOR CHAPTER SIX

This appendix covers the supplementary material for Chapter Six. This includes, higher resolution figures, diagrams, and detailed VHDL code.

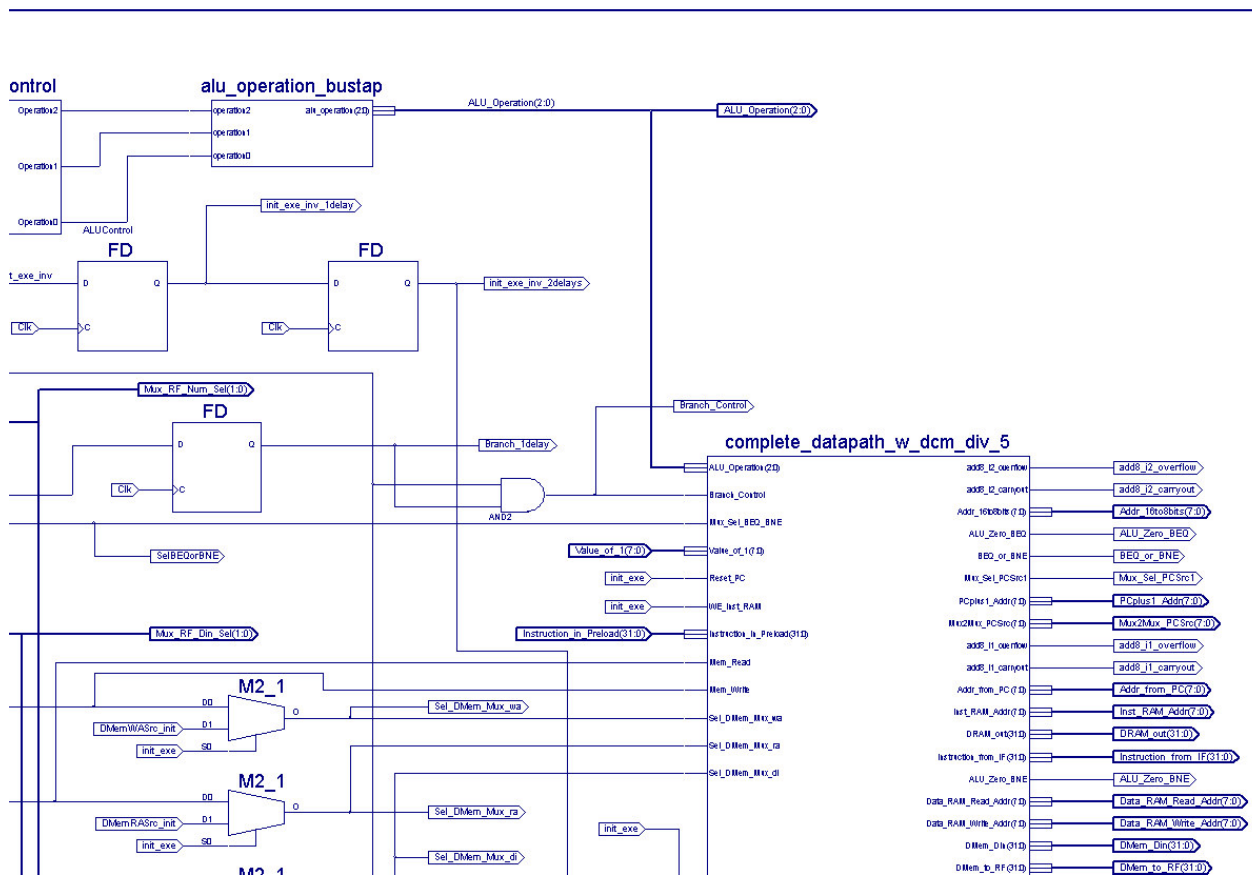
D.1 High Resolution Figure 6.3 (Full View Version)



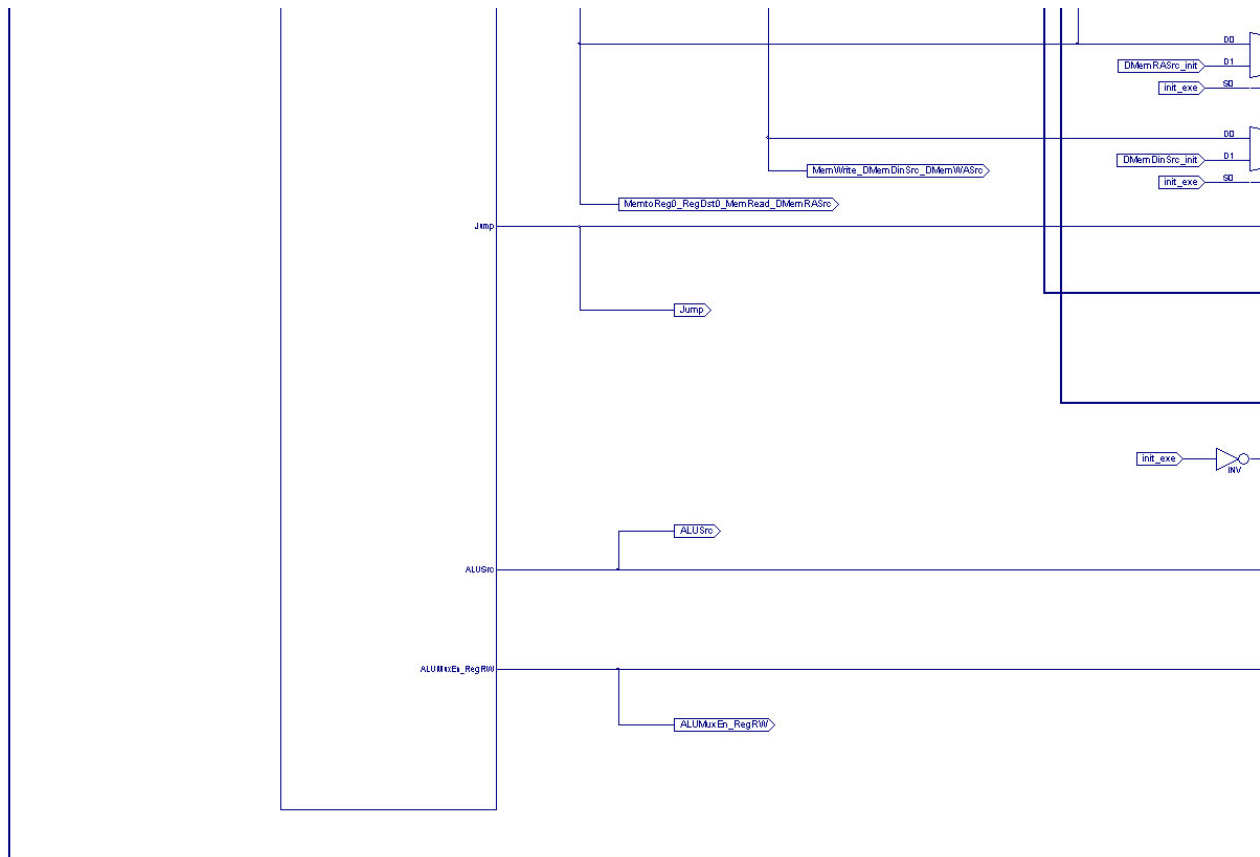
D.1A High Resolution Figure 6.3 (Magnified Top Left Quadrant)



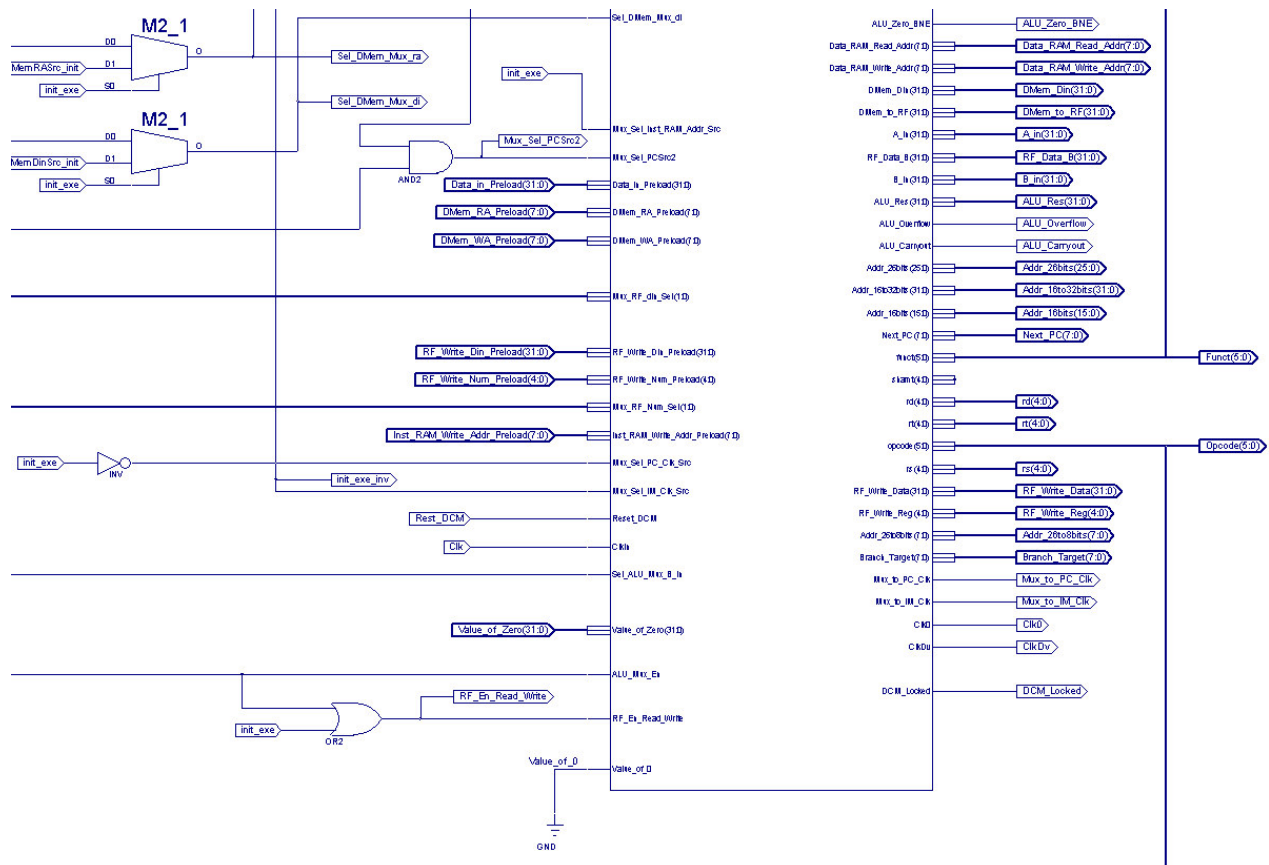
D.1B High Resolution Figure 6.3 (Magnified Top Right Quadrant)



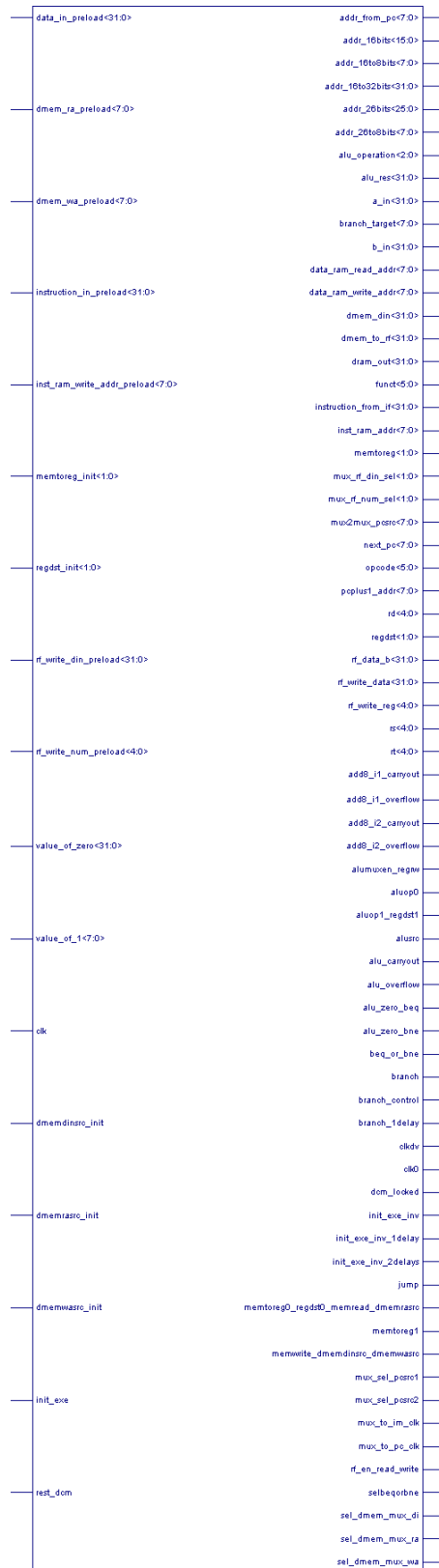
D.1C High Resolution Figure 6.3 (Magnified Bottom Left Quadrant)



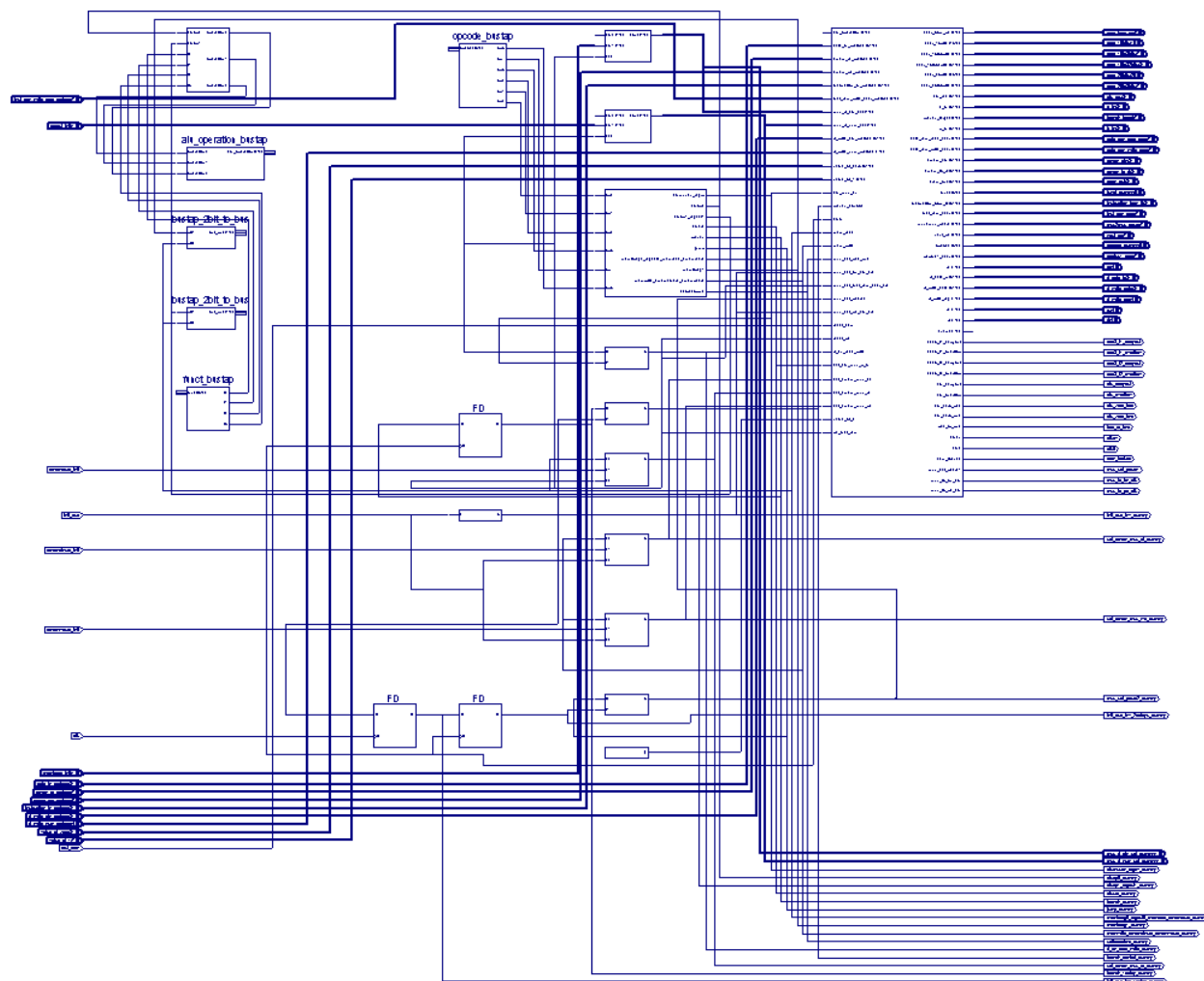
D.1D High Resolution Figure 6.3 (Magnified Bottom Right Quadrant)



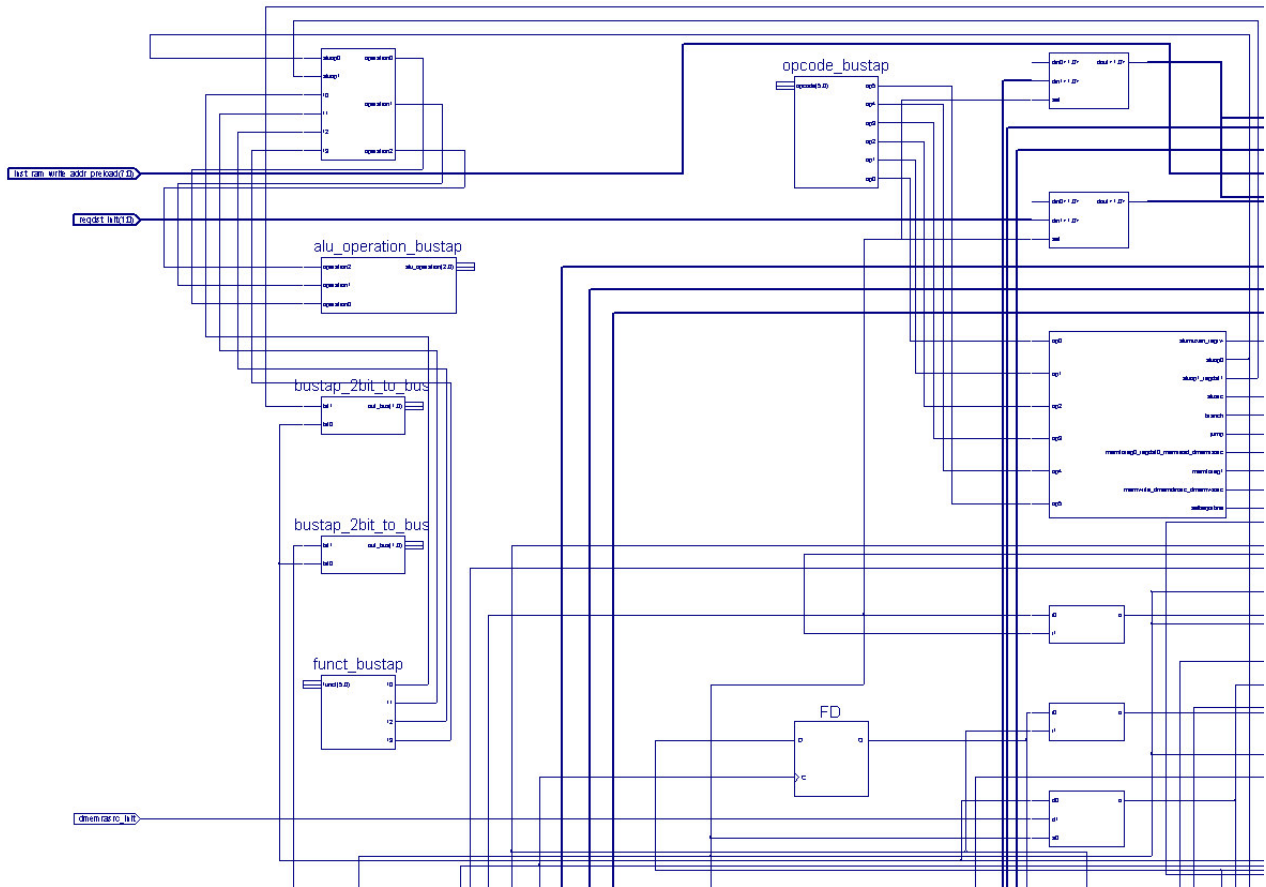
D.2 High Resolution Figure 6.4



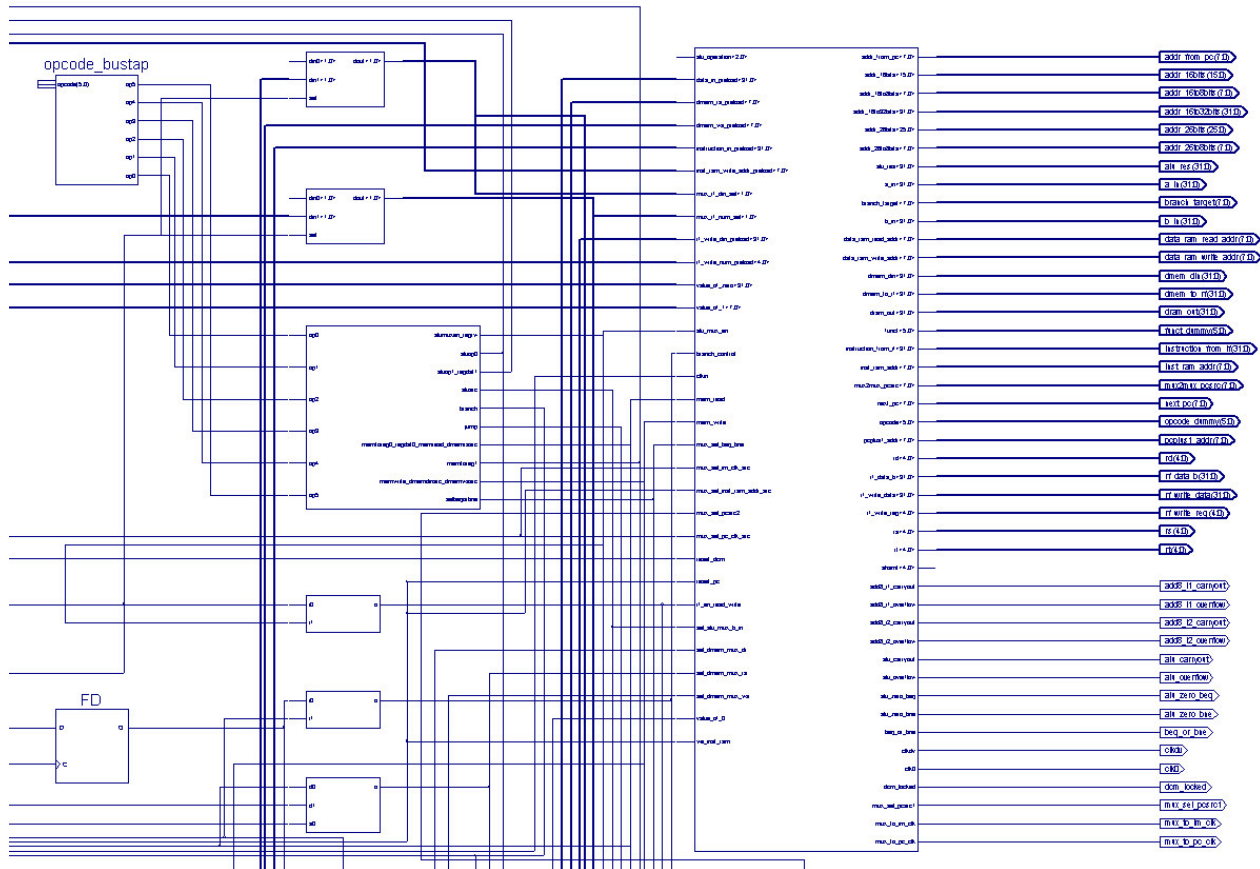
D.3 High Resolution Figure 6.5 (Full View Version)



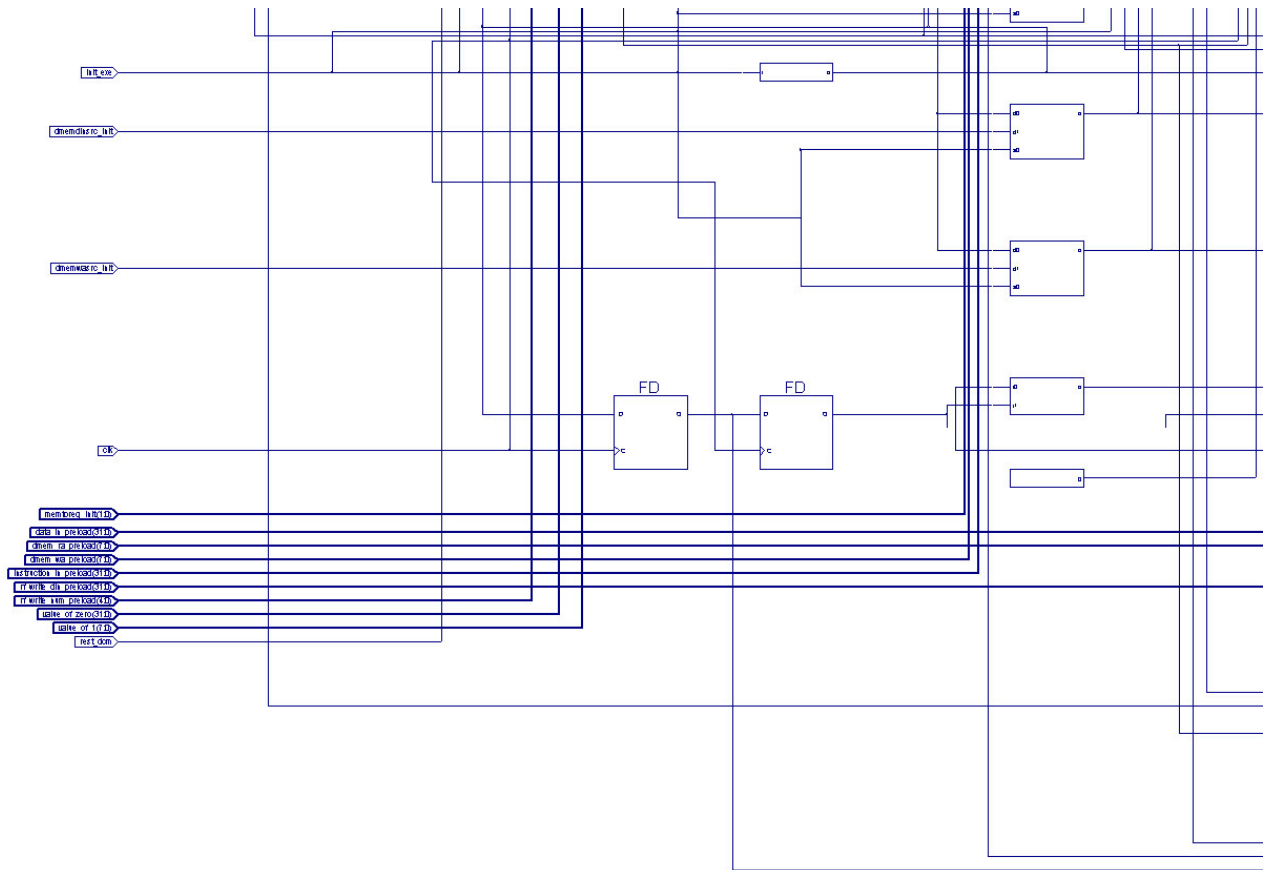
D.3A High Resolution Figure 6.5 (Magnified Top Left Quadrant)



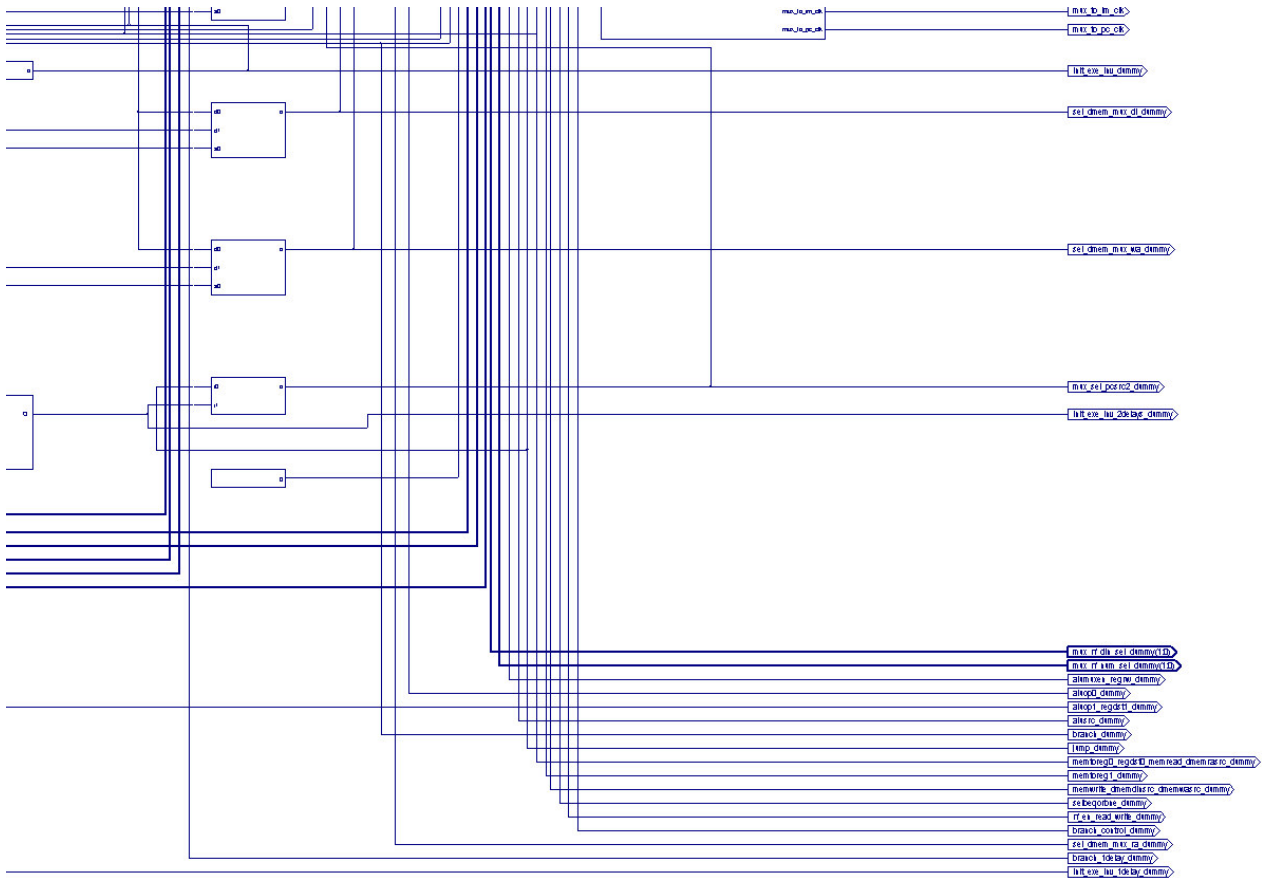
D.3B High Resolution Figure 6.5 (Magnified Top Right Quadrant)



D.3C High Resolution Figure 6.5 (Magnified Bottom Left Quadrant)



D.3D High Resolution Figure 6.5 (Magnified Bottom Right Quadrant)



D.4 VHDL Code for Section 6.3

```
-- Vhdl model created from schematic C:\Xilinx\virtex2\data\drawing\m2_1.sch - Thu Oct 19 08:23:13 2006
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
-- synopsys translate_off
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
-- synopsys translate_on

ENTITY M2_1_MXILINX_complete_datapath_and_control IS
    PORT ( D0: IN          STD_LOGIC;
           D1: IN          STD_LOGIC;
           S0: IN          STD_LOGIC;
           O:  OUT         STD_LOGIC);

end M2_1_MXILINX_complete_datapath_and_control;

ARCHITECTURE SCHEMATIC OF M2_1_MXILINX_complete_datapath_and_control IS
    SIGNAL M0:          STD_LOGIC;
    SIGNAL M1:          STD_LOGIC;

    ATTRIBUTE BOX_TYPE : STRING;

    COMPONENT AND2
        PORT ( I0: IN          STD_LOGIC;
               I1: IN          STD_LOGIC;
               O:  OUT         STD_LOGIC);
    END COMPONENT;

    ATTRIBUTE BOX_TYPE OF AND2 : COMPONENT IS "BLACK_BOX";
    COMPONENT AND2B1
        PORT ( I0: IN          STD_LOGIC;
               I1: IN          STD_LOGIC;
               O:  OUT         STD_LOGIC);
    END COMPONENT;

    ATTRIBUTE BOX_TYPE OF AND2B1 : COMPONENT IS "BLACK_BOX";
    COMPONENT OR2
        PORT ( I0: IN          STD_LOGIC;
               I1: IN          STD_LOGIC;
               O:  OUT         STD_LOGIC);
    END COMPONENT;

    ATTRIBUTE BOX_TYPE OF OR2 : COMPONENT IS "BLACK_BOX";
BEGIN
    I_36_9 : AND2
        PORT MAP (I0=>D1, I1=>S0, O=>M1);

    I_36_7 : AND2B1
        PORT MAP (I0=>S0, I1=>D0, O=>M0);

    I_36_8 : OR2
        PORT MAP (I0=>M1, I1=>M0, O=>O);

END SCHEMATIC;
```

```
-- Vhdl model created from schematic complete_datapath_and_control.sch - Thu Oct 19 08:23:30 2006
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
-- synopsys translate_off
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
-- synopsys translate_on
```

Appendix D – Supplementary Material for Chapter Six

ENTITY complete_datapath_and_control IS

```

PORT ( Clk:                                IN                                STD_LOGIC;
      DMemDinSrc_init:                     IN                                STD_LOGIC;
      DMemRASrc_init:                      IN                                STD_LOGIC;
      DMemWASrc_init:                     IN                                STD_LOGIC;
      DMem_RA_Preload:                     IN                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      DMem_WA_Preload:                     IN                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Data_in_Preload:                     IN                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      Inst_RAM_Write_Addr_Preload:         IN                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Instruction_in_Preload:              IN                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      MemtoReg_init:                      IN                                STD_LOGIC_VECTOR (1 DOWNTO 0);
      RF_Write_Din_Preload:               IN                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      RF_Write_Num_Preload:               IN                                STD_LOGIC_VECTOR (4 DOWNTO 0);
      RegDst_init:                        IN                                STD_LOGIC_VECTOR (1 DOWNTO 0);
      Rest_DCM:                           IN                                STD_LOGIC;
      Value_of_1:                          IN                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Value_of_Zero:                      IN                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      init_exe:                            IN                                STD_LOGIC;
      ALUMuxEn_RegRW:                     OUT                               STD_LOGIC;
      ALUOp0:                             OUT                               STD_LOGIC;
      ALUOp1_RegDst1:                     OUT                               STD_LOGIC;
      ALUSrc:                             OUT                               STD_LOGIC;
      ALU_Carryout:                       OUT                               STD_LOGIC;
      ALU_Operation:                      OUT                                STD_LOGIC_VECTOR (2 DOWNTO 0);
      ALU_Overflow:                       OUT                               STD_LOGIC;
      ALU_Res:                            OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      ALU_Zero_BEQ:                       OUT                               STD_LOGIC;
      ALU_Zero_BNE:                      OUT                               STD_LOGIC;
      A_in:                               OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      Addr_16bits:                        OUT                                STD_LOGIC_VECTOR (15 DOWNTO 0);
      Addr_16to32bits:                   OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      Addr_16to8bits:                    OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Addr_26bits:                       OUT                                STD_LOGIC_VECTOR (25 DOWNTO 0);
      Addr_26to8bits:                    OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Addr_from_PC:                      OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      BEQ_or_BNE:                        OUT                               STD_LOGIC;
      B_in:                               OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      Branch:                             OUT                               STD_LOGIC;
      Branch_1delay:                     OUT                               STD_LOGIC;
      Branch_Control:                    OUT                               STD_LOGIC;
      Branch_Target:                     OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Clk0:                              OUT                               STD_LOGIC;
      ClkDv:                             OUT                               STD_LOGIC;
      DCM_Locked:                        OUT                               STD_LOGIC;
      DMem_Din:                          OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      DMem_to_RF:                        OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      DRAM_out:                          OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      Data_RAM_Read_Addr:                 OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Data_RAM_Write_Addr:                 OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Funct:                              OUT                                STD_LOGIC_VECTOR (5 DOWNTO 0);
      Inst_RAM_Addr:                      OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Instruction_from_IF:                 OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      Jump:                              OUT                               STD_LOGIC;
      MemWrite_DMemDinSrc_DMemWASrc:     OUT                               STD_LOGIC;
      MemtoReg:                           OUT                                STD_LOGIC_VECTOR (1 DOWNTO 0);
      MemtoReg0_RegDst0_MemRead_DMemRASrc: OUT                               STD_LOGIC;
      MemtoReg1:                         OUT                               STD_LOGIC;
      Mux2Mux_PCSrc:                     OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Mux_RF_Din_Sel:                    OUT                                STD_LOGIC_VECTOR (1 DOWNTO 0);
      Mux_RF_Num_Sel:                    OUT                                STD_LOGIC_VECTOR (1 DOWNTO 0);
      Mux_Sel_PCSrc1:                    OUT                               STD_LOGIC;
      Mux_Sel_PCSrc2:                    OUT                               STD_LOGIC;
      Mux_to_IM_Clk:                     OUT                               STD_LOGIC;
      Mux_to_PC_Clk:                     OUT                               STD_LOGIC;
      Next_PC:                           OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Opcode:                            OUT                                STD_LOGIC_VECTOR (5 DOWNTO 0);
      PCplus1_Addr:                      OUT                                STD_LOGIC_VECTOR (7 DOWNTO 0);
      RF_Data_B:                         OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      RF_En_Read_Write:                  OUT                               STD_LOGIC;
      RF_Write_Data:                     OUT                                STD_LOGIC_VECTOR (31 DOWNTO 0);
      RF_Write_Reg:                      OUT                                STD_LOGIC_VECTOR (4 DOWNTO 0);
      RegDst:                            OUT                                STD_LOGIC_VECTOR (1 DOWNTO 0);

```

Appendix D – Supplementary Material for Chapter Six

```

SelBEQorBNE:          OUT          STD_LOGIC;
Sel_DMem_Mux_di:      OUT          STD_LOGIC;
Sel_DMem_Mux_ra:      OUT          STD_LOGIC;
Sel_DMem_Mux_wa:      OUT          STD_LOGIC;
add8_i1_carryout:     OUT          STD_LOGIC;
add8_i1_overflow:     OUT          STD_LOGIC;
add8_i2_carryout:     OUT          STD_LOGIC;
add8_i2_overflow:     OUT          STD_LOGIC;
init_exe_inv:         OUT          STD_LOGIC;
init_exe_inv_1delay:  OUT          STD_LOGIC;
init_exe_inv_2delays: OUT          STD_LOGIC;
rd:                   OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);
rs:                   OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);
rt:                   OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);

end complete_datapath_and_control;

ARCHITECTURE SCHEMATIC OF complete_datapath_and_control IS
    SIGNAL ALUMuxEn_RegRW_DUMMY          : STD_LOGIC;
    SIGNAL ALUOp0_DUMMY                  : STD_LOGIC;
    SIGNAL ALUOp1_RegDst1_DUMMY          : STD_LOGIC;
    SIGNAL ALUSrc_DUMMY                  : STD_LOGIC;
    SIGNAL ALU_Operation_DUMMY           : STD_LOGIC_VECTOR (2 DOWNTO 0);
    SIGNAL Branch_1delay_DUMMY          : STD_LOGIC;
    SIGNAL Branch_Control_DUMMY          : STD_LOGIC;
    SIGNAL Branch_DUMMY                  : STD_LOGIC;
    SIGNAL F0                            : STD_LOGIC;
    SIGNAL F1                            : STD_LOGIC;
    SIGNAL F2                            : STD_LOGIC;
    SIGNAL F3                            : STD_LOGIC;
    SIGNAL Funct_DUMMY                   : STD_LOGIC_VECTOR (5 DOWNTO 0);
    SIGNAL Jump_DUMMY                    : STD_LOGIC;
    SIGNAL MemWrite_DMemDinSrc_DMemWASrc_DUMMY : STD_LOGIC;
    SIGNAL MemtoReg0_RegDst0_MemRead_DMemRASrc_DUMMY : STD_LOGIC;
    SIGNAL MemtoReg1_DUMMY               : STD_LOGIC;
    SIGNAL MemtoReg_DUMMY                 : STD_LOGIC_VECTOR (1 DOWNTO 0);
    SIGNAL Mux_RF_Din_Sel_DUMMY           : STD_LOGIC_VECTOR (1 DOWNTO 0);
    SIGNAL Mux_RF_Num_Sel_DUMMY           : STD_LOGIC_VECTOR (1 DOWNTO 0);
    SIGNAL Mux_Sel_PCSrc2_DUMMY           : STD_LOGIC;
    SIGNAL Op0                            : STD_LOGIC;
    SIGNAL Op1                            : STD_LOGIC;
    SIGNAL Op2                            : STD_LOGIC;
    SIGNAL Op3                            : STD_LOGIC;
    SIGNAL Op4                            : STD_LOGIC;
    SIGNAL Op5                            : STD_LOGIC;
    SIGNAL Opcode_DUMMY                   : STD_LOGIC_VECTOR (5 DOWNTO 0);
    SIGNAL Operation0                     : STD_LOGIC;
    SIGNAL Operation1                     : STD_LOGIC;
    SIGNAL Operation2                     : STD_LOGIC;
    SIGNAL RF_En_Read_Write_DUMMY         : STD_LOGIC;
    SIGNAL RegDst_DUMMY                   : STD_LOGIC_VECTOR (1 DOWNTO 0);
    SIGNAL SelBEQorBNE_DUMMY              : STD_LOGIC;
    SIGNAL Sel_DMem_Mux_di_DUMMY          : STD_LOGIC;
    SIGNAL Sel_DMem_Mux_ra_DUMMY          : STD_LOGIC;
    SIGNAL Sel_DMem_Mux_wa_DUMMY          : STD_LOGIC;
    SIGNAL Value_of_0                     : STD_LOGIC;
    SIGNAL init_exe_inv_1delay_DUMMY       : STD_LOGIC;
    SIGNAL init_exe_inv_2delays_DUMMY      : STD_LOGIC;
    SIGNAL init_exe_inv_DUMMY              : STD_LOGIC;

    ATTRIBUTE BOX_TYPE : STRING;
    ATTRIBUTE INIT : STRING ;
    ATTRIBUTE INIT OF FD3 : LABEL IS "0";
    ATTRIBUTE INIT OF FD1 : LABEL IS "0";
    ATTRIBUTE INIT OF FD2 : LABEL IS "0";
    ATTRIBUTE U_SET : STRING ;
    ATTRIBUTE U_SET OF M2_1_i2 : LABEL IS "M2_1_i2_2";
    ATTRIBUTE U_SET OF M2_1_i3 : LABEL IS "M2_1_i3_1";
    ATTRIBUTE U_SET OF M2_1_i1 : LABEL IS "M2_1_i1_0";

    COMPONENT alu_control

```

```

PORT ( F0:                IN                STD_LOGIC;
      F1:                IN                STD_LOGIC;
      F2:                IN                STD_LOGIC;
      F3:                IN                STD_LOGIC;
      ALUOp1:            IN                STD_LOGIC;
      ALUOp0:            IN                STD_LOGIC;
      Operation1:        OUT               STD_LOGIC;
      Operation2:        OUT               STD_LOGIC;
      Operation0:        OUT               STD_LOGIC);
END COMPONENT;

COMPONENT alu_operation_bustap
PORT ( operation2:        IN                STD_LOGIC;
      operation1:        IN                STD_LOGIC;
      operation0:        IN                STD_LOGIC;
      alu_operation:      OUT               STD_LOGIC_VECTOR (2 DOWNTO 0));
END COMPONENT;

COMPONENT AND2
PORT ( I0:                IN                STD_LOGIC;
      I1:                IN                STD_LOGIC;
      O:                 OUT               STD_LOGIC);
END COMPONENT;

ATTRIBUTE BOX_TYPE OF AND2 : COMPONENT IS "BLACK_BOX";
COMPONENT bustap_2bit_to_bus
PORT ( bit1:              IN                STD_LOGIC;
      bit0:              IN                STD_LOGIC;
      out_bus:            OUT               STD_LOGIC_VECTOR (1 DOWNTO 0));
END COMPONENT;

COMPONENT complete_datapath_w_dcm_div_5
PORT ( Value_of_0:        IN                STD_LOGIC;
      Branch_Control:    IN                STD_LOGIC;
      Mux_Sel_BEQ_BNE:   IN                STD_LOGIC;
      Value_of_1:        IN                STD_LOGIC_VECTOR (7 DOWNTO 0);
      Reset_PC:          IN                STD_LOGIC;
      WE_Inst_RAM:       IN                STD_LOGIC;
      Instruction_in_Preload: IN          STD_LOGIC_VECTOR (31 DOWNTO 0);
      Mem_Read:          IN                STD_LOGIC;
      Mem_Write:         IN                STD_LOGIC;
      Sel_DMem_Mux_wa:   IN                STD_LOGIC;
      Sel_DMem_Mux_ra:   IN                STD_LOGIC;
      Sel_DMem_Mux_di:   IN                STD_LOGIC;
      ALU_Mux_En:        IN                STD_LOGIC;
      Sel_ALU_Mux_B_in:  IN                STD_LOGIC;
      ALU_Operation:     IN                STD_LOGIC_VECTOR (2 DOWNTO 0);
      Mux_Sel_Inst_RAM_Addr_Src: IN        STD_LOGIC;
      Mux_Sel_PCSrc2:    IN                STD_LOGIC;
      Data_in_Preload:   IN                STD_LOGIC_VECTOR (31 DOWNTO 0);
      DMem_RA_Preload:   IN                STD_LOGIC_VECTOR (7 DOWNTO 0);
      DMem_WA_Preload:   IN                STD_LOGIC_VECTOR (7 DOWNTO 0);
      RF_En_Read_Write:  IN                STD_LOGIC;
      Mux_RF_din_Sel:    IN                STD_LOGIC_VECTOR (1 DOWNTO 0);
      Value_of_Zero:     IN                STD_LOGIC_VECTOR (31 DOWNTO 0);
      RF_Write_Din_Preload: IN          STD_LOGIC_VECTOR (31 DOWNTO 0);
      RF_Write_Num_Preload: IN          STD_LOGIC_VECTOR (4 DOWNTO 0);
      Mux_RF_Num_Sel:    IN                STD_LOGIC_VECTOR (1 DOWNTO 0);
      Inst_RAM_Write_Addr_Preload: IN      STD_LOGIC_VECTOR (7 DOWNTO 0);
      Mux_Sel_PC_Clk_Src: IN                STD_LOGIC;
      Mux_Sel_IM_Clk_Src: IN                STD_LOGIC;
      Reset_DCM:         IN                STD_LOGIC;
      Clkin:             IN                STD_LOGIC;
      add8_i2_overflow:   OUT               STD_LOGIC;
      add8_i2_carryout:   OUT               STD_LOGIC;
      Addr_16to8bits:     OUT               STD_LOGIC_VECTOR (7 DOWNTO 0);
      ALU_Zero_BEQ:       OUT               STD_LOGIC;
      BEQ_or_BNE:        OUT               STD_LOGIC;
      Mux_Sel_PCSrc1:     OUT               STD_LOGIC;
      PCplus1_Addr:       OUT               STD_LOGIC_VECTOR (7 DOWNTO 0);
      Mux2Mux_PCSrc:      OUT               STD_LOGIC_VECTOR (7 DOWNTO 0);
      add8_il_overflow:   OUT               STD_LOGIC;
      add8_il_carryout:   OUT               STD_LOGIC);

```



```

        Addr_from_PC:      OUT          STD_LOGIC_VECTOR (7 DOWNTO 0);
        Inst_RAM_Addr:     OUT          STD_LOGIC_VECTOR (7 DOWNTO 0);
        DRAM_out:          OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        Instruction_from_IF: OUT        STD_LOGIC_VECTOR (31 DOWNTO 0);
        ALU_Zero_BNE:      OUT          STD_LOGIC;
        Data_RAM_Read_Addr: OUT        STD_LOGIC_VECTOR (7 DOWNTO 0);
        Data_RAM_Write_Addr: OUT       STD_LOGIC_VECTOR (7 DOWNTO 0);
        DMem_Din:          OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        DMem_to_RF:        OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        A_in:              OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        RF_Data_B:         OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        B_in:              OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        ALU_Res:           OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        ALU_Overflow:      OUT          STD_LOGIC;
        ALU_Carryout:      OUT          STD_LOGIC;
        Addr_26bits:       OUT          STD_LOGIC_VECTOR (25 DOWNTO 0);
        Addr_16to32bits:   OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        Addr_16bits:       OUT          STD_LOGIC_VECTOR (15 DOWNTO 0);
        Next_PC:           OUT          STD_LOGIC_VECTOR (7 DOWNTO 0);
        funct:             OUT          STD_LOGIC_VECTOR (5 DOWNTO 0);
        shamt:             OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);
        rd:                OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);
        rt:                OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);
        opcode:            OUT          STD_LOGIC_VECTOR (5 DOWNTO 0);
        rs:                OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);
        RF_Write_Data:     OUT          STD_LOGIC_VECTOR (31 DOWNTO 0);
        RF_Write_Reg:      OUT          STD_LOGIC_VECTOR (4 DOWNTO 0);
        Addr_26to8bits:    OUT          STD_LOGIC_VECTOR (7 DOWNTO 0);
        Branch_Target:     OUT          STD_LOGIC_VECTOR (7 DOWNTO 0);
        Mux_to_PC_Clk:     OUT          STD_LOGIC;
        Mux_to_IM_Clk:     OUT          STD_LOGIC;
        Clk0:              OUT          STD_LOGIC;
        ClkDv:             OUT          STD_LOGIC;
        DCM_Locked:        OUT          STD_LOGIC);
END COMPONENT;

COMPONENT FD
-- synopsis translate_off
GENERIC(      INIT : BIT := '0' );
-- synopsis translate_on
PORT ( C:      IN          STD_LOGIC;
       D:      IN          STD_LOGIC;
       Q:      OUT         STD_LOGIC);
END COMPONENT;

ATTRIBUTE BOX_TYPE OF FD : COMPONENT IS "BLACK_BOX";
COMPONENT funct_bustap
PORT ( funct:      IN          STD_LOGIC_VECTOR (5 DOWNTO 0);
       f3:         OUT         STD_LOGIC;
       f2:         OUT         STD_LOGIC;
       f1:         OUT         STD_LOGIC;
       f0:         OUT         STD_LOGIC);
END COMPONENT;

COMPONENT GND
PORT ( G:      OUT          STD_LOGIC);
END COMPONENT;

ATTRIBUTE BOX_TYPE OF GND : COMPONENT IS "BLACK_BOX";
COMPONENT INV
PORT ( I:      IN          STD_LOGIC;
       O:      OUT         STD_LOGIC);
END COMPONENT;

ATTRIBUTE BOX_TYPE OF INV : COMPONENT IS "BLACK_BOX";
COMPONENT M2_1_MXILINX_complete_datapath_and_control
PORT ( D0:      IN          STD_LOGIC;
       D1:      IN          STD_LOGIC;
       S0:      IN          STD_LOGIC;
       O:      OUT         STD_LOGIC);
END COMPONENT;

COMPONENT main_control

```

```

PORT ( Op4:                IN                STD_LOGIC;
      Op5:                IN                STD_LOGIC;
      Op1:                IN                STD_LOGIC;
      Op0:                IN                STD_LOGIC;
      Op3:                IN                STD_LOGIC;
      Op2:                IN                STD_LOGIC;
      Jump:               OUT                STD_LOGIC;
      SelBEQorBNE:       OUT                STD_LOGIC;
      MemWrite_DMemDinSrc_DMemWASrc: OUT    STD_LOGIC;
      MemtoReg0_RegDst0_MemRead_DMemRASrc: OUT
                                STD_LOGIC;
      ALUOp1_RegDst1:     OUT                STD_LOGIC;
      ALUOp0:             OUT                STD_LOGIC;
      ALUSrc:             OUT                STD_LOGIC;
      Branch:             OUT                STD_LOGIC;
      ALUMuxEn_RegRW:     OUT                STD_LOGIC;
      MemtoReg1:         OUT                STD_LOGIC);
END COMPONENT;

COMPONENT mux2b_2to1
  PORT ( sel:              IN                STD_LOGIC;
        din0:             IN                STD_LOGIC_VECTOR (1 DOWNTO 0);
        din1:             IN                STD_LOGIC_VECTOR (1 DOWNTO 0);
        dout:             OUT                STD_LOGIC_VECTOR (1 DOWNTO 0));
END COMPONENT;

COMPONENT opcode_bustap
  PORT ( opcode:          IN                STD_LOGIC_VECTOR (5 DOWNTO 0);
        op5:             OUT                STD_LOGIC;
        op4:             OUT                STD_LOGIC;
        op3:             OUT                STD_LOGIC;
        op2:             OUT                STD_LOGIC;
        op1:             OUT                STD_LOGIC;
        op0:             OUT                STD_LOGIC);
END COMPONENT;

COMPONENT OR2
  PORT ( IO:              IN                STD_LOGIC;
        I1:              IN                STD_LOGIC;
        O:               OUT                STD_LOGIC);
END COMPONENT;

ATTRIBUTE BOX_TYPE OF OR2 : COMPONENT IS "BLACK_BOX";
BEGIN
  ALUMuxEn_RegRW <= ALUMuxEn_RegRW_DUMMY;
  ALUOp0 <= ALUOp0_DUMMY;
  ALUOp1_RegDst1 <= ALUOp1_RegDst1_DUMMY;
  ALUSrc <= ALUSrc_DUMMY;
  ALU_Operation <= ALU_Operation_DUMMY;
  Branch <= Branch_DUMMY;
  Branch_1delay <= Branch_1delay_DUMMY;
  Branch_Control <= Branch_Control_DUMMY;
  Funct <= Funct_DUMMY;
  Jump <= Jump_DUMMY;
  MemWrite_DMemDinSrc_DMemWASrc <= MemWrite_DMemDinSrc_DMemWASrc_DUMMY;
  MemtoReg <= MemtoReg_DUMMY;
  MemtoReg0_RegDst0_MemRead_DMemRASrc <=
    MemtoReg0_RegDst0_MemRead_DMemRASrc_DUMMY;
  MemtoReg1 <= MemtoReg1_DUMMY;
  Mux_RF_Din_Sel <= Mux_RF_Din_Sel_DUMMY;
  Mux_RF_Num_Sel <= Mux_RF_Num_Sel_DUMMY;
  Mux_Sel_PCSrc2 <= Mux_Sel_PCSrc2_DUMMY;
  Opcode <= Opcode_DUMMY;
  RF_En_Read_Write <= RF_En_Read_Write_DUMMY;
  RegDst <= RegDst_DUMMY;
  SelBEQorBNE <= SelBEQorBNE_DUMMY;
  Sel_DMem_Mux_di <= Sel_DMem_Mux_di_DUMMY;
  Sel_DMem_Mux_ra <= Sel_DMem_Mux_ra_DUMMY;
  Sel_DMem_Mux_wa <= Sel_DMem_Mux_wa_DUMMY;
  init_exe_inv <= init_exe_inv_DUMMY;
  init_exe_inv_1delay <= init_exe_inv_1delay_DUMMY;
  init_exe_inv_2delays <= init_exe_inv_2delays_DUMMY;

```

Appendix D – Supplementary Material for Chapter Six

```

ALUControl : alu_control
  PORT MAP (F0=>F0, F1=>F1, F2=>F2, F3=>F3, ALUOp1=>ALUOp1_RegDst1_DUMMY,
    ALUOp0=>ALUOp0_DUMMY, Operation1=>Operation1, Operation2=>Operation2,
    Operation0=>Operation0);

ALUOperationBT : alu_operation_bustap
  PORT MAP (operation2=>Operation2, operation1=>Operation1,
    operation0=>Operation0, alu_operation(2)>=>ALU_Operation_DUMMY(2),
    alu_operation(1)>=>ALU_Operation_DUMMY(1),
    alu_operation(0)>=>ALU_Operation_DUMMY(0));

AND2_Jump : AND2
  PORT MAP (I0=>Jump_DUMMY, I1=>init_exe_inv_2delays_DUMMY,
    O=>Mux_Sel_PCsrc2_DUMMY);

AND2_Branch : AND2
  PORT MAP (I0=>Branch_1delay_DUMMY, I1=>init_exe_inv_DUMMY,
    O=>Branch_Control_DUMMY);

Bustap_RegDst : bustap_2bit_to_bus
  PORT MAP (bit1=>ALUOp1_RegDst1_DUMMY,
    bit0=>MemtoReg0_RegDst0_MemRead_DMemRASrc_DUMMY,
    out_bus(1)>=>RegDst_DUMMY(1), out_bus(0)>=>RegDst_DUMMY(0));

Bustap_MemtoRF : bustap_2bit_to_bus
  PORT MAP (bit1=>MemtoReg1_DUMMY,
    bit0=>MemtoReg0_RegDst0_MemRead_DMemRASrc_DUMMY,
    out_bus(1)>=>MemtoReg_DUMMY(1), out_bus(0)>=>MemtoReg_DUMMY(0));

Complete_Datapath : complete_datapath_w_dcm_div_5
  PORT MAP (Value_of_0=>Value_of_0, Branch_Control=>Branch_Control_DUMMY,
    Mux_Sel_BEQ_BNE=>SelBEQorBNE_DUMMY, Value_of_1(7)>=>Value_of_1(7),
    Value_of_1(6)>=>Value_of_1(6), Value_of_1(5)>=>Value_of_1(5),
    Value_of_1(4)>=>Value_of_1(4), Value_of_1(3)>=>Value_of_1(3),
    Value_of_1(2)>=>Value_of_1(2), Value_of_1(1)>=>Value_of_1(1),
    Value_of_1(0)>=>Value_of_1(0), Reset_PC=>init_exe, WE_Inst_RAM=>init_exe,
    Instruction_in_Preload(31)>=>Instruction_in_Preload(31),
    Instruction_in_Preload(30)>=>Instruction_in_Preload(30),
    Instruction_in_Preload(29)>=>Instruction_in_Preload(29),
    Instruction_in_Preload(28)>=>Instruction_in_Preload(28),
    Instruction_in_Preload(27)>=>Instruction_in_Preload(27),
    Instruction_in_Preload(26)>=>Instruction_in_Preload(26),
    Instruction_in_Preload(25)>=>Instruction_in_Preload(25),
    Instruction_in_Preload(24)>=>Instruction_in_Preload(24),
    Instruction_in_Preload(23)>=>Instruction_in_Preload(23),
    Instruction_in_Preload(22)>=>Instruction_in_Preload(22),
    Instruction_in_Preload(21)>=>Instruction_in_Preload(21),
    Instruction_in_Preload(20)>=>Instruction_in_Preload(20),
    Instruction_in_Preload(19)>=>Instruction_in_Preload(19),
    Instruction_in_Preload(18)>=>Instruction_in_Preload(18),
    Instruction_in_Preload(17)>=>Instruction_in_Preload(17),
    Instruction_in_Preload(16)>=>Instruction_in_Preload(16),
    Instruction_in_Preload(15)>=>Instruction_in_Preload(15),
    Instruction_in_Preload(14)>=>Instruction_in_Preload(14),
    Instruction_in_Preload(13)>=>Instruction_in_Preload(13),
    Instruction_in_Preload(12)>=>Instruction_in_Preload(12),
    Instruction_in_Preload(11)>=>Instruction_in_Preload(11),
    Instruction_in_Preload(10)>=>Instruction_in_Preload(10),
    Instruction_in_Preload(9)>=>Instruction_in_Preload(9),
    Instruction_in_Preload(8)>=>Instruction_in_Preload(8),
    Instruction_in_Preload(7)>=>Instruction_in_Preload(7),
    Instruction_in_Preload(6)>=>Instruction_in_Preload(6),
    Instruction_in_Preload(5)>=>Instruction_in_Preload(5),
    Instruction_in_Preload(4)>=>Instruction_in_Preload(4),
    Instruction_in_Preload(3)>=>Instruction_in_Preload(3),
    Instruction_in_Preload(2)>=>Instruction_in_Preload(2),
    Instruction_in_Preload(1)>=>Instruction_in_Preload(1),
    Instruction_in_Preload(0)>=>Instruction_in_Preload(0),
    Mem_Read=>MemtoReg0_RegDst0_MemRead_DMemRASrc_DUMMY,
    Mem_Write=>MemWrite_DMemDinSrc_DMemWASrc_DUMMY,
    Sel_DMem_Mux_wa=>Sel_DMem_Mux_wa_DUMMY,
    Sel_DMem_Mux_ra=>Sel_DMem_Mux_ra_DUMMY,
    Sel_DMem_Mux_di=>Sel_DMem_Mux_di_DUMMY, ALU_Mux_En=>ALUMuxEn_RegRW_DUMMY,

```

```

Sel_ALU_Mux_B_in=>ALUSrc_DUMMY, ALU_Operation(2)=>ALU_Operation_DUMMY(2),
ALU_Operation(1)=>ALU_Operation_DUMMY(1),
ALU_Operation(0)=>ALU_Operation_DUMMY(0),
Mux_Sel_Inst_RAM_Addr_Src=>init_exe,
Mux_Sel_PCSrc2=>Mux_Sel_PCSrc2_DUMMY,
Data_in_Preload(31)=>Data_in_Preload(31),
Data_in_Preload(30)=>Data_in_Preload(30),
Data_in_Preload(29)=>Data_in_Preload(29),
Data_in_Preload(28)=>Data_in_Preload(28),
Data_in_Preload(27)=>Data_in_Preload(27),
Data_in_Preload(26)=>Data_in_Preload(26),
Data_in_Preload(25)=>Data_in_Preload(25),
Data_in_Preload(24)=>Data_in_Preload(24),
Data_in_Preload(23)=>Data_in_Preload(23),
Data_in_Preload(22)=>Data_in_Preload(22),
Data_in_Preload(21)=>Data_in_Preload(21),
Data_in_Preload(20)=>Data_in_Preload(20),
Data_in_Preload(19)=>Data_in_Preload(19),
Data_in_Preload(18)=>Data_in_Preload(18),
Data_in_Preload(17)=>Data_in_Preload(17),
Data_in_Preload(16)=>Data_in_Preload(16),
Data_in_Preload(15)=>Data_in_Preload(15),
Data_in_Preload(14)=>Data_in_Preload(14),
Data_in_Preload(13)=>Data_in_Preload(13),
Data_in_Preload(12)=>Data_in_Preload(12),
Data_in_Preload(11)=>Data_in_Preload(11),
Data_in_Preload(10)=>Data_in_Preload(10),
Data_in_Preload(9)=>Data_in_Preload(9),
Data_in_Preload(8)=>Data_in_Preload(8),
Data_in_Preload(7)=>Data_in_Preload(7),
Data_in_Preload(6)=>Data_in_Preload(6),
Data_in_Preload(5)=>Data_in_Preload(5),
Data_in_Preload(4)=>Data_in_Preload(4),
Data_in_Preload(3)=>Data_in_Preload(3),
Data_in_Preload(2)=>Data_in_Preload(2),
Data_in_Preload(1)=>Data_in_Preload(1),
Data_in_Preload(0)=>Data_in_Preload(0),
DMem_RA_Preload(7)=>DMem_RA_Preload(7),
DMem_RA_Preload(6)=>DMem_RA_Preload(6),
DMem_RA_Preload(5)=>DMem_RA_Preload(5),
DMem_RA_Preload(4)=>DMem_RA_Preload(4),
DMem_RA_Preload(3)=>DMem_RA_Preload(3),
DMem_RA_Preload(2)=>DMem_RA_Preload(2),
DMem_RA_Preload(1)=>DMem_RA_Preload(1),
DMem_RA_Preload(0)=>DMem_RA_Preload(0),
DMem_WA_Preload(7)=>DMem_WA_Preload(7),
DMem_WA_Preload(6)=>DMem_WA_Preload(6),
DMem_WA_Preload(5)=>DMem_WA_Preload(5),
DMem_WA_Preload(4)=>DMem_WA_Preload(4),
DMem_WA_Preload(3)=>DMem_WA_Preload(3),
DMem_WA_Preload(2)=>DMem_WA_Preload(2),
DMem_WA_Preload(1)=>DMem_WA_Preload(1),
DMem_WA_Preload(0)=>DMem_WA_Preload(0),
RF_En_Read_Write=>RF_En_Read_Write_DUMMY,
Mux_RF_din_Sel(1)=>Mux_RF_Din_Sel_DUMMY(1),
Mux_RF_din_Sel(0)=>Mux_RF_Din_Sel_DUMMY(0),
Value_of_Zero(31)=>Value_of_Zero(31),
Value_of_Zero(30)=>Value_of_Zero(30),
Value_of_Zero(29)=>Value_of_Zero(29),
Value_of_Zero(28)=>Value_of_Zero(28),
Value_of_Zero(27)=>Value_of_Zero(27),
Value_of_Zero(26)=>Value_of_Zero(26),
Value_of_Zero(25)=>Value_of_Zero(25),
Value_of_Zero(24)=>Value_of_Zero(24),
Value_of_Zero(23)=>Value_of_Zero(23),
Value_of_Zero(22)=>Value_of_Zero(22),
Value_of_Zero(21)=>Value_of_Zero(21),
Value_of_Zero(20)=>Value_of_Zero(20),
Value_of_Zero(19)=>Value_of_Zero(19),
Value_of_Zero(18)=>Value_of_Zero(18),
Value_of_Zero(17)=>Value_of_Zero(17),
Value_of_Zero(16)=>Value_of_Zero(16),
Value_of_Zero(15)=>Value_of_Zero(15),

```

```

Value_of_Zero(14)=>Value_of_Zero(14),
Value_of_Zero(13)=>Value_of_Zero(13),
Value_of_Zero(12)=>Value_of_Zero(12),
Value_of_Zero(11)=>Value_of_Zero(11),
Value_of_Zero(10)=>Value_of_Zero(10), Value_of_Zero(9)=>Value_of_Zero(9),
Value_of_Zero(8)=>Value_of_Zero(8), Value_of_Zero(7)=>Value_of_Zero(7),
Value_of_Zero(6)=>Value_of_Zero(6), Value_of_Zero(5)=>Value_of_Zero(5),
Value_of_Zero(4)=>Value_of_Zero(4), Value_of_Zero(3)=>Value_of_Zero(3),
Value_of_Zero(2)=>Value_of_Zero(2), Value_of_Zero(1)=>Value_of_Zero(1),
Value_of_Zero(0)=>Value_of_Zero(0),
RF_Write_Din_Preload(31)=>RF_Write_Din_Preload(31),
RF_Write_Din_Preload(30)=>RF_Write_Din_Preload(30),
RF_Write_Din_Preload(29)=>RF_Write_Din_Preload(29),
RF_Write_Din_Preload(28)=>RF_Write_Din_Preload(28),
RF_Write_Din_Preload(27)=>RF_Write_Din_Preload(27),
RF_Write_Din_Preload(26)=>RF_Write_Din_Preload(26),
RF_Write_Din_Preload(25)=>RF_Write_Din_Preload(25),
RF_Write_Din_Preload(24)=>RF_Write_Din_Preload(24),
RF_Write_Din_Preload(23)=>RF_Write_Din_Preload(23),
RF_Write_Din_Preload(22)=>RF_Write_Din_Preload(22),
RF_Write_Din_Preload(21)=>RF_Write_Din_Preload(21),
RF_Write_Din_Preload(20)=>RF_Write_Din_Preload(20),
RF_Write_Din_Preload(19)=>RF_Write_Din_Preload(19),
RF_Write_Din_Preload(18)=>RF_Write_Din_Preload(18),
RF_Write_Din_Preload(17)=>RF_Write_Din_Preload(17),
RF_Write_Din_Preload(16)=>RF_Write_Din_Preload(16),
RF_Write_Din_Preload(15)=>RF_Write_Din_Preload(15),
RF_Write_Din_Preload(14)=>RF_Write_Din_Preload(14),
RF_Write_Din_Preload(13)=>RF_Write_Din_Preload(13),
RF_Write_Din_Preload(12)=>RF_Write_Din_Preload(12),
RF_Write_Din_Preload(11)=>RF_Write_Din_Preload(11),
RF_Write_Din_Preload(10)=>RF_Write_Din_Preload(10),
RF_Write_Din_Preload(9)=>RF_Write_Din_Preload(9),
RF_Write_Din_Preload(8)=>RF_Write_Din_Preload(8),
RF_Write_Din_Preload(7)=>RF_Write_Din_Preload(7),
RF_Write_Din_Preload(6)=>RF_Write_Din_Preload(6),
RF_Write_Din_Preload(5)=>RF_Write_Din_Preload(5),
RF_Write_Din_Preload(4)=>RF_Write_Din_Preload(4),
RF_Write_Din_Preload(3)=>RF_Write_Din_Preload(3),
RF_Write_Din_Preload(2)=>RF_Write_Din_Preload(2),
RF_Write_Din_Preload(1)=>RF_Write_Din_Preload(1),
RF_Write_Din_Preload(0)=>RF_Write_Din_Preload(0),
RF_Write_Num_Preload(4)=>RF_Write_Num_Preload(4),
RF_Write_Num_Preload(3)=>RF_Write_Num_Preload(3),
RF_Write_Num_Preload(2)=>RF_Write_Num_Preload(2),
RF_Write_Num_Preload(1)=>RF_Write_Num_Preload(1),
RF_Write_Num_Preload(0)=>RF_Write_Num_Preload(0),
Mux_RF_Num_Sel(1)=>Mux_RF_Num_Sel_DUMMY(1),
Mux_RF_Num_Sel(0)=>Mux_RF_Num_Sel_DUMMY(0),
Inst_RAM_Write_Addr_Preload(7)=>Inst_RAM_Write_Addr_Preload(7),
Inst_RAM_Write_Addr_Preload(6)=>Inst_RAM_Write_Addr_Preload(6),
Inst_RAM_Write_Addr_Preload(5)=>Inst_RAM_Write_Addr_Preload(5),
Inst_RAM_Write_Addr_Preload(4)=>Inst_RAM_Write_Addr_Preload(4),
Inst_RAM_Write_Addr_Preload(3)=>Inst_RAM_Write_Addr_Preload(3),
Inst_RAM_Write_Addr_Preload(2)=>Inst_RAM_Write_Addr_Preload(2),
Inst_RAM_Write_Addr_Preload(1)=>Inst_RAM_Write_Addr_Preload(1),
Inst_RAM_Write_Addr_Preload(0)=>Inst_RAM_Write_Addr_Preload(0),
Mux_Sel_PC_Clk_Src=>init_exe_inv_DUMMY,
Mux_Sel_IM_Clk_Src=>init_exe_inv_DUMMY, Reset_DCM=>Rest_DCM, Clkin=>Clk,
add8_i2_overflow=>add8_i2_overflow, add8_i2_carryout=>add8_i2_carryout,
Addr_16to8bits(7)=>Addr_16to8bits(7),
Addr_16to8bits(6)=>Addr_16to8bits(6),
Addr_16to8bits(5)=>Addr_16to8bits(5),
Addr_16to8bits(4)=>Addr_16to8bits(4),
Addr_16to8bits(3)=>Addr_16to8bits(3),
Addr_16to8bits(2)=>Addr_16to8bits(2),
Addr_16to8bits(1)=>Addr_16to8bits(1),
Addr_16to8bits(0)=>Addr_16to8bits(0), ALU_Zero_BEQ=>ALU_Zero_BEQ,
BEQ_or_BNE=>BEQ_or_BNE, Mux_Sel_PCSrc1=>Mux_Sel_PCSrc1,
PCplus1_Addr(7)=>PCplus1_Addr(7), PCplus1_Addr(6)=>PCplus1_Addr(6),
PCplus1_Addr(5)=>PCplus1_Addr(5), PCplus1_Addr(4)=>PCplus1_Addr(4),
PCplus1_Addr(3)=>PCplus1_Addr(3), PCplus1_Addr(2)=>PCplus1_Addr(2),
PCplus1_Addr(1)=>PCplus1_Addr(1), PCplus1_Addr(0)=>PCplus1_Addr(0),

```

```

Mux2Mux_PCSrc(7)=>Mux2Mux_PCSrc(7), Mux2Mux_PCSrc(6)=>Mux2Mux_PCSrc(6),
Mux2Mux_PCSrc(5)=>Mux2Mux_PCSrc(5), Mux2Mux_PCSrc(4)=>Mux2Mux_PCSrc(4),
Mux2Mux_PCSrc(3)=>Mux2Mux_PCSrc(3), Mux2Mux_PCSrc(2)=>Mux2Mux_PCSrc(2),
Mux2Mux_PCSrc(1)=>Mux2Mux_PCSrc(1), Mux2Mux_PCSrc(0)=>Mux2Mux_PCSrc(0),
add8_il_overflow=>add8_il_overflow, add8_il_carryout=>add8_il_carryout,
Addr_from_PC(7)=>Addr_from_PC(7), Addr_from_PC(6)=>Addr_from_PC(6),
Addr_from_PC(5)=>Addr_from_PC(5), Addr_from_PC(4)=>Addr_from_PC(4),
Addr_from_PC(3)=>Addr_from_PC(3), Addr_from_PC(2)=>Addr_from_PC(2),
Addr_from_PC(1)=>Addr_from_PC(1), Addr_from_PC(0)=>Addr_from_PC(0),
Inst_RAM_Addr(7)=>Inst_RAM_Addr(7), Inst_RAM_Addr(6)=>Inst_RAM_Addr(6),
Inst_RAM_Addr(5)=>Inst_RAM_Addr(5), Inst_RAM_Addr(4)=>Inst_RAM_Addr(4),
Inst_RAM_Addr(3)=>Inst_RAM_Addr(3), Inst_RAM_Addr(2)=>Inst_RAM_Addr(2),
Inst_RAM_Addr(1)=>Inst_RAM_Addr(1), Inst_RAM_Addr(0)=>Inst_RAM_Addr(0),
DRAM_out(31)=>DRAM_out(31), DRAM_out(30)=>DRAM_out(30),
DRAM_out(29)=>DRAM_out(29), DRAM_out(28)=>DRAM_out(28),
DRAM_out(27)=>DRAM_out(27), DRAM_out(26)=>DRAM_out(26),
DRAM_out(25)=>DRAM_out(25), DRAM_out(24)=>DRAM_out(24),
DRAM_out(23)=>DRAM_out(23), DRAM_out(22)=>DRAM_out(22),
DRAM_out(21)=>DRAM_out(21), DRAM_out(20)=>DRAM_out(20),
DRAM_out(19)=>DRAM_out(19), DRAM_out(18)=>DRAM_out(18),
DRAM_out(17)=>DRAM_out(17), DRAM_out(16)=>DRAM_out(16),
DRAM_out(15)=>DRAM_out(15), DRAM_out(14)=>DRAM_out(14),
DRAM_out(13)=>DRAM_out(13), DRAM_out(12)=>DRAM_out(12),
DRAM_out(11)=>DRAM_out(11), DRAM_out(10)=>DRAM_out(10),
DRAM_out(9)=>DRAM_out(9), DRAM_out(8)=>DRAM_out(8),
DRAM_out(7)=>DRAM_out(7), DRAM_out(6)=>DRAM_out(6),
DRAM_out(5)=>DRAM_out(5), DRAM_out(4)=>DRAM_out(4),
DRAM_out(3)=>DRAM_out(3), DRAM_out(2)=>DRAM_out(2),
DRAM_out(1)=>DRAM_out(1), DRAM_out(0)=>DRAM_out(0),
Instruction_from_IF(31)=>Instruction_from_IF(31),
Instruction_from_IF(30)=>Instruction_from_IF(30),
Instruction_from_IF(29)=>Instruction_from_IF(29),
Instruction_from_IF(28)=>Instruction_from_IF(28),
Instruction_from_IF(27)=>Instruction_from_IF(27),
Instruction_from_IF(26)=>Instruction_from_IF(26),
Instruction_from_IF(25)=>Instruction_from_IF(25),
Instruction_from_IF(24)=>Instruction_from_IF(24),
Instruction_from_IF(23)=>Instruction_from_IF(23),
Instruction_from_IF(22)=>Instruction_from_IF(22),
Instruction_from_IF(21)=>Instruction_from_IF(21),
Instruction_from_IF(20)=>Instruction_from_IF(20),
Instruction_from_IF(19)=>Instruction_from_IF(19),
Instruction_from_IF(18)=>Instruction_from_IF(18),
Instruction_from_IF(17)=>Instruction_from_IF(17),
Instruction_from_IF(16)=>Instruction_from_IF(16),
Instruction_from_IF(15)=>Instruction_from_IF(15),
Instruction_from_IF(14)=>Instruction_from_IF(14),
Instruction_from_IF(13)=>Instruction_from_IF(13),
Instruction_from_IF(12)=>Instruction_from_IF(12),
Instruction_from_IF(11)=>Instruction_from_IF(11),
Instruction_from_IF(10)=>Instruction_from_IF(10),
Instruction_from_IF(9)=>Instruction_from_IF(9),
Instruction_from_IF(8)=>Instruction_from_IF(8),
Instruction_from_IF(7)=>Instruction_from_IF(7),
Instruction_from_IF(6)=>Instruction_from_IF(6),
Instruction_from_IF(5)=>Instruction_from_IF(5),
Instruction_from_IF(4)=>Instruction_from_IF(4),
Instruction_from_IF(3)=>Instruction_from_IF(3),
Instruction_from_IF(2)=>Instruction_from_IF(2),
Instruction_from_IF(1)=>Instruction_from_IF(1),
Instruction_from_IF(0)=>Instruction_from_IF(0),
ALU_Zero_BNE=>ALU_Zero_BNE, Data_RAM_Read_Addr(7)=>Data_RAM_Read_Addr(7),
Data_RAM_Read_Addr(6)=>Data_RAM_Read_Addr(6),
Data_RAM_Read_Addr(5)=>Data_RAM_Read_Addr(5),
Data_RAM_Read_Addr(4)=>Data_RAM_Read_Addr(4),
Data_RAM_Read_Addr(3)=>Data_RAM_Read_Addr(3),
Data_RAM_Read_Addr(2)=>Data_RAM_Read_Addr(2),
Data_RAM_Read_Addr(1)=>Data_RAM_Read_Addr(1),
Data_RAM_Read_Addr(0)=>Data_RAM_Read_Addr(0),
Data_RAM_Write_Addr(7)=>Data_RAM_Write_Addr(7),
Data_RAM_Write_Addr(6)=>Data_RAM_Write_Addr(6),
Data_RAM_Write_Addr(5)=>Data_RAM_Write_Addr(5),
Data_RAM_Write_Addr(4)=>Data_RAM_Write_Addr(4),

```

```

Data_RAM_Write_Addr(3)=>Data_RAM_Write_Addr(3),
Data_RAM_Write_Addr(2)=>Data_RAM_Write_Addr(2),
Data_RAM_Write_Addr(1)=>Data_RAM_Write_Addr(1),
Data_RAM_Write_Addr(0)=>Data_RAM_Write_Addr(0),
DMem_Din(31)=>DMem_Din(31), DMem_Din(30)=>DMem_Din(30),
DMem_Din(29)=>DMem_Din(29), DMem_Din(28)=>DMem_Din(28),
DMem_Din(27)=>DMem_Din(27), DMem_Din(26)=>DMem_Din(26),
DMem_Din(25)=>DMem_Din(25), DMem_Din(24)=>DMem_Din(24),
DMem_Din(23)=>DMem_Din(23), DMem_Din(22)=>DMem_Din(22),
DMem_Din(21)=>DMem_Din(21), DMem_Din(20)=>DMem_Din(20),
DMem_Din(19)=>DMem_Din(19), DMem_Din(18)=>DMem_Din(18),
DMem_Din(17)=>DMem_Din(17), DMem_Din(16)=>DMem_Din(16),
DMem_Din(15)=>DMem_Din(15), DMem_Din(14)=>DMem_Din(14),
DMem_Din(13)=>DMem_Din(13), DMem_Din(12)=>DMem_Din(12),
DMem_Din(11)=>DMem_Din(11), DMem_Din(10)=>DMem_Din(10),
DMem_Din(9)=>DMem_Din(9), DMem_Din(8)=>DMem_Din(8),
DMem_Din(7)=>DMem_Din(7), DMem_Din(6)=>DMem_Din(6),
DMem_Din(5)=>DMem_Din(5), DMem_Din(4)=>DMem_Din(4),
DMem_Din(3)=>DMem_Din(3), DMem_Din(2)=>DMem_Din(2),
DMem_Din(1)=>DMem_Din(1), DMem_Din(0)=>DMem_Din(0),
DMem_to_RF(31)=>DMem_to_RF(31), DMem_to_RF(30)=>DMem_to_RF(30),
DMem_to_RF(29)=>DMem_to_RF(29), DMem_to_RF(28)=>DMem_to_RF(28),
DMem_to_RF(27)=>DMem_to_RF(27), DMem_to_RF(26)=>DMem_to_RF(26),
DMem_to_RF(25)=>DMem_to_RF(25), DMem_to_RF(24)=>DMem_to_RF(24),
DMem_to_RF(23)=>DMem_to_RF(23), DMem_to_RF(22)=>DMem_to_RF(22),
DMem_to_RF(21)=>DMem_to_RF(21), DMem_to_RF(20)=>DMem_to_RF(20),
DMem_to_RF(19)=>DMem_to_RF(19), DMem_to_RF(18)=>DMem_to_RF(18),
DMem_to_RF(17)=>DMem_to_RF(17), DMem_to_RF(16)=>DMem_to_RF(16),
DMem_to_RF(15)=>DMem_to_RF(15), DMem_to_RF(14)=>DMem_to_RF(14),
DMem_to_RF(13)=>DMem_to_RF(13), DMem_to_RF(12)=>DMem_to_RF(12),
DMem_to_RF(11)=>DMem_to_RF(11), DMem_to_RF(10)=>DMem_to_RF(10),
DMem_to_RF(9)=>DMem_to_RF(9), DMem_to_RF(8)=>DMem_to_RF(8),
DMem_to_RF(7)=>DMem_to_RF(7), DMem_to_RF(6)=>DMem_to_RF(6),
DMem_to_RF(5)=>DMem_to_RF(5), DMem_to_RF(4)=>DMem_to_RF(4),
DMem_to_RF(3)=>DMem_to_RF(3), DMem_to_RF(2)=>DMem_to_RF(2),
DMem_to_RF(1)=>DMem_to_RF(1), DMem_to_RF(0)=>DMem_to_RF(0),
A_in(31)=>A_in(31), A_in(30)=>A_in(30), A_in(29)=>A_in(29),
A_in(28)=>A_in(28), A_in(27)=>A_in(27), A_in(26)=>A_in(26),
A_in(25)=>A_in(25), A_in(24)=>A_in(24), A_in(23)=>A_in(23),
A_in(22)=>A_in(22), A_in(21)=>A_in(21), A_in(20)=>A_in(20),
A_in(19)=>A_in(19), A_in(18)=>A_in(18), A_in(17)=>A_in(17),
A_in(16)=>A_in(16), A_in(15)=>A_in(15), A_in(14)=>A_in(14),
A_in(13)=>A_in(13), A_in(12)=>A_in(12), A_in(11)=>A_in(11),
A_in(10)=>A_in(10), A_in(9)=>A_in(9), A_in(8)=>A_in(8), A_in(7)=>A_in(7),
A_in(6)=>A_in(6), A_in(5)=>A_in(5), A_in(4)=>A_in(4), A_in(3)=>A_in(3),
A_in(2)=>A_in(2), A_in(1)=>A_in(1), A_in(0)=>A_in(0),
RF_Data_B(31)=>RF_Data_B(31), RF_Data_B(30)=>RF_Data_B(30),
RF_Data_B(29)=>RF_Data_B(29), RF_Data_B(28)=>RF_Data_B(28),
RF_Data_B(27)=>RF_Data_B(27), RF_Data_B(26)=>RF_Data_B(26),
RF_Data_B(25)=>RF_Data_B(25), RF_Data_B(24)=>RF_Data_B(24),
RF_Data_B(23)=>RF_Data_B(23), RF_Data_B(22)=>RF_Data_B(22),
RF_Data_B(21)=>RF_Data_B(21), RF_Data_B(20)=>RF_Data_B(20),
RF_Data_B(19)=>RF_Data_B(19), RF_Data_B(18)=>RF_Data_B(18),
RF_Data_B(17)=>RF_Data_B(17), RF_Data_B(16)=>RF_Data_B(16),
RF_Data_B(15)=>RF_Data_B(15), RF_Data_B(14)=>RF_Data_B(14),
RF_Data_B(13)=>RF_Data_B(13), RF_Data_B(12)=>RF_Data_B(12),
RF_Data_B(11)=>RF_Data_B(11), RF_Data_B(10)=>RF_Data_B(10),
RF_Data_B(9)=>RF_Data_B(9), RF_Data_B(8)=>RF_Data_B(8),
RF_Data_B(7)=>RF_Data_B(7), RF_Data_B(6)=>RF_Data_B(6),
RF_Data_B(5)=>RF_Data_B(5), RF_Data_B(4)=>RF_Data_B(4),
RF_Data_B(3)=>RF_Data_B(3), RF_Data_B(2)=>RF_Data_B(2),
RF_Data_B(1)=>RF_Data_B(1), RF_Data_B(0)=>RF_Data_B(0),
B_in(31)=>B_in(31), B_in(30)=>B_in(30), B_in(29)=>B_in(29),
B_in(28)=>B_in(28), B_in(27)=>B_in(27), B_in(26)=>B_in(26),
B_in(25)=>B_in(25), B_in(24)=>B_in(24), B_in(23)=>B_in(23),
B_in(22)=>B_in(22), B_in(21)=>B_in(21), B_in(20)=>B_in(20),
B_in(19)=>B_in(19), B_in(18)=>B_in(18), B_in(17)=>B_in(17),
B_in(16)=>B_in(16), B_in(15)=>B_in(15), B_in(14)=>B_in(14),
B_in(13)=>B_in(13), B_in(12)=>B_in(12), B_in(11)=>B_in(11),
B_in(10)=>B_in(10), B_in(9)=>B_in(9), B_in(8)=>B_in(8), B_in(7)=>B_in(7),
B_in(6)=>B_in(6), B_in(5)=>B_in(5), B_in(4)=>B_in(4), B_in(3)=>B_in(3),
B_in(2)=>B_in(2), B_in(1)=>B_in(1), B_in(0)=>B_in(0),
ALU_Res(31)=>ALU_Res(31), ALU_Res(30)=>ALU_Res(30),

```

```

ALU_Res(29)=>ALU_Res(29), ALU_Res(28)=>ALU_Res(28),
ALU_Res(27)=>ALU_Res(27), ALU_Res(26)=>ALU_Res(26),
ALU_Res(25)=>ALU_Res(25), ALU_Res(24)=>ALU_Res(24),
ALU_Res(23)=>ALU_Res(23), ALU_Res(22)=>ALU_Res(22),
ALU_Res(21)=>ALU_Res(21), ALU_Res(20)=>ALU_Res(20),
ALU_Res(19)=>ALU_Res(19), ALU_Res(18)=>ALU_Res(18),
ALU_Res(17)=>ALU_Res(17), ALU_Res(16)=>ALU_Res(16),
ALU_Res(15)=>ALU_Res(15), ALU_Res(14)=>ALU_Res(14),
ALU_Res(13)=>ALU_Res(13), ALU_Res(12)=>ALU_Res(12),
ALU_Res(11)=>ALU_Res(11), ALU_Res(10)=>ALU_Res(10),
ALU_Res(9)=>ALU_Res(9), ALU_Res(8)=>ALU_Res(8), ALU_Res(7)=>ALU_Res(7),
ALU_Res(6)=>ALU_Res(6), ALU_Res(5)=>ALU_Res(5), ALU_Res(4)=>ALU_Res(4),
ALU_Res(3)=>ALU_Res(3), ALU_Res(2)=>ALU_Res(2), ALU_Res(1)=>ALU_Res(1),
ALU_Res(0)=>ALU_Res(0), ALU_Overflow=>ALU_Overflow,
ALU_Carryout=>ALU_Carryout, Addr_26bits(25)=>Addr_26bits(25),
Addr_26bits(24)=>Addr_26bits(24), Addr_26bits(23)=>Addr_26bits(23),
Addr_26bits(22)=>Addr_26bits(22), Addr_26bits(21)=>Addr_26bits(21),
Addr_26bits(20)=>Addr_26bits(20), Addr_26bits(19)=>Addr_26bits(19),
Addr_26bits(18)=>Addr_26bits(18), Addr_26bits(17)=>Addr_26bits(17),
Addr_26bits(16)=>Addr_26bits(16), Addr_26bits(15)=>Addr_26bits(15),
Addr_26bits(14)=>Addr_26bits(14), Addr_26bits(13)=>Addr_26bits(13),
Addr_26bits(12)=>Addr_26bits(12), Addr_26bits(11)=>Addr_26bits(11),
Addr_26bits(10)=>Addr_26bits(10), Addr_26bits(9)=>Addr_26bits(9),
Addr_26bits(8)=>Addr_26bits(8), Addr_26bits(7)=>Addr_26bits(7),
Addr_26bits(6)=>Addr_26bits(6), Addr_26bits(5)=>Addr_26bits(5),
Addr_26bits(4)=>Addr_26bits(4), Addr_26bits(3)=>Addr_26bits(3),
Addr_26bits(2)=>Addr_26bits(2), Addr_26bits(1)=>Addr_26bits(1),
Addr_26bits(0)=>Addr_26bits(0), Addr_16to32bits(31)=>Addr_16to32bits(31),
Addr_16to32bits(30)=>Addr_16to32bits(30),
Addr_16to32bits(29)=>Addr_16to32bits(29),
Addr_16to32bits(28)=>Addr_16to32bits(28),
Addr_16to32bits(27)=>Addr_16to32bits(27),
Addr_16to32bits(26)=>Addr_16to32bits(26),
Addr_16to32bits(25)=>Addr_16to32bits(25),
Addr_16to32bits(24)=>Addr_16to32bits(24),
Addr_16to32bits(23)=>Addr_16to32bits(23),
Addr_16to32bits(22)=>Addr_16to32bits(22),
Addr_16to32bits(21)=>Addr_16to32bits(21),
Addr_16to32bits(20)=>Addr_16to32bits(20),
Addr_16to32bits(19)=>Addr_16to32bits(19),
Addr_16to32bits(18)=>Addr_16to32bits(18),
Addr_16to32bits(17)=>Addr_16to32bits(17),
Addr_16to32bits(16)=>Addr_16to32bits(16),
Addr_16to32bits(15)=>Addr_16to32bits(15),
Addr_16to32bits(14)=>Addr_16to32bits(14),
Addr_16to32bits(13)=>Addr_16to32bits(13),
Addr_16to32bits(12)=>Addr_16to32bits(12),
Addr_16to32bits(11)=>Addr_16to32bits(11),
Addr_16to32bits(10)=>Addr_16to32bits(10),
Addr_16to32bits(9)=>Addr_16to32bits(9),
Addr_16to32bits(8)=>Addr_16to32bits(8),
Addr_16to32bits(7)=>Addr_16to32bits(7),
Addr_16to32bits(6)=>Addr_16to32bits(6),
Addr_16to32bits(5)=>Addr_16to32bits(5),
Addr_16to32bits(4)=>Addr_16to32bits(4),
Addr_16to32bits(3)=>Addr_16to32bits(3),
Addr_16to32bits(2)=>Addr_16to32bits(2),
Addr_16to32bits(1)=>Addr_16to32bits(1),
Addr_16to32bits(0)=>Addr_16to32bits(0), Addr_16bits(15)=>Addr_16bits(15),
Addr_16bits(14)=>Addr_16bits(14), Addr_16bits(13)=>Addr_16bits(13),
Addr_16bits(12)=>Addr_16bits(12), Addr_16bits(11)=>Addr_16bits(11),
Addr_16bits(10)=>Addr_16bits(10), Addr_16bits(9)=>Addr_16bits(9),
Addr_16bits(8)=>Addr_16bits(8), Addr_16bits(7)=>Addr_16bits(7),
Addr_16bits(6)=>Addr_16bits(6), Addr_16bits(5)=>Addr_16bits(5),
Addr_16bits(4)=>Addr_16bits(4), Addr_16bits(3)=>Addr_16bits(3),
Addr_16bits(2)=>Addr_16bits(2), Addr_16bits(1)=>Addr_16bits(1),
Addr_16bits(0)=>Addr_16bits(0), Next_PC(7)=>Next_PC(7),
Next_PC(6)=>Next_PC(6), Next_PC(5)=>Next_PC(5), Next_PC(4)=>Next_PC(4),
Next_PC(3)=>Next_PC(3), Next_PC(2)=>Next_PC(2), Next_PC(1)=>Next_PC(1),
Next_PC(0)=>Next_PC(0), funct(5)=>Funct_DUMMY(5),
funct(4)=>Funct_DUMMY(4), funct(3)=>Funct_DUMMY(3),
funct(2)=>Funct_DUMMY(2), funct(1)=>Funct_DUMMY(1),
funct(0)=>Funct_DUMMY(0), shamt=>open, rd(4)=>rd(4), rd(3)=>rd(3),

```



```

rd(2)=>rd(2), rd(1)=>rd(1), rd(0)=>rd(0), rt(4)=>rt(4), rt(3)=>rt(3),
rt(2)=>rt(2), rt(1)=>rt(1), rt(0)=>rt(0), opcode(5)=>Opcode_DUMMY(5),
opcode(4)=>Opcode_DUMMY(4), opcode(3)=>Opcode_DUMMY(3),
opcode(2)=>Opcode_DUMMY(2), opcode(1)=>Opcode_DUMMY(1),
opcode(0)=>Opcode_DUMMY(0), rs(4)=>rs(4), rs(3)=>rs(3), rs(2)=>rs(2),
rs(1)=>rs(1), rs(0)=>rs(0), RF_Write_Data(31)=>RF_Write_Data(31),
RF_Write_Data(30)=>RF_Write_Data(30),
RF_Write_Data(29)=>RF_Write_Data(29),
RF_Write_Data(28)=>RF_Write_Data(28),
RF_Write_Data(27)=>RF_Write_Data(27),
RF_Write_Data(26)=>RF_Write_Data(26),
RF_Write_Data(25)=>RF_Write_Data(25),
RF_Write_Data(24)=>RF_Write_Data(24),
RF_Write_Data(23)=>RF_Write_Data(23),
RF_Write_Data(22)=>RF_Write_Data(22),
RF_Write_Data(21)=>RF_Write_Data(21),
RF_Write_Data(20)=>RF_Write_Data(20),
RF_Write_Data(19)=>RF_Write_Data(19),
RF_Write_Data(18)=>RF_Write_Data(18),
RF_Write_Data(17)=>RF_Write_Data(17),
RF_Write_Data(16)=>RF_Write_Data(16),
RF_Write_Data(15)=>RF_Write_Data(15),
RF_Write_Data(14)=>RF_Write_Data(14),
RF_Write_Data(13)=>RF_Write_Data(13),
RF_Write_Data(12)=>RF_Write_Data(12),
RF_Write_Data(11)=>RF_Write_Data(11),
RF_Write_Data(10)=>RF_Write_Data(10), RF_Write_Data(9)=>RF_Write_Data(9),
RF_Write_Data(8)=>RF_Write_Data(8), RF_Write_Data(7)=>RF_Write_Data(7),
RF_Write_Data(6)=>RF_Write_Data(6), RF_Write_Data(5)=>RF_Write_Data(5),
RF_Write_Data(4)=>RF_Write_Data(4), RF_Write_Data(3)=>RF_Write_Data(3),
RF_Write_Data(2)=>RF_Write_Data(2), RF_Write_Data(1)=>RF_Write_Data(1),
RF_Write_Data(0)=>RF_Write_Data(0), RF_Write_Reg(4)=>RF_Write_Reg(4),
RF_Write_Reg(3)=>RF_Write_Reg(3), RF_Write_Reg(2)=>RF_Write_Reg(2),
RF_Write_Reg(1)=>RF_Write_Reg(1), RF_Write_Reg(0)=>RF_Write_Reg(0),
Addr_26to8bits(7)=>Addr_26to8bits(7),
Addr_26to8bits(6)=>Addr_26to8bits(6),
Addr_26to8bits(5)=>Addr_26to8bits(5),
Addr_26to8bits(4)=>Addr_26to8bits(4),
Addr_26to8bits(3)=>Addr_26to8bits(3),
Addr_26to8bits(2)=>Addr_26to8bits(2),
Addr_26to8bits(1)=>Addr_26to8bits(1),
Addr_26to8bits(0)=>Addr_26to8bits(0), Branch_Target(7)=>Branch_Target(7),
Branch_Target(6)=>Branch_Target(6), Branch_Target(5)=>Branch_Target(5),
Branch_Target(4)=>Branch_Target(4), Branch_Target(3)=>Branch_Target(3),
Branch_Target(2)=>Branch_Target(2), Branch_Target(1)=>Branch_Target(1),
Branch_Target(0)=>Branch_Target(0), Mux_to_PC_Clk=>Mux_to_PC_Clk,
Mux_to_IM_Clk=>Mux_to_IM_Clk, Clk0=>Clk0, ClkDv=>ClkDv,
DCM_Locked=>DCM_Locked);

FD3 : FD
-- synopsys translate_off
GENERIC MAP (      INIT => '0'      )
-- synopsys translate_on
PORT MAP (C=>Clk, D=>Branch_DUMMY, Q=>Branch_1delay_DUMMY);

FD1 : FD
-- synopsys translate_off
GENERIC MAP (      INIT => '0'      )
-- synopsys translate_on
PORT MAP (C=>Clk, D=>init_exe_inv_DUMMY, Q=>init_exe_inv_1delay_DUMMY);

FD2 : FD
-- synopsys translate_off
GENERIC MAP (      INIT => '0'      )
-- synopsys translate_on
PORT MAP (C=>Clk, D=>init_exe_inv_1delay_DUMMY,
Q=>init_exe_inv_2delays_DUMMY);

FunctBustap : funct_bustap
PORT MAP (funct(5)=>Funct_DUMMY(5), funct(4)=>Funct_DUMMY(4),
funct(3)=>Funct_DUMMY(3), funct(2)=>Funct_DUMMY(2),
funct(1)=>Funct_DUMMY(1), funct(0)=>Funct_DUMMY(0), f3=>F3, f2=>F2,
f1=>F1, f0=>F0);

```

```

Gnd_i1 : GND
  PORT MAP (G=>Value_of_0);

XLXI_82 : INV
  PORT MAP (I=>init_exe, O=>init_exe_inv_DUMMY);

M2_1_i2 : M2_1_MXILINX_complete_datapath_and_control
  PORT MAP (D0=>MemtoReg0_RegDst0_MemRead_DMemRASrc_DUMMY,
    D1=>DMemRASrc_init, S0=>init_exe, O=>Sel_DMem_Mux_ra_DUMMY);

M2_1_i3 : M2_1_MXILINX_complete_datapath_and_control
  PORT MAP (D0=>MemWrite_DMemDinSrc_DMemWASrc_DUMMY, D1=>DMemDinSrc_init,
    S0=>init_exe, O=>Sel_DMem_Mux_di_DUMMY);

M2_1_i1 : M2_1_MXILINX_complete_datapath_and_control
  PORT MAP (D0=>MemWrite_DMemDinSrc_DMemWASrc_DUMMY, D1=>DMemWASrc_init,
    S0=>init_exe, O=>Sel_DMem_Mux_wa_DUMMY);

MainControl : main_control
  PORT MAP (Op4=>Op4, Op5=>Op5, Op1=>Op1, Op0=>Op0, Op3=>Op3, Op2=>Op2,
    Jump=>Jump_DUMMY, SelBEQorBNE=>SelBEQorBNE_DUMMY,
    MemWrite_DMemDinSrc_DMemWASrc=>MemWrite_DMemDinSrc_DMemWASrc_DUMMY,

    MemtoReg0_RegDst0_MemRead_DMemRASrc=>MemtoReg0_RegDst0_MemRead_DMemRASrc_DUMMY,
    ALUOp1_RegDst1=>ALUOp1_RegDst1_DUMMY, ALUOp0=>ALUOp0_DUMMY,
    ALUSrc=>ALUSrc_DUMMY, Branch=>Branch_DUMMY,
    ALUMuxEn_RegRW=>ALUMuxEn_RegRW_DUMMY, MemtoReg1=>MemtoReg1_DUMMY);

mux2b_2to1_i2 : mux2b_2to1
  PORT MAP (sel=>init_exe, din0(1)=>MemtoReg_DUMMY(1),
    din0(0)=>MemtoReg_DUMMY(0), din1(1)=>MemtoReg_init(1),
    din1(0)=>MemtoReg_init(0), dout(1)=>Mux_RF_Din_Sel_DUMMY(1),
    dout(0)=>Mux_RF_Din_Sel_DUMMY(0));

mux2b_2to1_i1 : mux2b_2to1
  PORT MAP (sel=>init_exe, din0(1)=>RegDst_DUMMY(1),
    din0(0)=>RegDst_DUMMY(0), din1(1)=>RegDst_init(1),
    din1(0)=>RegDst_init(0), dout(1)=>Mux_RF_Num_Sel_DUMMY(1),
    dout(0)=>Mux_RF_Num_Sel_DUMMY(0));

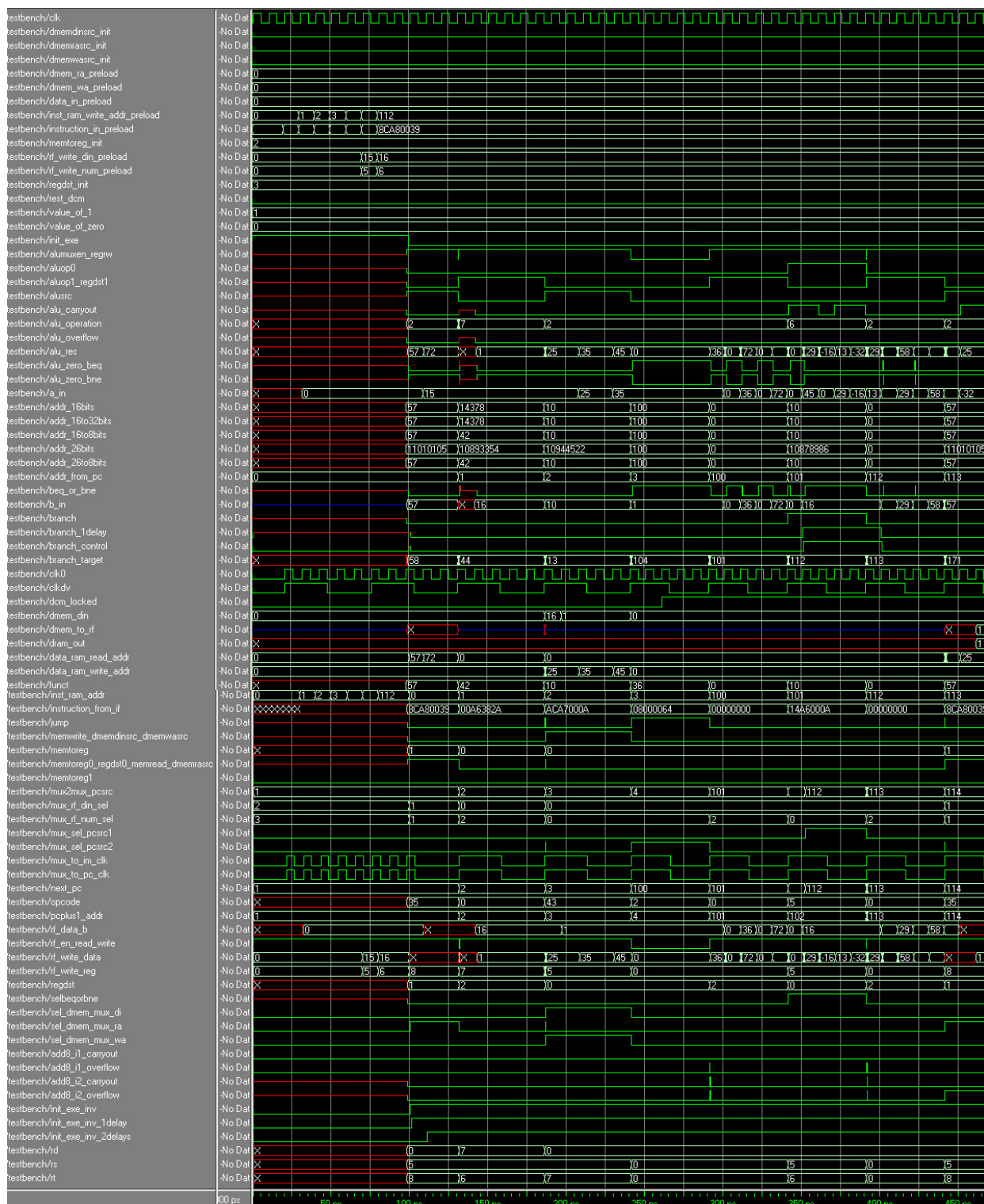
OpcodeBustap : opcode_bustap
  PORT MAP (opcode(5)=>Opcode_DUMMY(5), opcode(4)=>Opcode_DUMMY(4),
    opcode(3)=>Opcode_DUMMY(3), opcode(2)=>Opcode_DUMMY(2),
    opcode(1)=>Opcode_DUMMY(1), opcode(0)=>Opcode_DUMMY(0), op5=>Op5,
    op4=>Op4, op3=>Op3, op2=>Op2, op1=>Op1, op0=>Op0);

OR2_i1 : OR2
  PORT MAP (I0=>init_exe, I1=>ALUMuxEn_RegRW_DUMMY,
    O=>RF_En_Read_Write_DUMMY);

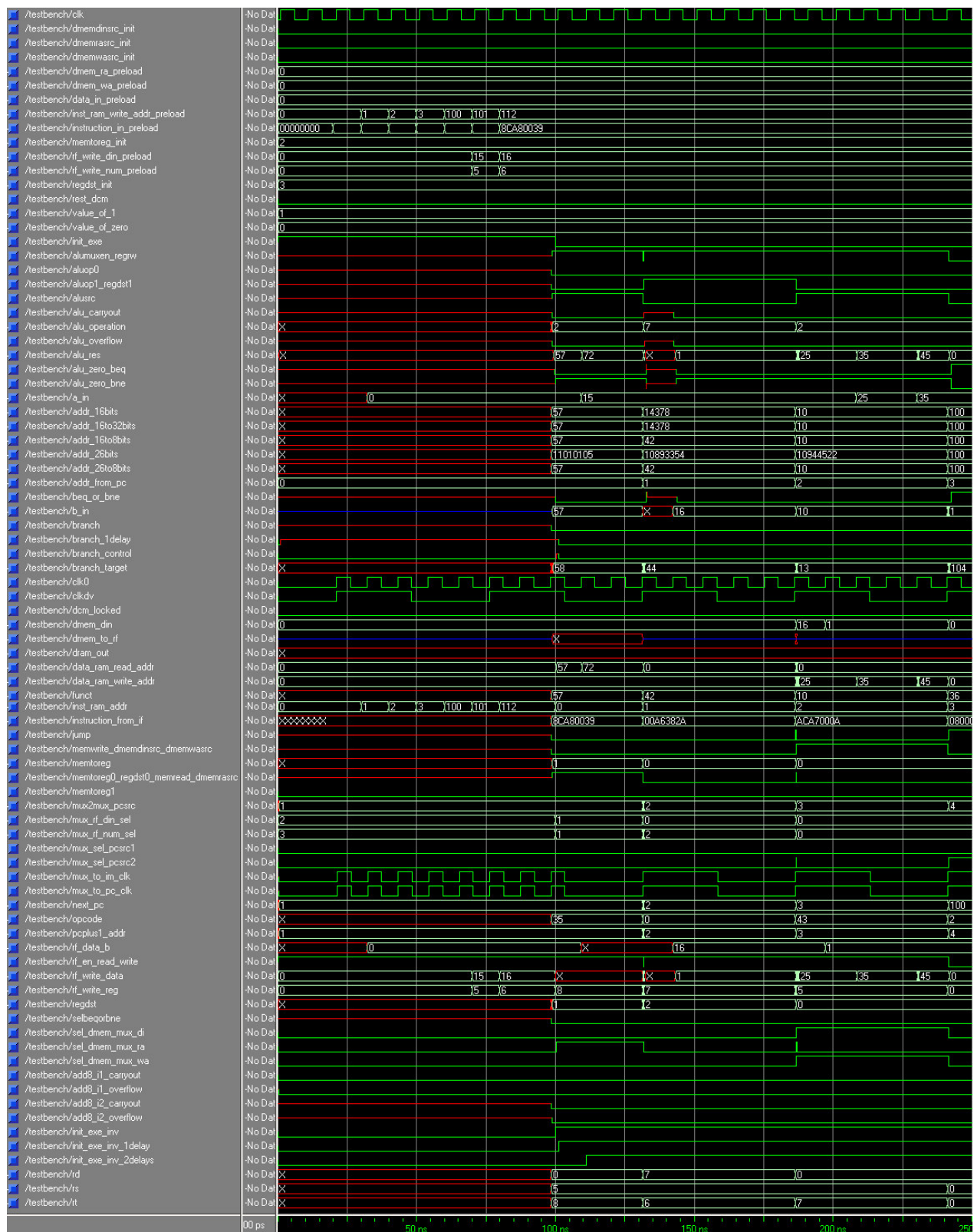
END SCHEMATIC;

```

D.5 High Resolution Figure 6.6 (Full View Version)



D.5A High Resolution Figure 6.6A



D.5B High Resolution Figure 6.6B

