VLSI Design II

Tutorial for the Semi Custom Part of the Image Processing System

Torsten Mähne

22 March to 5 April 2007
Laboratoire de Systèmes Microélectroniques (LSM)

Contents

1. Introduction .................................................. 1
   1.1. Preparation of the Project Directory .................. 2

2. Single Port Random Access Memory (RAM) .................. 3

3. Histogram Calculation Block ................................ 3
   3.1. Session 1: Creation and Simulation of the VHDL model of the histogram block .......... 3
   3.2. Session 2: Synthesis of the Gate-Level Netlist of the Histogram Block .................. 5
   3.3. Session 3: Placement & Routing of the Histogram Block .................................. 6

A. Single Port Random Access Memory (RAM) ................ 6

B. Histogram Block ........................................... 10

1. Introduction

The goal of this tutorial is to get familiar with the Electronic Design Automation (EDA) tools used in a typical semi custom digital design flow. It is based on the tutorial [5] given in the semi-custom part of the practical laboratory for the EDA Based Design Course. It’s principal parts VHDL and Verilog simulation (Chapter 2), Logic synthesis (Chapter 3), and Standard cell placement and routing (Chapter 4) will be repeated for the histogram block as follows:

Session 1 (22 March 2007): Creation of the Register Transfer Level (RTL) description in VHDL of the histogram block and its simulation with Mentor Graphics ModelSim

Session 2 (29 March 2007): Logic synthesis of a gate-level netlist from the RTL description using Synopsys Design Vision

*Phone: +41(21)69-36922, E-Mail: torsten.maehne@epfl.ch
**Session 3 (5 April 2007):** Automatic placement & routing of the netlist into a layout of the histogram block using Cadence First Encounter


1.1. Preparation of the Project Directory

All files created by the designer or generated by EDA tools during the course of the Image Processing System (IPS) project should be organised in directory structure as described in Vachoux [5], Sections 1.2 to 1.4. Follow the following steps to create the project directory after logging into immsunsrv2.epfl.ch using one of the edatpx accounts (with x = 1…10):

1. Create the project hierarchy using the `create_eda_project` script:

   [1]edatp1@immsunsrv2-edatp1> mkdir vlsi2
   [2]edatp1@immsunsrv2-edatp1> cd vlsi2/
   [3]edatp1@immsunsrv2-vlsi2> create_eda_project IPS
   [4]edatp1@immsunsrv2-vlsi2> cd IPS/

2. Set-up the EDA tools to be used in the project by creating or modifying a file called `edadk.conf` in your home directory or the top-level of your project directory (supersedes `~/edadk.conf`). The `edadk.conf` file should contain the following lines:

   ```
   edadk.conf
   1 mgc msim 6.2d
   2 mgc ams 2006.2a
   3 snps syn 2005.09
   4 cds soce 4.1
   5 cds ic 5.1.41
   6 cds assura 3.1.6
   7 cds mgc_ams 2006.2a
   8 dk ams hk370
   ```

3. Install the AMS design kit for use with Synopsys Design Vision and Cadence First Encounter by issuing the following command from the top-level of your project directory:

   [5]edatp1@immsunsrv2-IPS> ams_setup -p c35b4 -t synopsys_dc
   [6]edatp1@immsunsrv2-IPS> ams_setup -p c35b4 -t cadence_soce

4. Install the AMS design kit for use with Cadence IC by launching it once with the technology option c35b4 from the layout subdirectory LAY/:

   [7]edatp1@immsunsrv2-IPS> cd LAY
   [8]edatp1@immsunsrv2-LAY> ams_cds -t c35b4 -m fb

   When you are asked for the exact process option, select C35B4M6. Quit Cadence IC by selecting **Exit...** from the **File** menu of the icfb.

5. Change back to the top-level of the project directory:

   [9]edatp1@immsunsrv2-LAY> cd ..

   **Remember:** All EDA tools need to be launched from the top-level of the project directory to find their configuration files except for Cadence IC, which needs to be launched always from the LAY/ subdirectory.
Table 1: Interface of the Random Access Memory (RAM)

(a) Generics

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBITS_WORD</td>
<td>positive</td>
<td>8</td>
<td>Bits of a word</td>
</tr>
<tr>
<td>SIZE</td>
<td>positive</td>
<td>256</td>
<td>Number of words</td>
</tr>
<tr>
<td>NBITS_ADDRESS</td>
<td>positive</td>
<td>8</td>
<td>Bits of address</td>
</tr>
</tbody>
</table>

(b) Ports

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Dir.</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Clock input</td>
</tr>
<tr>
<td>reset</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Asynchronous reset (active low)</td>
</tr>
<tr>
<td>enable</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Enable RAM operation</td>
</tr>
<tr>
<td>rw_select</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Select read '0' or write '1' mode</td>
</tr>
<tr>
<td>address</td>
<td>signal</td>
<td>in</td>
<td>std_logic_vector(NBITS_ADDRESS - 1 downto 0)</td>
<td>Address to read from or write to</td>
</tr>
<tr>
<td>data_in</td>
<td>signal</td>
<td>in</td>
<td>std_logic_vector(NBITS_WORD - 1 downto 0)</td>
<td>Data input</td>
</tr>
<tr>
<td>data_out</td>
<td>signal</td>
<td>out</td>
<td>std_logic_vector(NBITS_WORD - 1 downto 0)</td>
<td>Data output</td>
</tr>
</tbody>
</table>

2. Single Port Random Access Memory (RAM)

The histogram block needs a RAM to store the number of occurrences of each gray scale value of a frame. The RAM will be created by the full-custom group during the course of the IPS project, but still the semi-custom group needs a simulation model, which will also serve as a specification.

The interface of the RAM block consisting of generics to configure the block and the ports to communicate with the block is given in Table 1. The behaviour of the single port RAM is implemented using a synchronous process with asynchronous active-low reset. If the RAM is enabled with enable = '1' and rw_select = '0', then a data word is read from the specified address at each positive clock edge and output at data_out. If the RAM is enabled with enable = '1' and rw_select = '1', then the data from data_in is written to the specified address at each positive clock edge. The VDHL source code of the single port RAM is given in Listing 1.

The testbench single port RAM given in Listing 2 carries out the test in two steps. First a random pattern is written into each memory cell using a pseudo random number generator. Afterwards all memory cells are read out and their value is compared with the values produced by the reinitialised pseudo random number generator.

3. Histogram Calculation Block

The histogram block calculates the histogram from the serial data read from video_in at each positive edge of the video_clock. Its interface definition is given in Table 2. The calculation is restarted after a frame_start has been signalled. It then calculates the histogram for the whole frame consisting of width * height pixels and signals its completion using the done signal. The histogram block relies on an external single port RAM (Section 2) to store the histogram.

3.1. Session 1: Creation and Simulation of the VHDL model of the histogram block

The first task of this tutorial is to create the VHDL model of the histogram calculation block with the interface from Table 2. The block shall have a asynchronous active-low reset. The histogram calculation is controlled through a synchronous Finite State Machine (FSM) (Figure 1).
### Table 2: Interface of the histogram calculation block

#### (a) Generics

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIDTH</td>
<td>positive</td>
<td>320</td>
<td>Width of frame</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>positive</td>
<td>240</td>
<td>Height of frame</td>
</tr>
<tr>
<td>NBITS_PIXEL</td>
<td>positive</td>
<td>8</td>
<td>Bits per pixel</td>
</tr>
<tr>
<td>NBITS_HISTOGRAM</td>
<td>positive</td>
<td>18</td>
<td>Bits for each histogram entry</td>
</tr>
</tbody>
</table>

#### (b) Ports

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Dir.</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Clock</td>
</tr>
<tr>
<td>reset</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Reset (active low)</td>
</tr>
<tr>
<td>frame_start</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Frame start</td>
</tr>
<tr>
<td>video_clock</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Video clock</td>
</tr>
<tr>
<td>video_in</td>
<td>signal</td>
<td>in</td>
<td>std_logic_vector(NBITS_PIXEL - 1 downto 0)</td>
<td>Digitised video signal</td>
</tr>
<tr>
<td>done</td>
<td>signal</td>
<td>out</td>
<td>std_logic</td>
<td>Histogram done</td>
</tr>
<tr>
<td>read_request</td>
<td>signal</td>
<td>in</td>
<td>std_logic</td>
<td>Read request for the histogram data</td>
</tr>
<tr>
<td>address</td>
<td>signal</td>
<td>in</td>
<td>std_logic_vector(NBITS_PIXEL - 1 downto 0)</td>
<td>Address to read from</td>
</tr>
<tr>
<td>data_out</td>
<td>signal</td>
<td>out</td>
<td>std_logic_vector(NBITS_HISTOGRAM - 1 downto 0)</td>
<td>Output for histogram data</td>
</tr>
<tr>
<td>ram_rw_select</td>
<td>signal</td>
<td>out</td>
<td>std_logic</td>
<td>Read or write mode select for RAM</td>
</tr>
<tr>
<td>ram_enable</td>
<td>signal</td>
<td>out</td>
<td>std_logic</td>
<td>Enable signal for RAM</td>
</tr>
<tr>
<td>ram_address</td>
<td>signal</td>
<td>out</td>
<td>std_logic_vector(NBITS_PIXEL - 1 downto 0)</td>
<td>Address for RAM</td>
</tr>
<tr>
<td>ram_data_in</td>
<td>signal</td>
<td>in</td>
<td>std_logic_vector(NBITS_HISTOGRAM - 1 downto 0)</td>
<td>Data read from RAM</td>
</tr>
<tr>
<td>ram_data_out</td>
<td>signal</td>
<td>out</td>
<td>std_logic_vector(NBITS_HISTOGRAM - 1 downto 0)</td>
<td>Data written to RAM</td>
</tr>
<tr>
<td>ram_reset</td>
<td>signal</td>
<td>out</td>
<td>std_logic</td>
<td>Reset for RAM (active low)</td>
</tr>
</tbody>
</table>
State Description

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START_PIXEL</td>
<td>Wait for first frame pixel, read its value, and answer requests for histogram entries</td>
</tr>
<tr>
<td>READ_PIXEL</td>
<td>Wait for next pixel and read its value</td>
</tr>
<tr>
<td>READ_HISTOGRAM</td>
<td>Read the histogram entry corresponding to current pixel</td>
</tr>
<tr>
<td>WAIT_RAM</td>
<td>Wait for RAM to complete the read request</td>
</tr>
<tr>
<td>WRITE_HISTOGRAM</td>
<td>Write back the incremented histogram entry corresponding to the current pixel</td>
</tr>
</tbody>
</table>

Figure 1: State diagram of the histogram calculation block

The single port RAM is private to the synthesisable histogram calculation block, which also governs external accesses to the RAM. Therefore, both blocks will be encapsulated into a histogram top block (Listing 3).

The second task is to create the testbench for the histogram top block. The testing approach is similar to the one used for the single port RAM (Listing 2). The testbench shall feed the histogram block after the reset with a random video signal, from which it calculates in parallel the histogram. After the block signals that it finished the calculation of the histogram for the current frame, the histogram shall be read out and compared with the histogram calculated by the testbench. If there are discrepancies, an error message shall be generated by the simulator.

3.2. Session 2: Synthesis of the Gate-Level Netlist of the Histogram Block

The Synthesis of the gate-level netlist from the VHDL RTL model of the histogram block will be done according to the description in Vachoux [5], Chapter 3 using Synopsys Design Vision. A Tcl script SYN/BIN/histogram_syn.tcl (Listing 4, downloadable from the Moodle site) can be used to automatise the synthesis for different constraints. To run the Tcl script, execute the following command in a Unix shell from the IPS/ project directory:
The Tcl script can serve as a template for synthesis scripts for other digital blocks. An introduction for the usage of Tcl in the context of (Synopsys) EDA tools is given in Syn [4].

The post-synthesis simulation with Standard Delay Format (SDF) back annotation of the gate delays is done according to Vachoux [5], Section 2.3 using Mentor Graphics ModelSim. A new histogram_top_syn entity with accompanying testbench tb_histogram_top_syn needs to created as modified versions of histogram_top and tb_histogram_top, which now instantiate the synthesised histogram calculation block. The region, to which the SDF file SYN/TIM/histogram_WIDTH320_HEIGHT240_NBITS_PIXEL8_NBITS_HISTOGRAM18_clock10ns_share_mapped.sdf needs to be mapped, is dut:histogram_1 and should be entered in the SDF tab of the Start Simulation dialog of Mentor Graphics ModelSim.

3.3. Session 3: Placement & Routing of the Histogram Block

The placement and routing of the gate-level netlist of the histogram block will be done according to the description in Vachoux [5], Chapter 4 using Cadence First Encounter. A Tcl script PAR/BIN/histogram_par.tcl (Listing 6, downloadable from the Moodle site) can be used to automate the placement and routing. Before it can be executed, an Encounter input configuration file PAR/CONF/histogram_WIDTH320_HEIGHT240_NBITS_PIXEL8_NBITS_HISTOGRAM18.conf needs to be created using the Design Import... Dialog from the File menu of Cadence Encounter as described in Vachoux [5], Section 4.2. An IO file (Listing 5, downloadable from the Moodle site) PAR/CONF/histogram_WIDTH320_HEIGHT240_NBITS_PIXEL8_NBITS_HISTOGRAM18.io controls the placement of the pins and prevents their placement on one of the power ring segments, which would lead to Design Rule Checker (DRC) errors. To run the Tcl script, execute the following command in a Unix shell from the IPS/ project directory:

[13]edatp1@immsunsrv2-IPS> encounter -log PAR/LOG/encounter -overwrite \
-init PAR/BIN/histogram_par.tcl -win

The Tcl script can serve as a template for placement and routing scripts for other digital blocks.

The post-placement-and-routing simulation with SDF back annotation of the gate delays is done according to Vachoux [5], Section 2.4 using the Verilog module of the histogram block generated by Cadence First Encounter. A new histogram_top_par entity with accompanying testbench tb_histogram_top_par needs to created as modified versions of histogram_top and tb_histogram_top, which now instantiate the placed and routed histogram calculation block. The region, to which the SDF file PAR/TIM/histogram_WIDTH320_HEIGHT240_NBITS_PIXEL8_NBITS_HISTOGRAM18_clock10ns_cts-routed.sdf needs to be mapped, is dut:histogram_1 and should be entered in the SDF tab of the Start Simulation dialog of Mentor Graphics ModelSim.

A. Single Port Random Access Memory (RAM)

Listing 1: VHDL source code of the Single Port RAM

```vhdl
-- Title : Synchronous Single Port RAM
-- Project : Image Processing System (IPS)
-- File : single_port_ram.vhd
-- Author : Torsten Maehne <torsten.maehne@epfl.ch>
-- Company : EPFL/STI/IMM/LSM
-- Created : 2007-03-20
-- Last update: 2007-03-22
-- Platform : ModelSim 6.2d
-- Standard : VHDL '93/02, Math Packages
-- Depends :
-- Description:
-- The behaviour of the single port RAM is implemented using a synchronous
-- process with asynchronous active-low reset. If the RAM is enabled with
-- enable = '1' and rw_select = '0', then a data word is read from the
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity single_port_ram is
  generic (
    NBITS_WORD : positive := 8; -- bits of a word
    SIZE : positive := 256; -- number of words
    NBITS_ADDRESS : positive := 8); -- bits of address
  port (
    signal clock : in std_logic; -- clock input
    signal reset : in std_logic; -- asynchronous reset (active low)
    signal enable : in std_logic; -- enable RAM operation
    signal rw_select : in std_logic; -- select read '0' or write '1' mode
    signal address : in std_logic_vector(NBITS_ADDRESS - 1 downto 0); -- address to read from or write to
    signal data_in : in std_logic_vector(NBITS_WORD - 1 downto 0); -- data input
    signal data_out : out std_logic_vector(NBITS_WORD - 1 downto 0)); -- data output
  end entity single_port_ram;

architecture behavioural of single_port_ram is
begin -- architecture behavioural
  -- purpose: control the read and write process to the RAM
  -- type : sequential
  -- inputs : clock, reset, enable, rw_select, address
  -- outputs: data_out
  ram_control : process (clock, reset) is
    type word_vector is array (natural range <>) of std_logic_vector(NBITS_WORD - 1 downto 0);
    variable memory : word_vector(0 to SIZE - 1) := (others => (others => '0')); -- memory
  begin
    if reset = '0' then
      -- asynchronous reset (active low)
      memory := (others => (others => '0'));
      data_out <= (others => '0');
    elsif clock'event and clock = '1' then
      -- rising clock edge
      if enable = '1' and rw_select = '0' then
        -- read from RAM
        data_out <= memory(to_integer(unsigned(address)));
      elsif enable = '1' and rw_select = '1' then
        -- write to RAM
        memory(to_integer(unsigned(address))) := data_in;
      end if;
  end if;
end process ram fsm;
end architecture behavioural;
Listing 2: Testbench for the Single Port RAM

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity tb_single_port_ram is
end entity tb_single_port_ram;

architecture testbench of tb_single_port_ram is

-- testbench constants
constant CLOCK_PERIOD : time := 10 ns; -- clock period
constant T_WAIT : time := 2 * CLOCK_PERIOD; -- wait time till start of test 
  pattern generation

-- random generator
constant INITIAL_SEED_1 : positive := 1; -- initial seed 1
constant INITIAL_SEED_2 : positive := 1; -- initial seed 2
```

```vhdl
begin
	extbf{null;}

end if;

end process ram_control;

end architecture behavioural;

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

-- Title : Testbench for design "single_port_ram"
-- Project : Image Processing System (IPS)
-- File : tb_single_port_ram.vhd
-- Author : Torsten Maehne <torsten.maehne@epfl.ch>
-- Company : EPFL/STI/IMM/LSM
-- Created : 2007-03-20
-- Last update : 2007-03-23
-- Platform : ModelSim 6.2d
-- Standard : VHDL '93/02, Math Packages
-- Depends : single_port_ram

-- Description:
-- The single port RAM is tested in two steps. First a random pattern is
-- written into each memory cell using a pseudo random number generator.
-- Afterwards all memory cells are read out and their value is compared with
-- the values produced by the reinitialised pseudo random generator.

-- Copyright (c) 2007 EPFL/STI/IMM/LSM

-- Date Version Author Description
-- 2007-03-20 1.0 maehne Created

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

-- testbench constants
constant CLOCK_PERIOD : time := 10 ns; -- clock period
constant T_WAIT : time := 2 * CLOCK_PERIOD; -- wait time till start of test 
  pattern generation

-- random generator
constant INITIAL_SEED_1 : positive := 1; -- initial seed 1
constant INITIAL_SEED_2 : positive := 1; -- initial seed 2

```
-- component generics
constant NBITS_WORD : positive := 8;
constant SIZE : positive := 256;
constant NBITS_ADDRESS : positive := 8;

-- component ports
signal rw_select : std_logic;
signal address : std_logic_vector(NBITS_ADDRESS - 1 downto 0);
signal data_in : std_logic_vector(NBITS_WORD - 1 downto 0);
signal data_out : std_logic_vector(NBITS_WORD - 1 downto 0);
signal enable : std_logic;
signal clock : std_logic := '1';
signal reset : std_logic;

begin
  -- architecture testbench
  -- component instantiation
  DUT: entity work.single_port_ram
  generic map
    (NBITS_WORD => NBITS_WORD,
     SIZE => SIZE,
     NBITS_ADDRESS => NBITS_ADDRESS)
  port map
    (rw_select => rw_select,
     address => address,
     data_in => data_in,
     data_out => data_out,
     enable => enable,
     clock => clock,
     reset => reset);

  -- clock generation
clock <= not clock after CLOCK_PERIOD / 2;

  -- reset generation
reset <= '0' after CLOCK_PERIOD / 4,
       '1' after (3 * CLOCK_PERIOD) / 4;

  -- test pattern generation
  testpattern_generator : process
    variable value_real : real := 0.0; -- real value
    variable value_unsigned : unsigned(NBITS_WORD - 1 downto 0) := ";
    to_unsigned(0, NBITS_WORD); -- unsigned value
    variable seed_1 : positive := INITIAL_SEED_1; -- random seed 1
    variable seed_2 : positive := INITIAL_SEED_2; -- random seed 2
  begin
    wait for T_WAIT;
    -- fill RAM with pseudo random pattern
    seed_1 := INITIAL_SEED_1;
    seed_2 := INITIAL_SEED_2;
    rw_select <= '1';
    for i in 0 to SIZE - 1 loop
      wait until falling_edge(clock);
      uniform(seed_1, seed_2, value_real);
      value_unsigned := to_unsigned(natural(value_real * real(2**NBITS_WORD) - 0.5), NBITS_WORD);
      address <= std_logic_vector(to_unsigned(i, NBITS_ADDRESS));
      data_in <= std_logic_vector(value_unsigned);
      enable <= '1';
end loop; -- i

wait until falling_edge(clock);
enable <= '0';

-- read from RAM and compare it with the expected pseudo random pattern
seed_1 := INITIAL_SEED_1;
seed_2 := INITIAL_SEED_2;
rw_select <= '0';
wait until rising_edge(clock);

for i in 0 to SIZE - 1 loop
uniform(seed_1, seed_2, value_real);
value_unsigned := to_unsigned(natural(value_real * real(2**NBITS_WORD) - 0.5), NBITS_WORD);
address <= std_logic_vector(to_unsigned(i, NBITS_ADDRESS));
enable <= '1';
wait until rising_edge(clock); -- read request processed by single port J RAM
wait until rising_edge(clock); -- correct value at data_out
assert unsigned(data_out) = value_unsigned
report "Read," & integer'image(to_integer(unsigned(data_out))) & " instead of expected," & integer'image(to_integer(value_unsigned)) & " from address," & integer'image(i) & " severity error;"
end loop; -- i

wait; -- forever
end process testpattern_generator;

end architecture testbench;

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

B. Histogram Block

Listing 3: VHDL source code of the histogram top block

----------------------------------------------------------------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity histogram_top is
  generic
    WIDTH : positive := 320; -- width of frame
    HEIGHT : positive := 240; -- height of frame
    NBITS_PIXEL : positive := 8; -- bits per pixel
    NBITS_HISTOGRAM : positive := 18); -- bits for each histogram entry
  port
    (clock : in std_logic; -- clock
    reset : in std_logic; -- reset
    frame_start : in std_logic; -- frame start
    video_clock : in std_logic; -- video clock
    video_in : in std_logic_vector(NBITS_PIXEL - 1 downto 0); -- digitised video signal
    done : out std_logic; -- histogram done
    read_request : in std_logic; -- read request for the histogram data
    address : in std_logic_vector(NBITS_PIXEL - 1 downto 0); -- address to read from
    data_out : out std_logic_vector(NBITS_HISTOGRAM - 1 downto 0)); -- output for histogram data
end entity histogram_top;

architecture structural of histogram_top is

begin
  histogram_1 : entity work.histogram
    generic map
      (WIDTH => WIDTH,
      HEIGHT => HEIGHT,
      NBITS_PIXEL => NBITS_PIXEL,
      NBITS_HISTOGRAM => NBITS_HISTOGRAM)
    port map
      (clock => clock,
      reset => reset,
      frame_start => frame_start,
      video_clock => video_clock,
      video_in => video_in,
      done => done,
      read_request => read_request,
      address => address,
      data_out => data_out,
single_port_ram_1: entity work.single_port_ram
    generic map (    
        NBITS_WORD => NBITS_HISTOGRAM,  
        SIZE => 2**NBITS_PIXEL,  
        NBITS_ADDRESS => NBITS_PIXEL)
    port map (    
        clock => clock,  
        enable => ram_enable,  
        reset => ram_reset,  
        rw_select => ram_rw_select,  
        address => ram_address,  
        data_in => ram_data_out,  
        data_out => ram_data_in);

end architecture structural;

Listing 4: Synopsys DC Tcl script for the histogram calculation block
set NBITS_PIXEL 8
set NBITS_HISTOGRAM 18
set CLK_NAME clock
# all time values are in ns
set CLK_PERIOD 10;
set INPUT_DELAY 2;
set OUTPUT_DELAY 2;
set OPERATING_COND WORST-IND

# Flags that drive the script behavior (can be changed)
#
# DB_FORMAT (db | ddc)
# if db, use the old DB format to store design information
# if ddc, use the new XG format to store design information (recommended)
# SHARE_RESOURCES (0 | 1)
# if 1, force the tool to share resources as much as possible
# if 0, no resource sharing
# COMPILE_SIMPLE (0 | 1)
# if 1, only do a single compile with default arguments
# if 0, do a two-step compilation with ungrouping in between
# OPT (string)
# can be used to generate different mapped file names

set DB_FORMAT ddc
set SHARE_RESOURCES 1
set COMPILE_SIMPLE 1
set OPT "_clock10ns_share"

# File names
#
set SOURCE_FILE_NAME ${VHDL_ENTITY}
set ROOT_FILE_NAME

$(VHDL_ENTITY)_WIDTH${WIDTH}_HEIGHT${HEIGHT}_NBITS_PIXEL${NBITS_PIXEL}_NBITS_HISTOGRAM${NBITS_HISTOGRAM}

set VHDL_SOURCE_FILE_NAME 

$(SOURCE_FILE_NAME).vhd
set ELAB_FILE_NAME $(ROOT_FILE_NAME)$(OPT)_elab
set MAPPED_FILE_NAME $(ROOT_FILE_NAME)$(OPT)_mapped
set DB_ELAB_FILE_NAME $(ELAB_FILE_NAME).$DB_FORMAT
set DB_MAPPED_FILE_NAME $(MAPPED_FILE_NAME).$DB_FORMAT
set VHDL_NETLIST_FILE_NAME $(MAPPED_FILE_NAME).vhd
set VLOG_NETLIST_FILE_NAME $(MAPPED_FILE_NAME).v
set SDF_FILE_NAME $(MAPPED_FILE_NAME).adf
set SDC_FILE_NAME $(MAPPED_FILE_NAME).sdc
set RPT_AREA_FILE_NAME $(MAPPED_FILE_NAME)_area.rpt
set RPT_TIMING_FILE_NAME $(MAPPED_FILE_NAME)_timing.rpt
set RPT_RESOURCES_FILE_NAME $(MAPPED_FILE_NAME)_resources.rpt
set RPT_REFERENCES_FILE_NAME $(MAPPED_FILE_NAME)_references.rpt
set RPT_CELLS_FILE_NAME $(MAPPED_FILE_NAME)_cells.rpt

# Absolute paths
#
set VHDL_SOURCE_FILE $(PROJECT_DIR)/HDL/RTL/${VHDL_SOURCE_FILE_NAME}
set VHDL_NETLIST_FILE $(PROJECT_DIR)/HDL/GATE/${VHDL_NETLIST_FILE_NAME}
set VLOG_NETLIST_FILE $(PROJECT_DIR)/HDL/GATE/${VLOG_NETLIST_FILE_NAME}
set DB_ELAB_FILE $(PROJECT_DIR)/SYN/DB/${DB_ELAB_FILE_NAME}
set DB_MAPPED_FILE $(PROJECT_DIR)/SYN/DB/${DB_MAPPED_FILE_NAME}
set SDF_FILE $(PROJECT_DIR)/SYN/TIM/${SDF_FILE_NAME}
set SDC_FILE $(PROJECT_DIR)/SYN/SDC/${SDC_FILE_NAME}
set RPT_AREA_FILE $(PROJECT_DIR)/SYN/RPT/${RPT_AREA_FILE_NAME}
set RPT_TIMING_FILE $(PROJECT_DIR)/SYN/RPT/${RPT_TIMING_FILE_NAME}
set RPT_RESOURCES_FILE \ ${PROJECT_DIR}/SYN/RPT/${RPT_RESOURCES_FILE_NAME}
set RPT_REFERENCES_FILE \ ${PROJECT_DIR}/SYN/RPT/${RPT_REFERENCES_FILE_NAME}
set RPT_CELLS_FILE \ ${PROJECT_DIR}/SYN/RPT/${RPT_CELLS_FILE_NAME}

# Analyze RTL source
analyze -format vhdl -lib WORK $VHDL_SOURCE_FILE

# Elaborate design
elaborate $VHDL_ENTITY \ -arch $VHDL_ARCH \ -lib DEFAULT -update \ -param "WIDTH=${WIDTH}, HEIGHT=${HEIGHT}, NBITS_PIXEL=${NBITS_PIXEL}, NBITS_HISTOGRAM=${NBITS_HISTOGRAM}"

check_design

# Define environment
set_operating_conditions -library c35_CORELIB $OPERATING_COND

# Define constraints
create_clock -name $CLK_NAME -period $CLK_PERIOD [get_ports $CLK_NAME]
set_input_delay $INPUT_DELAY -clock $CLK_NAME [list [all_inputs]]
set_output_delay $OUTPUT_DELAY -clock $CLK_NAME [list [all_outputs]]

set_max_area 0

# Use only plain DFF cells
set_dont_use [list c35_CORELIB.db:c35_CORELIB/DFE* c35_CORELIB.db:c35_CORELIB/DFS* c35_CORELIB.db:c35_CORELIB/TF* c35_CORELIB.db:c35_CORELIB/JK*]

set_fix_multiple_port_nets -all

# Set resource allocation and implementation
if { $SHARE_RESOURCES } {
    set_resource_allocation area_only
} else {
    set_resource_allocation none
}

# Save elaborated design and constraints
write -hierarchy -format $DB_FORMAT -output $DB_ELAB_FILE

# Map design to gates
if { $COMPILE_SIMPLE } {
```bash
compile
} else {
    compile -map_effort medium -area_effort medium
    ungroup -all -flatten
    compile -incremental -map_effort high
}

# Save mapped design
write -hierarchy -format $DB_FORMAT -output $DB_MAPPED_FILE

# Generate reports
report_area -nosplit > $RPT_AREA_FILE
report_timing -path full \ 
    -delay max \ 
    -nworst 1 \ 
    -max_paths 1 \ 
    -significant_digits 2 \ 
    -nosplit \ 
    -sort_by_group \ 
    > $RPT_TIMING_FILE
report_resources -nosplit -hierarchy > $RPT_RESOURCES_FILE
report_reference -nosplit > $RPT_REFERENCES_FILE
report_cell -nosplit > $RPT_CELLS_FILE

# Generate VHDL netlist
change_names -rule vhdl -hierarchy -verbose
write -format vhdl -hierarchy -output $VHDL_NETLIST_FILE

# Generate SDF data
write_sdf -version 2.1 $SDF_FILE

# Generate Verilog netlist
# The design is reloaded from scratch to avoid potential naming problems
# when using the netlist for placement and routing
remove_design -all
read_file -format $DB_FORMAT $DB_MAPPED_FILE
change_names -rule verilog -hierarchy -verbose
write -format verilog -hierarchy -output $VLOG_NETLIST_FILE

# Save system constraints
write_sdc -nosplit $SDC_FILE
```

Listing 5: Cadence First Encounter IO assignment file for the histogram calculation block
# Syntax:
# Pin: <pin-name> <orientation>
# where <orientation> may be either one of:
# n north (top)
# e east (right)
# s south (bottom)
# w west (left)

Offset: 16.0
Pin: clock w
Pin: reset w
Pin: frame_start w
Pin: video_clock w
Pin: video_in[7] w
Pin: video_in[6] w
Pin: video_in[5] w
Pin: video_in[3] w
Pin: video_in[2] w
Pin: video_in[1] w
Pin: video_in[0] w
Pin: read_request w
Pin: address[7] w
Pin: address[6] w
Pin: address[5] w
Pin: address[4] w
Pin: address[3] w
Pin: address[2] w
Pin: address[1] w
Offset: 211.0
Pin: address[0] w

Offset: 16.0
Pin: done e
Pin: data_out[17] e
Pin: data_out[16] e
Pin: data_out[14] e
Pin: data_out[12] e
Pin: data_out[10] e
Pin: data_out[9] e
Pin: data_out[8] e
Pin: data_out[1] e
Offset: 211.0
Pin: data_out[0] e

Offset: 16.0
Pin: ram_reset n
Pin: ram_enable n
Pin: ram_rw_select n
Pin: ram_data_in[17] n
Pin: ram_data_in[16] n
Pin: ram_data_in[14] n
Pin: ram_data_in[12] n
Pin: ram_data_in[10] n
Pin: ram_data_in[9] n
Pin: ram_data_in[8] n
Pin: ram_data_in[7] n
Pin: ram_data_in[3] n
Pin: ram_data_in[1] n
Pin: ram_data_in[0] n
Pin: ram_data_out[17] n
Pin: ram_data_out[16] n
Pin: ram_data_out[14] n
Pin: ram_data_out[12] n
Pin: ram_data_out[10] n
Pin: ram_data_out[9] n
Pin: ram_data_out[8] n
Pin: ram_data_out[1] n
Pin: ram_data_out[0] n

Listing 6: Cadence First Encounter Tcl script for the histogram calculation block

# Title : Cadence Encounter Tcl script for the histogram calculation block
# Project : Image Processing System
# File : histogram_par.tcl
# Author : Torsten Maehne <torsten.maehne@epfl.ch>
# Company : EPFL/STI/IMM/LSM
# Created : 2007-03-25
# Platform : Cadence Encounter 5.2
# Process : AMS 0.35u CMOS (C35), Hit-Kit 3.70
# Depends : histogram_WIDTH320_HEIGHT240_NBITS_PIXELS_NBITS_HISTOGRAM18.v
# histogram_WIDTH320_HEIGHT240_NBITS_PIXELS_NBITS_HISTOGRAM18.conf
# histogram_WIDTH320_HEIGHT240_NBITS_PIXELS_NBITS_HISTOGRAM18.io
# Last update: 2007-03-25
# Copyright (c) 2007 EPFL/STI/IMM/LSM
# Revisions : $Id: histogram_par.tcl 2 2007-03-27 13:10:06Z maehne $
set PROJECT_DIR [pwd]

# Design related information (can be changed)
set DESIGN histogram_WIDTH320_HEIGHT240_NBITS_PIXEL8_NBITS_HISTOGRAM18

set TIM_LIBRARY C35_CORELIB
set TIM_OC_MAX WORST-IND ;# TYPICAL | WORST | WORST-IND
set TIM_OC_MIN BEST-IND ;# TYPICAL | BEST | BEST-IND

# Floorplan settings
set FP_ASPECT_RATIO 1.0
set FP_ROW_DENSITY 0.85 ;# percent
set FP_CORE2IO 16 ;# micron

# Power ring and settings
set PR_WIDTH 4 ;# micron
set PR_SPACING 0.6 ;# micron
set PR_LAYER_TB MET1 ;# top and bottom layer
set PR_LAYER_LR MET2 ;# left and right layer

# Power stripe settings
set ST_NUM_SETS 1 ;# number of sets
set ST_SPACING 1 ;# micron
set ST_LAYER_V $PR_LAYER_LR
set ST_WIDTH 2 ;# micron
set ST_XOFS_R 100 ;# micron
set ST_XOFS_L 100 ;# micron

# Placement settings
set PL_EFFORT -high ;# -low / -medium / -high

# Clock tree synthesis settings
set CTS_BUFFER BUF2
set CTS_INV INV0

# Filler cells
set FILLER_CELLS "FILLRT25 FILLRT10 FILLRT5 FILLRT2 FILLRT1 FILL25 FILL10 FILL5 FILL2 FILL1"

# Flags that drive the script behavior (can be changed)
# ADD_STRIPES (0 | 1)
# if 1, add stripes
# PLACE_TIMING (0 | 1)
# if 1, do a timing driven placement
# CLOCK_TREE (0 | 1)
# if 1, create a clock tree
# CTS_CREATE_SPEC (0 | 1)
# if 1, create a clock tree specification file with default values
# ROUTE_TIMING (0 | 1)
# if 1, do a timing driven routing
# OPT (string)
# can be used to have different generated file names
#
set ADD_STRIPES 1
set PLACE_TIMING 1
set CLOCK_TREE 1
set CTS_CREATE_SPEC 0
set ROUTE_TIMING 1
set OPT "_clock10ns_cts"
#
# File names
#
set CONF_FILE_NAME ${DESIGN}.conf
set IO_FILE_NAME ${DESIGN}.io
set DESIGN_NAME ${DESIGN}${OPT}
set SAVE_DESIGN_FP_NAME ${DESIGN_NAME}-fplan.enc
set SAVE_DESIGN_PR_NAME ${DESIGN_NAME}-printr.enc
set SAVE_DESIGN_PL_NAME ${DESIGN_NAME}-placed.enc
set SAVE_DESIGN_PF_NAME ${DESIGN_NAME}-placed_filled.enc
set SAVE_DESIGN_CT_NAME ${DESIGN_NAME}-cts.enc
set SAVE_DESIGN_RO_NAME ${DESIGN_NAME}-routed.enc
set TIM_RCDB_NAME ${DESIGN_NAME}.rcdb
set SDF_FILE_NAME ${DESIGN_NAME}-routed.sdf
set SPEF_FILE_NAME ${DESIGN_NAME}-routed.spef
set RPT_CHECK_TA_NAME ${DESIGN_NAME}-checkta.rpt
set RPT_REPORT_TA_NAME ${DESIGN_NAME}-ta.rpt
set RPT_SLACK_NAME ${DESIGN_NAME}-slack.rpt
set RPT_GATE_COUNT_NAME ${DESIGN_NAME}-gate_count.rpt
set RPT_NOTCH_NAME ${DESIGN_NAME}-notch.rpt
set RPT_CONN_NAME ${DESIGN_NAME}-conn.rpt
set RPT_GEOM_NAME ${DESIGN_NAME}-geom.rpt
set RPT_DENSITY_NAME ${DESIGN_NAME}-density.rpt
set VLOG_NETLIST_SIM_NAME ${DESIGN_NAME}-routed.v
set VLOG_NETLIST_LVS_NAME ${DESIGN_NAME}-routed_lvs.v
set CTS_SPEC_NAME ${DESIGN_NAME}-spec.cts
set CTS_RGUIDE_NAME ${DESIGN_NAME}-guide.cts
set CTS_RPT_NAME ${DESIGN_NAME}-cts.rpt
set GDS_FILE_NAME ${DESIGN_NAME}.gds
#
# Absolute paths
#
set CONF_FILE ${PROJECT_DIR}/PAR/CONF/${CONF_FILE_NAME}
set IO_FILE ${PROJECT_DIR}/PAR/CONF/${IO_FILE_NAME}
set SAVE_DESIGN_FP_FILE ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_FP_NAME}
set SAVE_DESIGN_PR_FILE ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_PR_NAME}
set SAVE_DESIGN_PL_FILE ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_PL_NAME}
set SAVE_DESIGN_PF_FILE ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_PF_NAME}
set SAVE_DESIGN_CT_FILE ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_CT_NAME}
set SAVE_DESIGN_RO_FILE ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_RO_NAME}
set SDF_FILE ${PROJECT_DIR}/PAR/TIM/${SDF_FILE_NAME}
set SPEF_FILE ${PROJECT_DIR}/PAR/TIM/${SPEF_FILE_NAME}
set TIM_RCDB_FILE ${PROJECT_DIR}/PAR/TIM/${TIM_RCDB_NAME}
set RPT_CHECK_TA_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_CHECK_TA_NAME}
set RPT_REPORT_TA_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_REPORT_TA_NAME}
set RPT_SLACK_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_SLACK_NAME}
set RPT_GATE_COUNT_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_GATE_COUNT_NAME}
set RPT_NOTCH_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_NOTCH_NAME}
set RPT_CONN_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_CONN_NAME}
set RPT_GEOM_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_GEOM_NAME}
set RPT_DENSITY_FILE ${PROJECT_DIR}/PAR/RPT/${RPT_DENSITY_NAME}
set VLOG_NETLIST_SIM_FILE ${PROJECT_DIR}/HDL/GATE/${VLOG_NETLIST_SIM_NAME}
set VLOG_NETLIST_LVS_FILE ${PROJECT_DIR}/HDL/GATE/${VLOG_NETLIST_LVS_NAME}
set CTS_SPEC_FILE ${PROJECT_DIR}/PAR/CTS/${CTS_SPEC_NAME}
set CTS_RGUIDE_FILE ${PROJECT_DIR}/PAR/CTS/${CTS_RGUIDE_NAME}
set CTS_RPT_FILE ${PROJECT_DIR}/PAR/RPT/${CTS_RPT_NAME}
set GDS_FILE ${PROJECT_DIR}/PAR/DEX/${GDS_FILE_NAME}
set GDS_MAP_FILE ${PROJECT_DIR}/PAR/DEX/gds2.map

# -----------------------------------------------------------------------------
# Procedures
# -----------------------------------------------------------------------------
proc make_clock_tree create_spec {
  global CTS_BUFFER CTS_INV CTS_SPEC_FILE CTS_RGUIDE_FILE CTS_RPT_FILE
  if { $create_spec || ![file exists $CTS_SPEC_FILE] } {
    createClockTreeSpec \ 
    -bufFootprint $CTS_BUFFER \ 
    -invFootprint $CTS_INV \ 
    -output $CTS_SPEC_FILE
  }
  specifyClockTree -clkfile $CTS_SPEC_FILE
  ckSynthesis \ 
  -rguide $CTS_RGUIDE_FILE \ 
  -report $CTS_RPT_FILE
  optDesign -postCTS -setup -drv -outDir PAR/RPT
}

# -----------------------------------------------------------------------------
# Load configuration file
# -----------------------------------------------------------------------------
loadConfig $CONF_FILE 0
commitConfig

# -----------------------------------------------------------------------------
# Load IO file
# -----------------------------------------------------------------------------
loadIoFile $IO_FILE

# -----------------------------------------------------------------------------
# Set operating conditions
# -----------------------------------------------------------------------------
setOpCond \ 
  -maxLibrary $TIM_LIBRARY -max $TIM_OC_MAX \ 
  -minLibrary $TIM_LIBRARY -min $TIM_OC_MIN

# -----------------------------------------------------------------------------
# Set user grids
# -----------------------------------------------------------------------------
setPreference ConstraintUserXGrid 0.1
setPreference ConstraintUserYGrid 0.1
setPreference ConstraintUserXOffset 0.1
setPreference ConstraintUserYOffset 0.1
setPreference SnapAllCorners 1
setPreference BlockSnapRule 2
# Define global Power nets - make global connections

clearGlobalNets
globalNetConnect vdd! -type ppin -pin vdd! -inst * -module {} -verbose
globalNetConnect gnd! -type ppin -pin gnd! -inst * -module {} -verbose
# globalNetConnect vdd3o! -type ppin -pin vdd3o! -inst * -module {} -verbose
# globalNetConnect vdd3r1! -type ppin -pin vdd3r1! -inst * -module {} -verbose
# globalNetConnect vdd3r2! -type ppin -pin vdd3r2! -inst * -module {} -verbose
# globalNetConnect gnd3o! -type ppin -pin gnd3o! -inst * -module {} -verbose
# globalNetConnect gnd3r! -type ppin -pin gnd3r! -inst * -module {} -verbose

# Initialize floorplan

floorPlan -r $FP_ASPECT_RATIO \ $FP_ROW_DENSITY \ $FP_CORE2IO $FP_CORE2IO $FP_CORE2IO $FP_CORE2IO fit saveDesign $SAVE_DESIGN_FP_FILE

# Create and route power rings and power stripes

addRing \ -around core \ -nets { gnd! vdd! } \ -width_bottom $PR_WIDTH -width_top $PR_WIDTH \ -width_left $PR_WIDTH -width_right $PR_WIDTH \ -spacing_bottom $PR_SPACING -spacing_top $PR_SPACING \ -spacing_left $PR_SPACING -spacing_right $PR_SPACING \ -layer_bottom $PR_LAYER_TB -layer_top $PR_LAYER_TB \ -layer_left $PR_LAYER_LR -layer_right $PR_LAYER_LR \ -center 1 \ -tl 1 -tr 1 -bl 1 -br 1 -lt 1 -lb 1 -rt 1 -rb 1 \ -stacked_via_bottom_layer MET1 -stacked_via_top_layer MET4 \ -threshold 0.7
if { $ADD_STRIPES } {
  addStripe \ -nets { gnd! vdd! } \ -number_of_sets $ST_NUM_SETS \ -spacing $ST_SPACING \ -layer $ST_LAYER_V \ -width $ST_WIDTH \ -xleft_offset $ST_XOFS_L
}
sroute \ -jogControl { preferWithChanges differentLayer } \ -nets { gnd! vdd! }
saveDesign $SAVE_DESIGN_PR_FILE

# Add CAP cells

addEndCap -preCap ENDCAPL -postCap ENDCAPR -prefix ENDCAP

# Core cell placement

if { $PLACE_TIMING } {
  amoebaPlace $PL_EFFORT -timingdriven
} else {
  amoebaPlace $PL_EFFORT
}

setDrawMode place
saveDesign $SAVE_DESIGN_PL_FILE

# Create clock tree (optional)
#
if { $CLOCK_TREE } {
  make_clock_tree $CTS_CREATE_SPEC
  saveDesign $SAVE_DESIGN_CT_FILE
}

# Route design (Nanoroute)
#
if { $ROUTE_TIMING } {
  setNanoRouteMode -quiet -timingEngine CTE
  setNanoRouteMode -quiet -routeWithTimingDriven true
  setNanoRouteMode -quiet -routeTdrEffort 0
}
globalDetailRoute
optDesign -postRoute -setup -drv -outDir PAR/RPT
saveDesign $SAVE_DESIGN_RO_FILE
setDrawMode place

# Add filler cells
#
addFiller -cell $FILLER_CELLS -prefix FILLER
saveDesign $SAVE_DESIGN_PF_FILE

# Verifications
#
fillNotch -report $RPT_NOTCH_FILE
verifyConnectivity \
  -type all \n  -error 1000 \n  -warning 50 \n  -report $RPT_CONN_FILE
verifyGeometry \n  -allowSameCellViols \n  -allowRoutingBkgPinOverlap \n  -allowRoutingCellBkgOverlap \n  -report $RPT_GEOM_FILE
verifyMetalDensity \n  -detailed \n  -report $RPT_DENSITY_FILE

# Extract parasitics
#
setExtractRCMode \n  -detail \n  -rcdb $TIM_RCDB_FILE \n  -relative_c_t 0.01 \n  -total_c_t 5.0 \n  -reduce 5
extractRC
# Generate RC and timing files
rcOut -spef $SPEF_FILE
delayCal -sdf $SDF_FILE

# Generate reports
reportGateCount -outfile $RPT_GATE_COUNT_FILE

# Timings
setCteReport
setAnalysisMode -setup -async -skew -noClockTree -sequentialConstProp
buildTimingGraph
checkTA -verbose > $RPT_CHECK_TA_FILE
reportTA -format { hpin arc cell delay arrival required slew fanout load } -late -max_points 10 -net > $RPT_REPORT_TA_FILE

# Save netlist
saveNetlist -excludeLeafCell $VLOG_NETLIST_SIM_FILE
saveNetlist -physical $VLOG_NETLIST_LVS_FILE

# Save GDS2
streamOut $GDS_FILE -mapFile $GDS_MAP_FILE -libName ADDSUB -structureName $DESIGN_NAME -stripes $ST_NUM_SETS -units 1000 -mode ALL

References


