Report of Lab 2.2

Lalit Meena,2019CS50439

For this Lab 2 we are building microarchitecture for subset of arm instructions.

This stage involves putting together stage 1 modules, to form a simple processor which can execute the following subset of ARM instructions with limited variants/features. {add, sub, cmp, mov, ldr, str, beq, bne, b}

In previous modules ,made almost no changes in them but in ALU module changed opc type code as enumeration type instead of nibble.In stage-1,have made 4 module design files , 4 testbench file for each module and two helping files.

Here in stage 2,additionally made three module design file and 1 testbench file for processor.

Zip folder name – L2.2_2019CS50439

<u>Submission folder, **L2.1_2019CS50439** contains-</u>

1. **alu.vhd –** alu module design file
2. **data_mem.vhd -** data memory 64x32 module design file
3. **prog_mem.vhd –** program memory 64x32 module design file
4. **rfile.vhd -** design file for module register file 16x32
5. **others.vhd-**contains additional modules require for stage-2**(Instruction Decoder, Condition Checker, Program Counter)**
6. **flags.vhd-** design file for module flags
7. **Simple.vhd-** SIMPLE SINGLE CYCLE PROCESSOR

package.vhd -package to declare some types-words,bytes,etc

run.do- edaplayground file to help in syntheses of modules to give useful info about resources

<u>4 testbench files-</u>

testbench_alu.vhd – testbench file for alu module

testbench_dm.vhd – testbench file for data memory 64x32 module

testbench_pm.vhd — testbench file for program memory 64x32 module

testbench_rf.vhd - testbench file for module register file 16x32

testbench_simple.vhd - testbench file for module simple processor

Report of Lab 2.2-lab 2 stage 2 description and test cases(also screenshots)

More details are in next pages of each module-

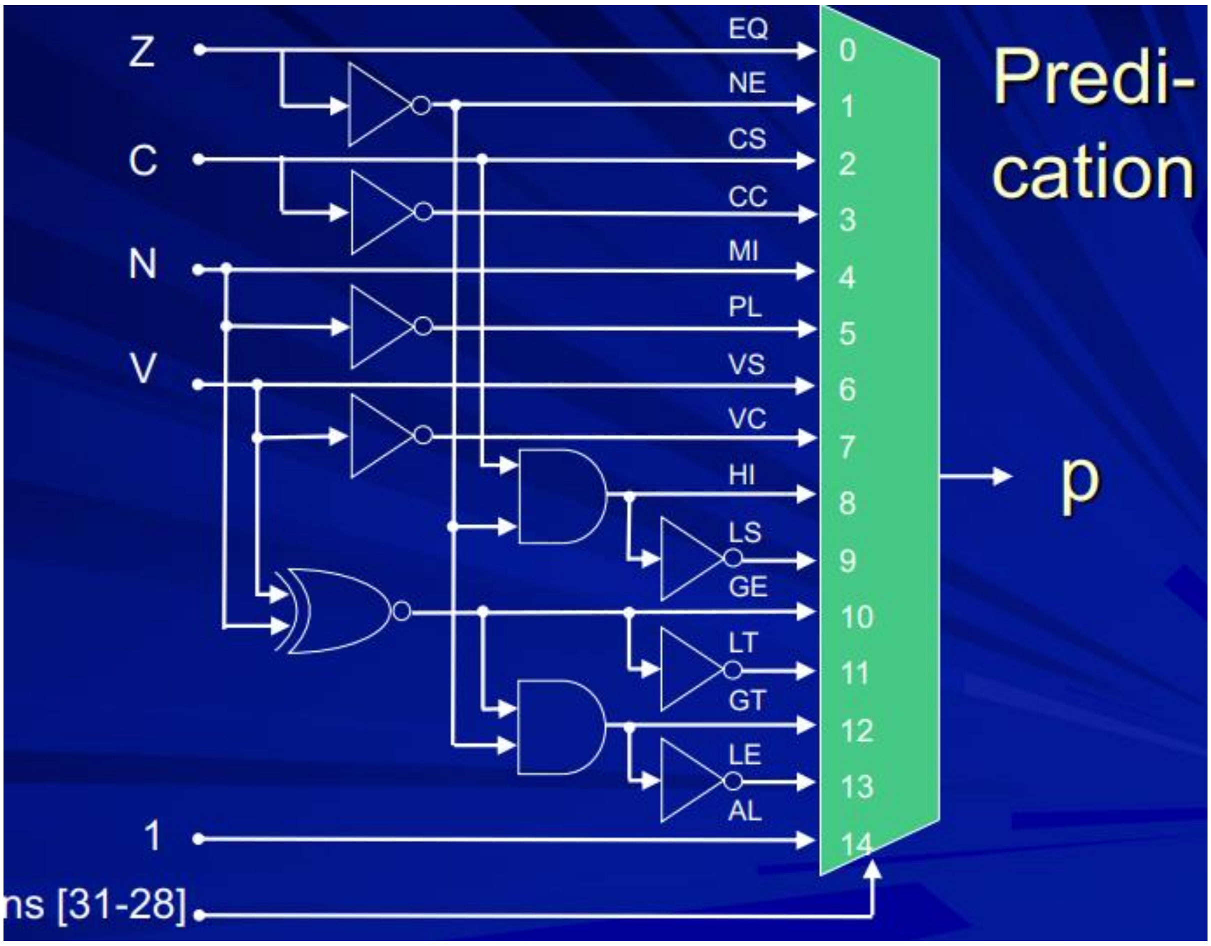
5. **others.vhd –** data memory 64x32 module design file

explanation-

This file contains contains additional modules require for stage-2(**Instruction Decoder, Condition Checker, Program Counter)**

Decoder-its like combinational control path which takes fetched instruction from PMem and generates control signals for main processor data path and other components

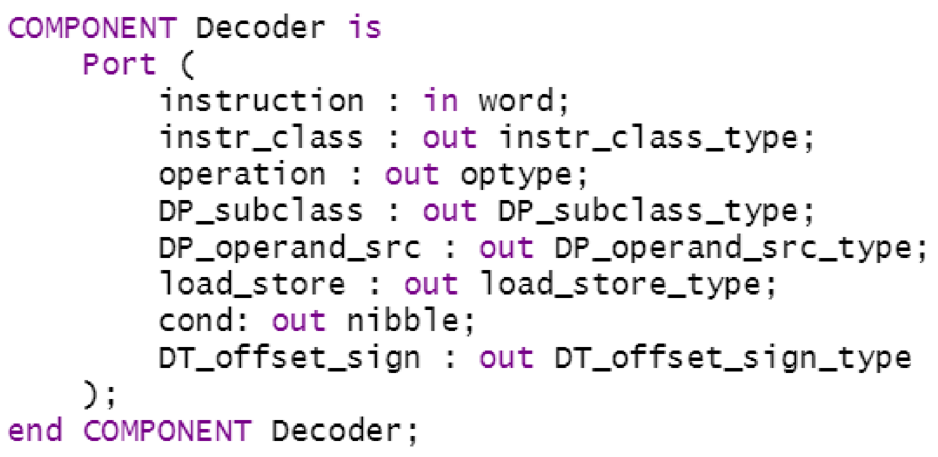
Condition Checker- This is a combinational circuit that looks at the Flags and condition field of the instruction to decide whether the specified condition is true or not-predicate result. Here now designed for EQ and NE conditions which are relevant for this stage, this is a simple circuit and works on below logic-



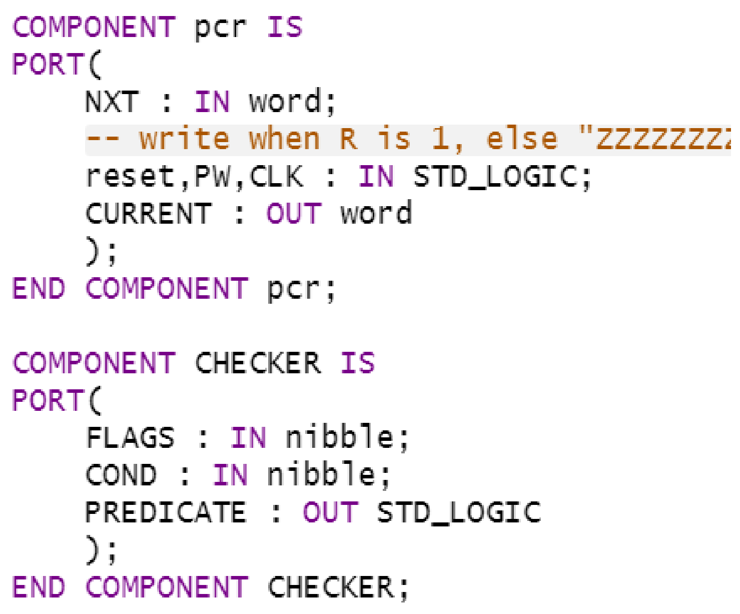
Program Counter-This normally updates the Program Counter on every clock by adding 4.if the current instruction is a branch instruction, and the specified condition (predicate result)is true, it adds appropriate offset to the Program Counter.

On reset it get set to zero.PC is the thing which drives our processor.

Components and ports info-



```
COMPONENT Decoder is
    Port (
        instruction : in word;
        instr_class : out instr_class_type;
        operation : out optype;
        DP_subclass : out DP_subclass_type;
        DP_operand_src : out DP_operand_src_type;
        load_store : out load_store_type;
        cond: out nibble;
        DT_offset_sign : out DT_offset_sign_type
    );
end COMPONENT Decoder;
```



```
COMPONENT pcr IS
PORT(
    NXT : IN word;
    -- write when R is 1, else "ZZZZZZZ
    reset,PW,CLK : IN STD_LOGIC;
    CURRENT : OUT word
    );
END COMPONENT pcr;

COMPONENT CHECKER IS
PORT(
    FLAGS : IN nibble;
    COND : IN nibble;
    PREDICATE : OUT STD_LOGIC
    );
END COMPONENT CHECKER;
```

6. flags.vhd –flags

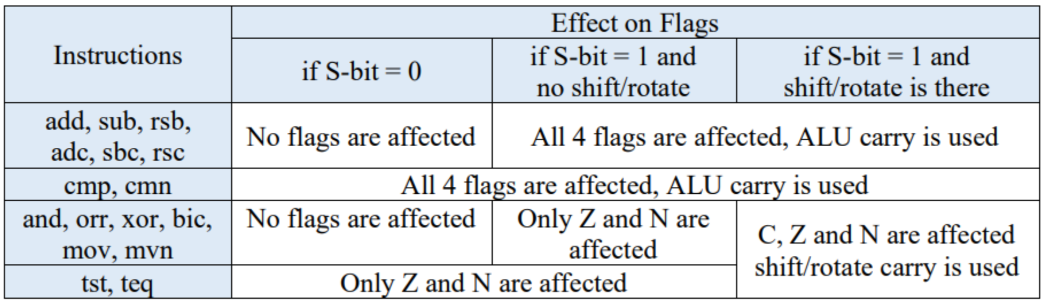
explanation-

It have an edge triggered 4-bit flags register (or 4 edge triggered flip flops) that will be set according to required conditions and clocked.

Flags-cvzn can be read unclocked.

Logic and implementation is same as given material uploaded on “Circuit for maintaining Flags”

How DP instructions affect Flags C, V, Z, N



Instructions	Effect on Flags		
	if S-bit = 0	if S-bit = 1 and no shift/rotate	if S-bit = 1 and shift/rotate is there
add, sub, rsb, adc, sbc, rsc	No flags are affected	All 4 flags are affected, ALU carry is used	
cmp, cmn	All 4 flags are affected, ALU carry is used		
and, orr, xor, bic, mov, mvn	No flags are affected	Only Z and N are affected	C, Z and N are affected shift/rotate carry is used
tst, teq	Only Z and N are affected		

entity and ports info-

```
-- entity
entity flag is
port(
    instr_class : in instr_class_type;
    opc:in optype;
    DP_subclass : IN DP_subclass_type;
    s,shiftr:in std_logic;
    c_alu,c_shiftr,CLK:in std_logic;
    a,b:in std_logic;--msb bits
    result: in word;
    cvzn:out nibble);
end entity;
```

7. **Simple.vhd-** SIMPLE SINGLE CYCLE PROCESSOR

Required Assumptions has been taken as stated and taught.

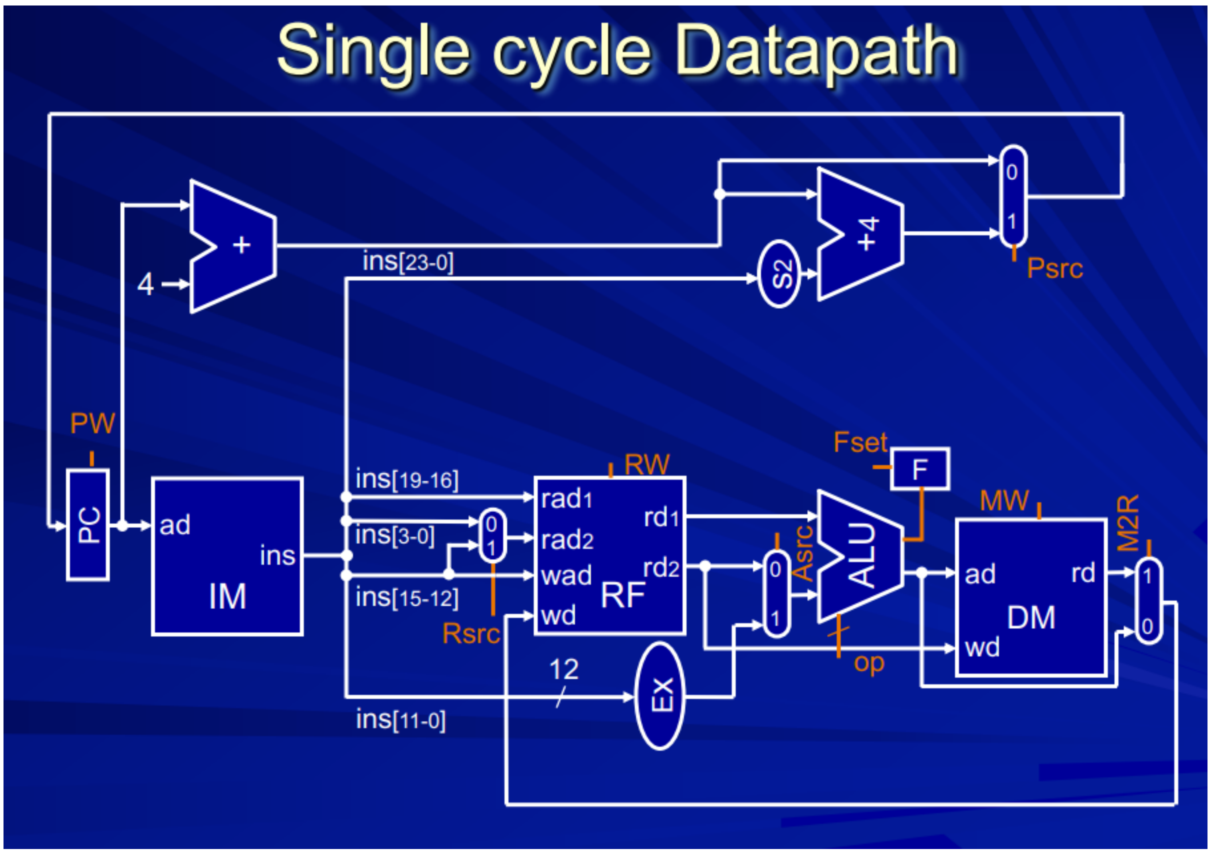
explanation-

It is main file which have data path for simple single cycle processor.With help of Decoder -control circuit we able to complete required processor. This also include all the "glue logic" which in this case includes some multiplexers and control logic.

Here all required and stated components are initiated,which are all the state elements (register file, data memory, flags, program counter and many more)

This file contains concurrent combinational statements like component instantiations,selecting required parameters,etc

Here evey cycle one fetched instruction by PC get executed.PC get increment conditionally by driven clock.



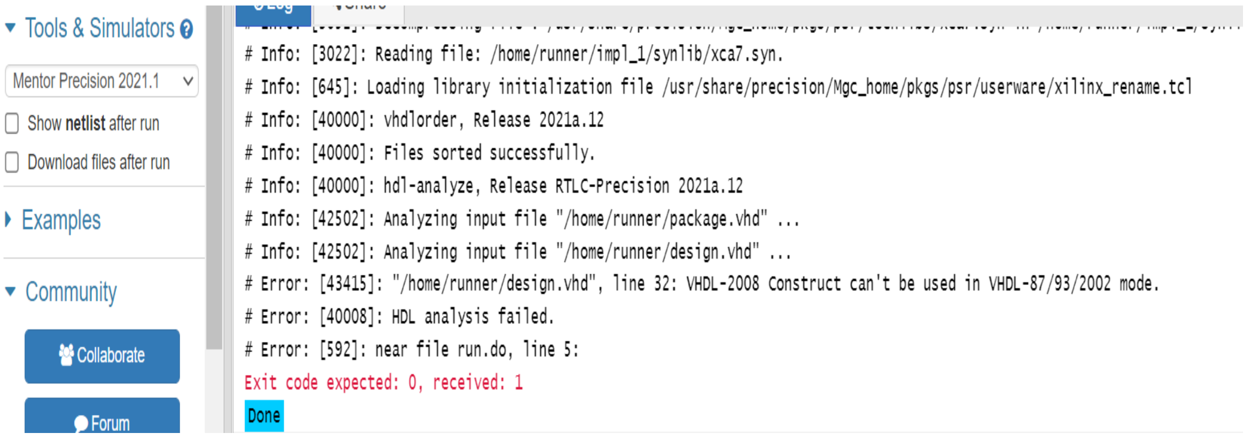
entity and ports info-

```
entity processor is
 port (
 clock, reset : in std_logic
 );
end processor;
```

Synthesis related

In Stage-2 ,used some new concurrent type of statements like when-else,with-select.This may be reason for encountering this message while synthesis for these using told run.do(Xilinx -family Artix-7 -part 7A100TCSG324) file for Mentor Precision 2021.

Also read on Piazza By sir that synthesis is not necessary.



```
# Info: [3022]: Reading file: /home/runner/impl_1/synlib/xca7.syn.
# Info: [645]: Loading library initialization file /usr/share/precision/Mgc_home/pkgs/psr/userware/xilinx_rename.tcl
# Info: [40000]: vhdlorder, Release 2021a.12
# Info: [40000]: Files sorted successfully.
# Info: [40000]: hdl-analyze, Release RTLC-Precision 2021a.12
# Info: [42502]: Analyzing input file "/home/runner/package.vhd" ...
# Info: [42502]: Analyzing input file "/home/runner/design.vhd" ...
# Error: [43415]: "/home/runner/design.vhd", line 32: VHDL-2008 Construct can't be used in VHDL-87/93/2002 mode.
# Error: [40008]: HDL analysis failed.
# Error: [592]: near file run.do, line 5:
Exit code expected: 0, received: 1
Done
```

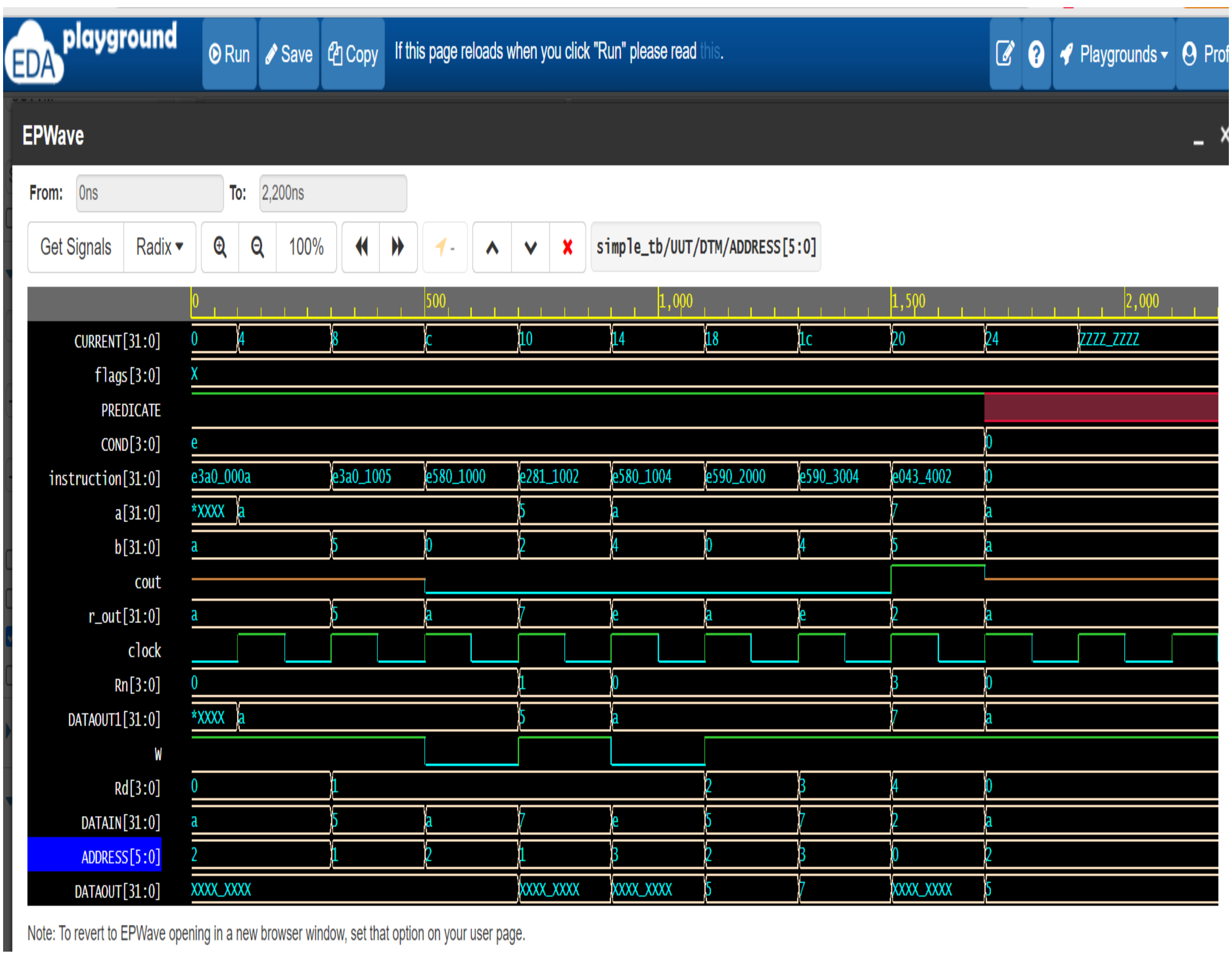
simulated with help of testbench_simple.vhd-

state change like flags and registers at end of program or step can be checked by signal like Rd,Rn,DATAIN,etc.

Example-1

```
signal mem_array : memory_type :=
                        (0 => X"E3A0000A",
                         1 => X"E3A01005",
                         2 => X"E5801000",
                         3 => X"E2811002",
                         4 => X"E5801004",
                         5 => X"E5902000",
                         6 => X"E5903004",
                         7 => X"E0434002",
                         others => X"00000000"
                        );
```

```
.text
mov r0, #10
mov r1, #5
str r1, [r0]
add r1, r1, #2
str r1, [r0, #4]
ldr r2, [r0]
ldr r3, [r0, #4]
sub r4, r3, r2
.end
```

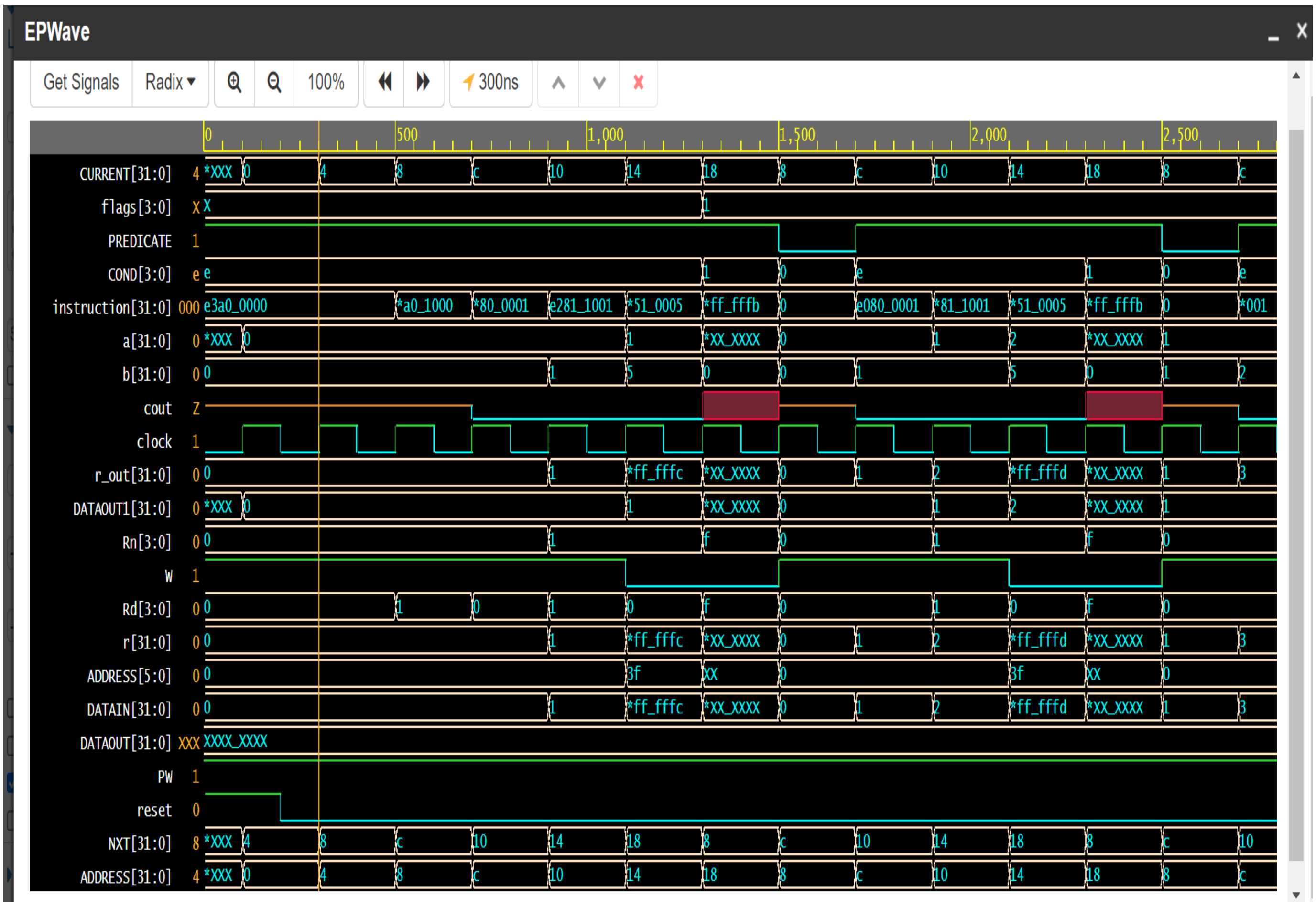


Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Example-2

```
signal mem_array : memory_type :=
                    (0 => X"E3A00000",
                     1 => X"E3A01000",
                     2 => X"E0800001",
                     3 => X"E2811001",
                     4 => X"E3510005",
                     5 => X"1AFFFFFB",
                     others => X"00000000"
                    );
```

```
        .text
        mov r0, #0
        mov r1, #0
Loop:   add r0, r0, r1
        add r1, r1, #1
        cmp r1, #5
        bne Loop
        .end
```

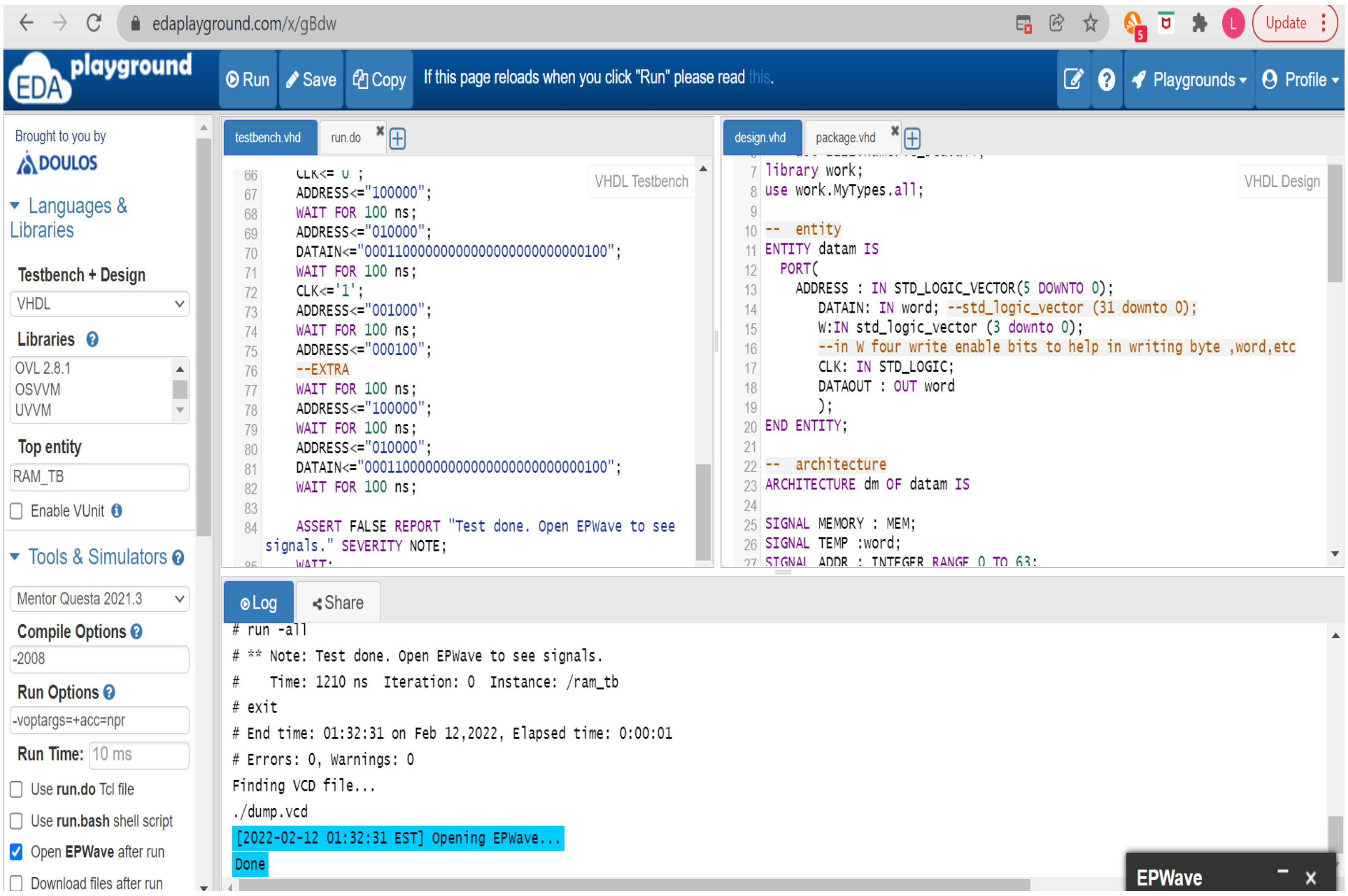


For each module,work and analysis I have put files in edaplayground like below-

Used for simulation- Mentor Questa 2021.3

Used for synthesis- Mentor Precision 2021.1

simulation settings For eg data mem-



Languages & Libraries

Testbench + Design: VHDL

Libraries: OVL 2.8.1, OSVVM, UVVM

Top entity: RAM_TB

☐ Enable VUnit

Tools & Simulators: Mentor Questa 2021.3

Compile Options: -2008

Run Options: -voptargs=+acc=npr

Run Time: 10 ms

☐ Use run.do Tcl file

☐ Use run.bash shell script

☑ Open EPWave after run

☐ Download files after run

```
    CLK<='0';
    ADDRESS<="100000";
    WAIT FOR 100 ns;
    ADDRESS<="010000";
    DATAIN<="00011000000000000000000000000100";
    WAIT FOR 100 ns;
    CLK<='1';
    ADDRESS<="001000";
    WAIT FOR 100 ns;
    ADDRESS<="000100";
    --EXTRA
    WAIT FOR 100 ns;
    ADDRESS<="100000";
    WAIT FOR 100 ns;
    ADDRESS<="010000";
    DATAIN<="00011000000000000000000000000100";
    WAIT FOR 100 ns;

    ASSERT FALSE REPORT "Test done. Open EPWave to see
signals." SEVERITY NOTE;
```

```
library work;
use work.MyTypes.all;

--  entity
ENTITY datam IS
  PORT(
    ADDRESS : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
       DATAIN: IN word; --std_logic_vector (31 downto 0);
       W:IN std_logic_vector (3 downto 0);
       --in W four write enable bits to help in writing byte ,word,etc
       CLK: IN STD_LOGIC;
       DATAOUT : OUT word
       );
END ENTITY;

--  architecture
ARCHITECTURE dm OF datam IS

SIGNAL MEMORY : MEM;
SIGNAL TEMP :word;
SIGNAL ADDR : INTEGER RANGE 0 TO 63;
```

```
# run -all
# ** Note: Test done. Open EPWave to see signals.
#    Time: 1210 ns  Iteration: 0  Instance: /ram_tb
# exit
# End time: 01:32:31 on Feb 12,2022, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
Finding VCD file...
./dump.vcd
[2022-02-12 01:32:31 EST] Opening EPWave...
Done
```

For more testcases and live running ,edaplayground can be played in demo for this stage and public link of it can be shared.