

## Lab Intro and Objectives

The main objective of this lab is to take the pipelined CPU we have made in the last lab and bring in some new hardware, a forwarding unit and hazard detection unit, to create our final (though not feature complete) CPU! These units are entirely combinational (dataflow), and require only minor modifications to our existing design, more on this in the next section.

## Brief Component Design Descriptions

The new components are as follows: forwardingmux, forwardingunit, hazarddetector, and hazardmux. In addition, other port map changes and additional signals were added to certain pipeline registers and muxes. One major change is to ID\_EX, where now an enable signal is required to pass values through the pipeline.

All new components use dataflow modeling, as they only detect conditions and give a corresponding output.

## A Brief Explainer on My Testbenches

All testbenches are modeled using structural as they require their to be tested components in their architecture. In this case, as pipecpu1.tb is a “black box” in terms of its outputs, only giving debug outputs. It runs a program from IMEM, which then allows me to run GTKWave to be able to see signals within the CPU for debugging.

## Results and Waveforms

### Test Program 1: p1, testing new instructions

LDUR	X9, [XZR, 0]	11111000010000000000001111101001
ADD	X9, X9, X9	10001011000010010000000100101001
ADD	X10, X9, X9	10001011000010010000000100101010
SUB	X11, X10, X9	11001011000010010000000101001011
STUR	X11, [XZR, 8]	11111000000000001000001111101011
STUR	X11, [XZR, 16]	1111100000000000100000001111101011

```

NOP                                00000000000000000000000000000000
NOP                                00000000000000000000000000000000
NOP                                00000000000000000000000000000000
NOP                                00000000000000000000000000000000

```

Registers (all 64-bit, unrepresented zeros for readability) are initialized to:

- $X9 = 0x01$
- All other registers ( $X10$  to  $X12$ ,  $X19$  to  $X22$ ) are initialized to  $0x00$ .

Data Memory contents has 64-bit values initialized to:

- $DMEM(0x00) = 0x0...01$
- $DMEM(0x08) = 0x0...00$
- $DMEM(0x10) = 0x0...00$
- $DMEM(0x18) = 0x0...00$
- $DMEM(0x2E) = 0x0...01$
- All other DMEM are initialized to  $0x00$ .

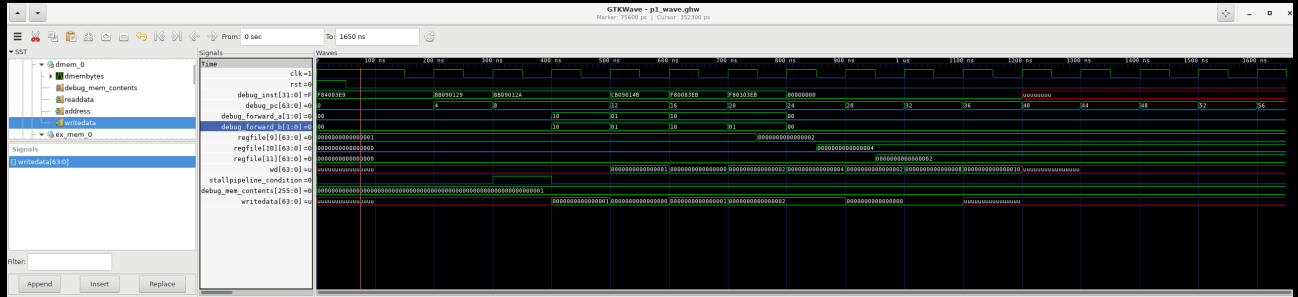


Figure 1: Test p1

A breakdown of the new instruction tests in **Figure 1**:

1.  $X9 = M[XZR, 0] = DMEM(0x00) = 0x0...01$ . Writeback can be seen during  $debug\_pc_{10}=12$ ,  $debug\_inst=0xCB09014B$ , where we see the signal  $wd_{10}=1$ . As  $X9$  was initialized to  $0x01$ , the same value, there is no change in  $regfile[9]$  in GTKWave.

2.  $X9 = X9 + X9 = (\text{Stalled})\ 0x01 + 0x01 = 0x02$ . A hazard is detected and a stall occurs during this instruction, at 300ns where `stallpipeline_condition` goes high as a result of X9 being read for this instruction, and X9 being the register destination in the last instruction. This allows for the ID stage to forward X9 from the EX stage of the last instruction.

Writeback of this instruction can be seen during `debug_pc10=20` and `debug_inst=0xF80103EB`, where now `regfile[9]=0x02`.

3.  $X10 = X9 + X9 = (\text{Forwarded})\ 0x02 + 0x02 = 0x04$ . Forwarding at the EX/MEM stage can be seen during `debug_pc10=12`, `debug_inst=0xCB09014B`, where both `debug_forward` signals are 0b01. Writeback occurs during `debug_pc10=24` when `regfile[10]=0x04`.
4.  $X11 = X10 - X9 = (\text{Forwarded})\ 0x04 - 0x02 = 0x02$ . Forwarding at the MEM/WB stage can be seen during `debug_pc10=16`, `debug_inst=0xF80083EB`, where both `debug_forward` signals are 0b10. Writeback occurs during `debug_pc10=28` when `regfile[11]=0x02`.
5. Store contents within X11 at DMEM[0x08]. This instruction forwards the contents of X11 from the previous instruction at the ID/EX stage, as seen at `debug_pc10=20`, `debug_inst=0xF80103EB`, where `debug_forward_a=0b10`. However, this instruction does not complete as a result of the following instruction below.
6. Store contents within X11 at DMEM[0x10]. As a result of this instruction following the previous one, a structural hazard within the CPU, as both instructions require the contents of X11.

In our current implementation, as we don't have forwarding hardware at the MEM/WB stage, we cannot resolve this hazard as the contents of X11 cannot be forwarded at MEM/WB. As a result our DMEM has uninitialized values as seen in DMEM's `write_data` at `debug_pc10=32` and nothing is written to DMEM. ( could be wrong ): )

7. 4 NOPs in a row. We can see that on the `debug_pc` continues to increment by 4, but nothing is affected indicating a proper NOP. Afterwards we can see `debug_inst` is undefined as there are no instructions left to execute in IMEM.

The waveform displays the following signals and their values over time:

- clk=0**: A regular square wave clock signal.
- debug\_pc[63:0]=5**: A constant signal with the value 5.
- debug\_inst[31:0]=0**: A constant signal with the value 0.
- ifid\_imemrdata.out[31:0]=0**: A constant signal with the value 0.
- regfile[9][63:0]=0**: A constant signal with the value 0.
- regfile[10][63:0]=0**: A constant signal with the value 0.
- regfile[11][63:0]=0**: A constant signal with the value 0.
- regfile[12][63:0]=0**: A constant signal with the value 0.
- regfile[19][63:0]=0**: A constant signal with the value 0.
- debug\_mem\_contents[255:0]=0**: A constant signal with the value 0.
- regfile[20][63:0]=0**: A constant signal with the value 0.
- regfile[21][63:0]=0**: A constant signal with the value 0.
- dmembytes[0][7:0]=01**: A signal that changes from 01 to 02 at approximately 400 ns and back to 01 at approximately 800 ns.
- dmembytes[16][7:0]=04**: A signal that changes from 04 to 08 at approximately 800 ns and back to 04 at approximately 1200 ns.
- dmembytes[18][7:0]=00**: A signal that changes from 00 to 0A at approximately 800 ns and back to 00 at approximately 1200 ns.
- dmembytes[31][7:0]=00**: A signal that changes from 00 to DE at approximately 1200 ns and back to 00 at approximately 1600 ns.
- dmembytes[2][7:0]=AD**: A signal that changes from AD to BE at approximately 1200 ns and back to AD at approximately 1600 ns.
- dmembytes[1][7:0]=BE**: A signal that changes from BE to EF at approximately 1200 ns and back to BE at approximately 1600 ns.
- dmembytes[0][7:0]=01**: A signal that changes from 01 to 02 at approximately 400 ns and back to 01 at approximately 800 ns.

- 4

