Politecnico di Torino

Microelectronic Systems

DLX Microprocessor: Design & Development
Final Project Report

Master degree in Computer Engineering

Referents: Prof. Mariagrazia Graziano, Giovanna Turvani

Authors: group 50

Angione Francesco
s262620

GitHub Repository

September 14, 2020

Contents

CHAPTER 1

Introduction

The DeLuXe (aka DLX) is a RISC processor architecture designed by John L. Hennessy and David A. Patterson. It is a modernized and simplified 32-bit load/store big endian architecture of the MIPS CPU, primarly intended for teaching purposes.

In the following pages the ASIC design flow has been applied starting from the basic requirements of DLX, it has been developed accordingly with basic features, i.e. the pipeline and the basic instruction set. As next step the integer multiplication has been added to the basic instruction set. For doing that, the Booth's multiplier studied during the course and developed during the laboratories has been pipelined in order to obtain a pipelined multiplier on eight clock cycles.
An important aspect of Digal Desing, in the development step, is to verify the functional behaviour of the components. In order to do that, testbenches in System Verilog have been used (resorting to a mixed language simulation) in order to maximixe the efficiency and speeding up the development time, it has also allowed to removed bugs during the early functional testing of the stages composing the pipeline by the mean of defining temporal properties on signals. Moreover, at the end, another testbench using the Universal Verification Methodology has been developed, which is a de facto standard in the Functional Verification Testing in the industry[1].

As last step of the design flow, the design is synthesized with different approaches in order to move the CPU in different design point of the space (area and clock frequency). After that, a subset of the synthesized designs is physically designed. Starting with the routing of the power supply system and ending with the placed and routed of the actual transistors.

CHAPTER 2

Functional Verification Testing

In Digital Design, functional verification testing is the task of verifying that the logic design is conformed to its specifications, it answers to the question *Does it do what it is intended do?*. The answer to this question is a complex task and takes the majority of time in integrated circuit design.

Thanks to the Moore's law [4] the complexity of integrated circuit has grow exponentially, putting more and more peripheral and modules in the same die's area it has leaded to the SoC area. This has lead to very complex circuits and it comes naturally that the old test benches architectures are no more suitable for checking the intended behaviour.

The task of Functional verification testing can be done through different methods:

- Logic simulation, it simulates the logic before it is built.
- Simulation acceleration, itapplies special purpose hardware to the logic simulation problem.
- Emulation, it builds a version of system using programmable logic.
- Formal verification, it attempts to prove mathematically that certain requirements are met.
- Intelligent verification, it uses automation to change testbenches at RTL level.

From now on, the logic simulation based functional verification will be used. In this approach the stimulus are provided by a testbench which emulates a meaningfull scenario in order to check that for a certain input, the related output is generated as the specifications indicates.
A test bench can be divided in sevela componets, each one of them with a specific purpose:

- the generator, it generates stimulis according to specifications and the expected environment. Stimulus are manually generated. On the other hand, a modern solution may be to create random stimuli that are statistically driven in order to verify part of the unit under test.
- the driver, it translates translate the stimuli produced by the generator into the actual inputs for the design under verification.
- the monitor, it stores the state and the outputs of the design into a database.
- the checker, it validates wrt to the specifications the stored values of the monitor.
- the arbitration manager, it manages all the above components together.

2.1 Testbenches with System Verilog

The usage of System Verilog as preferred language for testbenches has become increased more and more in the last years. One of the motivation is its C-like features, and easyness of usage. Moreover, it offers a lot of randomization of values features, the OOP paradigm and the possibility of defining interfaces (which are also synthesizable construct).
It has been developed as language that encapsulates the HDL and the HDL. Therefore, the usage of System Verilog as language for verification testbenches has allowed to use a different architecture for functionally testing the DLX, the UVM approach.
Another important aspect is the possibility of using assertions (SVA), they allow to specify the expected behavior of a design with temporal property on signals speeding up the developing process and the checking of correct behaviour(see Appendix E for properties definition related to the DLX top level entity). The usage of the SVAs has allowed the speed-up, in this particular case, of pipeline property for each subunit of the DLX. This has been done in order to reduce the overhead of checking and fixing problems when integration comes.

Following the previous mentioned ideas, the memories and their interfaces have been implemented in System Verilog (while the DLX has been described in VHDL). At the end, the testbench has been writtein in System Verilog with the following architecture:

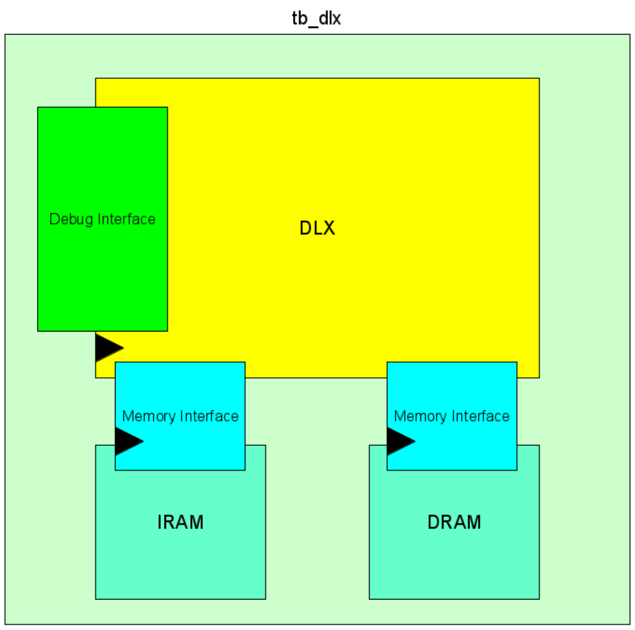


Figure 2.1: Testbench architecture

In Figure 2.1 are missing the clock interconnections and the reset just for a matter of readability. In this case the:

- DLX, it is the microprocessor under test.
- IRAM: it is the instruction ram which loads the program (compiled offline) for the unit under test.
- DRAM: it is the data ram which loads at the begginning a file with random values and at every write operation it refresh the content of another file.

- Memory interface: it is the interface connecting the memories with the microprocessor. It has been defined as a single interface for both the memories, the only difference is in the modport configuration, where it can be distringuished between a read-only memory and a read-write memory (see Appendix B).
- Debug interface: it is an interface that will be removed during the synthesis, its only purpose is to made visible the control signals of the CU in order to check if the instruction-dependent properties are covered (see Appendix E).

2.1.1 Functional Coverage

Functional coverage is a coverage metric defined to asses that the design has been adequately exercised[2]. Observation points can be inserted at every granularity and in different poitns, external to the design (on the interfaces), inside the design or both of them.
As it can be seen in Figure 2.2 the observation point has been added on the interfaces and they observe the read and write operation on memory as the memory address range(see Appendix B).

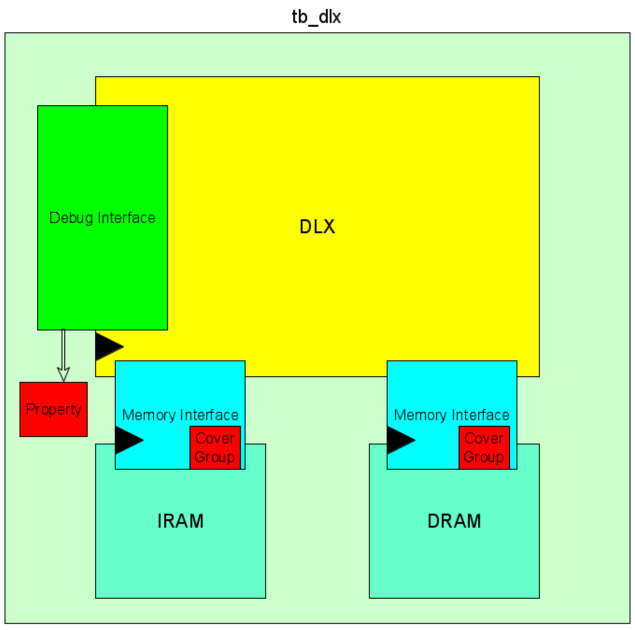


Figure 2.2: Testbench architecture with coverpoint(in red)

Instead of using covergroup, another approach is to reuse the property used for SVA. This approach has allowed to reuse the property defined for checking the correct behaviour of the control unit, moving them outside the DLX and checking if those property are covered by the Debug interface signals (which are the control signals from the control unit to the datapath). It is important to understand that the cover of those properties is program-dependet since they check the behaviour for each tipe of instruction (r-type,i-type and j-type). Therefore, a test program able to test the whole ISA has been developed.

It is worth to mention that memories are never fully covered since their address range is very large and they are exercised with test programs that use only a narrow subset of the memory range.

2.2 Universal Verification Methodology

The Universal Verification Methodology (UVM) is a standardized methodology for verifying integrated circuit designs[3]. UVM is derived mainly from the OVM (Open Verification Methodology). The UVM class library brings much automation to the SystemVerilog language such as sequences and data automation features (packing, copy, compare), and unlike the previous methodologies it is developed independently by the simulator vendors.
It exploits the OOP paradigm, therefore for each one of the entities that can be seen in Figure 2.3, they represent a class with a specific function.

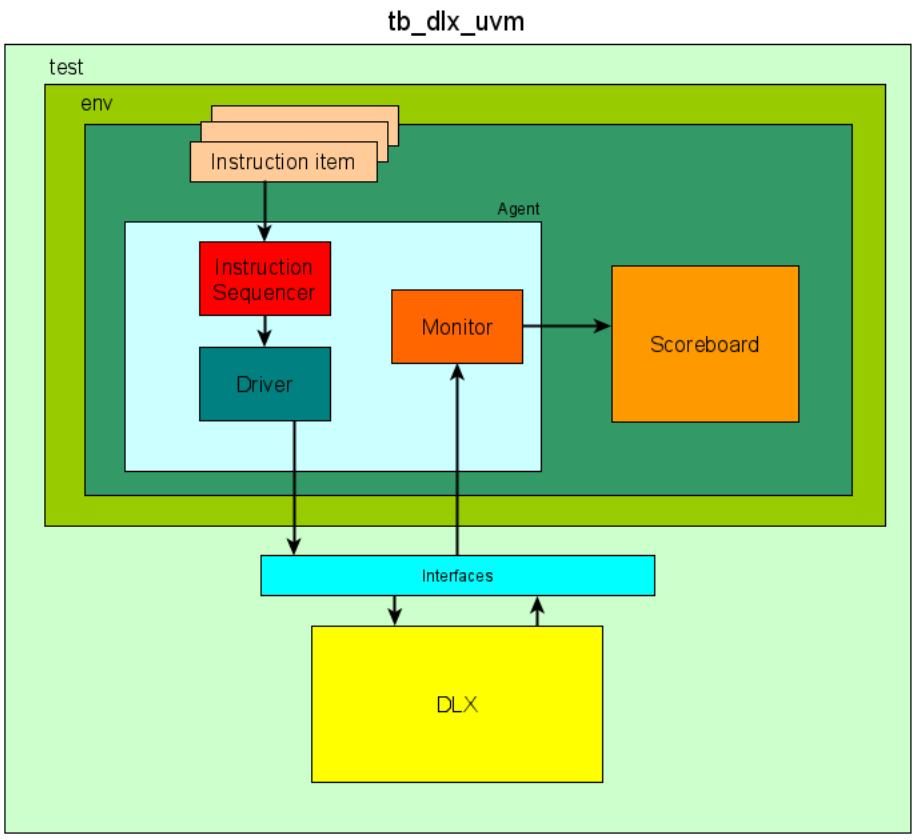


Figure 2.3: UVM testbench architecture

Each class in Figure 2.3 inherits the functions and tasks of the relative UVM related class. For this specific case, the UVM classes function are (see Appendix **??**):

- Instruction item, it is the basic instruction for the microprocessor plus some additional function, such as its conversion to string, the retrieve of the only opcode and/or the opcode ALU function. It is important to mention that it includes the random variables for the opcode, opcode alu function, r_d, rs_1, rs_2 , immediate and jump address(which are randomized when calling the relative function in instruction sequence) plus the constraint on the register, such as the one that the r_0 cannot be a destination register. Moreover, depening on the current opcode, it composes the instruction accordingly, i.e. the jump address is not needed when composing a add instruction and viceversa, even if all the variables are randomized everytime.

- Instruction sequence, it creates a given number of random instruction item.

- Instruction sequencer, it only gives the instruction item from the instruction sequence to the driver one by one.
- Driver, it retrieves the next instrction item from the sequencer and gives it to the DUT . It basically behaves as the IRAM of Figure 2.1.
- Monitor, this class is only in charge of sampling the Debug signals from the DUV and its current instruction.
- Agent, since it is active it is in charge of creating the sequencer,driver and monitor classes and connects the driver port to the export port of the sequencer.
- Scoreboard, it stores for each executed instruction if it passes the test or not and how many time it has been executed. It also checks if the correct signals(sampled from the Debug interface) have been asserted for the specific instruction. For example, *add* r_1,r_3,r_4, it will access to both the port of register file and then it will compute the addition between the accessed values and as last step it will write back the value in the register file. The scoreboard stores the signals from when the instruction is given to the microprocessor up to the 5-th clock cycle and then it compares the signals with an internal signature.
- Environment, it instantias the agent and the scoreboard and connects the analysis port of the scoreboard to the output port of the agent.
- Test, it is only in charge of instantiate the environment class, it also applies a preliminary reset to all systems and then it creates the sequence of random instructions.
- tb_dlx_uvm, it is the top level module which is in charge of instantiating the DUV, the interfaces. It internconnects the interfaces with DUV and starts the test (see Appendix F).

CHAPTER 3

A Top down view of the architecture

Every Digital System is always divided in two big blocks, as it can be also observed in Figure 3.1:

- Control Unit, it is the brain of the microprossesor and it is in charge of handling the syncrhonization between stages asserting the proper signals.
- Datapath, it is the actual brawn of the microprossesor. It is composed by 5 functional units (meaning that it is a 5 stage pipelined microprocessor) that perform data processing operations on data.

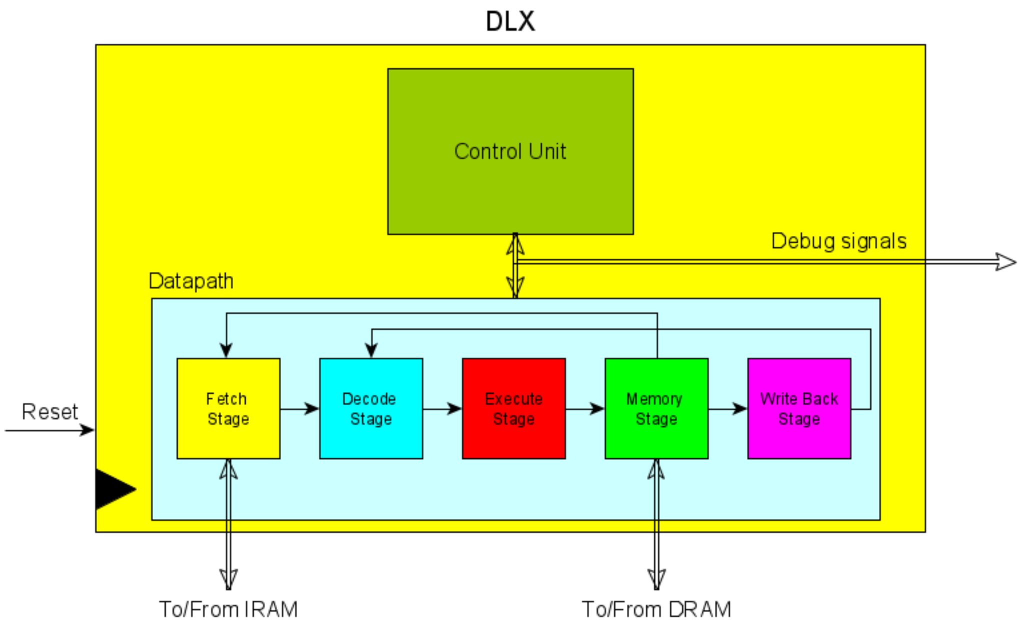


Figure 3.1: DLX top level entity

Notice that the clock and reset signal are routed to every units (interconnections missing in Figure 3.1 for increasing readability). Moreover, the debug signals are present only for simulation purposes, they are removed during the synthesis by means of synthesis pragma.

3.1 Control Unit

The brain of the microprosessor is a simple control unit based on two states, fetch and decode. However, the fetch state is executed only at the reset, leading to have a nop operation in the pipeline operation. Meanwhile, when the microprocessor is fully operational, it is always in the decode stage.

The control unit is based on the hardwired approach, meaning that for each one of the instruction there is a predefined signature for signals to be asserted (except for particular case, such as the sub instruction, where the carry in must be set to 1 since the adder is shared among the addition and subtraction). All the predefined signals for given instruction are activated only one. Therefore, for correctly synchronizing the pipeline they have to be properly delayed by mean of registers.
The control unit is also in charge of selecting the proper operation for the ALU. During the execution of the integer multiplication, it stalls the pipeline for the whole duration of the operation in order to avoid hazards or it may restore the pipeline behaviour in the casa that one of the multiplication operands is zero and/or it is bigger than $2^{16} - 1$.

3.2 Datapath

The datapath of the DLX is composed by 5 stages, as in Figure 3.2:

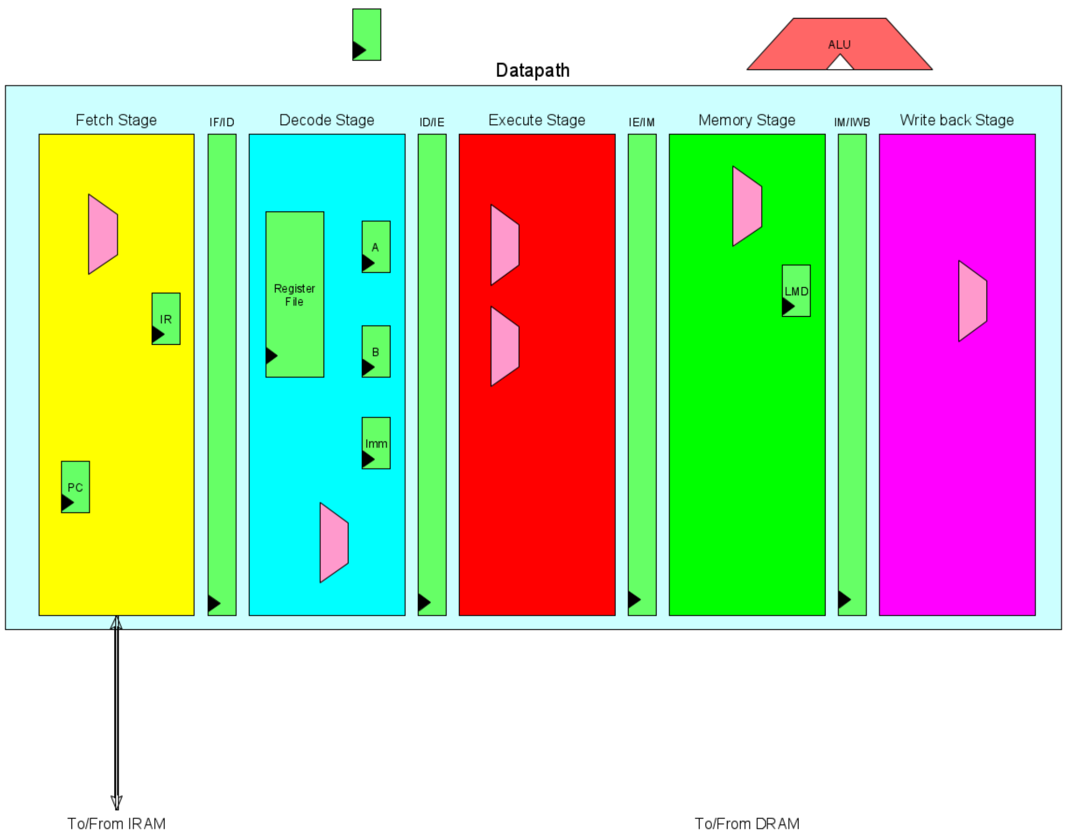


Figure 3.2: Datapath [1]

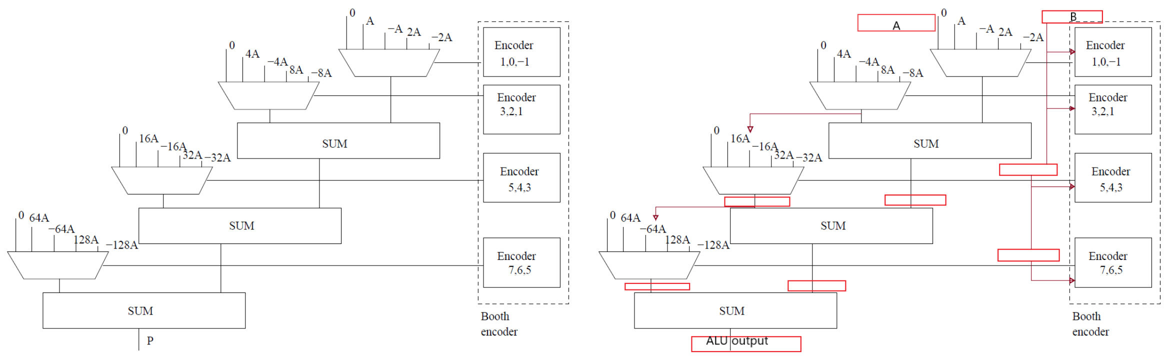
[1]Control Unit signals are missing for increasing readability

- Fetch Stage: it uses as address for instruction memory the value of the PC, while the data(instruction) coming from the memory are saved into the IR. It also computes the new value of the PC, a plus 4 during normal operation, and a loopback of the PC value in the case the pipeline is in stall.
- Decode Stage: it decode the instruction and depending on the instruction type, it selects the correct values for accessing to the register file and save them in the A and B registers or it extends the value of the immediate field (from 16 bit to 32 bit in case of an immediate instruction or from 26 to 32 bit in case of jump instruction).
- Execute Stage: The ALU operates on its input, depending on the current operation, and in this stage the condition for taking the branch is eventually evaluated. It is based of an enhanced version of the ALU developed during the laboratories, in which has been added the missing operations of comparison. In addition, the current adder has also been developed during the lab and it is based on the Pentium 4 adder which is shared among the addition and subtraction exploiting the properties of two's complement binary representation.
- Memory Stage: it is in charge of accessing the memory if needed or load the data from memory in LMD register. It also decides the value of the PC in the case a branch is taken.
- Write Back Stage: it writes back into the register file either data from ALU or data memory.

Moreover, the green registers between each stages (i.e. IF/ID, ID/IE, IE/IM and IM/IWB) are registers used for synchronization purposes of values, for example the register index for writing back into the register file need to be delayed by 3 clock cycle (it is needed in the write back stage). Nevertheless the logical view of Figure 3.2, they are actually implemented in the right stage (if not shown otherwise, such as for the A and B registers).

3.2.1 ALU Multiplier

Another important aspect of the ALU is the integeer multiplier.
It is based on the Booth's multiplier developed during the lab as in Figure 3.3a.



(a) Booth's multiplier (b) Pipelined Booth's multiplier[1]

Figure 3.3: ALU Multiplier

As it can be seen from Figure 3.3b[1], the previously developed unit has been enhanced pipelining it in order to have a 8 stage (6 inside the multiplier and 2 outside the unit, the A and B register and

[1] Registers are in red
[1] The Figure represents a 8-bit multiplier. However, the approach is the same, only the internal stages of the multiplier have been pipelined.

the ALU output register) multiplier, reducing the critical path and increasing the possible achieavable perfomances. It is worth to mention that since the unit is pipeline it can potentially execute in parallel up to 8 multiplication, one ofter the other without any data hazards and with a proper control unit to handle the specifici situation.
Moreover, the compiler script has been also modified for adding the integer multiplication and the assigned opcode can be seen in Appendix I.

CHAPTER 4

Synthesis

The synthesis of the design has been achieved by a script (Appendix L) which contains also the proposed synthesis algorithms. In addition, the script is also in charge of adjusting the environment variables and folders needed for the synthesis and later for the physical design script.

The synthesis has been done through an inductive approach. As first step, a simple design without any constraints has been synthesized and evaluated. For moving inside the design space, the next step has been using as constrained on the clock different percentage values of the non-constrained synthesized design clock. Different percentage has been used, from 1% up to 20% (increase of clock frequency wrt to the non-constrained synthesized clock). Moreover, in this case as synthesis strategy *compile_ultra* has been used for pushing more effort in general optimizaitons. The estimation of the area as much as possible to a real microprossessor has been achieved by the usage of Scan Flip Flops instead of the normal Flip Flops (avialable in the used library).
As next degree of freedom, in the design space, all the previous designs with different clock constraints have been synthesized putting a minimum area constraint.

4.1 Results

The results in terms of area, latency and area are collected and presented as graphs.
As first design space, the latency-area graph can be seen in Figure 4.1: In this design space, the best

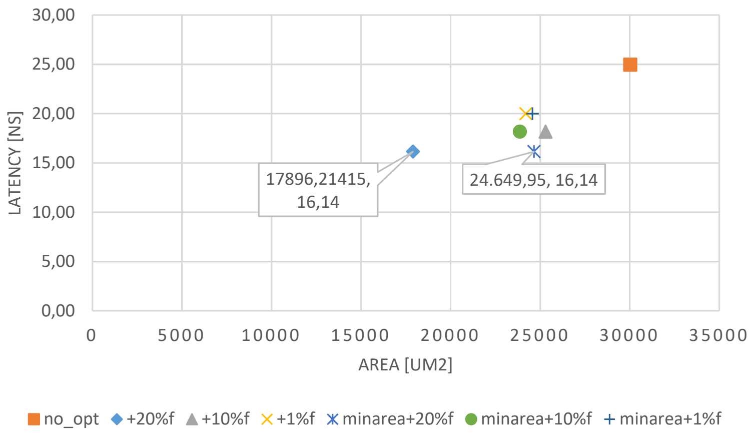


Figure 4.1: Design space: Area [1] vs Latency

design according to latency and area is the one synthesized with a clock frequency greater than the 20% of the non constrained design frequency and without no constraints on the area.

Moreover, a further exention of the previous design space may be the power consumption (where also constrains can be added). In Figure 4.2 it is presented the power consumption of the constraints on area and latency.

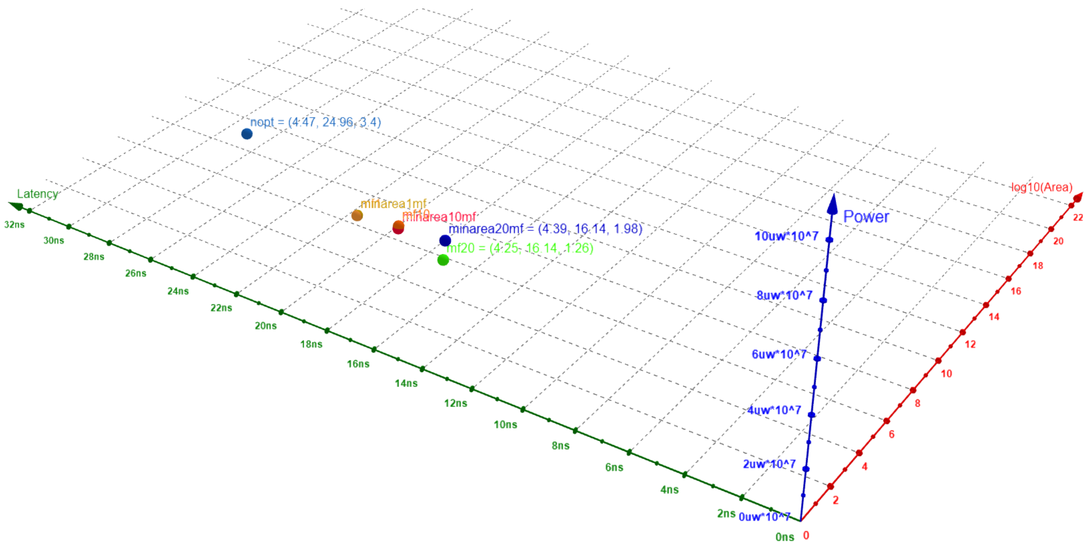


Figure 4.2: Design space: Area vs Latency vs Power

An as best design, it is still the one synthesized with 20% more of frequency. This result is probably due to the synthesis strategies and choices which lead to the best results of this design in terms of power,area and latency with the only constraint on the latency.

In the following table the results from synthesis reports are summarazied:

Design	Area [µm^2]	Latency [ns]	Power [µW]
No optimization	30034,85832	24,96	3,40E+07
+ 20% frequency	17896,21415	16,14	1,26E+07
+ 10% frequency	25306,70806	18,15	1,87E+07
+ 1% frequency	24199,08403	19,97	1,69E+07
+ 20% frequency and minarea	24649,95403	16,14	1,98E+07
+ 10% frequency and minarea	23870,84002	18,15	1,78E+07
+ 1% frequency and minarea	24569,62203	19,97	1,69E+07

Table 4.1: Area-Latency-Power points in Design space

[1]Cell area

CHAPTER 5

Physical Design

The physical design of the unit has been achieved by the usage of a script (Appendix M). As for the synthesis, this script is in charge of preparing the environment and start the real script for the physical design.
This step is very well known as a computing intensive step, even more then the synthesis since the granularity of details is bigger than the one in the synthesis. Therefore, for boosting-up the performance the script automatically sets the usage of six threads instead of only one.

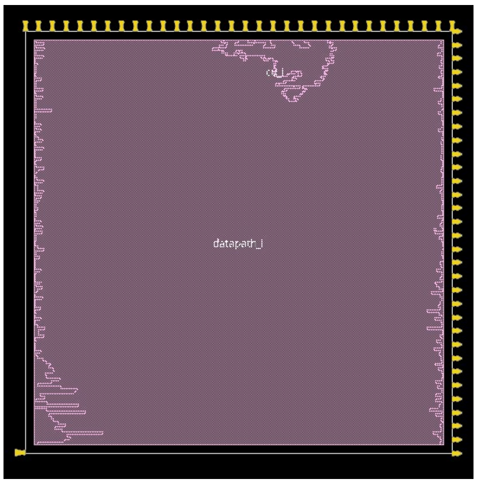
Nevertheless the variety in the design space, only a subset of them has been choosen for being placed on a die. In particular the unconstrained desing, the minimum area and a 10% less of the clock frequency and only the 10% less of the clock frequency designs. It is worth to mention that the physical design does not use the RTL description but it uses the gate-level netlist of the RTL, produced by the synthesis.

5.1 Results

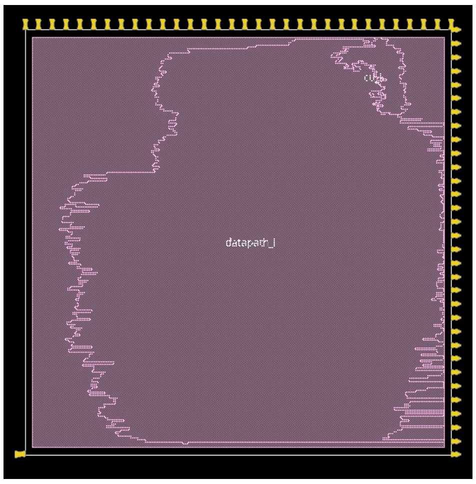
In the following pages, results are presented as images of the design on the same die.
As first result of physical design, it is worth to present the ameba view. It distinghuis between the control unit and the datapath of the DLX. As it can be seen from Figure 5.1a to Figure 5.1c, the area occupied by the control unit is always the same.

Moreover, from Figure 5.1 and Figure 5.2 it may seem that the pins are overlapping. However, this is not the case, it is only a matter of image resolution. Specifically, the pins on the top and right side (respectively for the IRAM and DRAM) are overlapped but on different level of the die. On the other hand, all the control signals from/to memories, clock and reset signals are on the bottom of the left corner.

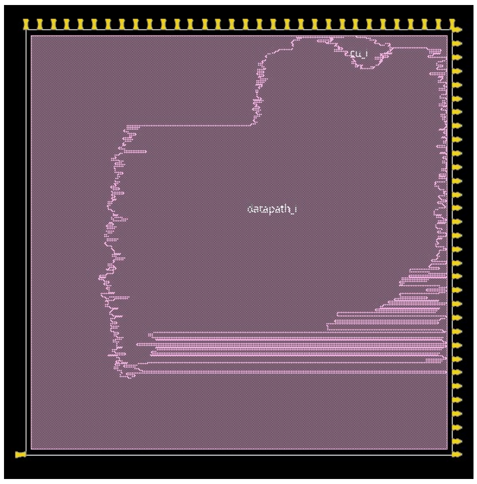
An important aspect is how the datapath area is changing, having in mind the boundaries of datapath from Figures 5.1 and looking at Figures 5.2. From A to C there is a consistent area reduction, the area is reducing by a factor of almost 2. Comparing the Figure 5.2a with the Figure 5.2b, even if there is a slight increase in the clock frequency, the synthesis strategies are different (it goes from a naive synthesis to a synthesis in which optimization efforts are done in order to reduce design metrics). The only difference with Figure 5.2c is the constraint on finding the design with the minimun area.



(a) Unconstrained design

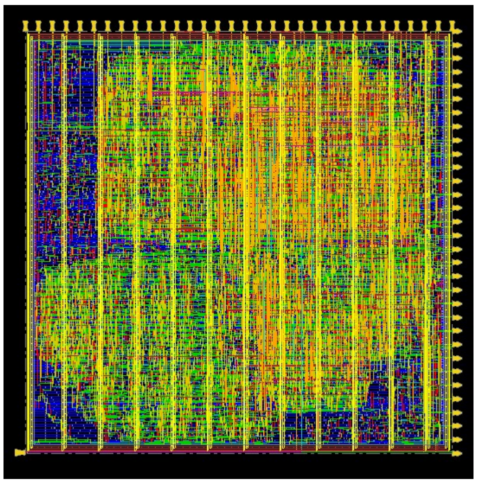


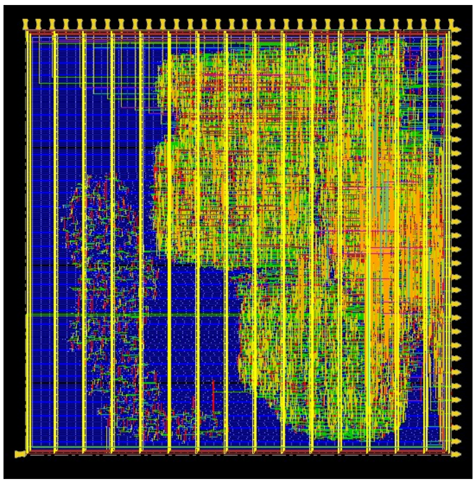
(b) 10% more on clock frequency



(c) 1% more on clock frequency and minimum area

Figure 5.1: Ameba view


(a) Unconstrained design


(b) 10% more on clock frequency

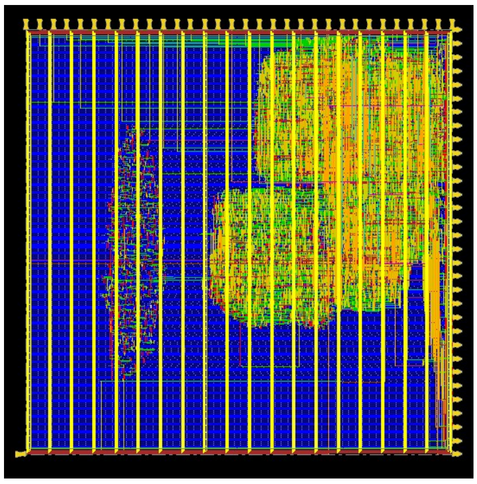

(c) 1% more on clock frequency and minu-
mum area

Figure 5.2: Placement view

Bibliography

[1] *1800.2-2017 - IEEE Standard for Universal Verification Methodology Language Reference Manual.* `https://standards.ieee.org/standard/1800_2-2017.html`.

[2] Ashok B. Mehta. "SystemVerilog Assertions and Functional Coverage. Guide to Language, Methodology and Applications". In: *Springer* (29 June,2017).

[3] Ashok B. Mehta. "Universal Verification Methodology (UVM) 1.2 User's Guide". In: *Accelera* (October 8,2015).

[4] Gordon E. Moore. "Cramming more componentsonto integrated circuits". In: *Electronics, Volume 38, Number 8* (April 19, 1965).

APPENDIX A

DLX testbench in System Verilog

```
//
//                                          \\\ Sapere Aude ///
//
//
   --------------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   --------------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : tb_dlx.sv
// Create : 2020-07-21 19:00:18
// Revise : 2020-08-21 20:02:24
// Editor : sublime text3, tab size (4)
// Description:  top level test bench for DLX fuctional verification
   testing
//                       Simple architecture :
//                                                         - Driver :
   iram
//                                                         - Monitor :
   dram and debug signals
//
   --------------------------------------------------------------------------------


`timescale  1ns/1ps
`include "./003-global_defs.svh"
`include "./004-implemented_instructions.svh"
`include "./memories/005-memory_interfaces.svh"
```

```
program automatic test_dlx_top(input logic clk, output logic rst,
   input cu_state_t  curr_state,
                                                  input logic [
                                                     ‘IRAM_WORD_SIZE
                                                     -1:0]
                                                     current_instruction
                                                     );

// it may be encapsulated into a class which extends the covergroup
   class and implements manually all the methods ( new, sample, start
    and stop etc. )
       covergroup instruction_cover with function sample(
          instructions_opcode opcode,   instructions_regtype_opcode
          func) ;
               option.name          = "instruction_cover";
           option.comment       = "cover␣group␣for␣all␣the␣implemented
              ␣instruction␣in␣004-implemented_instructions.svh";
           option.per_instance = 1;
           option.goal         = 100;
           option.weight       = 90;

       instru_opcode: coverpoint opcode{
               bins regtype= {i_regtype};
               bins immtype[] = {i_addi,i_andi,i_lw,i_nop,i_sgei,
                  i_slei,i_slli,i_snei,i_srli,i_subi,i_sw,i_ori,
                  i_xori};
               bins jump[]={i_j, i_jal};
               bins branch[]={i_beqz,i_benz}; // bnez beqz
       }

       instru_opcode_func: coverpoint func{
               bins alu_add =         {i_add};
               bins alu_and =      {i_and};
               bins alu_or =     {i_or};
               bins alu_sll =   {i_sll};
               bins alu_srl =    {i_srl};
               bins alu_sub =   {i_sub};
               bins alu_xor =  {i_xor};
               bins alu_sne =  {i_sne};
               bins alu_sle =         {i_sle};
               bins alu_sge =         {i_sge};
               bins alu_mul =         {i_mul};

       }


       endgroup : instruction_cover
```

```
        default clocking test_clk_prog @( posedge clk);
        endclocking

        instructions_opcode current_opcode;
        instructions_regtype_opcode current_opcode_func;


                // type_t'(x) cast x to type_t
assign  current_opcode=instructions_opcode'(current_instruction[
   `IRAM_WORD_SIZE-1:`IRAM_WORD_SIZE-`OP_CODE_SIZE]);
assign  current_opcode_func= (instructions_regtype_opcode'(
   current_instruction[`OP_CODE_SIZE-1:0]));


sequence end_seq;
        @(test_clk_prog)
                ((current_instruction==='Z || current_instruction==='x
                    ||current_instruction===i_j)[*12]);
endsequence;


        task automatic print_current_instruction (instructions_opcode
           opcode, instructions_regtype_opcode func);
                begin
                        if (opcode===i_regtype ) begin
                                $display("current␣instrution␣opcode␣%h
                                   ␣-->␣%s",current_instruction,
                                   enum_wrap_instruction#(
                                   instructions_regtype_opcode)::name
                                   (func));
                        end else begin
                                $display("current␣instrution␣%h␣-->␣%s
                                   ",current_instruction,
                                   enum_wrap_instruction#(
                                   instructions_opcode)::name(opcode)
                                   );
                        end
                end
        endtask : print_current_instruction


        initial begin
                instruction_cover cg_instruction = new(); //
                   instantiate the covegroupd
                // Set the database name sets the filename of the
                   coverage database into which coverage
                //information is saved at the end of a simulation run.
        $set_coverage_db_name("tb_dlx_coverage_instructions");
```

```
                $display("@%0dns␣Starting␣Test␣Program␣in␣System␣
                   Verilog",$time);
                $display("Reset",);
                rst=0;
                ##3;
                rst=1;
                cg_instruction.start();
                cg_instruction.set_inst_name("instruction␣coverage␣
                   group␣tb␣top␣dlx");
                fork
                        forever begin
                                wait($changed(current_instruction))
                                print_current_instruction(
                                   current_opcode,current_opcode_func
                                   );
                                cg_instruction.sample(current_opcode,
                                   current_opcode_func); // sample
                                   the covergroup
                                ##2;
                        end
                join_none
        // wait unti the end of the program in iram
        forever begin
                wait (end_seq.triggered)
                if(end_seq.triggered) begin
                        $display("Program␣in␣IRAM␣has␣ended␣@␣%d",
                           $time());
                        cg_instruction.stop();
                          // Display the coverage
                         $display("Total␣coverage␣of␣instructions␣%e",
                            cg_instruction.get_coverage());
                        $finish();
                end else begin
                ##10; // execute 10 more cc
                end
        end
        end



endprogram : test_dlx_top

module tb_dlx ();
        localparam clock_period= 10ns;
        // it needs the absolute path

        logic rst;
        logic clk;
        logic [$clog2(‘CU_STATES)-1:0] curr_state_debug_i;
        cu_state_t  curr_state_debug;
        logic [7:0] csr;
        logic DEBUG_iram_ready_cu;
        logic DEBUG_iram_enable_cu;
        logic DEBUG_signed_notsigned;
        logic DEBUG_compute_sext;
        logic DEBUG_jump_sext;
        logic DEBUG_write_rf;
        logic [1:0]DEBUG_evaluate_branch;
        logic DEBUG_alu_cin;
        logic DEBUG_alu_overflow;
        logic DEBUG_zero_mul_detect;
        logic DEBUG_mul_exeception;
        logic DEBUG_dram_ready_cu;
        logic DEBUG_dram_r_nw_cu;
        logic DEBUG_enable_rf;
        logic DEBUG_read_rf_p1;
        logic DEBUG_read_rf_p2;
        logic DEBUG_rtype_itypen;
        logic DEBUG_update_pc_branch;
        logic DEBUG_dram_enable_cu;
        logic [0:0]DEBUG_sel_val_a;
        logic [0:0]DEBUG_sel_val_b;
        logic [0:0]DEBUG_select_wb;
        logic [3:0]DEBUG_alu_op_type_i;
        TYPE_OP_ALU_sv DEBUG_alu_op_type;

        assign curr_state_debug=cu_state_t’(curr_state_debug_i);
        assign DEBUG_alu_op_type=TYPE_OP_ALU_sv’(DEBUG_alu_op_type_i);

        initial begin
                clk = ’0;
                forever #(clock_period/2) clk = ~clk;
        end

        // Specify the default clocking
        default clocking test_dlx @ (posedge clk);
        endclocking     // clock


        initial begin
                $display("Attention!!",);
                $display("Starting␣simulated␣execution␣of␣the␣DLX",);
                $display("Starting␣an␣amazing␣and␣very␣fancy␣testbench
                   ␣in␣System␣Verilog",);
        end

‘include "./006-property_def.svh"
```

```
    //
       ////////////////////////////////////////////////////////////////////////////

    ///// instantiate property and coverage groupd defined in
       property_def.svh ///
    //
       ////////////////////////////////////////////////////////////////////////////

    instructions_regtype_opcode ireg_instr;
instructions_opcode imm_instru,jump_instr,lw_instr,sw_instr,
   b_instr;
// cast from bit to typedef of instruction
always_comb begin : proc_cast
            // type_t'(x) cast x to type_t
            ireg_instr=instructions_regtype_opcode'(iram_if.DATA[
               `OP_CODE_SIZE-1:0]);
            imm_instru=instructions_opcode'(iram_if.DATA[
               `IRAM_WORD_SIZE-1:`IRAM_WORD_SIZE-`OP_CODE_SIZE]);
            jump_instr=instructions_opcode'(iram_if.DATA[
               `IRAM_WORD_SIZE-1:`IRAM_WORD_SIZE-`OP_CODE_SIZE]);
            lw_instr=instructions_opcode'(iram_if.DATA[
               `IRAM_WORD_SIZE-1:`IRAM_WORD_SIZE-`OP_CODE_SIZE]);
            sw_instr=instructions_opcode'(iram_if.DATA[
               `IRAM_WORD_SIZE-1:`IRAM_WORD_SIZE-`OP_CODE_SIZE]);
            b_instr=instructions_opcode'(iram_if.DATA[
               `IRAM_WORD_SIZE-1:`IRAM_WORD_SIZE-`OP_CODE_SIZE]);

end

    // property instantiation
instruction_check_property_ireg : cover property (
   instruction_check_ireg);

instruction_check_property_i : cover property(instruction_check_i)
   ;

instruction_check_property_jump : cover property(
   instruction_check_jump);

instruction_check_property_lw : cover property(
   instruction_check_lw) ;

instruction_check_property_sw : cover property(
   instruction_check_sw) ;

instruction_check_property_b : cover property(instruction_check_b)
    ;
```

```
multiplication_stall_check_property : cover property(
   multiplication_stall) ;

    ////////////////////////////////////
    // instantiate the components ///////
    ////////////////////////////////////

    // Instruction memory
    // instantiate the interface
    mem_interface #(.ADDRESS_SIZE(`IRAM_ADDRESS_SIZE),
            .WORD_SIZE(`IRAM_WORD_SIZE))
    iram_if (.clk(clk));

    // instantiate the dut and connect the interface
    romem #(.FILE_PATH   (`PATH_TO_IMEM),
            .WORD_SIZE   (`IRAM_WORD_SIZE),
            .ADDRESS_SIZE(`IRAM_ADDRESS_SIZE/2),
            .DATA_DELAY  (2))
            iram
    (.mif(iram_if.ro));



    mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                    .WORD_SIZE(`DRAM_WORD_SIZE))
    dram_if (.clk(clk));
    // Data memory
    rwmem #(
            .FILE_PATH     (`PATH_TO_DMEM_FINAL),
            .FILE_PATH_INIT(`PATH_TO_DMEM),
            .WORD_SIZE     (`DRAM_WORD_SIZE),
            .ADDRESS_SIZE  (`DRAM_ADDRESS_SIZE/2),
            .DATA_DELAY    (2))
    dram ( .memif(dram_if.rw));


    //DLX top level entity
    DLX #(
  .IR_SIZE(`IRAM_WORD_SIZE),
  .PC_SIZE(`IRAM_ADDRESS_SIZE)
) dlx_uut (
    // verbose assignmento of the interfaces signals
    // Inputs
  .CLK                              (clk),
  .RST                              (rst),
  // Instruction memory interface
  .IRAM_ADDRESS             (iram_if.ADDRESS),
  .IRAM_ENABLE              (iram_if.ENABLE),
  .IRAM_READY               (iram_if.DATA_READY),
    .IRAM_DATA                  (iram_if.DATA),
    // Data memory Interface
    .DRAM_ADDRESS               (dram_if.ADDRESS),
    .DRAM_ENABLE        (dram_if.ENABLE),
    .DRAM_READNOTWRITE  (dram_if.READNOTWRITE),
    .DRAM_READY         (dram_if.DATA_READY),
    .DRAM_DATA                  (dram_if.INOUT_DATA)
    // simulation debug signals
    //synthesis_translate off
    ,
    .STATE_CU                   (curr_state_debug_i),
    .csr                (csr),
    //all the control unit signals
    //used for system verilog verification
    .DEBUG_iram_ready_cu(DEBUG_iram_ready_cu),
    .DEBUG_iram_enable_cu(DEBUG_iram_enable_cu),
    .DEBUG_signed_notsigned(DEBUG_signed_notsigned),
    .DEBUG_compute_sext(DEBUG_compute_sext),
    .DEBUG_jump_sext(DEBUG_jump_sext),
    .DEBUG_write_rf(DEBUG_write_rf),
    .DEBUG_evaluate_branch(DEBUG_evaluate_branch),
    .DEBUG_alu_cin(DEBUG_alu_cin),
    .DEBUG_alu_overflow(DEBUG_alu_overflow),
    .DEBUG_zero_mul_detect(DEBUG_zero_mul_detect),
    .DEBUG_mul_exeception(DEBUG_mul_exeception),
    .DEBUG_dram_ready_cu(DEBUG_dram_ready_cu),
    .DEBUG_dram_r_nw_cu(DEBUG_dram_r_nw_cu),
    .DEBUG_enable_rf(DEBUG_enable_rf),
    .DEBUG_read_rf_p1(DEBUG_read_rf_p1),
    .DEBUG_read_rf_p2(DEBUG_read_rf_p2),
    .DEBUG_rtype_itypen(DEBUG_rtype_itypen),
        .DEBUG_sel_val_a(DEBUG_sel_val_a),
    .DEBUG_sel_val_b(DEBUG_sel_val_b),
        .DEBUG_update_pc_branch(DEBUG_update_pc_branch),
    .DEBUG_select_wb(DEBUG_select_wb),
    .DEBUG_alu_op_type(DEBUG_alu_op_type_i),
    .DEBUG_dram_enable_cu(DEBUG_dram_enable_cu)
    //synthesis_translate on
  );

//assign same reset signal to both interfaces
assign iram_if.rst=rst;
assign dram_if.rst=rst;

 test_dlx_top test_prog(.clk(clk),
                                                  .rst(rst),
                                                  .curr_state(
                                                     curr_state_debug),
                                                  .current_instruction(
                                                     iram_if.DATA));
endmodule
```

APPENDIX B

Memories interfaces

```
`ifndef __MEMORY_INTERFACES__VH
`define __MEMORY_INTERFACES__VH

`timescale 1ns/1ps

interface mem_interface
        #(parameter ADDRESS_SIZE=16,
        WORD_SIZE=32)
         ( input wire clk);

         logic rst;  //  reset active low
         logic  [ADDRESS_SIZE-1:0] ADDRESS;
         logic  ENABLE;
         wire   DATA_READY;
         wire   [WORD_SIZE-1:0]DATA;
         logic  READNOTWRITE;
         wire   [WORD_SIZE-1:0] INOUT_DATA;


        // for umv tb
        logic [WORD_SIZE-1:0]DATA_UVM;
        logic [WORD_SIZE-1:0]INOUT_DATA_UVM;


    clocking ram_interface @(posedge clk);
       input   #1  ADDRESS,ENABLE; // sampled after 1 time resoltuon
          see `timescale
       output  #1  DATA_READY,DATA;
    endclocking

    // automatic bind the cover points to the address
                covergroup address_cov () @ (posedge ENABLE iff rst);
                    addr : coverpoint ADDRESS {
                       option.auto_bin_max = 10; // max number of bins
                    }
                  endgroup

                 // explicitly bind the coverpoitn
      covergroup memory_rw @ (posedge READNOTWRITE or negedge
         READNOTWRITE);
       read_write : coverpoint READNOTWRITE {
        bins  read  = {0};
        bins  write = {1};
      }
    endgroup
    //=======
    // instantiate cover group

    address_cov cov_address = new();
    memory_rw cov_rw = new();

modport rw (input ADDRESS, ENABLE, READNOTWRITE,rst,clk, inout
   INOUT_DATA, output DATA_READY); // read write memory interface
modport ro (input ADDRESS, ENABLE, rst,clk , output DATA_READY, DATA);
    // read only memory interface

endinterface



`endif//__MEMORY_INTERFACES__VH
```

APPENDIX C

Global Definitions

```
//
//                                              \\\ Sapere Aude ///
//
//
   ------------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   ------------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : global_defs.svh
// Create : 2020-07-27 17:47:25
// Revise : 2020-07-27 17:47:25
// Editor : sublime text3, tab size (4)
// Description: util global definition (mirror of 000-globals.vhd)
//
   ------------------------------------------------------------------------------

`ifndef  __GLOBAL_DEFS__SVH
`define  __GLOBAL_DEFS__SVH

`define  NUMBIT 32 // number of bits for the architecture
`define  RF_REGS 32 // number of regisster in the register file
`define  IRAM_WORD_SIZE 32
`define  IRAM_SIZE 2**32-1
`define  IRAM_ADDRESS_SIZE 32
`define  DRAM_WORD_SIZE 32
`define  DRAM_SIZE 2**32-1
`define  DRAM_ADDRESS_SIZE 32

`define CU_STATES 3
```

```
`define  ENDIANESS                       "big"; // for memory access
`define  OPCODE_LENGTH                   6  // length in the
   instruction
`define  REGISTER_ADDRESS_FIELD_LENGTH   5  // length in the
   instruction
`define  IMMEDIATE_LENGTH                16 //  for I-type instructions
`define  ALU_FUNCTION_LENGTH             11 // for R-type instructions
`define  JUMP_ADDRESS_LENGTH             26 // for J-type instruction
`define  OP_CODE_SIZE  6  // OPCODE field size
`define  FUNC_SIZE      11 // FUNC field size
`define  TOT_CU_SIGN  22  // number of total signoal (I/O) of control
   unit

typedef enum bit[$clog2(`CU_STATES)-1:0]{hang_error,fetch, decode }
   cu_state_t;

typedef enum  bit  [3:0]{ADD=4'h0,
                                                SUB=4'h1,
                                                MULT=4'h2,
                                                BITAND=4'h3,
                                                BITOR=4'h4,
                                                BITXOR=4'h5,
                                                FUNCLSL=4'h6,
                                                FUNCLSR=4'h7,
                                                GE=4'h8,
                                                LE=4'h9,
                                                NE=4'ha}
                                                   TYPE_OP_ALU_sv ;
                                                   // cause error in
                                                   importing the enum
                                                    from vhdl


// uncomment using vivado simulator 'cause some sv system call are not
    supported from vivavo simulator
//`define VIVADO_SIM 1 // comment using questa sim simulator


// iram and dram content path
`define PATH_TO_DMEM_FINAL "/home/ms20.50/Desktop/DLX_project/hardware
   /dlx/test_bench/memories/dram_out.txt"
`define PATH_TO_DMEM "/home/ms20.50/Desktop/DLX_project/hardware/dlx/
   test_bench/memories/dram.txt"
`define PATH_TO_IMEM "/home/ms20.50/Desktop/DLX_project/hardware/dlx/
   test_bench/memories/sbst_dump.txt"



// debug interfae for simulation purposes
```

```
interface DEBUG_interface ();
        logic iram_ready_cu;
        logic iram_enable_cu;
        logic signed_notsigned;
        logic compute_sext;
        logic jump_sext;
        logic write_rf;
        logic [1:0]evaluate_branch;
        logic alu_cin;
        logic alu_overflow;
        logic zero_mul_detect;
        logic mul_exeception;
        logic dram_ready_cu;
        logic dram_r_nw_cu;
        logic enable_rf;
        logic read_rf_p1;
        logic read_rf_p2;
        logic rtype_itypen;
        logic [0:0]sel_val_a;
        logic [0:0]sel_val_b;
        logic update_pc_branch;
        logic [0:0]select_wb;
        logic dram_enable_cu;
        logic [7:0] csr;
        logic rst;
        logic [$clog2(`CU_STATES)-1:0]  curr_state_debug;
    logic [3:0]alu_operation;


endinterface

`endif // __GLOBAL_DEFS__SVH
```

APPENDIX D

Implemented Instructions

```
//
//                                        \\\ Sapere Aude ///
//
//
   ------------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   ------------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : implemented_instructions.svh
// Create : 2020-07-22 20:15:37
// Revise : 2020-07-22 20:15:37
// Editor : sublime text3, tab size (4)
// Description:  List of current implemented instruction in the CU of
   the DLX
//
   ------------------------------------------------------------------------------

‘ifndef __implemented_instructions__svh
‘define __implemented_instructions__svh

‘define IMPLEMENTED_INSTR 28

typedef bit[5:0] bit_instr;


typedef enum bit_instr {
   i_regtype=6’h00,
   i_addi= 6’h08,
   i_andi= 6’h0c,
   i_beqz= 6’h04,
   i_benz= 6'h05,
   i_j= 6'h02,
   i_jal= 6'h03,
   i_lw= 6'h23,
   i_nop=6'h15,
   i_sgei= 6'h1d,
   i_slei= 6'h1c,
   i_slli= 6'h14,
   i_snei= 6'h19,
   i_srli= 6'h16,
   i_subi=6'h0a,
   i_sw= 6'h2b,
   i_ori=6'h0d,
   i_xori=6'h0e
} instructions_opcode;

typedef enum bit_instr {
   i_add=6'h20,
   i_and=  6'h24,
   i_or= 6'h25,
   i_sll= 6'h04,
   i_srl= 6'h06,
   i_sub= 6'h22,
   i_xor= 6'h26,
   i_sne= 6'h29,
   i_sle= 6'h2c,
   i_sge= 6'h2d,
   i_mul=6'h3f
} instructions_regtype_opcode; // in the function field

/*for printing instruction name*/
virtual class enum_wrap_instruction#(type T);
  static function string name(T obj);
    return obj.name();
  endfunction
endclass



//General instructions
/*j j,0x02  jal j,0x03  beqz b,0x04  bnez b,0x05  bfpt b0,0x06 bfpf b0
   ,0x07  addi i,0x08
addui i,0x09  subi i,0x0a  subui i,0x0b  andi i,0x0c ori i,0x0d  xori
   i,0x0e
lhi i1,0x0frfe n,0x10 trap t,0x11 jr jr,0x12 jalr jr,0x13slli i,0
   x14nop n,0x15srli i,0x16 srai i,0x17 seqi i,0x18 snei i,0x19
slti i,0x1a sgti i,0x1b slei i,0x1c sgei i,0x1d lb l,0x20 lh l,0x21lw
   l,0x23lbu l,0x24lhu l,0x25
lf l,0x26ld l,0x27sb s,0x28sh s,0x29sw s,0x2bsf s,0x2esd s,0x2fitlb n
```

```
    ,0x38sltui i,0x3a sgtui i,0x3b sleui i,0x3c sgeui i,0x3d
*/

// Register-register instructions
/*sll r,0x04 srl r,0x06sra r,0x07 add r,0x20 addu r,0x21sub r,0x22subu
     r,0x23
and r,0x24 or r,0x25xor r,0x26 seq r,0x28 sne r,0x29 slt r,0x2a sgt r
    ,0x2b sle r,0x2c sge r,0x2dmovi2s r2,0x30 movs2i r2,0x31 movf r2,0
    x32
movd r2,0x33 movfp2i r2,0x34 movi2fp r2,0x35  movi2t r,0x36 movt2i r,0
    x37  sltu r,0x3a  sgtu r,0x3b  sleu r,0x3c  sgeu r,0x3d
*/


//Floating-point instruction
/*addf f,0x00 subf f,0x01multf f,0x02divf f,0x03addd f,0x04subd f,0
    x05multd f,0x06divd f,0x07
cvtf2d fd,0x08cvtf2i fd,0x09cvtd2f fd,0x0acvtd2i fd,0x0bcvti2f fd,0
    x0ccvti2d fd,0x0dmult f,0x0e
div f,0x0f eqf f2,0x10 nef f2,0x11 ltf f2,0x12 gtf f2,0x13 lef f2,0x14
     gef f2,0x15 multu f,0x16
divu f,0x17 eqd f2,0x18 ned f2,0x19 ltd f2,0x1a gtd f2,0x1b led f2,0
    x1c ged f2,0x1d
*/

`endif // __implemented_instructions__svh
```

APPENDIX E

Properties definitions

```
//
//                                        \\\ Sapere Aude ///
//
//
   ----------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   ----------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : property_def.svh
// Create : 2020-08-01 16:14:31
// Revise : 2020-08-01 16:14:31
// Editor : sublime text3, tab size (4)
// Description: Defintion of all property and sequences for the DLX
   top level entity
//
   ----------------------------------------------------------------------------

`ifndef __PROPERTY_DEF_SVH
`define __PROPERTY_DEF_SVH

`include "./003-global_defs.svh"
`include "./004-implemented_instructions.svh"

        // property definition
    property multiplication_stall;
        @(test_dlx)
            disable iff(!rst && !DEBUG_zero_mul_detect && !
               DEBUG_mul_exeception )
                                (ireg_instr===i_mul) |-> !
                                   DEBUG_iram_enable_cu[*9];// no
```

```
                                       fetching for 9 cc
    endproperty;

  /* sequence for reg type instructions*/
    sequence ireg_decode;
        ##1  DEBUG_enable_rf && DEBUG_read_rf_p1 && DEBUG_read_rf_p2
           && DEBUG_rtype_itypen && !DEBUG_compute_sext && !
           DEBUG_jump_sext;
    endsequence ;

    sequence ireg_execute(cin);
        ##1 !DEBUG_sel_val_a[0] && !DEBUG_sel_val_b[0] && (
           DEBUG_alu_cin===cin) && !DEBUG_evaluate_branch[1] && !
           DEBUG_evaluate_branch[0] && DEBUG_signed_notsigned ;
    endsequence;

    sequence ireg_memory;
        ##1 !DEBUG_dram_enable_cu ;
    endsequence ;

    sequence ireg_wb;
        ##1 DEBUG_write_rf && DEBUG_select_wb[0];
    endsequence ;
/*sequence for immediate instruction */
    sequence itype_decode;
        ##1  DEBUG_enable_rf && DEBUG_read_rf_p1 && !DEBUG_read_rf_p2
           && !DEBUG_rtype_itypen && DEBUG_compute_sext && !
           DEBUG_jump_sext;
    endsequence ;

    sequence itype_execute;
        ##1 !DEBUG_sel_val_a[0] && DEBUG_sel_val_b[0]  && !
           DEBUG_evaluate_branch[1] && !DEBUG_evaluate_branch[0] &&
           DEBUG_signed_notsigned ;
    endsequence ;

    sequence itype_memory;
        ##1 !DEBUG_dram_enable_cu ;
    endsequence ;

    sequence itype_wb;
        ##1 DEBUG_write_rf && DEBUG_select_wb[0];
    endsequence ;
/*sequnce for lw*/
    sequence lw_decode;
        ##1  DEBUG_enable_rf && DEBUG_read_rf_p1 && !DEBUG_read_rf_p2
           && !DEBUG_rtype_itypen && DEBUG_compute_sext && !
           DEBUG_jump_sext;
    endsequence ;
    sequence lw_execute;
        ##1 !DEBUG_sel_val_a[0] && DEBUG_sel_val_b[0] && !
           DEBUG_alu_cin && !DEBUG_evaluate_branch[1] && !
           DEBUG_evaluate_branch[0] && DEBUG_signed_notsigned ;
    endsequence ;

    sequence lw_memory;
        ##1 DEBUG_dram_enable_cu  && DEBUG_dram_r_nw_cu;
    endsequence ;

    sequence lw_wb;
        ##1 DEBUG_write_rf && !DEBUG_select_wb[0];
    endsequence ;

/*sequnce for sw*/
    sequence sw_decode;
        ##1  DEBUG_enable_rf && DEBUG_read_rf_p1 && DEBUG_read_rf_p2
           && !DEBUG_rtype_itypen && DEBUG_compute_sext && !
           DEBUG_jump_sext;
    endsequence;

    sequence sw_execute;
        ##1 !DEBUG_sel_val_a[0] && DEBUG_sel_val_b[0] && !
           DEBUG_alu_cin && !DEBUG_evaluate_branch[1] && !
           DEBUG_evaluate_branch[0] && DEBUG_signed_notsigned ;
    endsequence ;

    sequence sw_memory;
        ##1 DEBUG_dram_enable_cu  && !DEBUG_dram_r_nw_cu;
    endsequence ;

    sequence sw_wb;
        ##1 !DEBUG_write_rf ;
    endsequence ;

/*sequnce for b*/
    sequence b_decode;
        ##1  DEBUG_enable_rf && DEBUG_read_rf_p1 && !DEBUG_read_rf_p2
           && !DEBUG_rtype_itypen && DEBUG_compute_sext && !
           DEBUG_jump_sext;
    endsequence ;

    sequence beqz_execute;

            ##1 DEBUG_sel_val_a[0] && DEBUG_sel_val_b[0] && !
               DEBUG_alu_cin && !DEBUG_evaluate_branch[1] &&
               DEBUG_evaluate_branch[0] && DEBUG_signed_notsigned ;
```

```
   endsequence ;

       sequence benz_execute;
                // i_benz
           ##1 DEBUG_sel_val_a[0] && DEBUG_sel_val_b[0] && !
              DEBUG_alu_cin && DEBUG_evaluate_branch[1] && !
              DEBUG_evaluate_branch[0] && DEBUG_signed_notsigned ;
       endsequence;
   sequence b_memory;
       ##1 !DEBUG_dram_enable_cu  ;
   endsequence ;

   sequence b_wb;
       ##1 !DEBUG_write_rf ;
   endsequence;
/*sequence for jump instruction*/
   sequence ijump_decode;
       ##1  !DEBUG_enable_rf && !DEBUG_read_rf_p1 && !
          DEBUG_read_rf_p2 && !DEBUG_rtype_itypen &&
          DEBUG_compute_sext &&  DEBUG_jump_sext;
   endsequence ;

   sequence ijumpal_decode;
       ##1 !DEBUG_enable_rf && !DEBUG_read_rf_p1 && !DEBUG_read_rf_p2
           && !DEBUG_rtype_itypen && DEBUG_compute_sext &&
          DEBUG_jump_sext;
   endsequence;

   sequence ijump_execute;
       ##1 DEBUG_sel_val_a[0] && DEBUG_sel_val_b[0] && !DEBUG_alu_cin
           && !DEBUG_evaluate_branch[1] && DEBUG_evaluate_branch[0]
          && DEBUG_signed_notsigned ;
   endsequence ;

   sequence ijump_memory;
       ##1 !DEBUG_dram_enable_cu ;
   endsequence ;

   sequence ijump_wb;
       ##1 !DEBUG_write_rf;
   endsequence ;

   sequence ijal_wb;
       ##1 DEBUG_write_rf && DEBUG_select_wb[0];
   endsequence ;

   property instruction_check_ireg;
       @(test_dlx)
       // iram enable cu is for the fetch stage
```

```
    disable iff (!rst && ireg_instr!==0)
        // reg type
                    if(ireg_instr===i_sub)
                            DEBUG_iram_enable_cu |-> ireg_decode
                                |-> ireg_execute(1) |->
                                ireg_memory |-> ireg_wb
                    else
                            DEBUG_iram_enable_cu |-> ireg_decode
                                |-> ireg_execute(0) |->
                                ireg_memory |-> ireg_wb
endproperty;

    property instruction_check_jump;
            @(test_dlx)
                    disable iff(!rst && (jump_instr!==i_j||
                        jump_instr!==i_jal))
        if (jump_instr===i_jal )
                            DEBUG_iram_enable_cu |->
                                ijumpal_decode |->  ijump_execute
                                |-> ijump_memory |->  ijal_wb
        else
            DEBUG_iram_enable_cu |-> ijump_decode |->
                ijump_execute |->  ijump_memory |->  ijump_wb;
    endproperty;

property instruction_check_lw;
                            @(test_dlx)
                    disable iff(!rst && lw_instr!==i_lw )
                            DEBUG_iram_enable_cu |-> lw_decode |->
                                 lw_execute |-> lw_memory |->
                                lw_wb;
    endproperty;


    property instruction_check_sw;
            @(test_dlx)
                    disable iff(!rst && sw_instr!==i_sw)
                        DEBUG_iram_enable_cu |-> sw_decode |->
                            sw_execute |->  sw_memory |->  sw_wb;
    endproperty;


    property instruction_check_b;
                    @(test_dlx)
                    disable iff(!rst && ( b_instr!==i_beqz||
                        b_instr!==i_benz ))
                                if ( b_instr===i_beqz )
                                 DEBUG_iram_enable_cu |->
                                    b_decode |->
```

```
                                                beqz_execute |->
                                                b_memory |->  b_wb
                                            else
                                             DEBUG_iram_enable_cu |->
                                                b_decode |->
                                                benz_execute |->
                                                b_memory |->  b_wb;
        endproperty;

     property instruction_check_i;
                @(test_dlx)
                        disable iff(!rst  && ( b_instr!==i_beqz||
                           b_instr!==i_benz )  && sw_instr!==i_sw  &&
                            sw_instr!==i_lw
                                             && (jump_instr!==i_j||
                                                jump_instr!==i_jal) &&
                                                ireg_instr!==0)
                DEBUG_iram_enable_cu |-> itype_decode |->
                   itype_execute |->  itype_memory |->  itype_wb; //
                   itype
        endproperty;


`endif //__PROPERTY_DEF_SVH
```

APPENDIX F

UVM Top testbench

```
//
//                                        \\\ Sapere Aude ///
//
//
   -----------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   -----------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : tb_dlx_uvm.sv
// Create : 2020-30-08 19:00:18
// Revise : 2020-08-21 20:02:24
// Editor : sublime text3, tab size (4)
// Description:  top level test bench for DLX fuctional verification
   testing
//                                                UVM architecture
//
   -----------------------------------------------------------------------------


`timescale  1ns/1ps
`include "./003-global_defs.svh"
`include "./004-implemented_instructions.svh"
`include "./memories/005-memory_interfaces.svh"

import uvm_pkg::*;

`include <uvm_macros.svh>
`include <uvm_pkg.sv>

// including uvm classes in a bottom up order
```

```
`include "./uvm_class_def/dlx_sequence_item.sv"
`include "./uvm_class_def/dlx_sequence.sv"
`include "./uvm_class_def/dlx_sequencer.sv"
`include "./uvm_class_def/dlx_driver.sv"
`include "./uvm_class_def/dlx_monitor.sv"
`include "./uvm_class_def/dlx_scoreboard.sv"
`include "./uvm_class_def/dlx_env.sv"
`include "./uvm_class_def/dlx_test.sv"

module tb_dlx_uvm ();
        localparam clock_period= 10ns;
        // it needs the absolute path

        logic clk;

        initial begin
                clk = '0;
                forever #(clock_period/2) clk = ~clk;
        end

        // Specify the default clocking
        default clocking test_dlx @ (posedge clk);
        endclocking     // clock


        initial begin
                $display("Attention!!",);
                $display("Starting␣simulated␣execution␣of␣the␣DLX␣
                    usign␣UVM␣architecture",);
        end

        /////////////////////////////////////
        // instantiate the components ///////
        /////////////////////////////////////

        // instantiate the interface
        mem_interface #(.ADDRESS_SIZE(`IRAM_ADDRESS_SIZE),
                .WORD_SIZE(`IRAM_WORD_SIZE))
        iram_if (.clk(clk));

        mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE))
        dram_if (.clk(clk));

        DEBUG_interface dbg_if ();

        dlx_wrapper uut_uuv (.clk(clk),
                                        .rst   (dbg_if.rst), // active
                                            low
                                        .mif_ro(iram_if),// memory
                                           interface clocked by clk
                                        .mif_rw(dram_if),// memory
                                           interface clocked by clk
                                        .dbg_if(dbg_if));

assign iram_if.DATA= iram_if.ENABLE ? iram_if.DATA_UVM : '0;
assign dram_if.INOUT_DATA= dram_if.ENABLE ? dram_if.INOUT_DATA_UVM : '
   z;
        initial begin
        // set interfaces
        uvm_config_db#( virtual DEBUG_interface)::set(null,"*","dbg_if
           ",dbg_if);
    uvm_config_db#( virtual mem_interface #(.ADDRESS_SIZE(
       `IRAM_ADDRESS_SIZE),
               .WORD_SIZE(`IRAM_WORD_SIZE)) )::set(null,"*","iram_if"
                  ,iram_if);
    uvm_config_db#( virtual mem_interface #(.ADDRESS_SIZE(
       `IRAM_ADDRESS_SIZE),
               .WORD_SIZE(`IRAM_WORD_SIZE)) )::set(null,"*","dram_if"
                  ,dram_if);
        $dumpfile("dump.vcd"); $dumpvars;   //enabling the wave dump
        end
  initial begin
        $display("Test␣is␣starting....",);
    run_test("test"); // start the test
    $display("And␣finished!␣:D␣",);

  end

endmodule
```

APPENDIX G

UVM classes

```
//
//                                          \\\ Sapere Aude ///
//
//
   ------------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   ------------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_sequence_item.sv
// Create : 2020-09-01 21:57:14
// Revise : 2020-09-04 16:48:39
// Editor : sublime text3, tab size (4)
// Description : it contains the test program
//
   ------------------------------------------------------------------------------



`ifndef __DLX_SEQUENCE_ITEM_SV
`define __DLX_SEQUENCE_ITEM_SV
`include "../003-global_defs.svh"
`include "../004-implemented_instructions.svh"

import uvm_pkg::*;

`include <uvm_macros.svh>
`include <uvm_pkg.sv>

class instruction_item extends uvm_sequence_item;
```

```
  bit[32-1:0] signals;
  bit[`IRAM_WORD_SIZE-1:0] current_instruction;
  rand instructions_opcode current_opcode;
  rand bit[5-1:0] rd;
  rand bit[5-1:0] rs1;
  rand bit[5-1:0] rs2;
  rand bit[15:0] immediate;
  rand bit[25:0] jump_address;
  rand instructions_regtype_opcode current_opcode_func;
`uvm_object_utils_begin(instruction_item)
        `uvm_field_enum(instructions_opcode,current_opcode,UVM_ALL_ON)
        `uvm_field_enum(instructions_regtype_opcode,
           current_opcode_func,UVM_ALL_ON)
`uvm_object_utils_end

    constraint c1 { rd>0 ; rd <31;}
    constraint c2 {rs2>=0 ; rs2 <30;}
    constraint c3 { rs1>=0 ; rs1 <30;}

  function void compose_instruction(); // it is actually the set
     curret instruction
        `uvm_info(get_type_name(),$sformatf("Composing␣instruction:␣
           opcode:0x%h␣rs1:␣%d␣rs2:%d␣rd:%d␣opcode_func:0x%h",this.
           current_opcode,this.rs1,this.rs2,this.rd,this.
           current_opcode_func),UVM_LOW)
        if(current_opcode==i_regtype) begin
                this.current_instruction= {current_opcode,rs1 ,rs2 ,rd
                   , current_opcode_func};
        end else if (this.current_opcode===i_j || this.current_opcode
           ===i_jal) begin
                this.current_instruction= {current_opcode,jump_address
                   };
        end else begin  // immediate or benq or sw lw
                this.current_instruction= {current_opcode,rs1 ,rd,
                   immediate};
        end
        endfunction

  function   bit[`IRAM_WORD_SIZE-1:0] get_current_instruction();
        return this.current_instruction;
  endfunction : get_current_instruction

  function instructions_opcode get_opcode ();
        return this.current_opcode;
  endfunction : get_opcode

  function instructions_regtype_opcode get_opcode_func ();
        return this.current_opcode_func;
  endfunction : get_opcode_func

  function void force_nop ();
        this.current_opcode=i_nop;
  endfunction : force_nop


  function new(string name = "instruction_item" );
    super.new(name);
        // deault instruction is a nop
/*      this.current_opcode=i_nop;
        this.rs1=5'h0;
        this.rs2=5'h0;
        this.rd=5'h0;
        this.immediate=0;
        this.current_opcode_func=i_add;*/
  endfunction


   function string convert2str ();
         if(this.current_opcode===i_regtype) begin
                return $sformatf("%s␣%d,%d,%d",  enum_wrap_instruction
                   #(instructions_regtype_opcode)::name(
                   current_opcode_func) ,rs1,rs2,rd);
        end else if (this.current_opcode===i_j || this.current_opcode
           ===i_jal) begin
                return $sformatf("%s␣%h",  enum_wrap_instruction#(
                   instructions_opcode)::name(current_opcode) ,
                   jump_address);
        end else begin  // immediate or benq or sw lw
                return $sformatf("%s␣%d␣%d␣%d",enum_wrap_instruction#(
                   instructions_opcode)::name(current_opcode),rs1,rd,
                   immediate);
        end
   endfunction : convert2str

   function string get_current_instruction_name();
         if(this.current_opcode===i_regtype) begin
                return enum_wrap_instruction#(
                   instructions_regtype_opcode)::name(
                   current_opcode_func);
         end else begin
                return enum_wrap_instruction#(instructions_opcode)::
                   name(current_opcode);
         end
   endfunction : get_current_instruction_name


   function integer get_signals ();
```

```
        return {this.signals[31:15], 15'd0};
endfunction : get_signals

function integer get_carry_in ();
        return this.signals[4];
endfunction : get_carry_in


function bit[3:0] get_alu_op ();
        //return this.signals[3:0];
        bit[3:0] tmp;
                if(current_opcode==i_regtype) begin
                        case(current_opcode_func)
                           i_and: tmp=BITAND;
                           i_or: tmp=BITOR;
                           i_sll: tmp=FUNCLSL;
                           i_srl: tmp=FUNCLSR;
                           i_sub: tmp=SUB;
                           i_xor: tmp=BITXOR;
                           i_sne: tmp=NE;
                           i_sle: tmp=LE;
                           i_sge: tmp=GE;
                           i_mul: tmp= MULT;
                        default  tmp=ADD;
                        endcase
                end else begin
                        case (current_opcode)
                          i_andi: tmp=BITAND;
                          i_sgei: tmp=GE;
                          i_slei: tmp=LE;
                          i_slli: tmp=FUNCLSL;
                          i_snei: tmp=NE;
                          i_srli: tmp=FUNCLSR;
                          i_subi: tmp=SUB;
                      i_ori:  tmp= BITOR;
                  i_xori:  tmp=BITXOR;
                        default  tmp=ADD; /*i_addi , i_nop, i_lw,
                           i_jal, i_j, i_benz, i_beqz*/
                        endcase
                end
        return tmp;
endfunction : get_alu_op

function void set_signals (int sample_val,int cc);
        if(cc==0) begin
        this.signals=sample_val;
        end else begin
        this.signals= this.signals | sample_val;
        end
```

```
endfunction : set_signals

function void collect_instruction( bit [`IRAM_WORD_SIZE-1:0] val);

this.current_opcode=instructions_opcode'(val[`IRAM_WORD_SIZE-1:
   `IRAM_WORD_SIZE-`OP_CODE_SIZE]);
        if(current_opcode==i_regtype) begin
                this.rs1=int'(val[`IRAM_WORD_SIZE-`OP_CODE_SIZE-1:
                   `IRAM_WORD_SIZE-`OP_CODE_SIZE-5]);
                this.rs2=int'(val[`IRAM_WORD_SIZE-`OP_CODE_SIZE-1-5:
                   `IRAM_WORD_SIZE-`OP_CODE_SIZE-10]);
                this.rd=int'(val[`IRAM_WORD_SIZE-`OP_CODE_SIZE-1-10:
                   `IRAM_WORD_SIZE-`OP_CODE_SIZE-15]);
                this.current_opcode_func=instructions_regtype_opcode'(
                   val[`OP_CODE_SIZE:0]);
        end else if (this.current_opcode===i_j || this.current_opcode
           ===i_jal) begin
                this.jump_address=val[`IRAM_WORD_SIZE-`OP_CODE_SIZE
                   -1:0];
        end else begin  // immediate or benq or sw lw
                this.rs1=int'(val[`IRAM_WORD_SIZE-`OP_CODE_SIZE-1:
                   `IRAM_WORD_SIZE-`OP_CODE_SIZE-5]);
                this.rd=int'(val[`IRAM_WORD_SIZE-`OP_CODE_SIZE-1-5:
                   `IRAM_WORD_SIZE-`OP_CODE_SIZE-10]);
                this.immediate=val[15:0];
        end

endfunction : collect_instruction

endclass

`endif

//
//                                        \\\ Sapere Aude ///
//
//
   -----------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   -----------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_sequence.sv
// Create : 2020-09-01 21:57:14
// Revise : 2020-09-04 16:48:39
// Editor : sublime text3, tab size (4)
// Description : it contains the test program
```

```
//
   ----------------------------------------------------------------------------



`ifndef __DLX_SEQUENCE_SV
`define __DLX_SEQUENCE_SV
`include "../003-global_defs.svh"
`include "../004-implemented_instructions.svh"
`include "dlx_sequencer.sv"
`include "./dlx_sequence_item.sv"

import uvm_pkg::*;

`include <uvm_macros.svh>
`include <uvm_pkg.sv>


class instruction_sequence extends uvm_sequence#(instruction_item);
        `uvm_object_utils(instruction_sequence)
        `uvm_declare_p_sequencer(instruction_sequencer)
//      `uvm_sequence_utils(instruction_sequence,instruction_sequencer
   )
  const int length_instr=30; //  number of instruciton to be executed
        instruction_item m_itemins;


  function new(string name="instruction␣sequence");
    super.new(name);
              // we may add the open of the file for the iram

  endfunction


  virtual task body();
          m_itemins=instruction_item::type_id::create("m_itemins");
        //for (int i = 0; i < length_instr; i ++) begin
        repeat(length_instr) begin
        `uvm_info("SEQ", $sformatf("Generate␣new␣item:␣"), UVM_LOW)
                start_item(m_itemins);
                if(!m_itemins.randomize()) begin
                        `uvm_error(get_type_name(), "FAILTED␣TO␣
                           RANDOMIZE")
                end else begin
                `uvm_info(get_type_name(), "Randomized␣correctly",
                   UVM_LOW)
                end
        finish_item(m_itemins);
    end
    `uvm_info("SEQ", $sformatf("Done␣generation␣of␣%0d␣items",
        length_instr), UVM_LOW)
  endtask



endclass

`endif

//
//                                        \\\ Sapere Aude ///
//
//
   ----------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   ----------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_sequencer.sv
// Create : 2020-09-01 21:57:14
// Revise : 2020-09-04 16:48:39
// Editor : sublime text3, tab size (4)
// Description : it contains the test program
//
   ----------------------------------------------------------------------------



`ifndef __DLX_SEQUENCER_SV
`define __DLX_SEQUENCER_SV
`include "../003-global_defs.svh"
`include "../004-implemented_instructions.svh"
`include "./dlx_sequence_item.sv"

import uvm_pkg::*;

`include <uvm_macros.svh>
`include <uvm_pkg.sv>


class instruction_sequencer extends uvm_sequencer#(instruction_item);
  `uvm_component_utils(instruction_sequencer)


  function new(string name="sequencer␣for␣instructions" ,uvm_component
      parent=null);
    super.new(name,parent);
//              `uvm_update_sequence_lib_and_item(instruction_item)
  endfunction
//      `uvm_sequencer_utils(instruction_sequencer)
endclass




`endif

//
//                                    \\\ Sapere Aude ///
//
//
   ---------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   ---------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_driver.sv
// Create : 2020-09-01 21:57:00
// Revise : 2020-09-05 18:01:43
// Editor : sublime text3, tab size (4)
// Description :
//
   ---------------------------------------------------------------------------




`ifndef __DLX_DRIVER_SV
`define __DLX_DRIVER_SV
`include "../memories/005-memory_interfaces.svh"
`include "./dlx_sequencer.sv"
`include "./dlx_sequence_item.sv"
import uvm_pkg::*;
`include <uvm_macros.svh>
`include <uvm_pkg.sv>


class driver extends uvm_driver #(instruction_item);
  `uvm_component_utils(driver)
  virtual       mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE))  iram_if;
  virtual       mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)) dram_if;
//  instruction_sequencer s0;
                instruction_item m_itemins;
  function new(string name = "driver", uvm_component parent=null);
    super.new(name, parent);
  endfunction

 virtual  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if (!uvm_config_db#(virtual         mem_interface #(.ADDRESS_SIZE(
       `DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)))::get(null, "", "
                           iram_if", iram_if))
      `uvm_fatal("DRV", "Could␣not␣get␣iram_if")
        if (!uvm_config_db#(virtual     mem_interface #(.ADDRESS_SIZE(
           `DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)))::get(null, "", "
                           dram_if", dram_if))
      `uvm_fatal("DRV", "Could␣not␣get␣dram_if")
  endfunction : build_phase

  virtual task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin

      `uvm_info("DRV", $sformatf("Wait␣for␣item␣from␣sequencer"),
         UVM_LOW)
                seq_item_port.get_next_item(m_itemins);
      drive_item(m_itemins);
      if(m_itemins.get_current_instruction()===i_lw || m_itemins.
         get_current_instruction()===i_sw) begin
        drive_item_dram();
      end
      seq_item_port.item_done();
    end
  endtask : run_phase

  virtual task drive_item(instruction_item m_item);
      @ (posedge iram_if.clk);
      while (!iram_if.ENABLE)  begin
        `uvm_info("DRV", "Wait␣until␣enable␣is␣high", UVM_LOW)
        @(posedge iram_if.clk);
      end
                m_item.compose_instruction();
      iram_if.DATA_UVM=m_item.get_current_instruction();
          repeat(2) @(posedge iram_if.clk);
  endtask : drive_item

  virtual task drive_item_dram();
          @ (posedge dram_if.clk);
      while (!dram_if.ENABLE)  begin
        `uvm_info("DRV", "Wait␣until␣enable␣is␣high␣for␣dram", UVM_LOW
           )
        @(posedge dram_if.clk);
      end
      if(dram_if.READNOTWRITE) begin
        dram_if.INOUT_DATA_UVM=$urandom_range(0,2**10);
      `uvm_info("DRV", "read␣operation␣dram", UVM_LOW)
          end else begin
      // skip the write operation
      `uvm_info("DRV", "write␣operation␣dram", UVM_LOW)
      end
           repeat(2) @(posedge dram_if.clk);
  endtask : drive_item_dram

endclass


`endif

//
//                                        \\\ Sapere Aude ///
//
//
   -----------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   -----------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_monitor.sv
// Create : 2020-09-01 21:56:48
// Revise : 2020-09-05 19:44:48
// Editor : sublime text3, tab size (4)
// Description :
//
   -----------------------------------------------------------------------------



`ifndef __DLX_MONITOR_SV
`define __DLX_MONITOR_SV
`include "../memories/005-memory_interfaces.svh"
`include "../003-global_defs.svh"
```

```
`include "./dlx_sequencer.sv"

import uvm_pkg::*;
`include <uvm_macros.svh>
`include <uvm_pkg.sv>

class monitor extends uvm_monitor;
  `uvm_component_utils(monitor)

  uvm_analysis_port  #(instruction_item) mon_analysis_port;  // Open
     analysis port to pass data to scoreboard
  virtual       mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)) iram_if;
  virtual       mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)) dram_if;
  virtual DEBUG_interface dbg_if;

        instruction_item collected_instr;

  function new(string name="monitor", uvm_component parent=null);
    super.new(name, parent);
  endfunction


 virtual  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    mon_analysis_port = new ("mon_analysis_port", this);
        collected_instr= new();
    if (!uvm_config_db#(virtual         mem_interface #(.ADDRESS_SIZE(
       `DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)))::get(this, "", "
                           iram_if", iram_if))
      `uvm_fatal("MON", "Could␣not␣get␣iram_if")

        if (!uvm_config_db#(virtual     mem_interface #(.ADDRESS_SIZE(
           `DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)))::get(this, "", "
                           dram_if", dram_if))
      `uvm_fatal("MON", "Could␣not␣get␣dram_if")

        if (!uvm_config_db#(virtual DEBUG_interface)::get(this, "", "
           dbg_if", dbg_if))
      `uvm_fatal("MON", "Could␣not␣get␣debug_if")


  endfunction

  virtual task run_phase(uvm_phase phase);
        int cc=0;
    super.run_phase(phase);
  forever begin
      @ (posedge iram_if.clk);
      if (iram_if.ENABLE) begin
        collected_instr.collect_instruction(iram_if.DATA_UVM);
        `uvm_info(get_type_name(), $sformatf("Monitor␣found␣
           instruction␣%s", collected_instr.convert2str()), UVM_LOW)
        // for the next 5 cc get the debug interface signals
        cc=1;
        repeat (2) @ (posedge iram_if.clk);
        collected_instr.set_signals( sample_dbg(cc),cc); // sample
           fetch decode
        cc=2;
        repeat (2) @ (posedge iram_if.clk);
        collected_instr.set_signals(sample_dbg(cc),cc); // sample
           execute
        cc=3;
        repeat (2) @ (posedge iram_if.clk);
        collected_instr.set_signals(sample_dbg(cc),cc); // sample
           memory
        cc=4;
        repeat (2) @ (posedge iram_if.clk);
        collected_instr.set_signals(sample_dbg(cc),cc); // sample wb
        cc=1;
        `uvm_info(get_type_name(),"Sampled␣debug␣signals", UVM_LOW)
      end
       mon_analysis_port.write(collected_instr); // pass the value to
          scoreboard
    end
  endtask

        function integer sample_dbg(integer cc_stage);
          bit[32-1:0] signals;
                      // depending on the current cc_state we sample
                         different part of DEBUG interface
        case (cc_stage)
                1:begin // fetch - decode
                      signals[31]=dbg_if.iram_ready_cu;
                      signals[30]=dbg_if.enable_rf;
                      signals[29]=dbg_if.read_rf_p1;
                      signals[28]=dbg_if.read_rf_p2;
                      signals[27]=dbg_if.rtype_itypen;
                      signals[26]=dbg_if.compute_sext;
                      signals[25]=dbg_if.jump_sext;
                end
                2:begin // execute
                      signals[24:24]=dbg_if.sel_val_a[0];
                      signals[23:23]=dbg_if.sel_val_b[0];
                      signals[22]=dbg_if.evaluate_branch[1];
```

```
                        signals[21]=dbg_if.evaluate_branch[0];
                        signals[20]=dbg_if.signed_notsigned;
                        signals[4]=dbg_if.alu_cin;
                        signals[3:0]=dbg_if.alu_operation;
                end
                3:begin // mem
                        signals[19]=dbg_if.dram_enable_cu;
                        signals[18]=dbg_if.dram_r_nw_cu;
                        signals[17]=dbg_if.update_pc_branch;
                end
                4:begin // write back
                        signals[16]=dbg_if.select_wb[0];
                        signals[15]=dbg_if.write_rf;

                end
                default : signals='0;
        endcase
        signals[14:7]= dbg_if.csr;
        signals[6]= dbg_if.rst;
        return integer'(signals);
        endfunction : sample_dbg

endclass


`endif

//
//                                      \\\ Sapere Aude ///
//
//
   ----------------------------------------------------------------------------

// Copyright (c) 2014-2020 All rights reserved
//
   ----------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_env.sv
// Create : 2020-09-01 21:55:56
// Revise : 2020-09-05 21:54:01
// Editor : sublime text3, tab size (4)
// Description :
//
   ----------------------------------------------------------------------------



`ifndef __DLX_ENV_SV
```

```
`define __DLX_ENV_SV
`include "./dlx_sequencer.sv"
`include "./dlx_driver.sv"
`include "./dlx_monitor.sv"
`include "./dlx_scoreboard.sv"

import uvm_pkg::*;
`include <uvm_macros.svh>
`include <uvm_pkg.sv>

// it is better to put sequencer driver and monitor into a component
   called agent ( agegnt is protocol dependent and can be put into
   different verification environment )
// active agent contains : sequencer, driver and monitor
// passive agent contains : monitor

class agent extends uvm_agent;
  `uvm_component_utils(agent)

  uvm_active_passive_enum  active = UVM_ACTIVE; // by default agent is
      active

  function new(string name="agent", uvm_component parent=null);
    super.new(name, parent);
  endfunction

  driver                d0;             // Driver handle
  monitor               m0;             // Monitor handle
  /*the sequence does not need ot be dirivef from seqeunce class  it
     can be instantiated directly in the env usign the object
     uvm_sequencer*/
  instruction_sequencer s0;             // Sequencer Handle

 virtual  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
// if agent is ACTIVE, then create monitor and sequencer, else create
   only monitor
      if (get_is_active() == UVM_ACTIVE) begin
        s0 = instruction_sequencer::type_id::create("s0", this);
        d0 = driver::type_id::create("d0", this);
      end
    m0 = monitor::type_id::create("m0", this);
  endfunction

  virtual function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
      // Connect Sequencer to Driver, if the agent is active
      if (get_is_active() == UVM_ACTIVE) begin
         d0.seq_item_port.connect (s0.seq_item_export);
      end
  endfunction
endclass



class env extends uvm_env;
  `uvm_component_utils(env)

  function new(string name="env", uvm_component parent=null);
    super.new(name, parent);
  endfunction

  agent                 a0;              // Agent handle
  scoreboard    sb0;            // Scoreboard handle

   function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    a0 = agent::type_id::create("a0", this);
    sb0 = scoreboard::type_id::create("sb0", this);
  endfunction

  virtual function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);
    a0.m0.mon_analysis_port.connect(sb0.m_analysis_imp);
  endfunction


        task run_phase (uvm_phase phase);
                set_report_verbosity_level (UVM_MEDIUM);
                uvm_report_info      (get_name(), $sformatf ("Hello␣
                   UVM␣!␣Simulation␣has␣started."), UVM_MEDIUM,
                   `__FILE__, `__LINE__);
                `uvm_info   (get_name(), $sformatf("Finishing␣up␣with␣
                   run_phase␣...␣"), UVM_LOW)
        endtask : run_phase

        function check_all_instruction ();
                return sb0.all_instruction_checked();
        endfunction : check_all_instruction

endclass


`endif

//
//                                        \\\ Sapere Aude ///
//
//
```

```
   ----------------------------------------------------------------------------
// Copyright (c) 2014-2020 All rights reserved
//
   ----------------------------------------------------------------------------
// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_scoreboard.sv
// Create : 2020-09-01 21:56:39
// Revise : 2020-09-06 16:02:49
// Editor : sublime text3, tab size (4)
// Description :
//
   ----------------------------------------------------------------------------



`ifndef __DLX_SCOREBOARD_SV
`define __DLX_SCOREBOARD_SV
`include "../004-implemented_instructions.svh"
`include "./dlx_sequencer.sv"
`include "./dlx_monitor.sv"

import uvm_pkg::*;
`include <uvm_macros.svh>
`include <uvm_pkg.sv>

typedef struct {
        /*instructions_opcode current_opcode;
        instructions_regtype_opcode current_opcode_func;*/
        string pass="no"; //default value for every instruction is no
        int executed_num=0; // how many times the instruction is
           executed
}instruction_info;

class scoreboard extends uvm_scoreboard;
  `uvm_component_utils(scoreboard)

   const int unsigned control_regtype;
   const int unsigned control_immediate;
   const int unsigned control_beqz;
   const int unsigned control_benz;
   const int unsigned control_j;
   const int unsigned control_jal;
   const int unsigned control_sw;
   const int unsigned control_lw;
  function new(string name="scoreboard", uvm_component parent=null);
    super.new(name, parent);
    // init signatures
    control_regtype=32'h7dfb_8000;
        control_immediate=32'h6491_8000;
        control_beqz=32'h65b3_8000;
        control_benz=32'h65f3_8000;
        control_j=32'h7ffb_8000;
        control_jal=32'h7ffb_8000;
        control_sw=32'h75fb_8000;
        control_lw=32'h7dff_8000;
  endfunction


  instruction_info info_array_instr[string]; // associative array
  instruction_item instr_queue[$];
// declaring port
  uvm_analysis_imp #(instruction_item, scoreboard) m_analysis_imp;

   function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    m_analysis_imp = new("m_analysis_imp", this);
  endfunction


  virtual function void write(instruction_item item);
    `uvm_info(get_type_name(), "Stored␣instruction␣item␣in␣scorebard",
        UVM_LOW)
        instr_queue.push_back(item);

  endfunction : write


virtual task run_phase(uvm_phase phase);

        instruction_item item;
        TYPE_OP_ALU_sv check_tmp;
        instructions_opcode tmp;
        instructions_regtype_opcode tmp_opcode;
        forever begin
        wait ( instr_queue.size() >0)

        item = instr_queue.pop_front();
  //called when something is transferred from monitor to scoreboard
    `uvm_info(get_type_name(), $sformatf("Current␣instruction:␣%s",
       item.convert2str()), UVM_LOW)
    // update the number of executed instruction to the related string
//if exists otherwise allocate memory for it
        if(item.get_current_instruction_name()!=="")begin
    if(!info_array_instr.exists(item.get_current_instruction_name()))
       begin
                instruction_info iitmp;
                iitmp.pass="no";
                iitmp.executed_num=0;
        info_array_instr[item.get_current_instruction_name()]= iitmp;
    end
        info_array_instr[item.get_current_instruction_name()].
           executed_num++;


        // check debug signals
        if(item.get_opcode()===i_regtype) begin
                // check the general signals
                if(item.get_signals()!==control_regtype ) begin  //
                   compare with a signature
                        `uvm_error(get_type_name(), $sformatf("FAILED␣
                           instruction:␣%s!",item.convert2str()))
                        info_array_instr[item.
                           get_current_instruction_name()].pass="no";
                end else begin
                        `uvm_info(get_type_name(), $sformatf("PASSED␣
                           instruction:␣%s!",item.convert2str()),
                           UVM_LOW)
                        info_array_instr[item.
                           get_current_instruction_name()].pass="yes"
                           ;
                end

           tmp_opcode= item.get_opcode_func();
           check_tmp=TYPE_OP_ALU_sv'(item.get_alu_op());
                // check the specific alu operation since internal alu
                    op type is different than instruction code optype
                case (tmp_opcode)
                   i_add:
                        begin
                                if(ADD!==check_tmp)begin
                                `uvm_error (get_type_name(),$sformatf(
                                   "FAILED␣instruction:␣%s␣,␣wrong␣
                                   alu␣opcode␣!␣Actual␣:␣%d␣,␣
                                   Excpeted␣:␣%d",item.convert2str(),
                                   check_tmp , ADD))
                                info_array_instr[item.
                                   get_current_instruction_name()].
                                   pass="no";
                                end else begin
                                        `uvm_info(get_type_name(),
                                             $sformatf("PASSED␣
                                            instruction:␣%s␣,␣correct␣
```

```
                              alu␣opcode␣!",item.
                              convert2str()), UVM_LOW)
                          info_array_instr[item.
                              get_current_instruction_name
                              ()].pass="yes";
                  end
          end
      i_and:begin
                  if(BITAND!==check_tmp)begin
                  `uvm_error (get_type_name(),$sformatf(
                      "FAILED␣instruction:␣%s␣,␣wrong␣
                      alu␣opcode␣!␣Actual␣:␣%d␣,␣
                      Excpeted␣:␣%d",item.convert2str(),
                      check_tmp , BITAND))
                  info_array_instr[item.
                      get_current_instruction_name().
                      pass="no";
                  end else begin
                          `uvm_info(get_type_name(),
                              $sformatf("PASSED␣
                              instruction:␣%s␣,␣correct␣
                              alu␣opcode␣!",item.
                              convert2str()), UVM_LOW)
                          info_array_instr[item.
                              get_current_instruction_name
                              ()].pass="yes";
                  end
          end
      i_or:begin
                  if(BITOR!==check_tmp)begin
                  `uvm_error (get_type_name(),$sformatf(
                      "FAILED␣instruction:␣%s␣,␣wrong␣
                      alu␣opcode␣!␣Actual␣:␣%d␣,␣
                      Excpeted␣:␣%d",item.convert2str(),
                      check_tmp , BITOR))
                  info_array_instr[item.
                      get_current_instruction_name().
                      pass="no";
                  end else begin
                          `uvm_info(get_type_name(),
                              $sformatf("PASSED␣
                              instruction:␣%s␣,␣correct␣
                              alu␣opcode␣!",item.
                              convert2str()), UVM_LOW)
                          info_array_instr[item.
                              get_current_instruction_name
                              ()].pass="yes";
                  end
          end
```

```
i_sll:begin
             if(FUNCLSL!==check_tmp)begin
             `uvm_error (get_type_name(),$sformatf(
                "FAILED␣instruction:␣%s␣,␣wrong␣
                alu␣opcode␣!␣Actual␣:␣%d␣,␣
                Excpeted␣:␣%d",item.convert2str(),
                check_tmp , FUNCLSL))
             info_array_instr[item.
                get_current_instruction_name()].
                pass="no";
             end else begin
                     `uvm_info(get_type_name(),
                        $sformatf("PASSED␣
                        instruction:␣%s␣,␣correct␣
                        alu␣opcode␣!",item.
                        convert2str()), UVM_LOW)
                     info_array_instr[item.
                        get_current_instruction_name
                        ()].pass="yes";
             end
     end
i_srl:begin
             if(FUNCLSR!==check_tmp)begin
             `uvm_error (get_type_name(),$sformatf(
                "FAILED␣instruction:␣%s␣,␣wrong␣
                alu␣opcode␣!␣Actual␣:␣%d␣,␣
                Excpeted␣:␣%d",item.convert2str(),
                check_tmp , FUNCLSR))
             info_array_instr[item.
                get_current_instruction_name()].
                pass="no";
             end else begin
                     `uvm_info(get_type_name(),
                        $sformatf("PASSED␣
                        instruction:␣%s␣,␣correct␣
                        alu␣opcode␣!",item.
                        convert2str()), UVM_LOW)
                     info_array_instr[item.
                        get_current_instruction_name
                        ()].pass="yes";
             end
     end
i_sub:begin
             if(SUB!==check_tmp && item.
                get_carry_in()!==1)begin
             `uvm_error (get_type_name(),$sformatf(
                "FAILED␣instruction:␣%s␣,␣wrong␣
                alu␣opcode␣!␣Actual␣:␣%d␣,␣
                Excpeted␣:␣%d",item.convert2str(),
                check_tmp , SUB))
             info_array_instr[item.
                get_current_instruction_name()].
                pass="no";
             end else begin
                     `uvm_info(get_type_name(),
                        $sformatf("PASSED␣
                        instruction:␣%s␣,␣correct␣
                        alu␣opcode␣!",item.
                        convert2str()), UVM_LOW)
                     info_array_instr[item.
                        get_current_instruction_name
                        ()].pass="yes";
             end
     end
 i_xor:begin
             if(BITXOR!==check_tmp)begin
             `uvm_error (get_type_name(),$sformatf(
                "FAILED␣instruction:␣%s␣,␣wrong␣
                alu␣opcode␣!␣Actual␣:␣%d␣,␣
                Excpeted␣:␣%d",item.convert2str(),
                check_tmp , BITXOR))
             info_array_instr[item.
                get_current_instruction_name()].
                pass="no";
             end else begin
                     `uvm_info(get_type_name(),
                        $sformatf("PASSED␣
                        instruction:␣%s␣,␣correct␣
                        alu␣opcode␣!",item.
                        convert2str()), UVM_LOW)
                     info_array_instr[item.
                        get_current_instruction_name
                        ()].pass="yes";
             end
     end
 i_sne:begin
             if(NE!==check_tmp)begin
             `uvm_error (get_type_name(),$sformatf(
                "FAILED␣instruction:␣%s␣,␣wrong␣
                alu␣opcode␣!␣Actual␣:␣%d␣,␣
                Excpeted␣:␣%d",item.convert2str(),
                check_tmp , NE))
             info_array_instr[item.
                get_current_instruction_name()].
                pass="no";
             end else begin
                     `uvm_info(get_type_name(),
                        $sformatf("PASSED␣
                        instruction:␣%s␣,␣correct␣
                        alu␣opcode␣!",item.
                        convert2str()), UVM_LOW)
                    info_array_instr[item.
                        get_current_instruction_name
                        ()].pass="yes";
            end
        end
    i_sle:begin
            if(LE!==check_tmp)begin
            `uvm_error (get_type_name(),$sformatf(
                "FAILED␣instruction:␣%s␣,␣wrong␣
                alu␣opcode␣!␣Actual␣:␣%d␣,␣
                Excpeted␣:␣%d",item.convert2str(),
                check_tmp , LE))
            info_array_instr[item.
                get_current_instruction_name().
                pass="no";
            end else begin
                    `uvm_info(get_type_name(),
                        $sformatf("PASSED␣
                        instruction:␣%s␣,␣correct␣
                        alu␣opcode␣!",item.
                        convert2str()), UVM_LOW)
                    info_array_instr[item.
                        get_current_instruction_name
                        ()].pass="yes";
            end
        end
    i_sge: begin
            if(GE!==check_tmp)begin
            `uvm_error (get_type_name(),$sformatf(
                "FAILED␣instruction:␣%s␣,␣wrong␣
                alu␣opcode␣!␣Actual␣:␣%d␣,␣
                Excpeted␣:␣%d",item.convert2str(),
                check_tmp , GE))
            info_array_instr[item.
                get_current_instruction_name().
                pass="no";
            end else begin
                    `uvm_info(get_type_name(),
                        $sformatf("PASSED␣
                        instruction:␣%s␣,␣correct␣
                        alu␣opcode␣!",item.
                        convert2str()), UVM_LOW)
                    info_array_instr[item.
                        get_current_instruction_name
                        ()].pass="yes";
            end
```

```
                    end
                i_mul: begin
                            if(MULT!==check_tmp)begin
                            `uvm_error (get_type_name(),$sformatf(
                               "FAILED␣instruction:␣%s␣,␣wrong␣
                               alu␣opcode␣!␣Actual␣:␣%d␣,␣
                               Excpeted␣:␣%d",item.convert2str(),
                               check_tmp , MULT))
                            info_array_instr[item.
                               get_current_instruction_name().
                               pass="no";
                            end else begin
                                  `uvm_info(get_type_name(),
                                     $sformatf("PASSED␣
                                     instruction:␣%s␣,␣correct␣
                                     alu␣opcode␣!",item.
                                     convert2str()), UVM_LOW)
                                  info_array_instr[item.
                                     get_current_instruction_name
                                     ()].pass="yes";
                            end
                    end
              endcase



        end else if (item.get_opcode()===i_beqz ) begin

              // check the general signals
              if(item.get_signals()!==control_beqz ) begin  //
                 compare with a signature
                    `uvm_error(get_type_name(), $sformatf("FAILED␣
                       instruction:␣%s!",item.convert2str()))
                    info_array_instr[item.
                       get_current_instruction_name()].pass="no";
              end else begin
                    `uvm_info(get_type_name(), $sformatf("PASSED␣
                       instruction:␣%s!",item.convert2str()),
                       UVM_LOW)
                    info_array_instr[item.
                       get_current_instruction_name()].pass="yes"
                       ;
              end

        end else if( item.get_opcode()===i_benz) begin

                    // check the general signals
              if(item.get_signals()!==control_benz ) begin  //
                 compare with a signature
                    `uvm_error(get_type_name(), $sformatf("FAILED␣
                       instruction:␣%s!",item.convert2str()))
                    info_array_instr[item.
                       get_current_instruction_name()].pass="no";
              end else begin
                    `uvm_info(get_type_name(), $sformatf("PASSED␣
                       instruction:␣%s!",item.convert2str()),
                       UVM_LOW)
                    info_array_instr[item.
                       get_current_instruction_name()].pass="yes"
                       ;
              end

        end else if (item.get_opcode()===i_j) begin
                    // check the general signals
              $display("%h␣%h----", item.get_signals(), control_j);
              if(item.get_signals()!=control_j ) begin  //compare
                 with a signature
                    `uvm_error(get_type_name(), $sformatf("FAILED␣
                       instruction:␣%s!",item.convert2str()))
                    info_array_instr[item.
                       get_current_instruction_name()].pass="no";
              end else begin
                    `uvm_info(get_type_name(), $sformatf("PASSED␣
                       instruction:␣%s!",item.convert2str()),
                       UVM_LOW)
                    info_array_instr[item.
                       get_current_instruction_name()].pass="yes"
                       ;
              end

        end else if(item.get_opcode()===i_jal) begin
        // check the general signals

              if(item.get_signals()!==control_jal) begin  //compare
                 with a signature
                    `uvm_error(get_type_name(), $sformatf("FAILED␣
                       instruction:␣%s!",item.convert2str()))
                    info_array_instr[item.
                       get_current_instruction_name()].pass="no";
              end else begin
                    `uvm_info(get_type_name(), $sformatf("PASSED␣
                       instruction:␣%s!",item.convert2str()),
                       UVM_LOW)
                    info_array_instr[item.
                       get_current_instruction_name()].pass="yes"
                       ;
              end
```

```
end else if (item.get_opcode()===i_sw ) begin
    // check the general signals

            if(item.get_signals()!==control_sw ) begin  //compare
                with a signature
                    `uvm_error(get_type_name(), $sformatf("FAILED␣
                        instruction:␣%s!",item.convert2str()))
                    info_array_instr[item.
                        get_current_instruction_name()].pass="no";
            end else begin
                    `uvm_info(get_type_name(), $sformatf("PASSED␣
                        instruction:␣%s!",item.convert2str()),
                        UVM_LOW)
                    info_array_instr[item.
                        get_current_instruction_name()].pass="yes"
                        ;
            end

    end else if (item.get_opcode()===i_lw) begin
    // check the general signals
            if(item.get_signals()!== control_lw ) begin  //compare
                 with a signature
                    `uvm_error(get_type_name(), $sformatf("FAILED␣
                        instruction:␣%s!",item.convert2str()))
                    info_array_instr[item.
                        get_current_instruction_name()].pass="no";
            end else begin
                    `uvm_info(get_type_name(), $sformatf("PASSED␣
                        instruction:␣%s!",item.convert2str()),
                        UVM_LOW)
                    info_array_instr[item.
                        get_current_instruction_name()].pass="yes"
                        ;
            end

    end else begin  // immediate
    // check the general signals
            tmp= item.get_opcode();
            if(item.get_signals()!==control_immediate && tmp!==
                i_nop) begin  //compare with a signature
                    `uvm_error(get_type_name(), $sformatf("FAILED␣
                        instruction:␣%s!",item.convert2str()))
                    info_array_instr[item.
                        get_current_instruction_name()].pass="no";
            end else begin
                    `uvm_info(get_type_name(), $sformatf("PASSED␣
                        instruction:␣%s!",item.convert2str()),
                        UVM_LOW)
                    info_array_instr[item.
                        get_current_instruction_name()].pass="yes"
                        ;
            end


            check_tmp= TYPE_OP_ALU_sv'(item.get_alu_op());
            // check the specific alu operation
            case (tmp)
                    i_addi : begin

            if(ADD!==check_tmp)begin
                        `uvm_error (get_type_name(),$sformatf(
                            "FAILED␣instruction:␣%s␣,␣wrong␣
                            alu␣opcode␣addi␣!␣Actual␣:␣%d␣,␣
                            Excpeted␣:␣%d",item.convert2str(),
                            check_tmp , ADD))
                        info_array_instr[item.
                            get_current_instruction_name()].
                            pass="no";
                        end else begin
                                `uvm_info(get_type_name(),
                                    $sformatf("PASSED␣
                                    instruction:␣%s␣,␣correct␣
                                    alu␣opcode␣addi!",item.
                                    convert2str()), UVM_LOW)
                                info_array_instr[item.
                                    get_current_instruction_name
                                    ()].pass="yes";
                        end
                    end
                    i_andi : begin
            if(BITAND!==check_tmp)begin
                        `uvm_error (get_type_name(),$sformatf(
                            "FAILED␣instruction:␣%s␣,␣wrong␣
                            alu␣opcode␣subi␣!␣Actual␣:␣%d␣,␣
                            Excpeted␣:␣%d",item.convert2str(),
                            check_tmp , BITAND))
                        info_array_instr[item.
                            get_current_instruction_name()].
                            pass="no";
                        end else begin
                                `uvm_info(get_type_name(),
                                    $sformatf("PASSED␣
                                    instruction:␣%s␣,␣correct␣
                                    alu␣opcode␣subi!",item.
                                    convert2str()), UVM_LOW)
                                info_array_instr[item.
                                    get_current_instruction_name
                                    ()].pass="yes";
                    end
                end
                i_nop : begin
        if(ADD!==check_tmp)begin
                    `uvm_error (get_type_name(),$sformatf(
                       "FAILED␣instruction:␣%s␣,␣wrong␣
                       alu␣opcode␣nop␣!␣Actual␣:␣%d␣,␣
                       Excpeted␣:␣%d",item.convert2str(),
                       check_tmp , ADD))
                    info_array_instr[item.
                       get_current_instruction_name()].
                       pass="no";
                    end else begin
                        `uvm_info(get_type_name(),
                           $sformatf("PASSED␣
                           instruction:␣%s␣,␣correct␣
                           alu␣opcode␣nop!",item.
                           convert2str()), UVM_LOW)
                        info_array_instr[item.
                           get_current_instruction_name
                           ()].pass="yes";
                    end
                end
                i_sgei : begin
        if(GE!==check_tmp)begin
                    `uvm_error (get_type_name(),$sformatf(
                       "FAILED␣instruction:␣%s␣,␣wrong␣
                       alu␣opcode␣SGEI␣!␣Actual␣:␣%d␣,␣
                       Excpeted␣:␣%d",item.convert2str(),
                       check_tmp , GE))
                    info_array_instr[item.
                       get_current_instruction_name()].
                       pass="no";
                    end else begin
                        `uvm_info(get_type_name(),
                           $sformatf("PASSED␣
                           instruction:␣%s␣,␣correct␣
                           alu␣opcode␣SGEI!",item.
                           convert2str()), UVM_LOW)
                        info_array_instr[item.
                           get_current_instruction_name
                           ()].pass="yes";
                    end
                end
                i_slei : begin
        if(LE!==check_tmp)begin
                    `uvm_error (get_type_name(),$sformatf(
                       "FAILED␣instruction:␣%s␣,␣wrong␣
                       alu␣opcode␣SLEI␣!␣Actual␣:␣%d␣,␣
```

```
                            Excpeted␣:␣%d",item.convert2str(),
                            check_tmp , LE))
                        info_array_instr[item.
                            get_current_instruction_name()].
                            pass="no";
                        end else begin
                                'uvm_info(get_type_name(),
                                    $sformatf("PASSED␣
                                    instruction:␣%s␣,␣correct␣
                                    alu␣opcode␣SLEI!",item.
                                    convert2str()), UVM_LOW)
                                info_array_instr[item.
                                    get_current_instruction_name
                                    ()].pass="yes";
                        end
                end
                i_slli : begin
        if(FUNCLSL!==check_tmp)begin
                        'uvm_error (get_type_name(),$sformatf(
                            "FAILED␣instruction:␣%s␣,␣wrong␣
                            alu␣opcode␣SLLI␣!␣Actual␣:␣%d␣,␣
                            Excpeted␣:␣%d",item.convert2str(),
                            check_tmp , FUNCLSL))
                        info_array_instr[item.
                            get_current_instruction_name()].
                            pass="no";
                        end else begin
                                'uvm_info(get_type_name(),
                                    $sformatf("PASSED␣
                                    instruction:␣%s␣,␣correct␣
                                    alu␣opcode␣SLLI!",item.
                                    convert2str()), UVM_LOW)
                                info_array_instr[item.
                                    get_current_instruction_name
                                    ()].pass="yes";
                        end
                end
                i_snei : begin
        if(NE!==check_tmp)begin
                        'uvm_error (get_type_name(),$sformatf(
                            "FAILED␣instruction:␣%s␣,␣wrong␣
                            alu␣opcode␣␣SNEI␣!␣Actual␣:␣%d␣,␣
                            Excpeted␣:␣%d",item.convert2str(),
                            check_tmp , NE))
                        info_array_instr[item.
                            get_current_instruction_name()].
                            pass="no";
                        end else begin
                                'uvm_info(get_type_name(),
                                        $sformatf("PASSED␣
                                        instruction:␣%s␣,␣correct␣
                                        alu␣opcode␣SNEI␣!",item.
                                        convert2str()), UVM_LOW)
                                    info_array_instr[item.
                                        get_current_instruction_name
                                        ()].pass="yes";
                          end
                end
                i_srli : begin
      if(FUNCLSR!==check_tmp)begin
                          `uvm_error (get_type_name(),$sformatf(
                              "FAILED␣instruction:␣%s␣,␣wrong␣
                              alu␣opcode␣SRLI␣!␣Actual␣:␣%d␣,␣
                              Excpeted␣:␣%d",item.convert2str(),
                              check_tmp , FUNCLSR))
                          info_array_instr[item.
                              get_current_instruction_name().
                              pass="no";
                          end else begin
                                    `uvm_info(get_type_name(),
                                        $sformatf("PASSED␣
                                        instruction:␣%s␣,␣correct␣
                                        alu␣opcode␣SRLI!",item.
                                        convert2str()), UVM_LOW)
                                    info_array_instr[item.
                                        get_current_instruction_name
                                        ()].pass="yes";
                          end
                end
                i_subi : begin
      if(SUB!==check_tmp && item.get_carry_in()!==1)begin
                          `uvm_error (get_type_name(),$sformatf(
                              "FAILED␣instruction:␣%s␣,␣wrong␣
                              alu␣opcode␣SUBI␣!␣Actual␣:␣%d␣,␣
                              Excpeted␣:␣%d",item.convert2str(),
                              check_tmp , SUB))
                          info_array_instr[item.
                              get_current_instruction_name().
                              pass="no";
                          end else begin
                                    `uvm_info(get_type_name(),
                                        $sformatf("PASSED␣
                                        instruction:␣%s␣,␣correct␣
                                        alu␣opcode␣SUBI!",item.
                                        convert2str()), UVM_LOW)
                                    info_array_instr[item.
                                        get_current_instruction_name
                                        ()].pass="yes";
                              end
                      end
                      i_ori : begin
              if(BITOR!==check_tmp)begin
                              `uvm_error (get_type_name(),$sformatf(
                                 "FAILED␣instruction:␣%s␣,␣wrong␣
                                 alu␣opcode␣␣ORI␣!␣Actual␣:␣%d␣,␣
                                 Excpeted␣:␣%d",item.convert2str(),
                                 check_tmp , BITOR))
                              info_array_instr[item.
                                 get_current_instruction_name()].
                                 pass="no";
                              end else begin
                                      `uvm_info(get_type_name(),
                                         $sformatf("PASSED␣
                                         instruction:␣%s␣,␣correct␣
                                         alu␣opcode␣ORI␣!",item.
                                         convert2str()), UVM_LOW)
                                      info_array_instr[item.
                                         get_current_instruction_name
                                         ()].pass="yes";
                              end
                      end
                      i_xori : begin
              if(BITXOR!==check_tmp)begin
                              `uvm_error (get_type_name(),$sformatf(
                                 "FAILED␣instruction:␣%s␣,␣wrong␣
                                 alu␣opcode␣XORI␣!␣Actual␣:␣%d␣,␣
                                 Excpeted␣:␣%d",item.convert2str(),
                                 check_tmp , BITXOR))
                              info_array_instr[item.
                                 get_current_instruction_name()].
                                 pass="no";
                              end else begin
                                      `uvm_info(get_type_name(),
                                         $sformatf("PASSED␣
                                         instruction:␣%s␣,␣correct␣
                                         alu␣opcode␣XORI!",item.
                                         convert2str()), UVM_LOW)
                                      info_array_instr[item.
                                         get_current_instruction_name
                                         ()].pass="yes";
                              end
                      end
              endcase
      end
      end
      end
      endtask : run_phase
```

```
        virtual function void report_phase(uvm_phase phase);
        super.report_phase(phase);
        `uvm_info(get_type_name(), "Verbose␣prinf␣of␣instructions␣
           scoreboard",UVM_LOW)
        foreach(info_array_instr[key_in])begin
        `uvm_info(get_type_name(), $sformatf("instruction␣%s,␣pass:%s␣
           ,␣executed:␣%d␣times",key_in,info_array_instr[key_in].pass
           ,info_array_instr[key_in].executed_num),UVM_LOW)
        end

        endfunction : report_phase


function integer all_instruction_checked ();
        int check=1;
        string i;

        if(info_array_instr.num()<=0) begin
        return 0;
        end

        if(! info_array_instr.first(i)) begin
                `uvm_error (get_type_name(),"ERROR␣instruction␣info␣
                   array␣is␣empty")
        end
        for (; !info_array_instr.exists(i);) begin
                if(info_array_instr[i].executed_num<=0) begin
                        check=0;
                end
                 if(!info_array_instr.next(i)) begin
                        `uvm_error( get_type_name(), $sformatf("error␣
                           instruction␣info␣array␣not␣found␣for␣%s",i
                           ))
                 end
        end
        return check;
endfunction : all_instruction_checked

endclass


`endif

//
//           \\\ Sapere Aude ///
//
//
   ----------------------------------------------------------------------------
```

```
// Copyright (c) 2014-2020 All rights reserved
//
   -----------------------------------------------------------------------------

// Author : Angione Francesco s262620@studenti.polito.it
   franout@Github.com
// File   : dlx_test.sv
// Create : 2020-09-01 21:55:38
// Revise : 2020-09-01 21:55:38
// Editor : sublime text3, tab size (2)
// Description :
//
   -----------------------------------------------------------------------------



`ifndef __DLX_TEST_SV
`define __DLX_TEST_SV
`include "../003-global_defs.svh"
`include "../memories/005-memory_interfaces.svh"
`include "dlx_env.sv"
`include "dlx_sequence.sv"

import uvm_pkg::*;
`include <uvm_macros.svh>
`include <uvm_pkg.sv>

class test extends uvm_test;
  `uvm_component_utils(test)

  function new(string name = "test␣dlx", uvm_component parent=null);
    super.new(name, parent);
  endfunction

  env e0;
  instruction_sequence seq;
  virtual DEBUG_interface dbg_if;
  virtual       mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE))  iram_if;
  virtual       mem_interface #(.ADDRESS_SIZE(`DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE))  dram_if;

  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    e0 = env::type_id::create("e0", this);
    seq = instruction_sequence::type_id::create("seq", this);
    if (!uvm_config_db#(virtual DEBUG_interface)::get(null, "", "
       dbg_if", dbg_if))
      `uvm_fatal("TEST", "Did␣not␣get␣dbg_if")
      //uvm_config_db#(virtual DEBUG_interface)::set(this, "e0.a0.*",
         "dbg_if", dbg_if);

      if (!uvm_config_db#(virtual       mem_interface #(.ADDRESS_SIZE(
         `DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)))::get(null, "", "
                           iram_if", iram_if))
      `uvm_fatal("TEST", "Did␣not␣get␣iram")

     // uvm_config_db#(virtual  mem_interface #(.ADDRESS_SIZE(
        `DRAM_ADDRESS_SIZE),.WORD_SIZE(`DRAM_WORD_SIZE)))::set(this,
        "e0.a0.*", "iram_if", iram_if);

      if (!uvm_config_db#(virtual       mem_interface #(.ADDRESS_SIZE(
         `DRAM_ADDRESS_SIZE),
                        .WORD_SIZE(`DRAM_WORD_SIZE)))::get(null, "", "
                           dram_if", dram_if))
      `uvm_fatal("TEST", "Did␣not␣get␣dram")

    //  uvm_config_db#(virtual  mem_interface #(.ADDRESS_SIZE(
       `DRAM_ADDRESS_SIZE),.WORD_SIZE(`DRAM_WORD_SIZE)))::set(this, "
       e0.a0.*", "dram_if", dram_if);

  endfunction

  virtual task run_phase(uvm_phase phase);
    instruction_sequence s0 = instruction_sequence::type_id::create("
       s0");
    phase.raise_objection(this);
    apply_reset();
    `uvm_info(get_name(),"Done␣reset␣of␣DUT_DUV", UVM_LOW)
         seq.start(e0.a0.s0);
                seq.randomize();
       #1025;// 1 cc for each instruction ( it generates 30
          instruction )
    $display("UVM␣test␣is␣ended␣@␣%d",$time());

       // we may add a
/*

       forever begin
                wait (e0.check_all_instruction)
                if(e0.check_all_instruction()
                        break();
                else
                        continue
*/
    phase.drop_objection(this);
  endtask

  virtual task apply_reset();
    dbg_if.rst <= 0;
    iram_if.rst <= 0;
    dram_if.rst <= 0;
    repeat(10) @ (posedge iram_if.clk);
    dbg_if.rst <= 1;
    iram_if.rst <= 1;
    dram_if.rst <= 1;
    repeat(2) @ (posedge iram_if.clk);
  endtask

// print topology of environment at the end of elaboration phase
 virtual function void end_of_elaboration_phase (uvm_phase phase);
               print();
  endfunction


        virtual function void report_phase (uvm_phase phase);
                uvm_report_server svr;
            super.report_phase(phase);
        svr = uvm_report_server::get_server();
                if(svr.get_severity_count(UVM_FATAL)+svr.
                   get_severity_count(UVM_ERROR)>0) begin
                `uvm_info(get_type_name(), "
                   ---------------------------------------", UVM_NONE
                   )
                    `uvm_info(get_type_name(), "----␣␣␣␣␣␣␣␣␣␣␣␣TEST␣
                       FAIL␣␣␣␣␣␣␣␣␣␣␣----", UVM_NONE)
                `uvm_info(get_type_name(), "
                   ---------------------------------------", UVM_NONE
                   )
        end
                else begin
                    `uvm_info(get_type_name(), "
                       ---------------------------------------",
                       UVM_NONE)
                    `uvm_info(get_type_name(), "----␣␣␣␣␣␣␣␣␣␣␣TEST␣
                       PASS␣␣␣␣␣␣␣␣␣␣␣␣----", UVM_NONE)
                    `uvm_info(get_type_name(), "
                       ---------------------------------------",
                       UVM_NONE)
        end
        endfunction : report_phase

endclass

`endif
```

APPENDIX H

Top level entity of DLX

```
--
   -----------------------------------------------------------------------------
-- Title        : DLX top
-- Project      : DLX for Microelectronic Systems
--
   -----------------------------------------------------------------------------
-- File         : a-DLX.vhd
-- Author       : Francesco Angione <s262620@studenti.polito.it>
   franout@github.com
-- Company      : Politecnico di Torino, Italy
-- Created      : Wed Jul 22 22:58:15 2020
-- Last update  : Sat Sep  5 22:31:23 2020
-- Platform     : Default Part Number
-- Standard     : VHDL-2008
--
   -----------------------------------------------------------------------------
-- Copyright (c) 2020 Politecnico di Torino, Italy
--
   ----------------------------------------------------------------------------
-- Description: top level entity of dlx without memories
--
   -----------------------------------------------------------------------------


library IEEE;
use IEEE.std_logic_1164.all;

use work.globals.all;
use work.constants.all;
```

```
entity DLX is
  generic (
    IR_SIZE : integer := 32; -- Instruction Register Size
    PC_SIZE : integer := 32  -- Program Counter Size
  );
  port (
    -- Inputs
    CLK : in std_logic; -- Clock
    RST : in std_logic; -- Reset:Active-low
                        -- Instruction memory interface
    IRAM_ADDRESS : out std_logic_vector( iram_address_size- 1 downto
       0);
    IRAM_ENABLE  : out std_logic;
    IRAM_READY   : in  std_logic;
    IRAM_DATA    : in  std_logic_vector(instr_length-1 downto 0);
    -- Data memory Interface
    DRAM_ADDRESS       : out   std_logic_vector(dram_address_size-1
       downto 0);
    DRAM_ENABLE        : out   std_logic;
    DRAM_READNOTWRITE  : out   std_logic;
    DRAM_READY         : in    std_logic;
    DRAM_DATA          : inout std_logic_vector(data_size-1 downto 0)
    --
       --------------------------------------------------------------------------

    -- cpu status signals in case of exception or hang
       --------------------------
    --
       --------------------------------------------------------------------------

    -- simulation debug signals and verification purposes
    --synopsys translate_off
    ;
    STATE_CU : out std_logic_vector(f_log2(tot_state)-1 downto 0);
    csr      : out std_logic_vector(7 downto 0);
    -- all the control unit signals
    -- used for system verilog verification
    DEBUG_iram_ready_cu     : out std_logic;
    DEBUG_iram_enable_cu    : out std_logic;
    DEBUG_signed_notsigned  : out std_logic;
    DEBUG_compute_sext      : out std_logic;
    DEBUG_jump_sext         : out std_logic;
    DEBUG_write_rf          : out std_logic;
    DEBUG_evaluate_branch   : out std_logic_vector(1 downto 0);
    DEBUG_alu_cin           : out std_logic;
    DEBUG_alu_overflow      : out std_logic;
    DEBUG_zero_mul_detect   : out std_logic;
    DEBUG_mul_exeception    : out std_logic;
    DEBUG_dram_ready_cu     : out std_logic;
    DEBUG_dram_r_nw_cu      : out std_logic;
    DEBUG_enable_rf         : out std_logic;
    DEBUG_read_rf_p1        : out std_logic;
    DEBUG_read_rf_p2        : out std_logic;
    DEBUG_rtype_itypen      : out std_logic;
    DEBUG_update_pc_branch  : out std_logic;
    DEBUG_sel_val_a         : out std_logic_vector(0 downto 0);
    DEBUG_sel_val_b         : out std_logic_vector(0 downto 0);
    DEBUG_select_wb         : out std_logic_vector(0 downto 0);
    DEBUG_alu_op_type       : out std_logic_vector(3 downto 0);
    DEBUG_dram_enable_cu    : out std_logic
  --synopsys translate_on
  );
end DLX;


architecture dlx_rtl of DLX is

  -- control unit
  component control_unit is
    generic (
      PC_SIZE : integer := 32;
      RF_REGS : integer := 32; -- number of register in register file
      IR_SIZE : integer := 32; -- Instruction Register Size
      CW_SIZE : integer := 15  -- Control Word Size
    );
    port (
      clk : in std_logic;
      rst : in std_logic; -- active low
                          -- for fetch stage
      iram_enable_cu          : out std_logic;
      iram_ready_cu           : in  std_logic;
      curr_instruction_to_cu  : in  std_logic_vector(IR_SIZE-1 downto
         0);
      stall_pip               : out std_logic;
      -- for decode stage
      enable_rf   : out std_logic; -- used as enable for sign26
         extention when equal to 0
      read_rf_p1  : out std_logic; -- if read_rf_p1/2 are both zero it
         is a jal instruction write address -> 31
      read_rf_p2  : out std_logic;

      rtype_itypen : out std_logic;
      compute_sext : out std_logic;
      jump_sext    : out std_logic;
      -- for execute stage
      alu_op_type      : out std_logic_vector(3 downto 0); --
```

```
         TYPE_OP_ALU ; for compatibility with sv
      sel_val_a         : out std_logic_vector(0 downto 0 );
      sel_val_b         : out std_logic_vector(0 downto 0 );
      signed_notsigned : out std_logic;
      alu_cin           : out std_logic;
      evaluate_branch   : out std_logic_vector(1 downto 0); -- msb for
         evaluate branch negated(!=0) lsb for evaluate branch (==0)
                                                            -- from
                                                              execute
                                                               stage
      alu_overflow : in std_logic;
      -- exception control logic for multiplication
      zero_mul_detect : in std_logic;
      mul_exeception  : in std_logic;
      -- for memory stage
      dram_enable_cu    : out std_logic;
      dram_r_nw_cu      : out std_logic;
      dram_ready_cu     : in  std_logic;
      update_pc_branch : out std_logic;
      -- for write back stage
      write_rf   : out std_logic;
      select_wb : out std_logic_vector(0 downto 0)
      -- simulation debug signals
      --synopsys translate_off
      ;
      STATE_CU : out std_logic_vector(f_log2(tot_state)-1 downto 0);
      csr       : out std_logic_vector(7 downto 0)
    --synopsys translate_on
    );
  end component control_unit;

  -- Datapath
  component DATAPATH is
    generic (
      N        : integer := 32;
      RF_REGS : integer := 32; -- number of registers in register file
          component
      IR_SIZE : integer := 32; -- Instruction register size
      PC_SIZE : integer := 32  -- Program Counter Size
    );
    port (
      clk : in std_logic;
      rst : in std_logic;
      -- iram interface
      IRAM_ADDRESS : out std_logic_vector( iram_address_size- 1 downto
          0);
      IRAM_ENABLE  : out std_logic;
      IRAM_READY   : in  std_logic;
      IRAM_DATA    : in  std_logic_vector(IR_SIZE-1 downto 0);
      -- dram interface
      DRAM_ADDRESS       : out   std_logic_vector(dram_address_size-1
         downto 0);
      DRAM_ENABLE        : out   std_logic;
      DRAM_READNOTWRITE  : out   std_logic;
      DRAM_READY         : in    std_logic;
      DRAM_DATA          : inout std_logic_vector(data_size-1 downto 0)
         ;
      -- control unit interface, signals from/to control unit
      -- for fetch stage
      iram_enable_cu          : in  std_logic;
      iram_ready_cu           : out std_logic;
      stall                   : in  std_logic;
      curr_instruction_to_cu  : out std_logic_vector(PC_SIZE-1 downto
         0);
      -- for decode stage
      enable_rf     : in std_logic;
      read_rf_p1    : in std_logic;
      read_rf_p2    : in std_logic;
      write_rf      : in std_logic;
      rtype_itypen  : in std_logic; -- =='1' rtype instrucion =='0'
         itype instructnions
      jump_sext     : in std_logic;
      compute_sext  : in std_logic;
      -- for execute stage
      alu_op_type       : in std_logic_vector(3 downto 0); --
         TYPE_OP_ALU ; for compatibility with sv
      sel_val_a         : in std_logic_vector(0 downto 0 );
      sel_val_b         : in std_logic_vector(0 downto 0 );
      alu_cin           : in std_logic;
      signed_notsigned  : in std_logic;
      evaluate_branch   : in std_logic_vector(1 downto 0);
      -- from execute stage
      alu_overflow : out std_logic;
      -- exception control logic for multiplication
      zero_mul_detect  : out std_logic;
      mul_exeception   : out std_logic;
      -- for memory stage
      dram_enable_cu    : in  std_logic;
      dram_r_nw_cu      : in  std_logic;
      dram_ready_cu     : out std_logic;
      update_pc_branch  : in  std_logic;
      -- for write back stage
      select_wb : in std_logic_vector(0 downto 0)

    );
  end component DATAPATH;
  -----------------------------------------------------------------
  -- Internconnection Signals Declaration
  -----------------------------------------------------------------
  signal iram_ready_cu_i,iram_enable_cu_i,signed_notsigned_i,
  compute_sext_i,write_rf_i,alu_cin_i,jump_sext_i,update_pc_branch_i,
  alu_overflow_i,zero_mul_detect_i,mul_exeception_i,stall_i,
  dram_ready_cu_i,dram_r_nw_cu_i,enable_rf_i,read_rf_p1_i,read_rf_p2_i
     ,rtype_itypen_i,
  dram_enable_cu_i                              : std_logic;
  signal evaluate_branch_i                      : std_logic_vector(1
     downto 0);
  signal curr_instruction_to_cu_i               : std_logic_vector(
     PC_SIZE-1 downto 0);
  signal sel_val_a_i,sel_val_b_i ,select_wb_i : std_logic_vector(0
     downto 0);
  signal alu_op_type_i                          : std_logic_vector(3
     downto 0); --TYPE_OP_ALU ; for compatibility with sv




begin -- DLX

  -- Control Unit Instantiation
  cu_i : control_unit
    generic map (
      PC_SIZE => PC_SIZE,
      RF_REGS => register_in_rf,
      IR_SIZE => instr_length,
      CW_SIZE => tot_cu_sign
    )
    port map (
      clk                     => clk,
      rst                     => rst,
      iram_enable_cu          => iram_enable_cu_i,
      iram_ready_cu           => iram_ready_cu_i,
      curr_instruction_to_cu => curr_instruction_to_cu_i,
      stall_pip               => stall_i,
      -- for decode stage
      enable_rf     => enable_rf_i,
      read_rf_p1    => read_rf_p1_i,
      read_rf_p2    => read_rf_p2_i,
      write_rf      => write_rf_i,
      rtype_itypen => rtype_itypen_i,
      jump_sext     => jump_sext_i,
      compute_sext => compute_sext_i,
      -- for execute stage
      alu_op_type       => alu_op_type_i ,
      sel_val_a         => sel_val_a_i ,
      sel_val_b          => sel_val_b_i ,
      evaluate_branch    => evaluate_branch_i,
      signed_notsigned => signed_notsigned_i,
      -- from execute stage
      alu_cin          => alu_cin_i,
      alu_overflow     => alu_overflow_i,
      zero_mul_detect => zero_mul_detect_i,
      mul_exeception   => mul_exeception_i,
      -- for memory stage
      dram_enable_cu     => dram_enable_cu_i,
      dram_r_nw_cu       => dram_r_nw_cu_i,
      dram_ready_cu      => dram_ready_cu_i,
      update_pc_branch => update_pc_branch_i,
      -- for write back stage
      select_wb => select_wb_i
      -- simulation debug signals
      --synopsys translate_off
      ,
      STATE_CU => STATE_CU,
      csr      => csr
      --synopsys translate_on

    );

    -- Datapath instantiation
    datapath_i : DATAPATH generic map (
      N        => data_size,
      RF_REGS => register_in_rf, -- number of registers in register
         file component
      IR_SIZE => IR_SIZE,          -- Instruction register size
      PC_SIZE => PC_SIZE           -- Program Counter Size
    )
    port map (
      clk => clk,
      rst => rst,
      -- iram interface
      IRAM_ADDRESS  => IRAM_ADDRESS,
      IRAM_ENABLE   => IRAM_ENABLE ,
      IRAM_READY    => IRAM_READY ,
      IRAM_DATA     => IRAM_DATA ,
      -- dram interface
      DRAM_ADDRESS        => DRAM_ADDRESS ,
      DRAM_ENABLE         => DRAM_ENABLE ,
      DRAM_READNOTWRITE => DRAM_READNOTWRITE ,
      DRAM_READY          => DRAM_READY ,
      DRAM_DATA           => DRAM_DATA ,
      -- control unit interface, signals from/to control unit
      -- for fetch stage
      iram_enable_cu           => iram_enable_cu_i,
      iram_ready_cu           => iram_ready_cu_i,
      stall                   => stall_i,
      curr_instruction_to_cu => curr_instruction_to_cu_i,
      -- for decode stage
      enable_rf     => enable_rf_i,
      read_rf_p1    => read_rf_p1_i,
      read_rf_p2    => read_rf_p2_i,
      write_rf      => write_rf_i,
      jump_sext     => jump_sext_i,
      rtype_itypen => rtype_itypen_i,
      compute_sext => compute_sext_i,
      -- for execute stage
      alu_op_type       => alu_op_type_i ,
      sel_val_a         => sel_val_a_i ,
      sel_val_b         => sel_val_b_i ,
      evaluate_branch   => evaluate_branch_i,
      signed_notsigned => signed_notsigned_i,
      -- from execute stage
      alu_cin          => alu_cin_i,
      alu_overflow     => alu_overflow_i,
      zero_mul_detect => zero_mul_detect_i,
      mul_exeception  => mul_exeception_i,
      -- for memory stage
      dram_enable_cu    => dram_enable_cu_i,
      dram_r_nw_cu      => dram_r_nw_cu_i,
      dram_ready_cu     => dram_ready_cu_i,
      update_pc_branch => update_pc_branch_i,
      -- for write back stage
      select_wb => select_wb_i
    );

  --synopsys translate_off
  DEBUG_iram_ready_cu     <= iram_ready_cu_i;
  DEBUG_iram_enable_cu    <= iram_enable_cu_i;
  DEBUG_signed_notsigned <= signed_notsigned_i;
  DEBUG_compute_sext      <= compute_sext_i;
  DEBUG_jump_sext         <= jump_sext_i;
  DEBUG_write_rf          <= write_rf_i;
  DEBUG_evaluate_branch   <= evaluate_branch_i;
  DEBUG_alu_cin           <= alu_cin_i;
  DEBUG_alu_overflow      <= alu_overflow_i;
  DEBUG_zero_mul_detect   <= zero_mul_detect_i;
  DEBUG_mul_exeception    <= mul_exeception_i;
  DEBUG_dram_ready_cu     <= dram_ready_cu_i;
  DEBUG_dram_r_nw_cu      <= dram_r_nw_cu_i;
  DEBUG_enable_rf         <= enable_rf_i;
  DEBUG_read_rf_p1        <= read_rf_p1_i;
  DEBUG_read_rf_p2        <= read_rf_p2_i;
  DEBUG_rtype_itypen      <= rtype_itypen_i;
```

```
  DEBUG_dram_enable_cu    <= dram_enable_cu_i;
  DEBUG_update_pc_branch <= update_pc_branch_i;
  DEBUG_sel_val_a         <= sel_val_a_i ;
  DEBUG_sel_val_b         <= sel_val_b_i ;
  DEBUG_select_wb         <= select_wb_i;
  DEBUG_alu_op_type       <= alu_op_type_i; --TYPE_OP_ALU ; for
     compatibility with sv

  --synopsys translate_on

end dlx_rtl;
```

APPENDIX I

Globals package

```
--
   ----------------------------------------------------------------------------

-- Title        : globals
-- Project      : DLX for Microelectronic Systems
--
   ----------------------------------------------------------------------------

-- File         : 000-globals.vhd
-- Author       : Francesco Angione <s262620@studenti.polito.it>
   franout@github.com
-- Company      : Politecnico di Torino, Italy
-- Created      : Wed Jul 22 22:56:54 2020
-- Last update : Sun Aug 30 22:58:17 2020
-- Platform     : Default Part Number
-- Standard     : VHDL-2008
--
   ----------------------------------------------------------------------------

-- Copyright (c) 2020 Politecnico di Torino, Italy
--
   ---------------------------------------------------------------------------

-- Description: global constants, functions and  definitions for the
   dlx
--
   ----------------------------------------------------------------------------



library ieee;
use ieee.std_logic_1164.all;
use work.constants.all; -- it contains f_log2 function
```

```
package globals is

    -- function for transforming from big endian to little endian
    function b2l_endian (a : in std_logic_vector)
        return std_logic_vector;

    -- function for transforming from little endian to big endian
    function l2b_endian(a : in std_logic_vector)
        return std_logic_vector;

    -- function for calculating the log2 of an integer
    --function f_log2 ( x: integer ) return integer; it is in the
       constants package from the labs

    --
       -----------------------------------------------------------------------------------

    -- for debug purpuses
    --synthesis_translate off
    constant tot_state : integer := 3;
    --synthesis_translate on
    --
       -----------------------------------------------------------------------------------


    -- definition for instruction
    constant endianess                     : string  := "big"; -- for
       memory access
    constant instr_length                  : integer := 32;    --
       number of bits for an instruction
    constant opcode_length                 : integer := 6;     --
       length in the instruction
    constant register_address_field_length : integer := 5;     --
       length in the instruction
    constant immediate_length              : integer := 16;    --  for
        I-type instructions
    constant alu_function_length           : integer := 11;    -- for
       R-type instructions
    constant jump_address_length           : integer := 26;    -- for
       J-type instruction
    constant OP_CODE_SIZE                  : integer := 6;     --
       OPCODE field size
    constant FUNC_SIZE                     : integer := 11;    -- FUNC
        field size
    constant tot_cu_sign                   : integer := 22;    --
       number of total signoal (I/O) of control unit

    -- definition for data
    constant data_size : integer := 32;
```

```
-- definition for register file
constant register_in_rf : integer := 32;

-- definition for memories size
--      constant dram_size         :  integer  := 2**32-1;
constant dram_address_size : integer := 32; --f_log2(dram_size);
                                            --      constant
                                               iram_size
                                               : integer  :=
                                               2**32-1;
constant iram_address_size : integer := 32; --f_log2(iram_size);

-- R-TYPE -> register to register operation
-- I-TYPE -> register and an immediate of ALU operation or load/
   store memory operatins
-- from lab defined in alu_types.vhd package
type TYPE_OP_ALU is (ADD, SUB, MULT, BITAND, BITOR, BITXOR,
   FUNCLSL, FUNCLSR,GE,LE,NE);

-- see also implemented_instruction.svh in ./test_bench
-- removed as attribute encoding for being compliant with
   synthesis and remove warning
--type instruction is (
--           i_regtype,i_addi ,i_andi ,i_beqz ,i_benz ,i_j ,i_jal
    ,i_lw ,i_nop
--          ,i_ori ,i_sgei ,i_slei ,i_slli ,i_snei ,
--           i_srli ,i_subi ,i_sw ,i_xori
--
--    );
--type ireg_instruction_funcode is(i_add,i_mul,i_and ,i_or,i_sge ,
   i_sle,i_sll ,i_sne,i_srl,i_sub,i_xor);




-- for being complaint with synthesis
constant i_regtype : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
   "00"&x"0";
constant i_addi    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
   "00"&x"8";
constant i_andi    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
   "00"&x"c";
constant i_beqz    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
   "00"&x"4";
constant i_benz    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
   "00"&x"5";
constant i_j       : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
   "00"&x"2";
constant i_jal     : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "00"&x"3";
    constant i_lw      : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"3";
    constant i_nop     : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "01"&x"5";
    constant i_ori     : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "00"&x"d";
    constant i_sgei    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "01"&x"d";
    constant i_slei    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "01"&x"c";
    constant i_slli    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "01"&x"4";
    constant i_snei    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "01"&x"9";
    constant i_srli    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "01"&x"6";
    constant i_subi    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "00"&x"a";
    constant i_sw      : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"b";
    constant i_xori    : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "00"&x"e";

    --- function opcode definition
    constant i_add : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"0";
    constant i_mul : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "11"&x"f";
    constant i_and : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"4";
    constant i_or  : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"5";
    constant i_sge : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"d";
    constant i_sle : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"c";
    constant i_sll : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "00"&x"4";
    constant i_sne : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"9";
    constant i_srl : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "00"&x"6";
    constant i_sub : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"2";
    constant i_xor : std_logic_vector(OP_CODE_SIZE-1 downto 0) :=
        "10"&x"6";

end globals;
```

```
package body globals is



    -- function for transforming from big endian to little endian
    function b2l_endian (a : in std_logic_vector)
        return std_logic_vector is
        variable result    : std_logic_vector(a'RANGE);
        constant cNumBytes : natural := a'length / 8;

    begin
        for i in 0 to cNumBytes-1 loop
            for j in 7 downto 0 loop
                result(8*i + j) := a(8*(cNumBytes-1-i) + j);
            end loop; -- j
        end loop;     -- i

        return result;
    end; -- function b2l_endian


    -- function for transforming from little endian to big endian  it
       is actually the same function
    function l2b_endian(a : in std_logic_vector)
        return std_logic_vector is
        variable result    : std_logic_vector(a'RANGE);
        constant cNumBytes : natural := a'length / 8;
    begin
        for i in 0 to cNumBytes-1 loop
            for j in 7 downto 0 loop
                result(8*i + j) := a(8*(cNumBytes-1-i) + j);
            end loop; -- j
        end loop;     -- i

        return result;
    end; -- function l2b_endian


end package body globals;
```

APPENDIX J

Globals components package from labs

```
--
   ---------------------------------------------------------------------------

-- Title       : global component
-- Project     : DLX for Microelectronic Systems
--
   ---------------------------------------------------------------------------

-- File        : 001-global_components.vhd
-- Author      : Francesco Angione <s262620@studenti.polito.it>
   franout@github.com
-- Company     : Politecnico di Torino, Italy
-- Created     : Wed Jul 22 22:56:15 2020
-- Last update : Sun Aug 30 22:58:11 2020
-- Platform    : Default Part Number
-- Standard    : VHDL-2008
--
   ---------------------------------------------------------------------------

-- Copyright (c) 2020 Politecnico di Torino, Italy
--
   --------------------------------------------------------------------------

-- Description: package which contains all basic components from the
   labs of MS
--                             the component files are in the folder
   ./global_components.package
--                             Suggestion: Open the folder of the dlx
    with Sublime Text v3 in order
--                                                 to see the declaration
    for each component when selected with the mouse
--
   ---------------------------------------------------------------------------
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use work.globals.all;
use work.constants.all; -- for avoiding compilation error
use work.alu_types.all;

package global_components is
        -- dummy alu declaration
        component ALU is
                generic (N   :     integer := numBit);
                port ( FUNC : IN TYPE_OP;
                        DATA1, DATA2 : IN  std_logic_vector(N-1 downto
                            0);
                        OUTALU       : OUT std_logic_vector(N-1 downto
                            0));
        end component ALU;

        --- a multiplexer for selection among a generic number of
           signals glued together
        component MUX_zbit_nbit IS
                Generic ( N : integer := NUMBIT;
                                                  -- number of bit for
                   every signal
                        Z : integer := 3);
                                                                      --
                          log2 of the number of signals in input,
                          baiscally it is the length of the
                          selection signal
                Port ( inputs : In std_logic_vector(0 TO ((2**Z)*(N))
                   -1 ); -- one single input with the concatenantion
                   of all input signals
                        SEL : In  std_logic_vector(Z-1 downto 0);
                        Y   : Out std_logic_vector(N-1 downto 0));
        end component MUX_zbit_nbit;

        -- register n bit
        component reg_nbit IS
                generic ( n      :     INTEGER := numbit);
                port (clk,reset : in std_logic;
                        d : in  std_logic_vector(N-1 DOWNTO 0);
                        Q : out std_logic_vector(N-1 downto 0));
        end component reg_nbit;

        -- windowed register file s
        component wrf is
                generic ( NBITREG : natural := 64;
                        M : natural := 8;  -- number of global
                          registers
```

```
                                    -- F and N must be power of
                                        two
                  N : natural := 4;  -- number of registers in
                     each in/out/local section
                  F : natural := 8); -- number of windows
          port ( CLK : IN std_logic;
                  RESET        : IN  std_logic;
                  ENABLE       : IN  std_logic;
                  RD1          : IN  std_logic;
                  RD2          : IN  std_logic;
                  WR           : IN  std_logic;
                  ADD_WR       : IN  std_logic_vector(f_log2(2**
                     f_log2(3*N) + M)-1 downto 0); -- the most
                     significant part of the address identifies
                      the global registers from the local one
                  ADD_RD1      : IN  std_logic_vector(f_log2(2**
                     f_log2(3*N) + M)-1 downto 0);
                  ADD_RD2      : IN  std_logic_vector(f_log2(2**
                     f_log2(3*N) + M)-1 downto 0);
                  DATAIN       : IN  std_logic_vector(NBITREG-1
                     downto 0);
                  OUT1         : OUT std_logic_vector(NBITREG-1
                     downto 0);
                  OUT2         : OUT std_logic_vector(NBITREG-1
                     downto 0);
                  SUB_CALL     : IN  std_logic;
                  SUB_RET      : IN  std_logic;
                  FILL         : OUT std_logic;
                  SPILL        : OUT std_logic;
                  DATA_MEM_IN  : IN  std_logic_vector(NBITREG-1
                     downto 0);
                  DATA_MEM_OUT : OUT std_logic_vector(NBITREG-1
                     downto 0);
                  READY        : OUT std_logic); -- signal for
                     the CU that defines when no memory
                     operaions are being performed
end component wrf;

-- register file
component register_file is
        generic ( NBITREG : natural := 64;
                  NBITADD : natural := 5);
        port ( CLK : IN std_logic;
                  RESET   : IN  std_logic;
                  ENABLE  : IN  std_logic;
                  RD1     : IN  std_logic;
                  RD2     : IN  std_logic;
                  WR      : IN  std_logic;
                  ADD_WR  : IN  std_logic_vector(NBITADD-1
                    downto 0);
                ADD_RD1 : IN  std_logic_vector(NBITADD-1
                    downto 0);
                ADD_RD2 : IN  std_logic_vector(NBITADD-1
                    downto 0);
                DATAIN  : IN  std_logic_vector(NBITREG-1
                    downto 0);
                OUT1    : OUT std_logic_vector(NBITREG-1
                    downto 0);
                OUT2    : OUT std_logic_vector(NBITREG-1
                    downto 0));
end component register_file;

-- p4 adder
-- hierarchy:
-- sparse tree carry generator:
-- propagate and generate network
-- general propagate  and generate block
-- general generate block
-- sum generator:
-- carry select block:
-- ripple carry adder x2:
-- full adder
component p4_adder IS
        GENERIC ( NBIT :    integer := 8 ); -- number of bits
            for the adder
        PORT( a,b      : IN std_logic_vector(NBIT -1 DOWNTO 0
            );
                cin  : IN  std_logic;
                s    : OUT std_logic_vector(NBIT-1 DOWNTO 0);
                cout : OUT std_logic);
END component p4_adder;

-- mux 2x1 on nbit
component mux21_nbit IS
        Generic (N :    integer := NUMBIT);
        Port ( A   : In std_logic_vector(N-1 downto 0) ;
                B   : In  std_logic_vector(N-1 downto 0);
                SEL : In  std_logic;
                Y   : Out std_logic_vector(N-1 downto 0));
end component mux21_nbit;

-- behavioural full adder
component FA is
        Port ( A : In std_logic;
                B  : In  std_logic;
                Ci : In  std_logic;
                S  : Out std_logic;
                Co : Out std_logic);
        end component FA;

        -- basic encoder
        component encoder IS
                PORT (y : IN std_logic_vector(2 DOWNTO 0);
                        sel : OUT std_logic_vector(2 DOWNTO 0));
        END component encoder;

        -- top level entity of the booth multiplier
        component boothmul IS
                GENERIC(N                     :    integer := 32);
                PORT( multiplier, multiplicand : IN std_logic_vector(N
                   -1 DOWNTO 0);
                        result : OUT std_logic_vector(2*N-1 DOWNTO 0))
                           ;
        END component boothmul;


        -- behavioural adder
        component adder IS
                GENERIC ( NBIT :    integer := 8 ); -- number of bits
                   for the adder
                PORT( a,b      : IN std_logic_vector(NBIT-1 DOWNTO 0 )
                   ;
                        cin : IN  std_logic;
                        s   : OUT std_logic_vector(NBIT DOWNTO 0));
        END component adder;


end package global_components;

package body global_components is

end package body global_components;
```

APPENDIX K

Simulation script

```
#!/usr/bin/bash

path_to_file="./hardware/dlx/"
## using questasim
source /software/scripts/init_questa10.7c

if [ -z $1 ]; then
        echo "usage␣./simulation.sh␣sbst.asm"
        exit
fi

path_to_hex_test_program="$1"

if [ ! -f $path_to_hex_test_program ] ; then
        echo "Input␣software␣file␣for␣IRAM␣not␣defined"
        exit
fi

echo "Moving␣the␣hex␣files␣into␣the␣memories␣folders"
cp "$path_to_hex_test_program" ../"$path_to_file"test_bench/memories/

echo "Starting␣initialization␣of␣simulation␣environment"
cd ..
if [ -d "./work" ] ;then
        rm -rf work
fi

vlib ./work # it also creates the folder
echo "Simulation␣ready␣to␣go!"
echo "Start␣hierarchical␣compilation"
echo "Compiling␣labs␣units"
vcom -2008 -check_synthesis -autoorder ${path_to_file}
   global_components.package/*.vhd
vcom -2008 -check_synthesis ${path_to_file}000-globals.vhd
vcom -2008 -check_synthesis ${path_to_file}001-global_components.vhd
# vlog for verilog -incr(mental) compilation
# vcom for vhdl-2008 and drc for synthesis (basic)


#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    adder.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    alu.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    alu_type.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    boothmul.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    complement2.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    constants.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    csb.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    encoder.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    fa.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    fd.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    gsb.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    mux.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    p4_adder.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    pgsb.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    rca.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    reg_nbit.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    registerfile.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    stcg.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    sum_gen.vhd
#vcom -2008 -check_synthesis ${path_to_file}global_components.package/
    pg.vhd


echo "Compilig␣memories"
vlog -incr ${path_to_file}test_bench/003-global_defs.svh
vlog -incr ${path_to_file}test_bench/004-implemented_instructions.svh
vlog -incr ${path_to_file}test_bench/006-property_def.svh
vlog -incr ${path_to_file}test_bench/memories/tb_memories.sv
vlog -incr ${path_to_file}test_bench/memories/005-memory_interfaces.
   svh
vlog -incr ${path_to_file}test_bench/memories/romem.sv
vlog -incr ${path_to_file}test_bench/memories/rwmem.sv

echo "Compiling␣stages,control␣unit␣and␣datapath"
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.e-
   Write_back.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.a-
   Fetch.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.b-
   Decode.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.d.
   Memory.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.b-
   Decode.stage/a.b.b.a-sign_extension.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.a-check_branch_logic.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.b-general_alu.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.a-mux.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.b-adder.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.c-complement2.
   vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.d-encoder.vhd
vcom -2008 -check_synthesis ${path_to_file}a.a-control_unit.vhd
vcom -2008 -check_synthesis ${path_to_file}a-DLX.vhd

echo "Compiling␣top␣level␣testbench␣for␣DLX"
vlog -incr ${path_to_file}test_bench/tb_dlx.sv

echo "Starting␣simulation␣of␣dlx␣top␣level␣entity"
## suppress novopt error and metavalue warnings
vsim  -suppress 12110 -suppress 8664 -novopt work.tb_dlx -do ./scripts
   /dlx_tb.do
```

```
echo "Do␣you␣want␣to␣see␣the␣DLX'tb␣with␣the␣Universal␣Verification␣
   Methodology␣architecture?␣y/n"
read answer
if [ answer=="y" ] ; then
export UVM_VERSION="1.1d"
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_sequence_item.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_sequence.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_sequencer.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_driver.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_monitor.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_scoreboard.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_env.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/uvm_class_def/dlx_test.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/dlx_wrapper.sv
vlog +incdir+/software/mentor/questa10.7c/questasim/uvm-"$UVM_VERSION"
   /../verilog_src/uvm-"$UVM_VERSION"/src -incr ${path_to_file}
   test_bench/tb_dlx_uvm.sv
vsim -c -sv_lib /software/mentor/questa10.7c/questasim/uvm-"
   $UVM_VERSION"/linux_x86_64/uvm_dpi +UVM_STACKTRACE  -suppress 8664
     work.tb_dlx_uvm -do ./scripts/dlx_uvm_tb.do
else
echo "Ok!␣you␣are␣missing␣a␣lot␣of␣fancy,␣amazing␣and␣astonishing␣
   things"
fi
rm -rf ./work
```

APPENDIX L

Synthesis script

```
#!/usr/bin/bash
echo "Starting␣synthesis␣setup"
cd ..
export path_to_file=$PWD"/hardware/dlx/"
cd project/


if [ -d "./synthesis" ] ;then
        cp -r ./synthesis ./synthesis.old
        echo "Previous␣synthesis␣run␣can␣be␣found␣in␣synthesis.old␣
           folder"
        rm -rf ./synthesis
fi
mkdir synthesis
cd synthesis
mkdir report
mkdir output_netlist
cp ../../scripts/.synopsys_dc.setup ./
cp ../../scripts/synthesis.tcl ./
source /software/scripts/init_synopsys_64.18
mkdir work
echo "Startig␣Design␣Vision␣executing␣synthesis.tcl␣script"
design_vision -f synthesis.tcl -no_gui
```

TCL script:

```
###############################################
#### analyzing and checking vhdl netlist  #####
###############################################
puts "Starting␣top␣down␣compilation"
set designer "Franout␣-␣Francesco␣Angione"
set company "Microelectronic␣Systems␣@␣Polito"
## maybe for vhdl2008 support
#set_property file_type {VHDL 2008} [get_files  {{D:/uni/2018-2019/
   Microelectronic systems/dlx_project/hardware/dlx/
   global_components.package/pg.vhd}}]
## path_to_file variable comes from bash script
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/constants.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/alu_type.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)000-globals.vhd"
analyze -library WORK -format vhdl "$env(path_to_file)001
   -global_components.vhd"

puts "Compiling␣labs␣units"
analyze -library WORK -format vhdl  "$env(path_to_file)
   global_components.package/wrf.vhd"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/adder.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/alu.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/boothmul.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/complement2.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/encoder.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/fa.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/fd.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/gsb.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/mult_gm.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/mux.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/mux21.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/mux21_nbit.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/p4_adder.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/pgsb.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/rca.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/csb.vhd␣"
```

```
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/reg_nbit.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/registerfile.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/stcg.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/sum_gen.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   global_components.package/pg.vhd"

puts "Compiling␣stages,control␣unit␣and␣datapath"
analyze -library WORK -format vhdl "$env(path_to_file)a.b-DataPath.vhd
   ␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.e-Write_back.stage.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.a-Fetch.stage.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.b-Decode.stage.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.d.Memory.stage.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.b-Decode.stage/a.b.b.a-sign_extension.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage/
   a.b.c.a-check_branch_logic.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage/a.b.c.b-general_alu.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage/
   a.b.c.c-boothmul_pipelined.vhd␣"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage/
   a.b.c.c-boothmul_pipelined.core/a.b.c.a-mux.vhd"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage/
   a.b.c.c-boothmul_pipelined.core/a.b.c.b-adder.vhd"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage/
   a.b.c.c-boothmul_pipelined.core/a.b.c.c-complement2.vhd"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.b-DataPath.core/a.b.c-Execute.stage/
   a.b.c.c-boothmul_pipelined.core/a.b.c.d-encoder.vhd"
analyze -library WORK -format vhdl "$env(path_to_file)
   a.a-control_unit.vhd␣"
analyze -autoread -library WORK -format vhdl "$env(path_to_file)
```

```
   a-DLX.vhd␣"


##############################################################
################ elaborating the top entity ###############
##############################################################
puts "Elaborating␣top␣level␣entity"
elaborate -update dlx -architecture dlx_rtl -library WORK -parameters
   "IR_SIZE=32,PC_SIZE=32"
current_design "DLX_IR_SIZE32_PC_SIZE32"

# define a clock name
set clockName "clk"
create_clock -name $clockName CLK

##########################################
# first compilation, without constraints #
##########################################
puts "Synthesis␣without␣constraints"
compile
# reporting riming and power after the first synthesis without
   constraints #
report_timing -nworst 10 > ./report/
   timing10worst_report_dlx_irsize32_pcsize32_nopt.rpt ; # 10 worse
   paths
report_timing > ./report/timing_report_dlx_irsize32_pcsize32b_nopt.rpt
report_area > ./report/area_report_dlx_irsize32_pcsize32_nopt.rpt
report_power > ./report/power_report_dlx_irsize32_pcsize32_nopt.rpt
report_clock > ./report/report_clock_dlx_irsize32_pcsize32_nopt.rpt

write -hierarchy -format ddc -output ./output_netlist/
   dlx_irsize32_pcsize32_nopt.ddc
write -hierarchy -format vhdl -output ./output_netlist/
   dlx_irsize32_pcsize32_nopt.vhdl
write -hierarchy -format verilog -output ./output_netlist/
   dlx_irsize32_pcsize32_nopt.v
write_sdc ./output_netlist/dlx_irsize32_pcsize32_nopt.sdc

puts "Synthesis␣push␣as␣much␣as␣possible␣the␣clock␣frequency␣i.e.␣
   until␣the␣slack␣is␣positive"
# forces a combinational max delay of REQUIRED_TIME from each of the
   inputs
# to each of th eoutput, that is a delay lower than the one found
   after
# the first compilation step #
# often this is the working clock period of your system #
#set a 20% lower required time( float )  than maxpath
set MAX_PATH_c [ get_timing_paths -delay_type max -nworst 1
   -include_hierarchical_pins ]
```

```
#calculating the value of max_path in ns
set mp_l [list]
foreach_in_collection path $MAX_PATH_c   {
        set mpi 0.0
    foreach_in_collection point [ get_attribute $path points ] {
        set mpi [ expr $mpi + [ get_attribute $point arrival ]]
        }
lappend mp_l $mpi
}

# since the path is critical there is only one object into the list
set REQUIRED_TIME [ expr $mp_l*0.80 ]
#########################
### clock constraints ###
#########################
create_clock -name $clockName -period $REQUIRED_TIME CLK
set_max_delay $REQUIRED_TIME -from [all_inputs] -to [all_outputs]
set_fix_hold $clockName

set max_transition_time 0.01
set_max_transition $max_transition_time [all_outputs]
set_min_delay 0.20 -from [all_inputs] -to [all_outputs]
set_input_delay 0.15 -clock $clockName [all_inputs]
set_output_delay 0.15 -clock $clockName [all_outputs]

optimize_registers -clock $clockName  -minimum_period_only
set_fix_hold $clockName

# optimize
# enable the scan insetion and evaluation of impact
compile_ultra -scan
# save report
report_timing > ./report/
   timing_report_dlx_irsize32_pcsize32_opt_20.rpt
report_area > ./report/area_report_dlx_irsize32_pcsize32_opt_20.rpt
report_power > ./report/power_report_dlx_irsize32_pcsize32_opt_20.rpt
report_clock > ./report/report_clock_dlx_irsize32_pcsize32_opt_20.rpt
# saving files
write -hierarchy -format ddc -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_20.ddc
write -hierarchy -format vhdl -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_20.vhdl
write -hierarchy -format verilog -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_20.v

#set a 10% lower required time( float )  than maxpath
set REQUIRED_TIME [ expr $mp_l*0.90 ]
#########################
```

```
### clock constraints ###
#########################
create_clock -name $clockName -period $REQUIRED_TIME CLK
set_max_delay $REQUIRED_TIME -from [all_inputs] -to [all_outputs]
set_fix_hold $clockName

set max_transition_time 0.01
set_max_transition $max_transition_time [all_outputs]
set_min_delay 0.20 -from [all_inputs] -to [all_outputs]
set_input_delay 0.15 -clock $clockName [all_inputs]
set_output_delay 0.15 -clock $clockName [all_outputs]

optimize_registers -clock $clockName  -minimum_period_only
set_fix_hold $clockName
# optimize
compile_ultra -scan
# save report
report_timing > ./report/
   timing_report_dlx_irsize32_pcsize32_opt_10.rpt
report_area > ./report/area_report_dlx_irsize32_pcsize32_opt_10.rpt
report_power > ./report/power_report_dlx_irsize32_pcsize32_opt_10.rpt
report_clock > ./report/report_clock_dlx_irsize32_pcsize32_opt_10.rpt

# saving files
write -hierarchy -format ddc -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_10.ddc
write -hierarchy -format vhdl -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_10.vhdl
write -hierarchy -format verilog -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_10.v
write_sdc ./output_netlist/dlx_irsize32_pcsize32_10.sdc


#set a 1% lower required time( float )  than maxpath
set REQUIRED_TIME [ expr $mp_l*0.99 ]
#########################
### clock constraints ###
#########################
create_clock -name $clockName -period $REQUIRED_TIME CLK
set_max_delay $REQUIRED_TIME -from [all_inputs] -to [all_outputs]
set_fix_hold $clockName

set max_transition_time 0.01
set_max_transition $max_transition_time [all_outputs]
set_min_delay 0.20 -from [all_inputs] -to [all_outputs]
set_input_delay 0.15 -clock $clockName [all_inputs]
set_output_delay 0.15 -clock $clockName [all_outputs]

optimize_registers -clock $clockName  -minimum_period_only
```

```
set_fix_hold $clockName
# optimize
compile_ultra -scan
# save report
report_timing > ./report/timing_report_dlx_irsize32_pcsize32_opt_1.rpt
report_area > ./report/area_report_dlx_irsize32_pcsize32_opt_1.rpt
report_power > ./report/power_report_dlx_irsize32_pcsize32_opt_1.rpt
report_clock > ./report/report_clock_dlx_irsize32_pcsize32_opt_1.rpt

# saving files
write -hierarchy -format ddc -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_1.ddc
write -hierarchy -format vhdl -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_1.vhdl
write -hierarchy -format verilog -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_1.v



# put constraint on finding the minimum possible areea for this design
set_max_area  0.0
puts "Synthesis␣push␣as␣much␣as␣possible␣the␣clock␣frequency␣i.e.␣
   until␣the␣slack␣is␣positive␣and␣push␣as␣much␣as␣possible␣the␣area"

# since the path is critical there is only one object into the list
set REQUIRED_TIME [ expr $mp_l*0.80 ]

#########################
### clock constraints ###
#########################
create_clock -name $clockName -period $REQUIRED_TIME CLK
set_max_delay $REQUIRED_TIME -from [all_inputs] -to [all_outputs]
set_fix_hold $clockName

set max_transition_time 0.01
set_max_transition $max_transition_time [all_outputs]
set_min_delay 0.20 -from [all_inputs] -to [all_outputs]
set_input_delay 0.15 -clock $clockName [all_inputs]
set_output_delay 0.15 -clock $clockName [all_outputs]

optimize_registers -clock $clockName  -minimum_period_only
set_fix_hold $clockName
# optimize
compile_ultra -scan
# save report
report_timing > ./report/
   timing_report_dlx_irsize32_pcsize32_opt_20_minarea.rpt
report_area > ./report/
   area_report_dlx_irsize32_pcsize32_opt_20_minarea.rpt
```

```
report_power > ./report/
   power_report_dlx_irsize32_pcsize32_opt_20_minarea.rpt
report_clock > ./report/
   report_clock_dlx_irsize32_pcsize32_opt_20_minarea.rpt

# saving files
write -hierarchy -format ddc -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_20_minarea.ddc
write -hierarchy -format vhdl -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_20_minarea.vhdl
write -hierarchy -format verilog -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_20_minarea.v

#set a 10% lower required time( float )  than maxpath
set REQUIRED_TIME [ expr $mp_l*0.90 ]

#########################
### clock constraints ###
#########################
create_clock -name $clockName -period $REQUIRED_TIME CLK
set_max_delay $REQUIRED_TIME -from [all_inputs] -to [all_outputs]
set_fix_hold $clockName

set max_transition_time 0.01
set_max_transition $max_transition_time [all_outputs]
set_min_delay 0.20 -from [all_inputs] -to [all_outputs]
set_input_delay 0.15 -clock $clockName [all_inputs]
set_output_delay 0.15 -clock $clockName [all_outputs]

optimize_registers -clock $clockName  -minimum_period_only
set_fix_hold $clockName
# optimize
compile_ultra -scan
# save report
report_timing > ./report/
   timing_report_dlx_irsize32_pcsize32_opt_10_minarea.rpt
report_area > ./report/
   area_report_dlx_irsize32_pcsize32_opt_10_minarea.rpt
report_power > ./report/
   power_report_dlx_irsize32_pcsize32_opt_10_minarea.rpt
report_clock > ./report/
   report_clock_dlx_irsize32_pcsize32_opt_10_minarea.rpt

# saving files
write -hierarchy -format ddc -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_10_minarea.ddc
write -hierarchy -format vhdl -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_10_minarea.vhdl
write -hierarchy -format verilog -output ./output_netlist/
```

```
   dlx_irsize32_pcsize32_topt_10_minarea.v


#set a 1% lower required time( float )  than maxpath
set REQUIRED_TIME [ expr $mp_1*0.99 ]

########################
### clock constraints ###
########################
create_clock -name $clockName -period $REQUIRED_TIME CLK
set_max_delay $REQUIRED_TIME -from [all_inputs] -to [all_outputs]
set_fix_hold $clockName

set max_transition_time 0.01
set_max_transition $max_transition_time [all_outputs]
set_min_delay 0.20 -from [all_inputs] -to [all_outputs]
set_input_delay 0.15 -clock $clockName [all_inputs]
set_output_delay 0.15 -clock $clockName [all_outputs]

optimize_registers -clock $clockName  -minimum_period_only
set_fix_hold $clockName
# optimize
compile_ultra -scan
# save report
report_timing > ./report/
   timing_report_dlx_irsize32_pcsize32_opt_1_minarea.rpt
report_area > ./report/
   area_report_dlx_irsize32_pcsize32_opt_1_minarea.rpt
report_power > ./report/
   power_report_dlx_irsize32_pcsize32_opt_1_minarea.rpt
report_clock > ./report/
   report_clock_dlx_irsize32_pcsize32_opt_1_minarea.rpt

# saving files
write -hierarchy -format ddc -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_1_minarea.ddc
write -hierarchy -format vhdl -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_1_minarea.vhdl
write -hierarchy -format verilog -output ./output_netlist/
   dlx_irsize32_pcsize32_topt_1_minarea.v
write_sdc ./output_netlist/dlx_irsize32_pcsize32_1_minarea.sdc

exit
```

APPENDIX M

Physical Design script

```
#!/usr/bin/bash
echo "Starting␣physical␣design"
cd ..

export path_to_file_synthesis=$PWD"/project/synthesis/"

gui_option=$1
if  [[ "$gui_option" != "no_gui" &&  "$gui_option" != "gui"  ]]  ;
   then
        echo "--->␣wrong␣parameter!␣Usage␣is␣./physical_design.sh␣gui/
           no_gui␣<---"
        exit
fi
cd project/
echo "Starting␣physical␣design␣setup"

if [ -d "./physical_design" ] ;then
        cp -r ./physical_design ./physical_design.old
        echo "Previous␣physical␣design␣run␣can␣be␣found␣in␣
           physical_design.old␣folder"
        rm -rf ./physical_design
fi
mkdir ./physical_design
cd ./physical_design
cp ../../scripts/physical_design.tcl ./
cp ../../scripts/Default_nopt.view ./
cp ../../scripts/Default_10.view ./
cp ../../scripts/Default_1_minarea.view ./
mkdir report
mkdir output_netlist
mkdir images_nopt
mkdir images_10
mkdir images_1_minarea
```

```
source /software/scripts/init_innovus17.11
# launch innovus
echo "Startig␣Innovus␣executing␣physical_design.tcl␣script␣with␣
   $gui_option"
if [ "$gui_option" == "gui" ] ; then
innovus -files physical_design.tcl
else
innovus -files physical_design.tcl -no_gui
fi

exit
```

TCL script:

```
puts "Configuring␣Innovus"
setMultiCpuUsage -cpuAutoAdjust {true} -localCpu 6
set_global _enable_mmmc_by_default_flow      $CTE::mmmc_default
suppressMessage ENCEXT-2799
loadWorkspace -name Physical
set defHierChar /
set delaycal_input_transition_delay 0.1ps
set fpIsMaxIoHeight 0
set init_gnd_net {gnd}
set init_mmmc_file {Default_nopt.view}
set init_oa_search_lib {}
set init_pwr_net {vdd}
set init_verilog "$env(path_to_file_synthesis)output_netlist/
   dlx_irsize32_pcsize32_nopt.v"
set init_lef_file /software/dk/nangate45/lef/
   NangateOpenCellLibrary.lef
set lsgOCPGainMult 1.000000
set LEF_DIR /software/dk/nangate45/lef
set LEF_list [list ${LEF_DIR}/NangateOpenCellLibrary.lef]
set init_lef_file "${LEF_list}"
set LIB_DIR /software/dk/nangate45/liberty
set MyTimingLibNom ${LIB_DIR}/
   NangateOpenCellLibrary_typical_ecsm_nowlm.lib
set MyTimingLibSlow ${LIB_DIR}/NangateOpenCellLibrary_slow_ecsm.lib
set MyTimingLibFast ${LIB_DIR}/NangateOpenCellLibrary_fast_ecsm.lib
set MycapTable $LEF_DIR/captables/NCSU_FreePDK_45nm.capTbl
puts "Floorplanning␣Innovus"
init_design
getIoFlowFlag
setIoFlowFlag 0
floorPlan -site FreePDK45_38x28_10R_NP_162NW_34O -r 1 0.6 5 5 5 5
getIoFlowFlag
```

```
## for gui run
uiSetTool select
getIoFlowFlag
fit
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0

puts "Power␣planning␣and␣routing␣Innovus"

setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0
   -avoid_short 0 -skip_crossing_trunks none -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size 1 -orthogonal_only true
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }
#addRing -nets {} -type core_rings -follow core -layer {top metal1
   bottom metal1 left metal2 right metal2} -width {top 1.8 bottom 1.8
    left 1.8 right 1.8} -spacing {top 1.8 bottom 1.8 left 1.8 right 1
   .8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8} -center 0
   -extend_corner {} -threshold 0 -jog_distance 0
   -snap_wire_center_to_grid None
setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0
   -avoid_short 0 -skip_crossing_trunks none -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size 1 -orthogonal_only true
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }
addRing -nets {gnd vdd} -type core_rings -follow core -layer {top
   metal9 bottom metal9 left metal10 right metal10} -width {top 0.8
   bottom 0.8 left 0.8 right 0.8} -spacing {top 0.8 bottom 0.8 left 0
   .8 right 0.8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8}
   -center 1 -extend_corner {} -threshold 0 -jog_distance 0
   -snap_wire_center_to_grid None
```

```
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
setAddStripeMode -ignore_block_check false -break_at none
   -route_over_rows_only false -rows_without_stripes_only false
   -extend_to_closest_target none -stop_at_last_wire_for_area false
   -partial_set_thru_domain false -ignore_nondefault_domains false
   -trim_antenna_back_to_shape none -spacing_type edge_to_edge
   -spacing_from_block 0 -stripe_min_length 0 -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size false -split_vias false
   -orthogonal_only true -allow_jog { padcore_ring  block_ring }
addStripe -nets {vdd gnd} -layer metal10 -direction vertical -width 0
   .8 -spacing 0.8 -set_to_set_distance 20 -start_from left
   -start_offset 15 -switch_layer_over_obs false
   -max_same_layer_jog_length 2 -padcore_ring_top_layer_limit metal10
    -padcore_ring_bottom_layer_limit metal1
   -block_ring_top_layer_limit metal10 -block_ring_bottom_layer_limit
    metal1 -use_wire_group 0 -snap_wire_center_to_grid None
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }

setSrouteMode -viaConnectToShape { noshape }
sroute -connect { blockPin padPin padRing corePin floatingStripe }
   -layerChangeRange { metal1(1) metal10(10) } -blockPinTarget {
   nearestTarget } -padPinPortConnect { allPort oneGeom }
   -padPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd
   } -floatingStripeTarget { blockring padring ring stripe ringpin
   blockpin followpin } -allowJogging 1 -crossoverViaLayerRange {
   metal1(1) metal10(10) } -nets { gnd vdd } -allowLayerChange 1
   -blockPin useLef -targetViaLayerRange { metal1(1) metal10(10) }
```

```
puts "␣Pin␣assignment␣"

getPinAssignMode -pinEditInBatch -quiet
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   CLK
setPinAssignMode -pinEditInBatch false
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   RST
setPinAssignMode -pinEditInBatch false


# iram
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   IRAM_ENABLE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   IRAM_READY
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Top -layer 1
   -spreadType side -pin {{IRAM_ADDRESS[0]} {IRAM_ADDRESS[1]} {
   IRAM_ADDRESS[2]} {IRAM_ADDRESS[3]} {IRAM_ADDRESS[4]} {IRAM_ADDRESS
   [5]} {IRAM_ADDRESS[6]} {IRAM_ADDRESS[7]} {IRAM_ADDRESS[8]} {
   IRAM_ADDRESS[9]} {IRAM_ADDRESS[10]} {IRAM_ADDRESS[11]} {
   IRAM_ADDRESS[12]} {IRAM_ADDRESS[13]} {IRAM_ADDRESS[14]} {
   IRAM_ADDRESS[15]} {IRAM_ADDRESS[16]} {IRAM_ADDRESS[17]} {
   IRAM_ADDRESS[18]} {IRAM_ADDRESS[19]} {IRAM_ADDRESS[20]} {
   IRAM_ADDRESS[21]} {IRAM_ADDRESS[22]} {IRAM_ADDRESS[23]} {
   IRAM_ADDRESS[24]} {IRAM_ADDRESS[25]} {IRAM_ADDRESS[26]} {
   IRAM_ADDRESS[27]} {IRAM_ADDRESS[28]} {IRAM_ADDRESS[29]} {
   IRAM_ADDRESS[30]} {IRAM_ADDRESS[31]}}
setPinAssignMode -pinEditInBatch true

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Top -layer 2
   -spreadType side -pin {{IRAM_DATA[0]} {IRAM_DATA[1]} {IRAM_DATA
   [2]} {IRAM_DATA[3]} {IRAM_DATA[4]} {IRAM_DATA[5]} {IRAM_DATA[6]} {
   IRAM_DATA[7]} {IRAM_DATA[8]} {IRAM_DATA[9]} {IRAM_DATA[10]} {
   IRAM_DATA[11]} {IRAM_DATA[12]} {IRAM_DATA[13]} {IRAM_DATA[14]} {
    IRAM_DATA[15]} {IRAM_DATA[16]} {IRAM_DATA[17]} {IRAM_DATA[18]} {
    IRAM_DATA[19]} {IRAM_DATA[20]} {IRAM_DATA[21]} {IRAM_DATA[22]} {
    IRAM_DATA[23]} {IRAM_DATA[24]} {IRAM_DATA[25]} {IRAM_DATA[26]} {
    IRAM_DATA[27]} {IRAM_DATA[28]} {IRAM_DATA[29]} {IRAM_DATA[30]} {
    IRAM_DATA[31]}}
setPinAssignMode -pinEditInBatch true

# dram
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
    Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
    DRAM_ENABLE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
    Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
    DRAM_READNOTWRITE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
    Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
    DRAM_READY
setPinAssignMode -pinEditInBatch false


setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Right -layer 1
    -spreadType side -pin {{DRAM_ADDRESS[0]} {DRAM_ADDRESS[1]} {
    DRAM_ADDRESS[2]} {DRAM_ADDRESS[3]} {DRAM_ADDRESS[4]} {DRAM_ADDRESS
    [5]} {DRAM_ADDRESS[6]} {DRAM_ADDRESS[7]} {DRAM_ADDRESS[8]} {
    DRAM_ADDRESS[9]} {DRAM_ADDRESS[10]} {DRAM_ADDRESS[11]} {
    DRAM_ADDRESS[12]} {DRAM_ADDRESS[13]} {DRAM_ADDRESS[14]} {
    DRAM_ADDRESS[15]} {DRAM_ADDRESS[16]} {DRAM_ADDRESS[17]} {
    DRAM_ADDRESS[18]} {DRAM_ADDRESS[19]} {DRAM_ADDRESS[20]} {
    DRAM_ADDRESS[21]} {DRAM_ADDRESS[22]} {DRAM_ADDRESS[23]} {
    DRAM_ADDRESS[24]} {DRAM_ADDRESS[25]} {DRAM_ADDRESS[26]} {
    DRAM_ADDRESS[27]} {DRAM_ADDRESS[28]} {DRAM_ADDRESS[29]} {
    DRAM_ADDRESS[30]} {DRAM_ADDRESS[31]}}
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Right -layer 2
    -spreadType side -pin {{DRAM_DATA[0]} {DRAM_DATA[1]} {DRAM_DATA
    [2]} {DRAM_DATA[3]} {DRAM_DATA[4]} {DRAM_DATA[5]} {DRAM_DATA[6]} {
    DRAM_DATA[7]} {DRAM_DATA[8]} {DRAM_DATA[9]} {DRAM_DATA[10]} {
    DRAM_DATA[11]} {DRAM_DATA[12]} {DRAM_DATA[13]} {DRAM_DATA[14]} {
    DRAM_DATA[15]} {DRAM_DATA[16]} {DRAM_DATA[17]} {DRAM_DATA[18]} {
```

```
   DRAM_DATA[19]} {DRAM_DATA[20]} {DRAM_DATA[21]} {DRAM_DATA[22]} {
   DRAM_DATA[23]} {DRAM_DATA[24]} {DRAM_DATA[25]} {DRAM_DATA[26]} {
   DRAM_DATA[27]} {DRAM_DATA[28]} {DRAM_DATA[29]} {DRAM_DATA[30]} {
   DRAM_DATA[31]}}
setPinAssignMode -pinEditInBatch false

## for gui run
setDrawView  fplan
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_fplan_postpin
setDrawView  ameba
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_ameba_postpin
setDrawView  place
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_place_postpin


puts "Cell␣placing␣Innovus"
placeDesign
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postCTS
optDesign -postCTS -hold
getFillerMode -quiet
addFiller -cell FILLCELL_X8 FILLCELL_X32 FILLCELL_X4 FILLCELL_X2
   FILLCELL_X16 FILLCELL_X1 -prefix FILLER

## for gui run
setDrawView  fplan
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_fplan_prerouting
setDrawView  ameba
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_ameba_prerouting
setDrawView  place
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_place_prerouting

puts "Signal␣routing␣␣Innovus"

setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -drouteStartIteration default
setNanoRouteMode -quiet -routeTopRoutingLayer default
setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail
```

```
setAnalysisMode -analysisType onChipVariation
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postRoute
optDesign -postRoute -hold
saveDesign DLX_IR_SIZE32_PC_SIZE32_nopt_saved
reset_parasitics
extractRC
redirect -quiet {set honorDomain [getAnalysisMode -honorClockDomains]}
     > /dev/null
timeDesign -postRoute -pathReports -drvReports -slackReports -numPaths
     50 -prefix DLX_IR_SIZE32_PC_SIZE32_nopt_postroute -outDir
   timingReports
get_time_unit
report_timing -machine_readable -max_paths 10000 -max_slack 0.75
   -path_exceptions all > DLX_IR_SIZE32_PC_SIZE32_nopt.mtarpt
load_timing_debug_report -name default_report
   DLX_IR_SIZE32_PC_SIZE32_nopt.mtarpt

puts "Verification"
verifyConnectivity -type all -error 1000 -warning 50
setVerifyGeometryMode -area { 0 0 0 0 } -minWidth true -minSpacing
   true -minArea true -sameNet true -short true -overlap true
   -offRGrid false -offMGrid true -mergedMGridCheck true -minHole
   true -implantCheck true -minimumCut true -minStep true
   -viaEnclosure true -antenna false -insuffMetalOverlap true
   -pinInBlkg false -diffCellViol true -sameCellViol false
   -padFillerCellsOverlap true -routingBlkgPinOverlap true
   -routingCellBlkgOverlap true -regRoutingOnly false
   -stackedViasOnRegNet false -wireExt true -useNonDefaultSpacing
   false -maxWidth true -maxNonPrefLength -1 -error 1000
verifyGeometry
setVerifyGeometryMode -area { 0 0 0 0 }
get_verify_drc_mode -disable_rules -quiet
get_verify_drc_mode -quiet -area
get_verify_drc_mode -quiet -layer_range
get_verify_drc_mode -check_implant -quiet
get_verify_drc_mode -check_implant_across_rows -quiet
get_verify_drc_mode -check_ndr_spacing -quiet
get_verify_drc_mode -check_only -quiet
get_verify_drc_mode -check_same_via_cell -quiet
get_verify_drc_mode -exclude_pg_net -quiet
get_verify_drc_mode -ignore_trial_route -quiet
get_verify_drc_mode -max_wrong_way_halo -quiet
get_verify_drc_mode -use_min_spacing_on_block_obs -quiet
get_verify_drc_mode -limit -quiet
set_verify_drc_mode -disable_rules {} -check_implant true
   -check_implant_across_rows false -check_ndr_spacing false
   -check_same_via_cell false -exclude_pg_net false
   -ignore_trial_route false -report
   DLX_IR_SIZE32_PC_SIZE32_nopt.drc.rpt -limit 1000
verify_drc

## dump images of various stages * congestion * density  floor
   placement       route_all  * route_C4   * route_C5   * route_C6  *
    route_M1  * route_M2        route_M3  * specialroute   *
   violations
## for gui run
setDrawView  fplan
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_fplan
setDrawView  ameba
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_ameba
setDrawView  place
fit
dumpToGIF ./images_nopt/DLX_IR_SIZE32_PC_SIZE32_nopt_place
dumpPictures -dir ./images_nopt/ -prefix [dbGet top.name] -fullScreen
saveFPlan ./DLX_IR_SIZE32_PC_SIZE32_nopt_floorplan
reportGateCount -level 5 -limit 100 -outfile
   DLX_IR_SIZE32_PC_SIZE32_nopt.gateCount
saveNetlist DLX_IR_SIZE32_PC_SIZE32_nopt.v
all_hold_analysis_views
all_setup_analysis_views
write_sdf  -ideal_clock_network DLX_IR_SIZE32_PC_SIZE32_nopt.sdf


puts "Starting␣with␣optimized␣design␣with␣a␣lower␣slack"

set defHierChar /
set delaycal_input_transition_delay 0.1ps
set fpIsMaxIoHeight 0
set init_gnd_net {gnd}
set init_mmmc_file Default_10.view
set init_oa_search_lib {}
set init_pwr_net {vdd}
set init_verilog "$env(path_to_file_synthesis)output_netlist/
   dlx_irsize32_pcsize32_topt_10.v"
set init_lef_file /software/dk/nangate45/lef/
   NangateOpenCellLibrary.lef
set lsgOCPGainMult 1.000000
set LEF_DIR /software/dk/nangate45/lef
set LEF_list [list ${LEF_DIR}/NangateOpenCellLibrary.lef]
set init_lef_file "${LEF_list}"
set LIB_DIR /software/dk/nangate45/liberty
set MyTimingLibNom ${LIB_DIR}/
   NangateOpenCellLibrary_typical_ecsm_nowlm.lib
set MyTimingLibSlow ${LIB_DIR}/NangateOpenCellLibrary_slow_ecsm.lib
set MyTimingLibFast ${LIB_DIR}/NangateOpenCellLibrary_fast_ecsm.lib
set MycapTable $LEF_DIR/captables/NCSU_FreePDK_45nm.capTbl

puts "Floorplanning␣Innovus"

init_design
getIoFlowFlag
setIoFlowFlag 0
floorPlan -site FreePDK45_38x28_10R_NP_162NW_34O -r 1 0.6 5 5 5 5
getIoFlowFlag
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0

puts "Power␣planning␣and␣routing␣Innovus"

setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0
   -avoid_short 0 -skip_crossing_trunks none -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size 1 -orthogonal_only true
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }
#addRing -nets {} -type core_rings -follow core -layer {top metal1
   bottom metal1 left metal2 right metal2} -width {top 1.8 bottom 1.8
    left 1.8 right 1.8} -spacing {top 1.8 bottom 1.8 left 1.8 right 1
   .8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8} -center 0
   -extend_corner {} -threshold 0 -jog_distance 0
   -snap_wire_center_to_grid None
setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0
   -avoid_short 0 -skip_crossing_trunks none -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size 1 -orthogonal_only true
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }
```

```
addRing -nets {gnd vdd} -type core_rings -follow core -layer {top
   metal9 bottom metal9 left metal10 right metal10} -width {top 0.8
   bottom 0.8 left 0.8 right 0.8} -spacing {top 0.8 bottom 0.8 left 0
   .8 right 0.8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8}
   -center 1 -extend_corner {} -threshold 0 -jog_distance 0
   -snap_wire_center_to_grid None
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
setAddStripeMode -ignore_block_check false -break_at none
   -route_over_rows_only false -rows_without_stripes_only false
   -extend_to_closest_target none -stop_at_last_wire_for_area false
   -partial_set_thru_domain false -ignore_nondefault_domains false
   -trim_antenna_back_to_shape none -spacing_type edge_to_edge
   -spacing_from_block 0 -stripe_min_length 0 -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size false -split_vias false
   -orthogonal_only true -allow_jog { padcore_ring  block_ring }
addStripe -nets {vdd gnd} -layer metal10 -direction vertical -width 0
   .8 -spacing 0.8 -set_to_set_distance 20 -start_from left
   -start_offset 15 -switch_layer_over_obs false
   -max_same_layer_jog_length 2 -padcore_ring_top_layer_limit metal10
    -padcore_ring_bottom_layer_limit metal1
   -block_ring_top_layer_limit metal10 -block_ring_bottom_layer_limit
    metal1 -use_wire_group 0 -snap_wire_center_to_grid None
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }

setSrouteMode -viaConnectToShape { noshape }
sroute -connect { blockPin padPin padRing corePin floatingStripe }
   -layerChangeRange { metal1(1) metal10(10) } -blockPinTarget {
   nearestTarget } -padPinPortConnect { allPort oneGeom }
   -padPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd
   } -floatingStripeTarget { blockring padring ring stripe ringpin
   blockpin followpin } -allowJogging 1 -crossoverViaLayerRange {
   metal1(1) metal10(10) } -nets { gnd vdd } -allowLayerChange 1
   -blockPin useLef -targetViaLayerRange { metal1(1) metal10(10) }


puts "␣Pin␣assignment␣"

getPinAssignMode -pinEditInBatch -quiet
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   CLK
setPinAssignMode -pinEditInBatch false
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   RST
setPinAssignMode -pinEditInBatch false


# iram
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   IRAM_ENABLE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   IRAM_READY
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Top -layer 1
   -spreadType side -pin {{IRAM_ADDRESS[0]} {IRAM_ADDRESS[1]} {
   IRAM_ADDRESS[2]} {IRAM_ADDRESS[3]} {IRAM_ADDRESS[4]} {IRAM_ADDRESS
   [5]} {IRAM_ADDRESS[6]} {IRAM_ADDRESS[7]} {IRAM_ADDRESS[8]} {
   IRAM_ADDRESS[9]} {IRAM_ADDRESS[10]} {IRAM_ADDRESS[11]} {
   IRAM_ADDRESS[12]} {IRAM_ADDRESS[13]} {IRAM_ADDRESS[14]} {
   IRAM_ADDRESS[15]} {IRAM_ADDRESS[16]} {IRAM_ADDRESS[17]} {
   IRAM_ADDRESS[18]} {IRAM_ADDRESS[19]} {IRAM_ADDRESS[20]} {
   IRAM_ADDRESS[21]} {IRAM_ADDRESS[22]} {IRAM_ADDRESS[23]} {
   IRAM_ADDRESS[24]} {IRAM_ADDRESS[25]} {IRAM_ADDRESS[26]} {
   IRAM_ADDRESS[27]} {IRAM_ADDRESS[28]} {IRAM_ADDRESS[29]} {
   IRAM_ADDRESS[30]} {IRAM_ADDRESS[31]}}
setPinAssignMode -pinEditInBatch true

setPinAssignMode -pinEditInBatch true
```

```
editPin -fixOverlap 1 -spreadDirection clockwise -side Top -layer 2
   -spreadType side -pin {{IRAM_DATA[0]} {IRAM_DATA[1]} {IRAM_DATA
   [2]} {IRAM_DATA[3]} {IRAM_DATA[4]} {IRAM_DATA[5]} {IRAM_DATA[6]} {
   IRAM_DATA[7]} {IRAM_DATA[8]} {IRAM_DATA[9]} {IRAM_DATA[10]} {
   IRAM_DATA[11]} {IRAM_DATA[12]} {IRAM_DATA[13]} {IRAM_DATA[14]} {
   IRAM_DATA[15]} {IRAM_DATA[16]} {IRAM_DATA[17]} {IRAM_DATA[18]} {
   IRAM_DATA[19]} {IRAM_DATA[20]} {IRAM_DATA[21]} {IRAM_DATA[22]} {
   IRAM_DATA[23]} {IRAM_DATA[24]} {IRAM_DATA[25]} {IRAM_DATA[26]} {
   IRAM_DATA[27]} {IRAM_DATA[28]} {IRAM_DATA[29]} {IRAM_DATA[30]} {
   IRAM_DATA[31]}}
setPinAssignMode -pinEditInBatch true

# dram
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   DRAM_ENABLE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   DRAM_READNOTWRITE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   DRAM_READY
setPinAssignMode -pinEditInBatch false


setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Right -layer 1
   -spreadType side -pin {{DRAM_ADDRESS[0]} {DRAM_ADDRESS[1]} {
   DRAM_ADDRESS[2]} {DRAM_ADDRESS[3]} {DRAM_ADDRESS[4]} {DRAM_ADDRESS
   [5]} {DRAM_ADDRESS[6]} {DRAM_ADDRESS[7]} {DRAM_ADDRESS[8]} {
   DRAM_ADDRESS[9]} {DRAM_ADDRESS[10]} {DRAM_ADDRESS[11]} {
   DRAM_ADDRESS[12]} {DRAM_ADDRESS[13]} {DRAM_ADDRESS[14]} {
   DRAM_ADDRESS[15]} {DRAM_ADDRESS[16]} {DRAM_ADDRESS[17]} {
   DRAM_ADDRESS[18]} {DRAM_ADDRESS[19]} {DRAM_ADDRESS[20]} {
   DRAM_ADDRESS[21]} {DRAM_ADDRESS[22]} {DRAM_ADDRESS[23]} {
   DRAM_ADDRESS[24]} {DRAM_ADDRESS[25]} {DRAM_ADDRESS[26]} {
   DRAM_ADDRESS[27]} {DRAM_ADDRESS[28]} {DRAM_ADDRESS[29]} {
   DRAM_ADDRESS[30]} {DRAM_ADDRESS[31]}}
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Right -layer 2
```

```
   -spreadType side -pin {{DRAM_DATA[0]} {DRAM_DATA[1]} {DRAM_DATA
   [2]} {DRAM_DATA[3]} {DRAM_DATA[4]} {DRAM_DATA[5]} {DRAM_DATA[6]} {
   DRAM_DATA[7]} {DRAM_DATA[8]} {DRAM_DATA[9]} {DRAM_DATA[10]} {
   DRAM_DATA[11]} {DRAM_DATA[12]} {DRAM_DATA[13]} {DRAM_DATA[14]} {
   DRAM_DATA[15]} {DRAM_DATA[16]} {DRAM_DATA[17]} {DRAM_DATA[18]} {
   DRAM_DATA[19]} {DRAM_DATA[20]} {DRAM_DATA[21]} {DRAM_DATA[22]} {
   DRAM_DATA[23]} {DRAM_DATA[24]} {DRAM_DATA[25]} {DRAM_DATA[26]} {
   DRAM_DATA[27]} {DRAM_DATA[28]} {DRAM_DATA[29]} {DRAM_DATA[30]} {
   DRAM_DATA[31]}}
setPinAssignMode -pinEditInBatch false


## for gui run
setDrawView  fplan
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_fplan_postpin
setDrawView  ameba
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_ameba_postpin
setDrawView  place
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_place_postpin


puts "Cell␣placing␣Innovus"
placeDesign
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postCTS
optDesign -postCTS -hold
getFillerMode -quiet
addFiller -cell FILLCELL_X8 FILLCELL_X32 FILLCELL_X4 FILLCELL_X2
   FILLCELL_X16 FILLCELL_X1 -prefix FILLER


## for gui run
setDrawView  fplan
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_fplan_prerouting
setDrawView  ameba
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_ameba_prerouting
setDrawView  place
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_place_prerouting
puts "Signal␣routing␣␣Innovus"

setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -drouteStartIteration default
setNanoRouteMode -quiet -routeTopRoutingLayer default
setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail
setAnalysisMode -analysisType onChipVariation
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postRoute
optDesign -postRoute -hold
saveDesign DLX_IR_SIZE32_PC_SIZE32_10_saved
reset_parasitics
extractRC
redirect -quiet {set honorDomain [getAnalysisMode -honorClockDomains]}
     > /dev/null
timeDesign -postRoute -pathReports -drvReports -slackReports -numPaths
     50 -prefix DLX_IR_SIZE32_PC_SIZE32_10_postroute -outDir
    timingReports
get_time_unit
report_timing -machine_readable -max_paths 10000 -max_slack 0.75
    -path_exceptions all > DLX_IR_SIZE32_PC_SIZE32_10.mtarpt
load_timing_debug_report -name default_report
    DLX_IR_SIZE32_PC_SIZE32_10.mtarpt

puts "Verification"
verifyConnectivity -type all -error 1000 -warning 50
setVerifyGeometryMode -area { 0 0 0 0 } -minWidth true -minSpacing
    true -minArea true -sameNet true -short true -overlap true
    -offRGrid false -offMGrid true -mergedMGridCheck true -minHole
    true -implantCheck true -minimumCut true -minStep true
    -viaEnclosure true -antenna false -insuffMetalOverlap true
    -pinInBlkg false -diffCellViol true -sameCellViol false
    -padFillerCellsOverlap true -routingBlkgPinOverlap true
    -routingCellBlkgOverlap true -regRoutingOnly false
    -stackedViasOnRegNet false -wireExt true -useNonDefaultSpacing
    false -maxWidth true -maxNonPrefLength -1 -error 1000
verifyGeometry
setVerifyGeometryMode -area { 0 0 0 0 }
get_verify_drc_mode -disable_rules -quiet
get_verify_drc_mode -quiet -area
get_verify_drc_mode -quiet -layer_range
get_verify_drc_mode -check_implant -quiet
get_verify_drc_mode -check_implant_across_rows -quiet
get_verify_drc_mode -check_ndr_spacing -quiet
get_verify_drc_mode -check_only -quiet
get_verify_drc_mode -check_same_via_cell -quiet
get_verify_drc_mode -exclude_pg_net -quiet
get_verify_drc_mode -ignore_trial_route -quiet
get_verify_drc_mode -max_wrong_way_halo -quiet
get_verify_drc_mode -use_min_spacing_on_block_obs -quiet
get_verify_drc_mode -limit -quiet
set_verify_drc_mode -disable_rules {} -check_implant true
   -check_implant_across_rows false -check_ndr_spacing false
   -check_same_via_cell false -exclude_pg_net false
   -ignore_trial_route false -report
   DLX_IR_SIZE32_PC_SIZE32_10.drc.rpt -limit 1000
verify_drc



setDrawView  fplan
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_fplan
setDrawView  ameba
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_ameba
setDrawView  place
fit
dumpToGIF ./images_10/DLX_IR_SIZE32_PC_SIZE32_10_place
dumpPictures -dir ./images_nopt/ -prefix [dbGet top.name] -fullScreen
saveFPlan ./DLX_IR_SIZE32_PC_SIZE32_10_floorplan

dumpPictures -dir ./images_10/ -prefix [dbGet top.name] -fullScreen
reportGateCount -level 5 -limit 100 -outfile
   DLX_IR_SIZE32_PC_SIZE32_10.gateCount
saveFPlan ./DLX_IR_SIZE32_PC_SIZE32_10_floorplan
saveNetlist DLX_IR_SIZE32_PC_SIZE32_10.v
all_hold_analysis_views
all_setup_analysis_views
write_sdf  -ideal_clock_network DLX_IR_SIZE32_PC_SIZE32_10.sdf




puts "Starting␣with␣optimized␣design␣with␣a␣lower␣slack␣and␣area"


set defHierChar /
set delaycal_input_transition_delay 0.1ps
set fpIsMaxIoHeight 0
set init_gnd_net {gnd}
set init_mmmc_file Default_10.view
set init_oa_search_lib {}
set init_pwr_net {vdd}
set init_verilog "$env(path_to_file_synthesis)output_netlist/
   dlx_irsize32_pcsize32_topt_1_minarea.v"
set init_lef_file /software/dk/nangate45/lef/
   NangateOpenCellLibrary.lef
```

```
set lsgOCPGainMult 1.000000
set LEF_DIR /software/dk/nangate45/lef
set LEF_list [list ${LEF_DIR}/NangateOpenCellLibrary.lef]
set init_lef_file "${LEF_list}"
set LIB_DIR /software/dk/nangate45/liberty
set MyTimingLibNom ${LIB_DIR}/
   NangateOpenCellLibrary_typical_ecsm_nowlm.lib
set MyTimingLibSlow ${LIB_DIR}/NangateOpenCellLibrary_slow_ecsm.lib
set MyTimingLibFast ${LIB_DIR}/NangateOpenCellLibrary_fast_ecsm.lib

set MycapTable $LEF_DIR/captables/NCSU_FreePDK_45nm.capTbl

puts "Floorplanning␣Innovus"

init_design
getIoFlowFlag
setIoFlowFlag 0
floorPlan -site FreePDK45_38x28_10R_NP_162NW_340 -r 1 0.6 5 5 5 5
getIoFlowFlag
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0

puts "Power␣planning␣and␣routing␣Innovus"

setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0
   -avoid_short 0 -skip_crossing_trunks none -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size 1 -orthogonal_only true
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }
#addRing -nets {} -type core_rings -follow core -layer {top metal1
   bottom metal1 left metal2 right metal2} -width {top 1.8 bottom 1.8
    left 1.8 right 1.8} -spacing {top 1.8 bottom 1.8 left 1.8 right 1
   .8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8} -center 0
   -extend_corner {} -threshold 0 -jog_distance 0
   -snap_wire_center_to_grid None
setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0
   -avoid_short 0 -skip_crossing_trunks none -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size 1 -orthogonal_only true
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }
addRing -nets {gnd vdd} -type core_rings -follow core -layer {top
   metal9 bottom metal9 left metal10 right metal10} -width {top 0.8
   bottom 0.8 left 0.8 right 0.8} -spacing {top 0.8 bottom 0.8 left 0
   .8 right 0.8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8}
   -center 1 -extend_corner {} -threshold 0 -jog_distance 0
   -snap_wire_center_to_grid None
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
setAddStripeMode -ignore_block_check false -break_at none
   -route_over_rows_only false -rows_without_stripes_only false
   -extend_to_closest_target none -stop_at_last_wire_for_area false
   -partial_set_thru_domain false -ignore_nondefault_domains false
   -trim_antenna_back_to_shape none -spacing_type edge_to_edge
   -spacing_from_block 0 -stripe_min_length 0 -stacked_via_top_layer
   metal10 -stacked_via_bottom_layer metal1
   -via_using_exact_crossover_size false -split_vias false
   -orthogonal_only true -allow_jog { padcore_ring  block_ring }
addStripe -nets {vdd gnd} -layer metal10 -direction vertical -width 0
   .8 -spacing 0.8 -set_to_set_distance 20 -start_from left
   -start_offset 15 -switch_layer_over_obs false
   -max_same_layer_jog_length 2 -padcore_ring_top_layer_limit metal10
    -padcore_ring_bottom_layer_limit metal1
   -block_ring_top_layer_limit metal10 -block_ring_bottom_layer_limit
    metal1 -use_wire_group 0 -snap_wire_center_to_grid None
   -skip_via_on_pin {  standardcell } -skip_via_on_wire_shape {
   noshape }

setSrouteMode -viaConnectToShape { noshape }
sroute -connect { blockPin padPin padRing corePin floatingStripe }
   -layerChangeRange { metal1(1) metal10(10) } -blockPinTarget {
   nearestTarget } -padPinPortConnect { allPort oneGeom }
   -padPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd
   } -floatingStripeTarget { blockring padring ring stripe ringpin
   blockpin followpin } -allowJogging 1 -crossoverViaLayerRange {
   metal1(1) metal10(10) } -nets { gnd vdd } -allowLayerChange 1
   -blockPin useLef -targetViaLayerRange { metal1(1) metal10(10) }


puts "␣Pin␣assignment␣"

getPinAssignMode -pinEditInBatch -quiet
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   CLK
setPinAssignMode -pinEditInBatch false
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   RST
setPinAssignMode -pinEditInBatch false


# iram
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   IRAM_ENABLE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   IRAM_READY
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Top -layer 1
   -spreadType side -pin {{IRAM_ADDRESS[0]} {IRAM_ADDRESS[1]} {
   IRAM_ADDRESS[2]} {IRAM_ADDRESS[3]} {IRAM_ADDRESS[4]} {IRAM_ADDRESS
   [5]} {IRAM_ADDRESS[6]} {IRAM_ADDRESS[7]} {IRAM_ADDRESS[8]} {
   IRAM_ADDRESS[9]} {IRAM_ADDRESS[10]} {IRAM_ADDRESS[11]} {
   IRAM_ADDRESS[12]} {IRAM_ADDRESS[13]} {IRAM_ADDRESS[14]} {
```

```
   IRAM_ADDRESS[15]} {IRAM_ADDRESS[16]} {IRAM_ADDRESS[17]} {
   IRAM_ADDRESS[18]} {IRAM_ADDRESS[19]} {IRAM_ADDRESS[20]} {
   IRAM_ADDRESS[21]} {IRAM_ADDRESS[22]} {IRAM_ADDRESS[23]} {
   IRAM_ADDRESS[24]} {IRAM_ADDRESS[25]} {IRAM_ADDRESS[26]} {
   IRAM_ADDRESS[27]} {IRAM_ADDRESS[28]} {IRAM_ADDRESS[29]} {
   IRAM_ADDRESS[30]} {IRAM_ADDRESS[31]}}
setPinAssignMode -pinEditInBatch true

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Top -layer 2
   -spreadType side -pin {{IRAM_DATA[0]} {IRAM_DATA[1]} {IRAM_DATA
   [2]} {IRAM_DATA[3]} {IRAM_DATA[4]} {IRAM_DATA[5]} {IRAM_DATA[6]} {
   IRAM_DATA[7]} {IRAM_DATA[8]} {IRAM_DATA[9]} {IRAM_DATA[10]} {
   IRAM_DATA[11]} {IRAM_DATA[12]} {IRAM_DATA[13]} {IRAM_DATA[14]} {
   IRAM_DATA[15]} {IRAM_DATA[16]} {IRAM_DATA[17]} {IRAM_DATA[18]} {
   IRAM_DATA[19]} {IRAM_DATA[20]} {IRAM_DATA[21]} {IRAM_DATA[22]} {
   IRAM_DATA[23]} {IRAM_DATA[24]} {IRAM_DATA[25]} {IRAM_DATA[26]} {
   IRAM_DATA[27]} {IRAM_DATA[28]} {IRAM_DATA[29]} {IRAM_DATA[30]} {
   IRAM_DATA[31]}}
setPinAssignMode -pinEditInBatch true

# dram
setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   DRAM_ENABLE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   DRAM_READNOTWRITE
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -unit MICRON -spreadDirection clockwise -side
   Left -layer 1 -spreadType start -spacing 0.14 -start 0.0 0.0 -pin
   DRAM_READY
setPinAssignMode -pinEditInBatch false


setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Right -layer 1
   -spreadType side -pin {{DRAM_ADDRESS[0]} {DRAM_ADDRESS[1]} {
   DRAM_ADDRESS[2]} {DRAM_ADDRESS[3]} {DRAM_ADDRESS[4]} {DRAM_ADDRESS
   [5]} {DRAM_ADDRESS[6]} {DRAM_ADDRESS[7]} {DRAM_ADDRESS[8]} {
   DRAM_ADDRESS[9]} {DRAM_ADDRESS[10]} {DRAM_ADDRESS[11]} {
   DRAM_ADDRESS[12]} {DRAM_ADDRESS[13]} {DRAM_ADDRESS[14]} {
   DRAM_ADDRESS[15]} {DRAM_ADDRESS[16]} {DRAM_ADDRESS[17]} {
    DRAM_ADDRESS[18]} {DRAM_ADDRESS[19]} {DRAM_ADDRESS[20]} {
    DRAM_ADDRESS[21]} {DRAM_ADDRESS[22]} {DRAM_ADDRESS[23]} {
    DRAM_ADDRESS[24]} {DRAM_ADDRESS[25]} {DRAM_ADDRESS[26]} {
    DRAM_ADDRESS[27]} {DRAM_ADDRESS[28]} {DRAM_ADDRESS[29]} {
    DRAM_ADDRESS[30]} {DRAM_ADDRESS[31]}}
setPinAssignMode -pinEditInBatch false

setPinAssignMode -pinEditInBatch true
editPin -fixOverlap 1 -spreadDirection clockwise -side Right -layer 2
    -spreadType side -pin {{DRAM_DATA[0]} {DRAM_DATA[1]} {DRAM_DATA
    [2]} {DRAM_DATA[3]} {DRAM_DATA[4]} {DRAM_DATA[5]} {DRAM_DATA[6]} {
    DRAM_DATA[7]} {DRAM_DATA[8]} {DRAM_DATA[9]} {DRAM_DATA[10]} {
    DRAM_DATA[11]} {DRAM_DATA[12]} {DRAM_DATA[13]} {DRAM_DATA[14]} {
    DRAM_DATA[15]} {DRAM_DATA[16]} {DRAM_DATA[17]} {DRAM_DATA[18]} {
    DRAM_DATA[19]} {DRAM_DATA[20]} {DRAM_DATA[21]} {DRAM_DATA[22]} {
    DRAM_DATA[23]} {DRAM_DATA[24]} {DRAM_DATA[25]} {DRAM_DATA[26]} {
    DRAM_DATA[27]} {DRAM_DATA[28]} {DRAM_DATA[29]} {DRAM_DATA[30]} {
    DRAM_DATA[31]}}
setPinAssignMode -pinEditInBatch false

## for gui run
setDrawView  fplan
fit
dumpToGIF ./images_1_minarea/
    DLX_IR_SIZE32_PC_SIZE32_1_minarea_fplan_postpin
setDrawView  ameba
fit
dumpToGIF ./images_1_minarea/
    DLX_IR_SIZE32_PC_SIZE32_1_minarea_ameba_postpin
setDrawView  place
fit
dumpToGIF ./images_1_minarea/
    DLX_IR_SIZE32_PC_SIZE32_1_minarea_place_postpin

puts "Cell␣placing␣Innovus"
placeDesign
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postCTS
optDesign -postCTS -hold
getFillerMode -quiet
addFiller -cell FILLCELL_X8 FILLCELL_X32 FILLCELL_X4 FILLCELL_X2
    FILLCELL_X16 FILLCELL_X1 -prefix FILLER
## for gui run
setDrawView  fplan
fit
dumpToGIF ./images_1_minarea/
    DLX_IR_SIZE32_PC_SIZE32_1_minarea_fplan_prerouting
setDrawView  ameba
fit
```

```
dumpToGIF ./images_1_minarea/
   DLX_IR_SIZE32_PC_SIZE32_1_minarea_ameba_prerouting
setDrawView  place
fit
dumpToGIF ./images_1_minarea/
   DLX_IR_SIZE32_PC_SIZE32_1_minarea_place_prerouting

puts "Signal␣routing␣␣Innovus"
setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -drouteStartIteration default
setNanoRouteMode -quiet -routeTopRoutingLayer default
setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail
setAnalysisMode -analysisType onChipVariation
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postRoute
optDesign -postRoute -hold
saveDesign DLX_IR_SIZE32_PC_SIZE32_1_minarea_saved
reset_parasitics
extractRC
redirect -quiet {set honorDomain [getAnalysisMode -honorClockDomains]}
    > /dev/null
timeDesign -postRoute -pathReports -drvReports -slackReports -numPaths
    50 -prefix DLX_IR_SIZE32_PC_SIZE32_1_minarea_postroute -outDir
   timingReports
get_time_unit
report_timing -machine_readable -max_paths 10000 -max_slack 0.75
   -path_exceptions all > DLX_IR_SIZE32_PC_SIZE32_1_minarea.mtarpt
load_timing_debug_report -name default_report
   DLX_IR_SIZE32_PC_SIZE32_1_minarea.mtarpt

puts "Verification"
verifyConnectivity -type all -error 1000 -warning 50
setVerifyGeometryMode -area { 0 0 0 0 } -minWidth true -minSpacing
   true -minArea true -sameNet true -short true -overlap true
   -offRGrid false -offMGrid true -mergedMGridCheck true -minHole
   true -implantCheck true -minimumCut true -minStep true
   -viaEnclosure true -antenna false -insuffMetalOverlap true
   -pinInBlkg false -diffCellViol true -sameCellViol false
   -padFillerCellsOverlap true -routingBlkgPinOverlap true
   -routingCellBlkgOverlap true -regRoutingOnly false
   -stackedViasOnRegNet false -wireExt true -useNonDefaultSpacing
   false -maxWidth true -maxNonPrefLength -1 -error 1000
verifyGeometry
setVerifyGeometryMode -area { 0 0 0 0 }
```

```
get_verify_drc_mode -disable_rules -quiet
get_verify_drc_mode -quiet -area
get_verify_drc_mode -quiet -layer_range
get_verify_drc_mode -check_implant -quiet
get_verify_drc_mode -check_implant_across_rows -quiet
get_verify_drc_mode -check_ndr_spacing -quiet
get_verify_drc_mode -check_only -quiet
get_verify_drc_mode -check_same_via_cell -quiet
get_verify_drc_mode -exclude_pg_net -quiet
get_verify_drc_mode -ignore_trial_route -quiet
get_verify_drc_mode -max_wrong_way_halo -quiet
get_verify_drc_mode -use_min_spacing_on_block_obs -quiet
get_verify_drc_mode -limit -quiet
set_verify_drc_mode -disable_rules {} -check_implant true
   -check_implant_across_rows false -check_ndr_spacing false
   -check_same_via_cell false -exclude_pg_net false
   -ignore_trial_route false -report
   DLX_IR_SIZE32_PC_SIZE32_1_minarea.drc.rpt -limit 1000
verify_drc
fit
setDrawView  fplan
dumpToGIF ./images_1_minarea/DLX_IR_SIZE32_PC_SIZE32_1_minarea_fplan
setDrawView  ameba
fit
dumpToGIF ./images_1_minarea/DLX_IR_SIZE32_PC_SIZE32_1_minarea_ameba
setDrawView  place
fit
dumpToGIF ./images_1_minarea/DLX_IR_SIZE32_PC_SIZE32_1_minarea_place
dumpPictures -dir ./images_nopt/ -prefix [dbGet top.name] -fullScreen
saveFPlan ./DLX_IR_SIZE32_PC_SIZE32_1_minarea_floorplan
dumpPictures -dir ./images_1_minarea/ -prefix [dbGet top.name]
   -fullScreen
reportGateCount -level 5 -limit 100 -outfile
   DLX_IR_SIZE32_PC_SIZE32_1_minarea.gateCount
saveFPlan ./DLX_IR_SIZE32_PC_SIZE32_1_minarea_floorplan
saveNetlist DLX_IR_SIZE32_PC_SIZE32_1_minarea.v
all_hold_analysis_views
all_setup_analysis_views
write_sdf  -ideal_clock_network DLX_IR_SIZE32_PC_SIZE32_1_minarea.sdf
exit
```

APPENDIX N

Regression Test script

```
#!/usr/bin/bash
echo "##########################################"
echo "##␣------>␣Boot␣regression␣test␣<------␣##"
echo "##########################################"
path_to_file="./hardware/dlx/"

echo "Starting␣initialization␣of␣simulation␣environment"
## using questasim
source /software/scripts/init_questa10.7c
cd .. # go in parent folder
if [ -d "./work" ] ;then
        rm -rf work
fi
## add fsm debug
echo "Starting␣regression␣functional␣tests"
vlib ./work # it also creates the folder
echo "Simulation␣ready␣to␣go!"
echo "Start␣hierarchical␣compilation"
vcom -2008 -check_synthesis -autoorder ${path_to_file}
   global_components.package/constants.vhd
vcom -2008 -check_synthesis -autoorder ${path_to_file}
   global_components.package/alu_type.vhd
vcom -2008 -check_synthesis -autoorder ${path_to_file}000-globals.vhd
vcom -2008 -check_synthesis -autoorder ${path_to_file}001-
   global_components.vhd
# vlog for verilog -incr(mental) compilation
# vcom for vhdl-2008 and drc for synthesis (basic)

echo "Compiling␣labs␣units" ## compile the entire folder
vcom -2008 -check_synthesis -autoorder ${path_to_file}
   global_components.package/*.vhd

vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/
   tb_sum_gen.vhd
```

```
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/
   MULTIPLIER_tb.vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/tb_alu.
   vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/tb_csb.
   vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/tb_fd.
   vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/
   tb_mux21_generic.vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/
   tb_p4adder.vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/tb_rca.
   vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/
   tb_registerfile.vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/
   tb_registerfile_wrf.vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/tb_regN.
   vhd
vcom -2008  ${path_to_file}test_bench/testbench_subunits_labs/tb_stcg.
   vhd


echo "Compilig␣memories"
vlog -incr ${path_to_file}test_bench/003-global_defs.svh
vlog -incr ${path_to_file}test_bench/004-implemented_instructions.svh
vlog -incr ${path_to_file}test_bench/006-property_def.svh
vlog -incr ${path_to_file}test_bench/memories/tb_memories.sv
vlog -incr ${path_to_file}test_bench/memories/005-memory_interfaces.
   svh
vlog -incr ${path_to_file}test_bench/memories/romem.sv
vlog -incr ${path_to_file}test_bench/memories/rwmem.sv



echo "Compiling␣stages,control␣unit␣and␣datapath"

vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.e-
   Write_back.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.a-
   Fetch.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.b-
   Decode.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.d.
   Memory.stage.vhd
```

```
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.b-
   Decode.stage/a.b.b.a-sign_extension.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.a-check_branch_logic.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.b-general_alu.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.a-mux.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.b-adder.vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.c-complement2.
   vhd
vcom -2008 -check_synthesis ${path_to_file}a.b-DataPath.core/a.b.c-
   Execute.stage/a.b.c.c-boothmul_pipelined.core/a.b.c.d-encoder.vhd
vcom -2008 -check_synthesis ${path_to_file}a.a-control_unit.vhd
vcom -2008 -check_synthesis ${path_to_file}a-DLX.vhd



echo "Compiling␣testbenches␣subunits"
vlog -incr ${path_to_file}test_bench/testbench_subunits/
   tb_writeback_stage.sv
vlog -incr ${path_to_file}test_bench/testbench_subunits/tb_cu.sv
vlog -incr ${path_to_file}test_bench/testbench_subunits/
   tb_decode_stage.sv
vlog -incr ${path_to_file}test_bench/testbench_subunits/
   tb_execute_stage.sv
vlog -incr ${path_to_file}test_bench/testbench_subunits/tb_fetch_stage
   .sv
vlog -incr ${path_to_file}test_bench/testbench_subunits/
   tb_memory_stage.sv


echo "##########################################################"
echo "####--->␣Starting␣regression␣test␣for␣every␣units␣<---####"
echo "##########################################################"


if [ ! -z "$1"  ] ; then
echo "Starting␣simulation␣for␣$1"
vsim -suppress 12110 -novopt work.tb_"$1" -do ./scripts/"$1"_tb.do

else
echo "Starting␣simulation␣of␣memory␣units"
vsim -suppress 12110 -novopt work.tb_memories -do ./scripts/
   memories_tb.do
```

```
echo "Starting␣simulation␣of␣fetch␣␣stage␣unit"
#suppress warning using the -novopt
vsim -suppress 12110 -novopt work.tb_fetch_stage -do ./scripts/
   fetch_stage_tb.do
echo "Starting␣simulation␣of␣decode␣stage␣unit"
vsim -suppress 12110 -novopt work.tb_decode_stage -do ./scripts/
   decode_stage_tb.do
echo "Starting␣simulation␣of␣execute␣stage␣unit"
vsim -suppress 12110 -novopt work.tb_execute_stage -do ./scripts/
   execute_stage_tb.do
echo "Starting␣simulation␣of␣memory␣stage␣unit"
vsim -suppress 12110 -novopt work.tb_memory_stage -do ./scripts/
   memory_stage_tb.do
echo "Starting␣simulation␣of␣write␣stage␣unit"
vsim -suppress 12110 -novopt work.tb_writeback_stage -do ./scripts/
   writeback_stage_tb.do
echo "Starting␣simulation␣of␣cu␣unit"
vsim -suppress 12110 -novopt work.tb_cu -do ./scripts/cu_tb.do
fi
echo "[PASS]␣->␣Regression␣test␣passed␣,␣you␣can␣safely␣execute␣the␣
   simulation.sh␣script␣for␣testing␣the␣DLX␣top␣level␣entity"
rm -rf work
exit
```