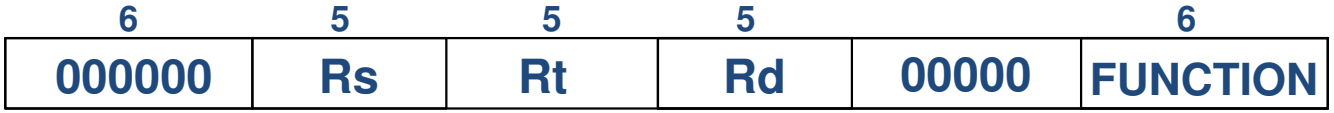
HW4 – “Rtype” MIPS

[capable of perfomming Rtype intstructions (without jr) and also addi, beq, bne, j]

HW4 instruction set

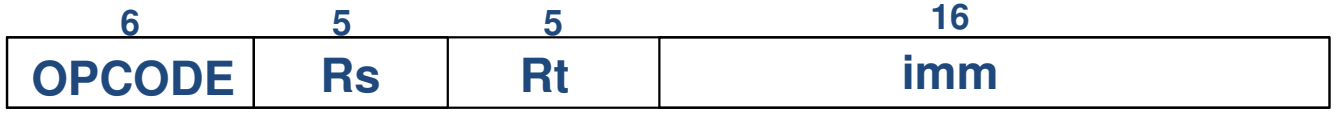
R-type

```
add  Rd, Rs, Rt      #  Rd=Rs+Rt
sub Rd, Rs, Rt       #  Rd=Rs-Rt
and Rd, Rs, Rt       #  Rd=Rs AND Rt
or Rd, Rs, Rt        #  Rd=Rs OR Rt
xor Rd, Rs, Rt       #  Rd=Rs XOR Rt
slt Rd, Rs, Rt       #  if Rs<Rt  Rd=1 else Rd=0
```



I-type

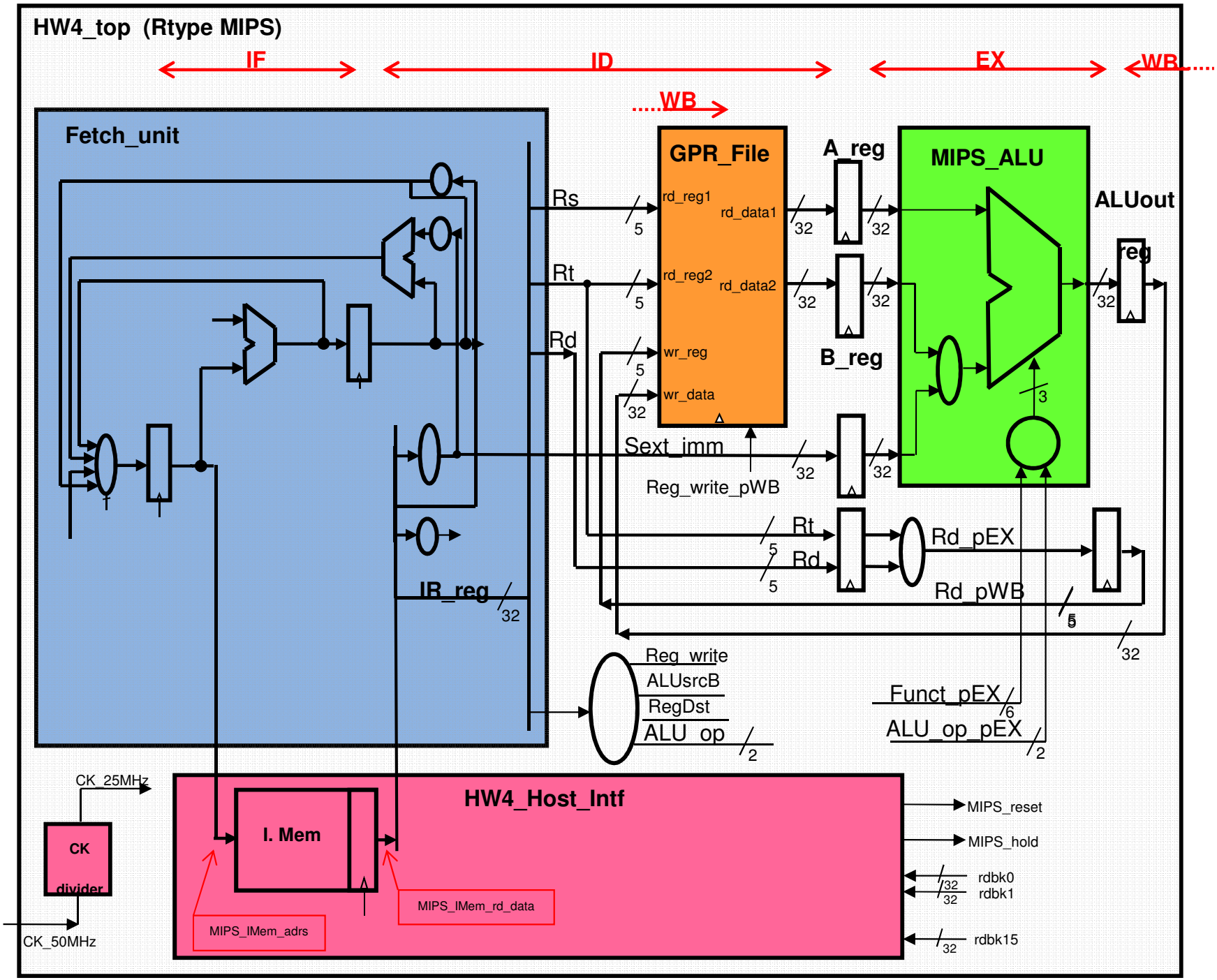
```
addi  Rt, Rs, imm    #  Rt=Rs+ sext(imm)
beq  Rs, Rt, label   #  if Rs==Rt,  PC=PC+4+ sext(imm)*4
                     #  else        PC=PC+4
bne  Rs, Rt, label   #  same as beq with cond of Rs≠Rt
```



j-type

```
j    imm             #  PC= imm*4          (no sext)
```

HW4_MIPS (Rtype MIPS)



The HW4_MIPS CPU has four phases.

IF – Instruction Fetch, which is carried out inside the Fetch Unit producing the instruction in the IR_reg at the rising edge of the clock which ends the IF phase and starts the ID phase.

ID – Instruction Decode, which is the stage in which we do the following:

Decode the instruction residing now at the IR_reg and decide what should be done. This means, we produce all control signals to be used by that instruction in all phases of this instruction – ID, ED and WB.

Read Rs into A_reg and Rt into B_reg

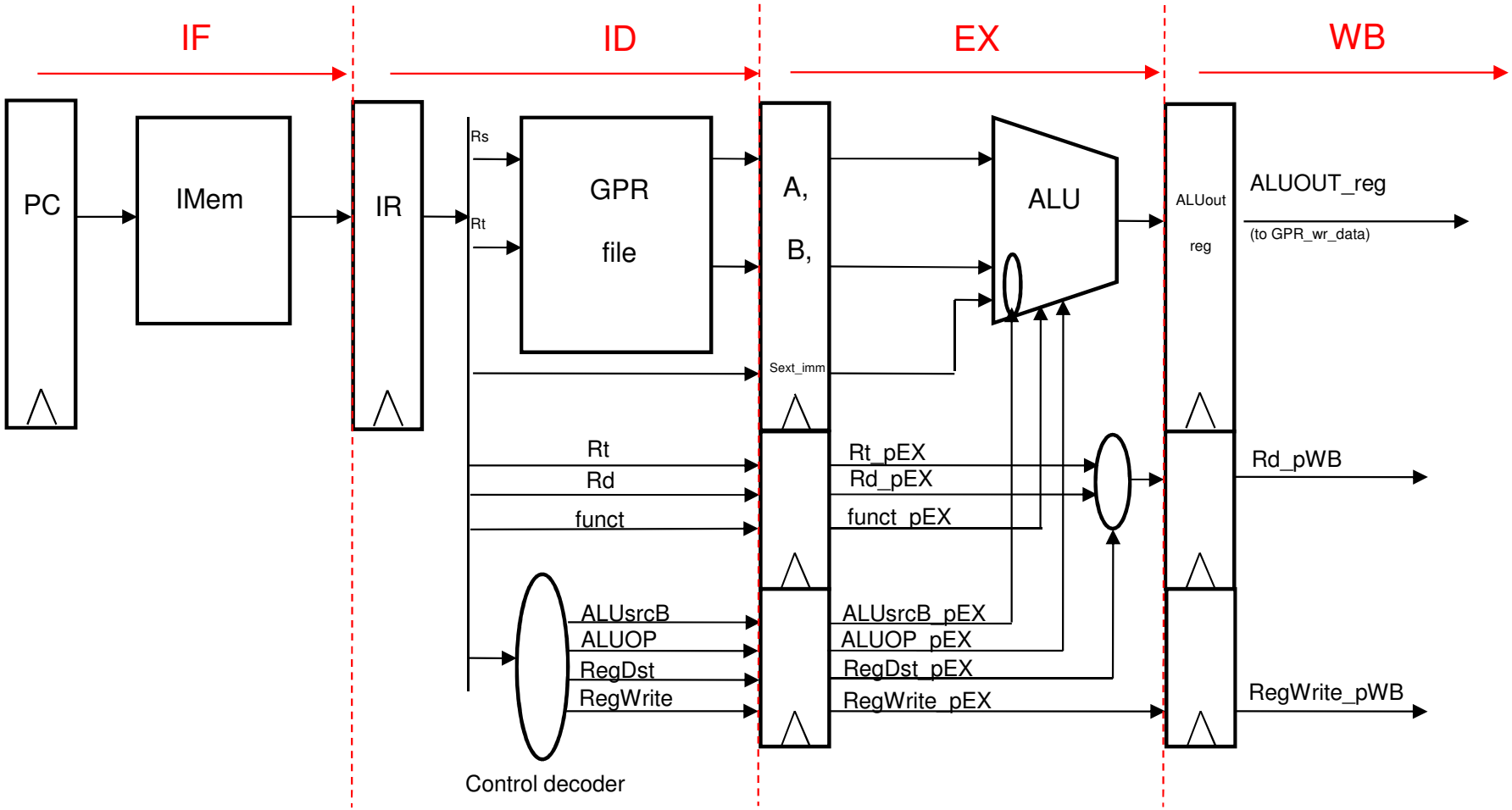
The rising edge of the clock sampling data into the A_reg and B_reg ends the ID phase and starts the EX phase.

EX – Execute, which is the phase in which the ALU calculates the result of A op B (in *Rtype* instructions) or A+sext_imm (in *addi* instructions). The result is sampled into the ALUout_reg at the rising edge of the clock which ends the EX phase and starts the WB phase.

In this phase we also select Rs or Rd as the GPR file destination register to be written into in the Write Back phase.

WB – Write Back, which is the final phase of the instruction. If this is an *Rtype* or *addi* instruction, then we write the ALUout_reg value into the GPR file. If this is a *j*, *beq* or *bne* instruction, we do nothing at that stage. The rising edge of the clock sampling data into the GPR File ends the WB phase and completes the instruction.

HW4_MIPS control scheme



Modifications required in the Fetch Unit

1. **We remove all rdbk0-15 output signals from the Fetch Unit**. We hope we won't need them since the Fetch Unit is already debugged and the changes we introduce are minor.

2. Instead **we add output signals coming out of the Fetch_Unit that should be used by the rest of the CPU**. These *output* signals are:

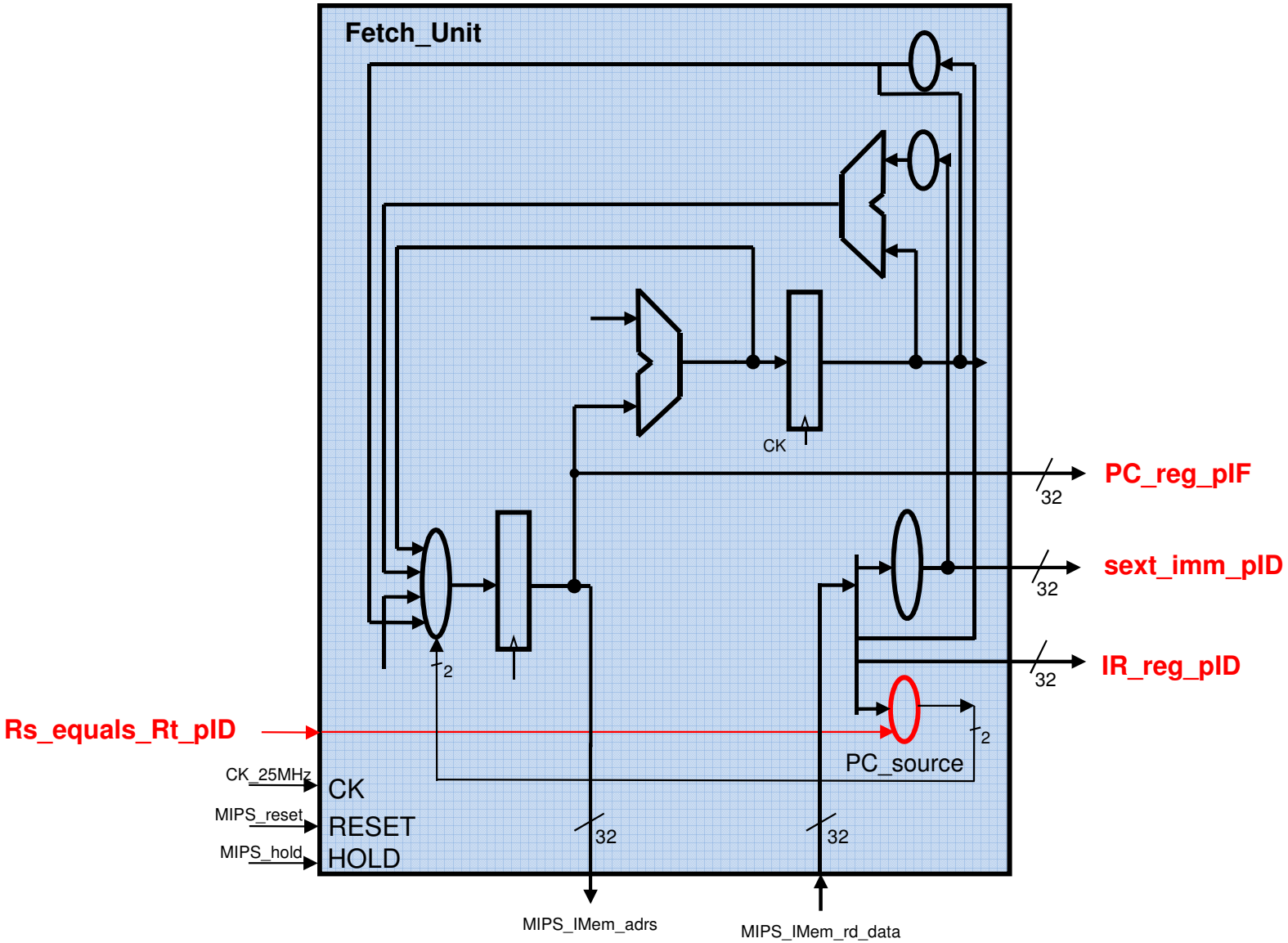
- IR_reg_pID - This is a 32 bit signal of the IR_reg (the instruction bits). We added pID to that signal name to indicate it is the IR_reg value at the ID phase.
- sext_imm_pID - Similarly, this is the 32 bit sext_imm signal we calculate at the ID phase. It is outputted from the Fetch Unit to be used later in the EX phase.
- PC_reg_pIF - this is the 32 bit PC_reg we use at the IF phase to access the IMem. It is outputted from the Fetch Unit to be used for verification purposes only.

3. Note that for TB purposes we output the CK_out_to_TB, RESET_out_to_TB, HOLD_out_to_TB signals from the **HW4_top_4sim.vhd** which is our top component (see section c below).

4. **We add the** Rs_equals_Rt_pID signal that tells us whether to branch in **beq** (if it is '1') or not (if it is '0'). This signal should come from comparing the two data outputs of the GPR File which resides outside the Fetch_Unit. You should modify the PC_source process so that the **beq** and **bne** instructions are properly performed.

5. Make sure that the **addi** instruction is also supported.

The modified Fetch_Unit



General signals in HW4_top

CK - The 25 MHz clock coming out of the Clock_driver.

RESET – coming out of the Host_Intf and is used as reset signal to all registers

HOLD –coming out of the Host_Intf and is used to freeze writing into all FFs & registers

ID phase signals in HW4_top

IR_reg- a 32 bit register that has the instruction we read from the IMem.

This signal is a rename of the IR_reg_pID signal coming out of the modified Fetch Unit

Opcode – the 6 MSBs of IR_reg. To be decoded and produce the control signals.

Rs – IR[25:21].

Rt – IR[20:16] .

Rd – IR[15:11].

Funct – IR[5:0].

sext_imm – renaming of sext_imm_pID coming out of the Fetch Unit.

GPR_rd_data1 – the 32 bit output of the rd_data1 of the GPR and input to A_reg.

GPR_rd_data2 – the 32 bit output of the rd_data2 of the GPR and input to B_reg.

Rs_equals_Rt – '1' if GPR_rd_data1== GPR_rd_data2, and '0' otherwise.

Used in branch instructions. That signal (renamed) is sent to the Fetch Unit.

ID control signals in HW4_top - These are created from decoding the opcode:

ALUsrcB – '1' when sext_imm is used (in addi insruction).

ALUOP – a 2 bit signal. "10" will cause the ALU to follow the Funct field – in Rtype Instructions, "01" means subtract (for beq & bne - compatible to the Single Cycle MIPS), ,"00" maens add (all other instructions including addi).

RegDst – '1' when WB is according to Rd (Rtype inst.); 0- according to Rt (all other inst.)

RegWrite – '1' when we WB (Rtype or addi inst.), '0' when we don't (j, beq & bne inst.)

EX phase signals in HW4_top

A_reg – a 32 bit register receiving the GPR_rd_data1 signal. Its value is used in EX phase

B_reg – a 32 bit register receiving the GPR_rd_data2 signal

sext_imm_reg – a 32 bit register receiving the sext_imm coming from the Fetch Unit

Rt_pEX – Rt delayed by 1 clock cycle

Rd_pEX – Rd delayed by 1 clock cycle

ALUoutput – a 32 bit signal of the output of the ALU (renaming of ALU_out signal coming out of the MIPS_ALU component).

EX phase control signals in HW4_top

ALUsrcB_pEX – ALUsrcB delayed by 1 clock cycle.

Funct_pEX – Funct delayed by 1 clock cycle.

ALUOP_pEX – ALUOP delayed by 1 clock cycle.

RegDst_pEX – RegDst delayed by 1 clock cycle.

RegWrite_pEX – RegWrite delayed by 1 clock cycle.

WB phase signals in HW4_top

ALUout_reg – a 32 bit register getting the ALUout signal at its input.

Rd_pWB – the output of RegDst mux selecting to which register the CPU writes in WB phase.

WB phase control signals in HW4_top

RegWrite_pWB – RegWrite_pEX delayed by 1 clock cycle.

You get a **HW4_top_4sim.empty** file in which you have all of these signals defined . You have to add your design of the HW4_top, i.e., write the equations of the top file. In this vhd file we use the **Fetch_Unit**, **GPR**, **MIPS_ALU**, **Clock_Driver** and the **HW4_Host_Intf_4sim**.

Description of the HW4_top_4sim project

1. **HW4_top_4sim.vhd** – This is your design of HW4. It uses the **GPR**, **MIPS_ALU** the updated **Fetch_Unit**, the **Clock_driver** and the **Host_Intf_4sim** components and all of the signals described in 2b.
2. **GPR.vhd** – your GPR File design you prepared in HW3.
3. **dual_port_memory.vhd** – part of the GPR File design you prepared in HW3.
4. **MIPS_ALU.vhd** – your MIPS_ALU design you prepared in HW3.
5. **Fetch_Unit.vhd** - The Fetch Unit you prepared in HW2 after the modifications detailed in section 2a above.

6. **BYOC_Clock_driver_4sim.vhd** – the CK divider & driver we use for simulation (also good for the Modelsim simulator)
7. **BYOC_Host_Intf_4sim.vhd** – The prepared components including the IMem and "pre-loaded" program and creating the reset & ck signals.

8. **SIM_HW4_TB.vhd** - The TB vhd file prepared in advance
9. **SIM_HW4_TB_data.dat** – this data file prepared in advance that is read by the SIM_HW4_TB and used to compare the simulation results to the expected ones.
10. **SIM_HW4_program.dat** - The program file for simulation.
11. **SIM_HW4_filenames.vhd** - The actual path information of the two dat files.

NOTE: In **SIM_HW4_filenames.vhd** we specified the path of the **dat** file. You should update that according to your simulation project actual path.

Simulation report

You should submit a single zip file for the Simulation and implementation phases. It should have two directories/folders. The first is called **Simulation**, the 2nd is called **Implementation**.

In the **Simulation** folder you will have 3 sub-folder of:

- **Src_4sim** – here you put all of the *.vhd sources and the *.dat file (to be used by the the TB)
- **Sim** – here you should have the HW4_4sim project created by the simulator you used
- **Docs** – Here you put your simulation report. The first few lines in the report will have your ID numbers (names are optional). See instructions in BYOC_HW4.doc for the rest of the simulation report.
  ***This should be a WORD file and not a PDF file – so remarks can be added when grading the report*.**

Later, in the Implementation phase you will add 3 sub-folders to the **Implementation** folder. These will be:

- **Src_4ISE** – here you put all of the *.vhd sources and the *.ucf file (and no TB file)
- **ISE** – here you should have the HW4_top project created by the Xilinx ISE SW.
- **Docs** - Here you put your implementation report. See instructions in BYOC_HW4.doc.

HW4_MIPS - Rtype only CPU - implementation

1. Rename **HW4_top_4sim.vhd** file to **HW4_top.vhd** & remove all of the TB signals (CK_out_to_TB, RESET_out_to_TB, HOLD_out_to_TB & rdbk0_out_to_TB-rdbk15_out_to_TB). Use the **HW4_top.empty** that is similar to the **HW4_top_4sim.empty** with the above mentioned changes.

2. You should replace the **HW4_Host_Intf_4sim.vhd** with a component that looks the same, the **HW4_Host_Intf.ngc**, which has the infra-structure that allows the PC to load data into the IMem via the RS232 by the **BYOCInterface** SW.

3. The files we will use to implement the design on the Nexys2 board are:

- **BYOC.ucf** - The file listing which signal are connected to which FPGA pins in the Nexys2 board.
- **HW4_top.vhd** – This is your design of HW4
- **Fetch_Unit.vhd** - The Fetch Unit you prepared in HW2 after modifications of section 2a.
- **GPR.vhd** – your GPR File design you prepared in HW3.
- **dual_port_memory.vhd** – part of the GPR File design you prepared in HW3.
- **MIPS_ALU.vhd** – your MIPS_ALU design you prepared in HW3.
- **Clock_driver.vhd** – the CK divider & driver we also used in HW2.
- **BYOC_Host_Intf.ngc** - The Host Interface infrastructure interfacing the PC.

4. Now run the Xilinx ISE SW, create a HW4_top.bit file and test it.

HW4_top – implementation report

You should submit a single zip file for the Simulation and implementation phases. It should have two directories/folders. The first is called **Simulation**, the 2nd is called **Implementation**. In the **Implementation** directory you should have 3 sub-directories:

- **Src_4ISE** – here you put all of the *.vhd sources and the *.ucf file (and no TB file)
- **ISE** – here you should have the HW4 project created by the Xilinx ISE SW.
- **Docs** - Here you put your implementation report. The first few lines in the report will have your ID numbers (names are optional). See instructions in BYOC_HW4.doc.

***This should be a WORD file and not a PDF file – so remarks can be added when grading the report*.**

As part of completing this part of the course you will have to show me how you run the design on the Nexys2 board in the lab. And maybe answer some questions.

Now it is your turn!

Thanks for listening!