TÉCNICO LISBOA

Bologna Master Degree in Electronics Engineering

Electronic Integrated Systems

2023/2024 - 1st semester, P1

Pulse Generator

Group 3:

Afonso Oliveira | 108271

Duarte Marques | 96523

Tiago Vaz | 110984

November 5th 2023

Contents

1 Introduction

The objective of this project is to design a digital integrated circuit that generates a pulse waveform, using the CMOS 130 nm technology available from United Microelectronics Corporation (UMC). Firstly, Verilog is used to implement the **pulse generator** Register Transfer Level (RTL) code, in which all signals are active in high-level (H) and inactive in low-level (L). The inputs correspond to the clock signal (`clk`), `INIT` and `PER`. The value of the output (`PULSE`) is determined in each rising edge of the clock according to the values of the other inputs, which determine if the pulse generator starts/restarts or whether a single pulse waveform or periodic signal are generated in the output. In both cases, a pulse is given by the parameters shown in Tab. 1, from which the clock frequency was defined. The waveform starts in low-level during T_I, after which it stays in high-level during T_W and returns to low-level during T_F.

Table 1: Time specifications and period of the clock signal (T_{clk}) for the pulse generator.

Parameter	T_{clk}[µs]	T_I[µs]	T_W[µs]	T_F[µs]
Value	0.1	2.0	0.3	1.5

Once the pulse generator module is designed in Verilog and simulated with a respective testbench, the circuit is implemented with logic gates using the software `genus`. The tool `innovus` is then used to perform place and route (P&R), the final step necessary to transform the pulse generator RTL description into a GDSII layout representation ready for fabrication submission.

2 Verilog RTL code implementation

The first step in the design of the pulse generator digital integrated circuit corresponds to its behaviour description in Verilog, which is included in Listing 1. Firstly, the single-bit inputs of the module are defined, along with the output `PULSE`, whose single-bit value needs to be saved in a register. Additionally, three 6-bit local parameters `n_I`, `n_W` and `n_F` are defined, since the value of `count`, saved in a 6-bit register, will be compared with these values to determine whether `PULSE` should be at L or H. A `localparam` can be used to protect a value from accidental or incorrect redefinition by a user (unlike a `parameter`, this value cannot be modified by parameter redefinition or by a `defparam` statement).

Due to the time values T_I, T_W and T_F shown in Tab. 1 - which characterize the pulse - a clock period of 0.1 µs, which corresponds to a clock frequency of 10 MHz, was defined. This clock period was selected since all these time parameters are multiples of 0.1 µs, therefore facilitating the use of the `localparam` values (which correspond to integer values) to define the pulse waveform by comparing them with the value of the counter. Additionally, the fact that the clock period is lower than those time parameters guarantees a proper functioning of the circuit; this higher clock frequency also contributes to a faster response. Taking into account these time values, in each pulse the output should remain at L during 20 clock cycles, change to H for 3 clock cycles, and then returning to L for another 15 cycles - these are registered by using the internal variable `count[5:0]`.

The other internal signal corresponds to aux_PER, which was created to store the value of PER when INIT changes to H (i.e., its bit is '1'). In this situation, the value of PER is put at zero, by choice (alternatively, the previously stored value in the register could have simply been maintained). The counter is also set to zero, in order to restart a proper single pulse or period signal when INIT changes back to L (i.e., its bit is '0'). In this case, if the value of PER (now registered in aux_PER) was '0', a single pulse waveform is generated. On the other hand, when the value of PER was '1', a periodic signal is continuously generated until INIT changes back to H.

```
`timescale 10ns / 1ps

////////////////////////////////////////////////////////////////////////////////
//  Authors: Afonso Oliveira (108271), Duarte Marques (96523), Tiago Vaz (110984)
//  Module Name: pulsegen
//  Description: Pulse generator
////////////////////////////////////////////////////////////////////////////////

module pulsegen(
  input clk,
  input INIT,
  input PER,
  output reg PULSE
  );

  localparam n_I=6'b010100, n_W=6'b010111, n_F=6'b100110; // Number of clock cycles
    (20, 23 and 38, respectively)
  reg[5:0] count; // Counter (maximum value 38, requires 6 bits)
  reg aux_PER; // Store value of PER when INIT==1

  always @(posedge clk) begin

    // Start/restart
    if (INIT == 1) begin
      count <= 6'b0;
      aux_PER <= PER;
      PULSE <= 0;
    end

    else begin
      if (INIT == 0) begin
        // Single pulse waveform
        if ((aux_PER == 0) && (count < n_F)) begin
          if ((count >= 6'b0) && (count < n_I))
            PULSE <= 0;
          else if ((count >= n_I) && (count < n_W))
            PULSE <= 1;
          else if ((count >= n_W) && (count < n_F))
            PULSE <= 0;
```

```
          count <= count+6'b1; // Increase value of the counter
        end

        // Periodic signal
        if (aux_PER == 1) begin
          if (count < (n_F-6'b1)) begin
            if ((count >= 6'b0) && (count < n_I))
              PULSE <= 0;
            else if ((count >= n_I) && (count < n_W))
              PULSE <= 1;
            else if ((count >= n_W) && (count < (n_F-6'b1)))
              PULSE <= 0;
            count <= count+6'b1; // Increase value of the counter
          end
          else
            count <= 6'b0; // PULSE remains at 0 in this iteration
        end

      end
    end
  end
endmodule
```

Listing 1: Verilog RTL code for the pulse generator.

In order to test if the pulse generator is working properly, the testbench shown in Listing 2 was created. In this code, a timescale with a time unit of 10 ns was defined, therefore 10 time units must pass in each clock cycle (since the clock period is 100 ns), which means that the value of `clk` is changed every 5 time units. After defining the inputs and output in the circuit, the values of the former are all initialized with '0'. Then, different situations are taken into account. Firstly, a single pulse waveform is created by setting `INIT` to '1' and then returning its value to '0'. Since each pulse lasts for 3.8 µs, it is necessary to wait more than 380 time units in order to fully observe the single pulse. Then, having `PER` changed to H, a similar change in `INIT` is made in order to show a periodic signal in the output, for 2000 time units, which means that 5 complete pulses should be seen. During this time, the value of `PER` is temporarily changed to prove that only when `INIT` is at '1' this change affects the pulse generator. Secondly, a change from the period signal to a single pulse is tested, with `PER` at '0' and by changing `INIT`. Finally, an incomplete single pulse is stopped midway while at H - by changing `INIT` to '1' after only 215 time units - and the simulation is finished after a complete single pulse is generated once again.

```
`timescale 10ns / 1ps

module pulsegen_tb;

  reg clk, INIT, PER; // Inputs
  wire PULSE; // Output
```

```
  pulsegen pulsegen (
      .clk(clk),
      .INIT(INIT),
      .PER(PER),
      .PULSE(PULSE)
      );

  initial begin
    clk=0;
    INIT=0;
    PER=0;
  end

  // Clock frequency 10MHz (0.1us period)
  always
    #5 clk = !clk;

  initial begin
    // Test single pulse waveform followed by periodic signal
    #100 INIT=1;
    #20 INIT=0;
    #400 PER=1;
    #20 INIT=1;
    #20 INIT=0;
    #900 PER=0; // Test change in PER without INIT=1
    #100 PER=1;
    #1000

    // Test single pulse waveform after periodic signal
    PER=0;
    #20 INIT=1;
    #20 INIT=0;
    #400

    // Stop single pulse midway then finish with complete single pulse
    INIT=1;
    #20 INIT=0;
    #215 INIT=1;
    #20 INIT=0;
    #400
    $finish;
  end
endmodule
```

Listing 2: Verilog RTL code for the pulse generator testbench.

By using Xcelium (an HDL functional simulator) and the Verilog codes shown in Listings 1 and 2, the simulation results shown in Figs. 1-5 prove that the desired behavior was achieved. In each case,

the cursors `Baseline` and `TimeA` delimit the pulse being taken into account, apart from Fig. 5, where they delimit the entire simulation time. In Fig. 1, it can be observed that the value of `count[5:0]` initially remains undefined, since `INIT` is initialized at '0'. Because of that, `aux_PER` also remains undefined, which means that no value is given to `PULSE`, according to the code presented in Listing 1. In the first rising edge of the clock right after `INIT` goes to '1', all these signals are given the value zero. Once `INIT` goes to '0' again, the single pulse is generated in the output. The output remains at L for 20 clock cycles, then remains at H when `count[5:0]` is 21, 22 or 23 (three clock cycles), finishing off at L until `count[5:0]` is 38. After that, the value of the counter remains fixed, and `PULSE` stays at L, because the single pulse had already been successfully generated.

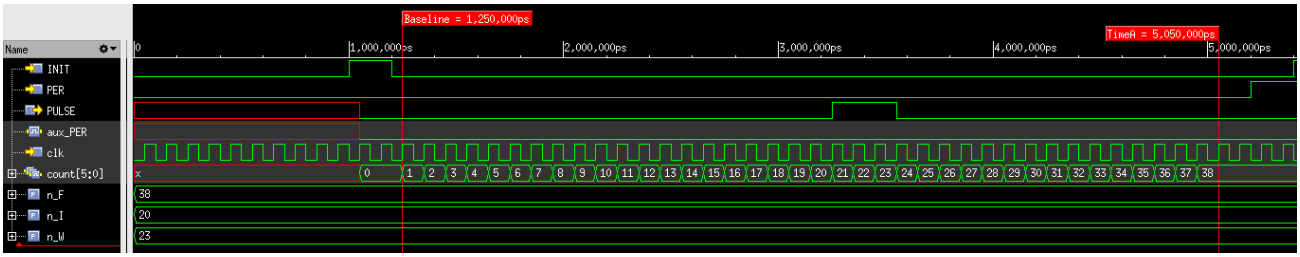

Figure 1: Initial simulation results - first single pulse waveform.

In the next two waveforms shown below, the first and second pulses in the periodic signal case are both shown, in order to specify the slight difference in the value of `count[5:0]` in each case. In both cases, the counter is initially at zero when the respective block in the code is accessed (since the counter is set to zero when `INIT` is at H), and in very first rising edge of the clock a value of '1' is given to `PULSE` and the counter increases by one. However, in the very first pulse, the value of `count[5:0]` was previously at 38 - as shown in Fig. 2 - since the single pulse depicted in Fig. 1 had been previously generated in the output, while 37 is the highest value in the counter seen for all other cases (as shown in Fig. 3). In each pulse in the periodic signal, `PULSE` is at L from 1 to 20 in `count[5:0]`, at H from 21 to 23, and again at L from 24 to 0 (included). This corresponds to the main difference to the single pulse mode, where the counter went from 24 to 38. Now, as described in the code of Listing 1, there needs to be a clock cycle where the counter is reset to zero for the next pulse; in this iteration, the pulse is still in its final stage (i.e., corresponding to T_F), with `PULSE` remaining at zero, thus completing the desired 15 clock cycles for this final part of each pulse.

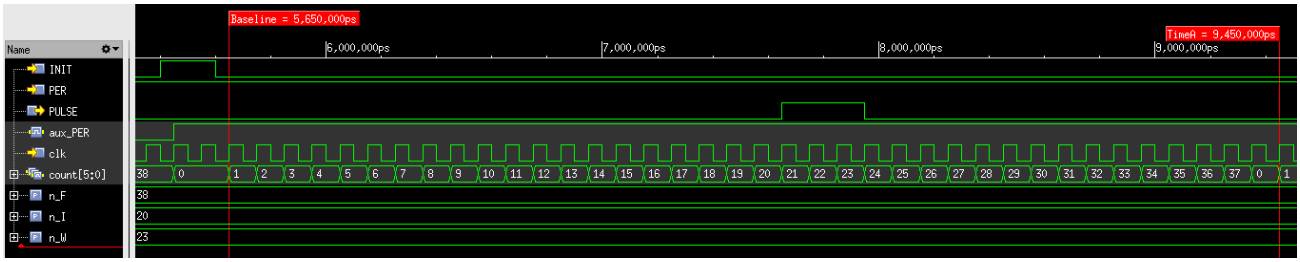

Figure 2: Initial simulation results - first pulse in the period signal, after the single pulse.

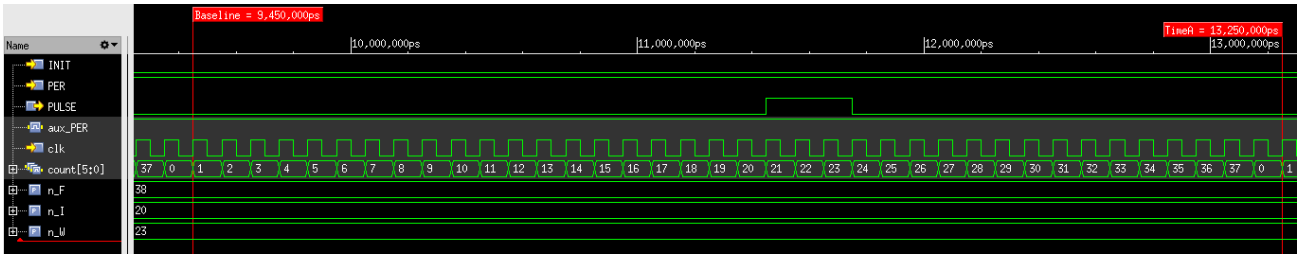

Figure 3: Initial simulation results - second pulse in the period signal, after the single pulse.

In Fig. 4, the periodic signal is stopped (in this case, when the counter was at 12) and a single pulse is successfully generated in the output. The full waveform, corresponding to a total simulation time of 36 550 ns, is shown in Fig. 5, where a single pulse stopped midway followed by a final complete single pulse can also be seen, as expected.

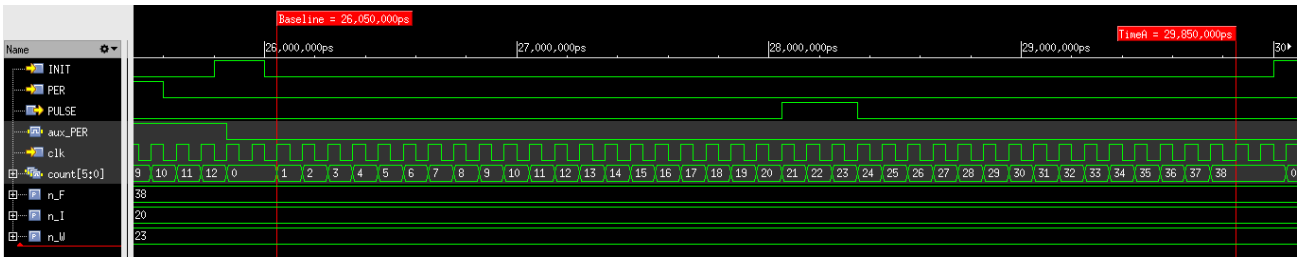

Figure 4: Initial simulation results - change from periodic signal to single pulse.

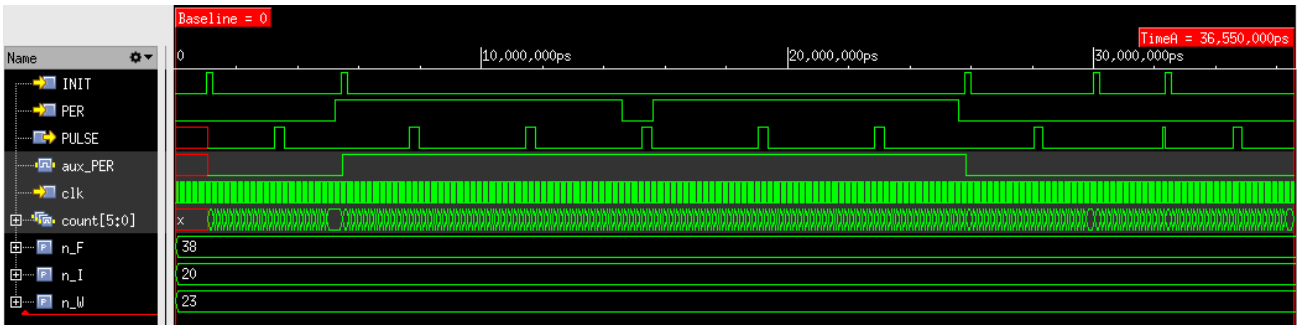

Figure 5: Initial simulation results - full waveform.

3 Verilog gate-level code with technology standard cells

Verilog was used to design a state machine that is synthesizable, i.e., can be implemented with logic gates. By using the software `genus`, a script was therefore ran in order to perform synthesis of the pulse generator module, previously implemented with the Verilog RTL code. For this purpose, it is necessary for the program to read the library files which describe the technology from which the gates are used to generate the standard cells netlist. In this script, it was also necessary to specify the 100 ns clock period.

After synthesis was complete, some commands were used to check the performance of the circuit,

having the reports shown in Listings 3-7 been obtained. The command `report_timing` performs timing analysis on the specified timing paths of the synthesized design. By default, the tool reports the timing path with the worst calculated slack (i.e., the difference between the time available for a signal to travel through a path and the time required for the signal to travel through that path). The information in Listing 3 includes the latency (time taken by a system to produce the output after the input is applied) at the starting point (`Src Latency`) and in the network between the starting and ending points (`Net Latency`). The time when the data signal arrives at the endpoint (`Arrival`) is also included in the report; the setup time requirement for the path corresponds to how early the data must be available before the clock edge. Since the time available to meet timing requirements (`Required Time`) is (much) higher than the time for the data path portion, a positive slack is obtained, and a significant time margin to meet timing requirements exists.

The final part of this report lists various sections along the path, including the cells in question, fan-out (maximum number of digital inputs that the output of a single logic gate can feed without disrupting the circuitry's operations), and associated delays in the correspondent path.

```
Path 1: MET (98711 ps) Setup Check with Pin count_reg[1]/CK->D
          Group: clk
     Startpoint: (R) count_reg[3]/CK
          Clock: (R) clk
       Endpoint: (R) count_reg[1]/D
          Clock: (R) clk

                     Capture       Launch
        Clock Edge:+  100000            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=  100000            0

             Setup:-     134
     Required Time:=   99866
      Launch Clock:-       0
         Data Path:-    1155
             Slack:=   98711

#-------------------------------------------------------------------------------
# TimingPoint Flags Arc Edge Cell Fanout Load Trans Delay Arrival InstanceLocation
#                                        (fF)  (ps)  (ps)   (ps)
#-------------------------------------------------------------------------------
count_reg[3]/CK -    -      R (arrival) 8  -     0     -     0       (-,-)
count_reg[3]/Q  -    CK->Q  F QDFFEHD   6  28.5  135   278   278     (-,-)
g1533__3680/O   -    I2->O  R NR2CHD    3  12.1  333   182   461     (-,-)
g1530/O         -    I->O   F INVDHD    4  18.2  160   122   583     (-,-)
g1519__1666/O   -    I2->O  R NR2CHD    3  13.3  362   201   784     (-,-)
g1507__1881/O   -    B2->O  R AO12EHD   3  12.8  168   168   952     (-,-)
g1496__8428/O   -    C1->O  F OAI112BHD 1  3.4   159   103   1055    (-,-)
```

```
g1488__2398/O   -   A1->O R OAI112BHD 1  3.2   198   100   1155    (-,-)
count_reg[1]/D  <<< -     R QDFFEHD   1  -     -     0     1155    (-,-)
#-------------------------------------------------------------------------------
```

Listing 3: Part of the report obtained by running `report_timing`.

The command `report_power` runs power analysis on the current design and reports details of power consumption based on the current operating conditions of the device, and the switching rates of the design. As further described in Sec. 4, the leakage power is the static or standby power consumed by the circuit even when not actively switching, and it has the lowest percentage of the total power. Internal power - whose contribution to the power consumption is the most significant - is the dynamic power consumed when signals transition within the digital logic gates and interconnections, while switching power is the power consumed during transitions in the output. The same relation in terms of percentages was verified in the place and route phase, thus further conclusions are inferred in Sec. 4. It can also be seen that, besides the clock itself, only registers and logic gates are implemented in the synthesized design, since only they contribute to power consumption.

```
------------------------------------------------------------------------
  Category          Leakage      Internal     Switching        Total     Row%
------------------------------------------------------------------------
    memory      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00%
  register      1.49695e-07   1.25770e-06   1.89736e-07   1.59713e-06  60.36%
     latch      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00%
     logic      2.21756e-07   2.66220e-06   4.05802e-07   8.93778e-07  33.78%
      bbox      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00%
     clock      0.00000e+00   0.00000e+00   1.55131e-07   1.55131e-07   5.86%
       pad      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00%
        pm      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00%
------------------------------------------------------------------------
  Subtotal      3.71451e-07   1.52392e-06   7.50669e-07   2.64604e-06 100.00%
Percentage           14.04%        57.59%        28.37%       100.00% 100.00%
------------------------------------------------------------------------
```

Listing 4: Part of the report obtained by running `report_power`.

The report shown below contains information about the various logic gates used in the design, including the number of instances, area, and the associated library for each one - which include, for example, AND, NOR or inverter gates. Additionally, it provides a breakdown of gate types and their respective total areas. This information is valuable for assessing the distribution of gate types and their respective areas within the design, which can be crucial for optimizing the design for factors like area efficiency and performance. The total area shown in this report will be useful while defining the floorplan core dimension in the P&R phase.

```
   Gate     Instances    Area                  Library
------------------------------------------------------------------
```

```
AN2EHD              1    6.400    fsc0h_d_generic_core_ss1p08v125c
AO12EHD             1    8.960    fsc0h_d_generic_core_ss1p08v125c
AO222EHD            1   15.360    fsc0h_d_generic_core_ss1p08v125c
AO22CHD             1   11.520    fsc0h_d_generic_core_ss1p08v125c
AOI12CHD            1    8.960    fsc0h_d_generic_core_ss1p08v125c
AOI22BHD            1    8.960    fsc0h_d_generic_core_ss1p08v125c
DFECHD              1   33.280    fsc0h_d_generic_core_ss1p08v125c
INVCKDHD            2    7.680    fsc0h_d_generic_core_ss1p08v125c
INVDHD              5   19.200    fsc0h_d_generic_core_ss1p08v125c
MOAI1CHD            1   11.520    fsc0h_d_generic_core_ss1p08v125c
ND2CHD              5   25.600    fsc0h_d_generic_core_ss1p08v125c
ND2DHD              2   10.240    fsc0h_d_generic_core_ss1p08v125c
ND3CHD              1    7.680    fsc0h_d_generic_core_ss1p08v125c
NR2CHD              9   46.080    fsc0h_d_generic_core_ss1p08v125c
OAI112BHD           6   53.760    fsc0h_d_generic_core_ss1p08v125c
OAI12CHD            1    7.680    fsc0h_d_generic_core_ss1p08v125c
OAI13BHD            2   17.920    fsc0h_d_generic_core_ss1p08v125c
OR2B1CHD            2   12.800    fsc0h_d_generic_core_ss1p08v125c
OR2EHD              1    6.400    fsc0h_d_generic_core_ss1p08v125c
OR3B2CHD            1    8.960    fsc0h_d_generic_core_ss1p08v125c
QDFFEHD             7  179.200    fsc0h_d_generic_core_ss1p08v125c
------------------------------------------------------------------
total              52  508.160

     Type       Instances    Area  Area %
----------------------------------------
sequential              8 212.480   41.8
inverter                7  26.880    5.3
logic                  37 268.800   52.9
physical_cells          0   0.000    0.0
----------------------------------------
total                  52 508.160  100.0
```

Listing 5: Part of the report obtained by running `report_gates`, which itself includes the main information that can be obtained from `report_area`.

The report obtained from `report_port *`, shown in Listing 6, provides information related to the external loads and drivers/slew settings for specific ports. In the first section, there are capacitance values for the input and output ports. Capacitance values are important for calculating signal delays and power consumption, and determining the overall timing of the design. The provided data is crucial for understanding how the external loads, driver characteristics, and slew settings impact the timing and behavior of the signals in the digital design. This report also indicates that small rise and fall slew-rates exist in the output `PULSE`, due to the capacitances in the circuit. These slew-rate values would be even higher if an external wire capacitance had been added in the synthesis script, has done for the 4-bits counter example.

```
External Loads
```

```
--------------
              Pin    Wire  Fanout
 Port    Dir  Cap    Cap    Load
--------------------------------
 clk     in   13.3    0.0
 INIT    in    9.2   10.0
 PER     in    1.6    1.8
 PULSE   out   1.6    3.9

External Driver/Slew
--------------------
 Port    Dir    [...]    Slew Max-Rise    Slew Max-Fall    [...]
-------------------------------------------------------------
 clk     in              0.0              0.0
 INIT    in              0.0              0.0
 PER     in              0.0              0.0
 PULSE   out             85.0             53.3
```

Listing 6: Part of the report obtained by running `report_port *`.

All of the units considered in the previous reports are depicted in Listing 7. Finally, a graphical interface was called with the command `gui_show` in order to show the netlist schematic built by the synthesis procedure. This schematic, shown in Fig. 6, includes all the gates described in Listing 5, along with the ports described in Listing 6, and all the wires connecting the various instances in the circuit. The clock signal, which corresponds to the lowest input on the left-hand side, is directly connected to all the flip-flops in this schematic, something that will be addressed in Sec. 4.

```
LIBRARY                 fsc0h_d_generic_core_ss1p08v125c
Time_unit               :1000ps
Capacitive_load_unit    :1000.0fF
Resistance_unit         :1kohm
Voltage_unit            :1V
Current_unit            :1mA
Power_unit              :nW
```

Listing 7: Report obtained by running `report_units`.

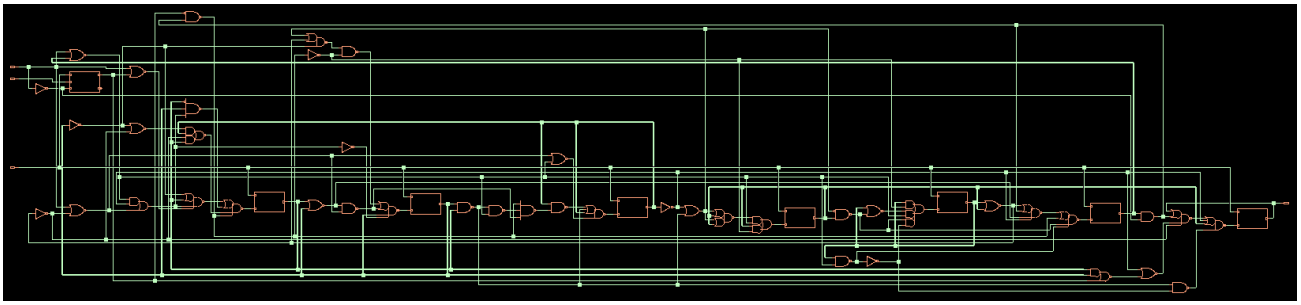

Figure 6: Pulse generator schematic implemented by the Genus synthesizer.

Similarly to the procedure done in Sec. 2, a simulation with the testbench shown in Listing 2 was performed using Xcelium, but this time the synthesized Verilog netlist with standard cells (named `pulsegen_synth.v`) was used. The script ran to test the synthesized code also needed to read the library file of the technology used to implement the circuit. As desired, very similar results were obtained in this case, as shown in Figs. 7-10, whose interpretation is the exact same as before. It is worth noting, however, that in these figures the local parameters `n_I`, `n_W` and `n_F` were not plotted, since in the synthesis procedure they are implemented with numerous wires (since each parameter had 6 bits), which would therefore unnecessarily obscure the main results depicted in these waveforms.

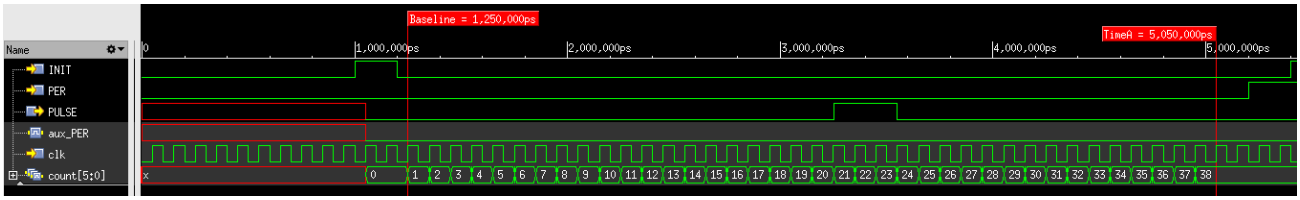

Figure 7: Simulation results after synthesis - first single pulse waveform.

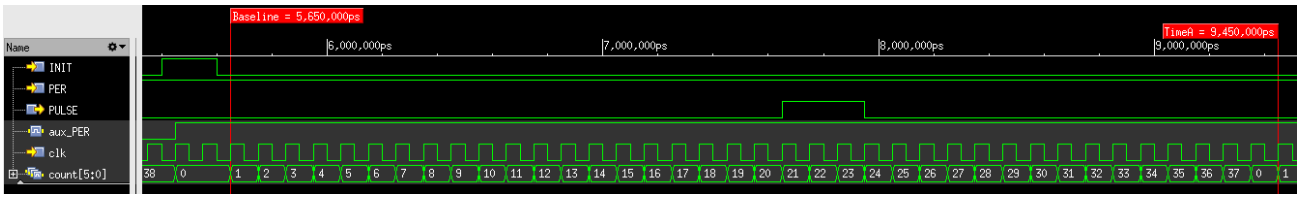

Figure 8: Simulation results after synthesis - first pulse in the periodic signal, after the single pulse.

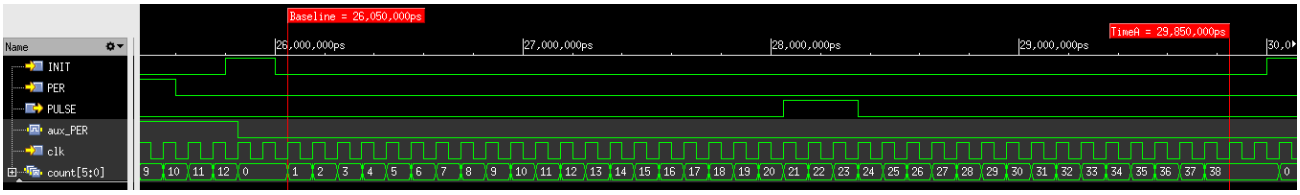

Figure 9: Simulation results after synthesis - change from periodic signal to single pulse.

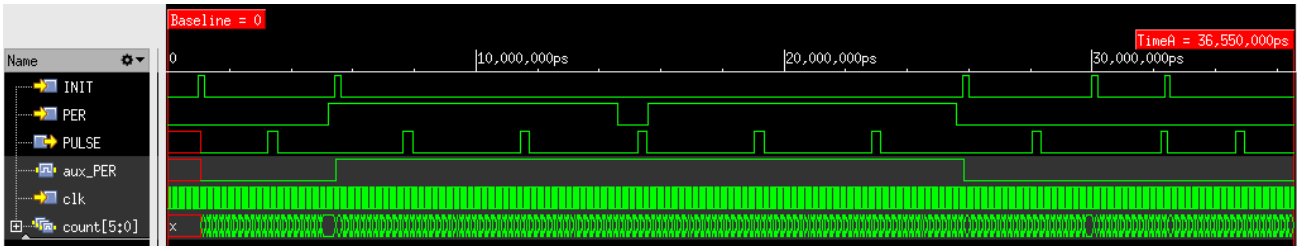

Figure 10: Simulation results after synthesis - full waveform.

The major difference observed after synthesis is the aspect depicted in Fig. 11, in which a change in all the bits of `count[5:0]` occurs after a rising edge of the clock. Because of the delays inherent to

a physical implementation of the pulse generator, the response of the counter occurs 120 ps after the rising edge of the clock, instead of simultaneously. Additionally, the change in the most significant bit in the counter (from L to H) occurs 6 ps before the change in the other bits (from H to L, at the same time - at least considering the time precision of 1 ps in the simulation), which leads to a shortly-lived intermediate state in the counter. However, this does not affect the overall functioning of the pulse generator design, not only due to the very short time in which this occurs, but also because the value of the counter is only changed, in each iteration, after the value of `PULSE` is registered, as depicted in Listing 1. It is also worth noting that, even though there is now also a slew-rate associated with `PULSE`, this phenomenon was not apparent in the waveforms, due to its very low value when compared with the time precision of the simulation.

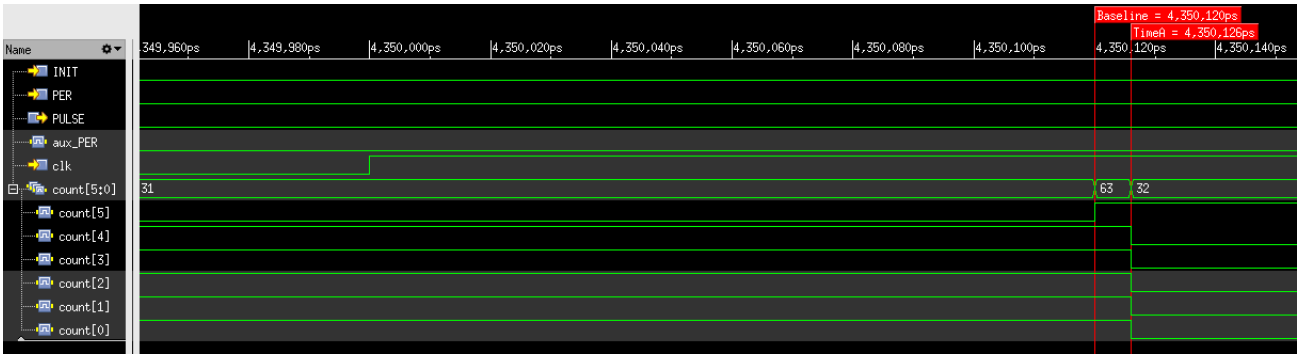

Figure 11: Simulation results after synthesis - detail of the delays between changes in the signals.

4 Place and Route

Having synthesized the Verilog file with a netlist of standard cells (`pulsegen_synth.v`), it is necessary to perform the design flow of **place and route (P&R)**. For this purpose, it is also necessary to utilize the file `pulsegen_synth.sdc` (generated in synthesis), with some constraints information for this task. The Verilog file must be indicated when importing the design, since it includes a netlist of standard/macro cells (gatelist) used in the pulse generator implementation, while also presenting information regarding the inputs, output and wires in the circuit. This phase also requires the technology LEF files which describe aspects such as layout views of standard/macro cells, electrical properties and design rules. The Multi Mode Multi Corner (MMMC) configuration file (`pulsegen.view`) is necessary to include the information regarding the P&R constraints, RC corners (through the `captable.cap` file), library sets and delay corners.

When specifying the floorplan, an important aspect to take into account is the **core dimension**. According to the information in Listing 5, the total area used in the synthesized circuit corresponded to 508.160 μm^2. It is advisable to maintain a core utilization of around 60% to 70%, mostly to make sure that there is enough space for the Clock Tree Synthesis (CTS) phase. In order to determine the width and height of the core, an initial ratio of 65% was considered for the existing standard cells, and the same aspect ratio used in the 4-bits counter example was also taken into account (by choice, in order to simplify this problem). By solving this system of two equations, while also taking into

account that the **core height** should be a multiple of 3.2 µm and the **core width** should be a multiple of 0.4 µm, the final core dimensions of 25.6 µm and 30 µm (respectively) were selected, leading to a total area of 768 μm^2 - this ended up corresponding to a core utilization of 66%. Finally, the core to IO margins (which should be multiples of 0.4 µm) were defined to be 2.4 µm.

The rest of the procedure was done in a similar way to the 4-bits counter. After performing the special route, the command `verifyGeometry` indicated no violations nor warnings in the design. Regarding the **pins**, the inputs `INIT` and `PER` (`metal3` pins) were placed on the left side, with a spacing of 1.2 µm between them, since this spacing should be a multiple of 0.4 µm, in order to facilitate routing. The `clk` pin (in `metal4`) was placed on the bottom and the output pin `PULSE` (in `metal3`) was placed on the right. A depth of 0.52 µm and a width of 0.2 µm were used for all pins. Vertical stripes were not added since these are more useful in designs with larger area.

Being in the Pre-Place design phase, a Setup timing analysis led to a positive **Worst Negative Slack (WNS)** and null **Total Negative Slack (TNS)**, as intended, since WNS corresponds to the difference between the clock period and the delay between a pair of registers; a positive worst case setup time slack means the constraint is met and a negative slack means that the longest path has a path delay longer than the clock period of the circuit. Moreover, TNS is the sum of all negative slacks. For all the stages in the P&R procedure, the expected results were always obtained in the respective timing analyses, therefore optimization was never required; however, the optimization step was always made regardless (in Pre-CTS, Post-CTS and Post-Route).

The **CTS** phase is mandatory in synchronous designs and minimizes the clock skew (since the clock should arrive ideally at the same time at each flip-flop), to minimize setup and hold violations. By using the GUI interface for the clock tree and the Schematic Viewer, the results shown in Figs. 12 and 13 were obtained, respectively.

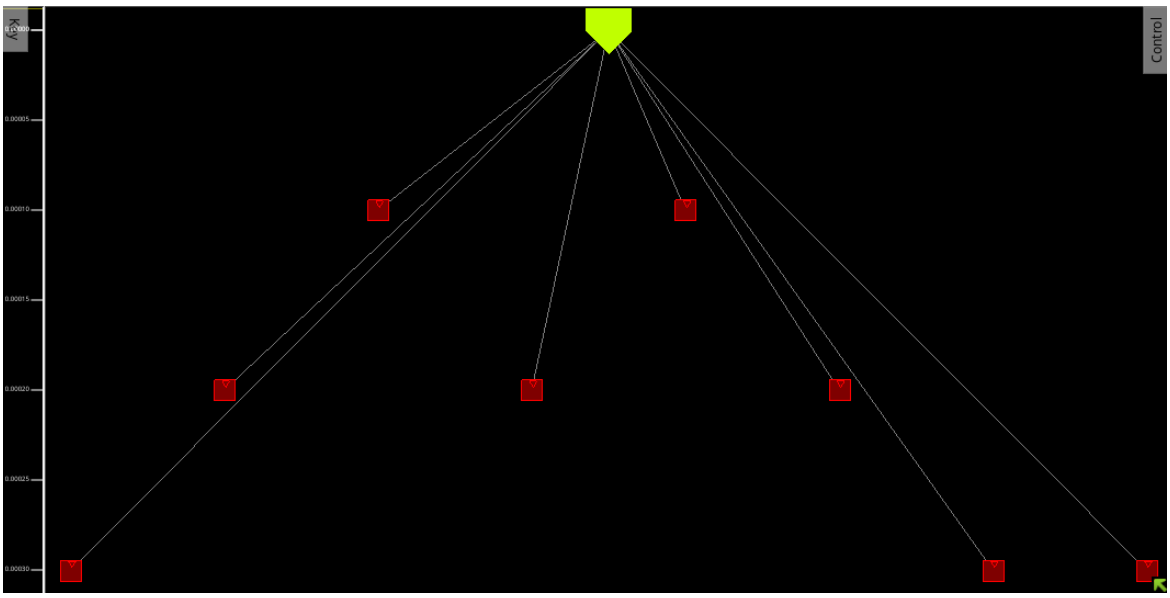

Figure 12: GUI available in the CCOpt Clock Tree Debugger menu with the results after CTS.

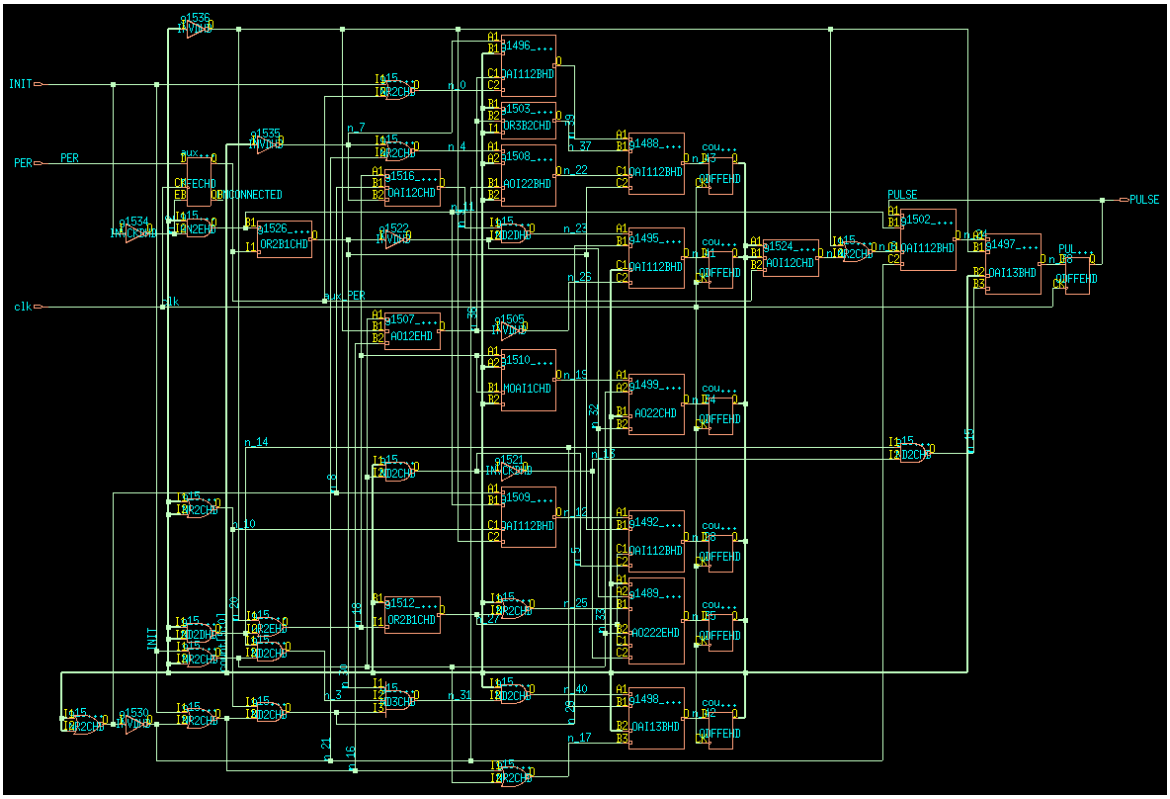

Figure 13: Pulse generator circuit, shown in the Schematic Viewer.

As seen in Fig. 12, no additional cells were placed between the `clk` pin and the flip-flops in the design. Due to its relative simplicity, the distance between them is not sufficiently large to justify the inclusion of buffers, inverter or delay cells to cope with fan-out and skew. In fact, it can be observed in Fig. 13 that the clock is directly connected to eight different cells. Similar information was obtained by running the commands `report_ccopt_clock_trees` and `report_ccopt_clock_tree_structure`.

Finally, nano route was essential to finish performing the interconnections as defined in the Verilog file, whereas filler cells were added since clock tree buffers do not fill all empty spaces between standard cells, leaving holes that can result in space violations; these filler cells, with different widths, ensure continuity of supply rails and help biasing n-well and p-well on empty spaces. After these final tasks in the P&R route procedure, a series of analyses and verification steps were made, having the reports shown in Listings 8-15 been obtained. Firstly, the commands `verifyGeometry` and `verify_drc` (which have similar verification purposes) indicated no DRC errors in the design. No errors were also found by `checkDesign -all`, but only three common warnings (also encountered in the 4-bits counter example) related with one of the many pins in the circuit. The command `verify_connectivity` (which verifies, for instance, if there are unconnected pins) also found no problems or warnings.

```
VERIFY GEOMETRY ..... Starting Verification
VERIFY GEOMETRY ..... Initializing
[...]
Begin Summary ...
```

```
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.
```

Listing 8: Part of the report obtained by running `verifyGeometry`.

```
  VERIFY DRC ..... Starting Verification
  VERIFY DRC ..... Initializing
  VERIFY DRC ..... Deleting Existing Violations
  VERIFY DRC ..... Creating Sub-Areas
  VERIFY DRC ..... Using new threading
  VERIFY DRC ..... Sub-Area: {0.000 0.000 34.800 30.400} 1 of 1
  VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.

  Verification Complete : 0 Viols.
```

Listing 9: Part of the report obtained by running `verify_drc`.

```
Design: pulsegen
------ Design Summary:
Total Standard Cell Number   (cells) : 123
[...]
Total Standard Cell Area     ( um^2) : 768.00
[...]
------ Design Statistics:
Number of Instances            : 123
Number of Non-uniquified Insts : 111
Number of Nets                 : 58
Average number of Pins per Net : 3.21
Maximum number of Pins in Net  : 9
------ I/O Port summary
Number of Primary I/O Ports    : 4
Number of Input Ports          : 3
Number of Output Ports         : 1
[...]
*** Summary of all messages that are not suppressed in this session:
Severity  ID               Count  Summary
WARNING   IMPREPO-202          1  There are %d Ports connected to core ins...
WARNING   IMPREPO-213          1  There are %d I/O Pins connected to Non-I...
WARNING   IMPREPO-216          1  There are %d Instances with input pins t...
*** Message Summary: 3 warning(s), 0 error(s)
```

Listing 10: Part of the report obtained by running `checkDesign -all`.

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Mon Oct 30 18:12:36 2023

Design Name: pulsegen
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (34.8000, 30.4000)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary
[...]
******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
```

Listing 11: Part of the report obtained by running `verify_connectivity`.

Regarding the setup and hold timing analyses, only positive WNS and null TNS values were obtained, along with no Design Rule Violations (DRVs), as desired. The command `reportGateCount` provided the value of the area for each gate in the layout, which would need to be multiplied by the number of gates in order to determine the total area occupied by them. Finally, the command `report_power` indicated three distinct power values in the design. Switching power results from the changes from H to L (and vice-versa) in the output - i.e., it is power dissipated by the charging and discharging of the load capacitance at the output of the cell - whereas the leakage power (whose relative contributions is the smallest) is due to power consumed even when the transistors are OFF. The highest value corresponds to internal power, which results from the charging or discharging of any existing capacitance **internal** to the cell; the definition of internal power includes power dissipated by a momentary short-circuit between the P and N transistors of a gate, called short-circuit power. It makes sense that internal power is the most significant, since the internal signals in the circuit change much more often than the output. Another interesting aspect to notice is that `count_reg[0]` had the highest average power, which is due to the fact that the least significant bit in `count[5:0]` changes every clock cycle when the counter is altered. Additionally, the register for `aux_PER` has the highest leakage power, perhaps because it is almost directly connected to an input pin (in this case, `PER`). It can also be verified that a voltage of 1.08 V was indicated for `dvdd` (most likely resulting from information in the technology files), instead of the value 1.2 V mentioned in the laboratory guide.

```
#  Generated by:      Cadence Innovus 20.11-s130_1 [...]
#  Design:            pulsegen
#  Command:           timeDesign -signoff [...]
+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 98.852  | 98.852  | 99.070  |
```

```
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|    9    |    7    |    9    |
+--------------------+---------+---------+---------+
+----------------+-----------------------------------+------------------+
|                |              Real                 |       Total      |
|    DRVs        +------------------+----------------+------------------|
|                |  Nr nets(terms)  | Worst Vio      |  Nr nets(terms)  |
+----------------+------------------+----------------+------------------+
|   max_cap      |      0 (0)       |   0.000        |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000        |      0 (0)       |
|   max_fanout   |      0 (0)       |     0          |      0 (0)       |
|   max_length   |      0 (0)       |     0          |      0 (0)       |
+----------------+------------------+----------------+------------------+
Density: 66.167% (100.000% with Fillers)
Total number of glitch violations: 0
```

Listing 12: Part of the report obtained in the setup timing analysis (Sign-Off stage).

```
#  Generated by:      Cadence Innovus 20.11-s130_1 [...]
#  Design:            pulsegen
#  Command:           timeDesign -signoff -hold [...]
+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.020  |  0.140  |  0.020  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|    9    |    7    |    9    |
+--------------------+---------+---------+---------+
Density: 66.167% (100.000% with Fillers)
```

Listing 13: Part of the report obtained in the hold timing analysis (Sign-Off stage).

```
Gate area 3.8400 um^2
```

Listing 14: Report obtained by running `reportGateCount`.

```
Using Power View: WC_AN.
Load RC corner of view WC_AN
Begin Boundary Leakage Calculation
[...]
Begin Static Power Report Generation
[...]
* Liberty Libraries used:
*         WC_AN: ../../SYNTHESIS/synthesis_libs/fsc0h_d_generic_core_ss1p08v125c.lib
[...]
* Power Units = 1mW
*
```

```
* Time Units = 1e-09 secs
[...]
Total Power
-----------------------------------------------------------
Total Internal Power:         0.00154783      66.4541%
Total Switching Power:        0.00041142      17.6638%
Total Leakage Power:          0.00036992      15.8820%
Total Power:                  0.00232917
-----------------------------------------------------------

Group                   Internal   Switching   Leakage     Total      Percentage
                        Power      Power       Power       Power      (%)
-------------------------------------------------------------------------------
Sequential              0.001279   0.0001168   0.0001494   0.001545   66.35
Macro                   0          0           0           0          0
IO                      0          0           0           0          0
Combinational           0.0002687  0.0002946   0.0002205   0.0007838  33.65
Clock (Combinational)   0          0           0           0          0
Clock (Sequential)      0          0           0           0          0
-------------------------------------------------------------------------------
Total                   0.001548   0.0004114   0.0003699   0.002329   100
-------------------------------------------------------------------------------

Rail   Voltage   Internal   Switching   Leakage     Total      Percentage
                            Power       Power       Power      Power(%)
-------------------------------------------------------------------------
dvdd   1.08      0.001548   0.0004114   0.0003699   0.002329   100

-------------------------------------------------------------
* Power Distribution Summary:
*      Highest Average Power: count_reg[0] (QDFFEHD): 0.0002069
*      Highest Leakage Power: aux_PER_reg (DFECHD): 1.878e-05
*      Total Cap: 3.71004e-13 F
*      Total instances in design: 52
*      Total instances in design with no power: 0
*      Total instances in design with no activty: 0
```

Listing 15: Part of the report obtained by running `report_power`.

Having finished the entire P&R flow, the final physical layout shown in Fig. 14 was achieved. By using the tech file `/verilog_libs/fsc0h_d_generic_core_30.lib` (similarly to the synthesis phase), along with the gate netlist `pulsegen_pr.v` (generated after P&R), the simulated waveforms shown in Figs. 15-19 were obtained. These show the same results as in Sec. 3, having the desired functionality been achieved - a single pulse waveform or a periodic signal is generated in the output depending on the inputs of the circuit. Once again, delays occur between the rising edge of the clock and changes in the ports in the circuit, and also between different internal variables, as seen in Fig. 19.

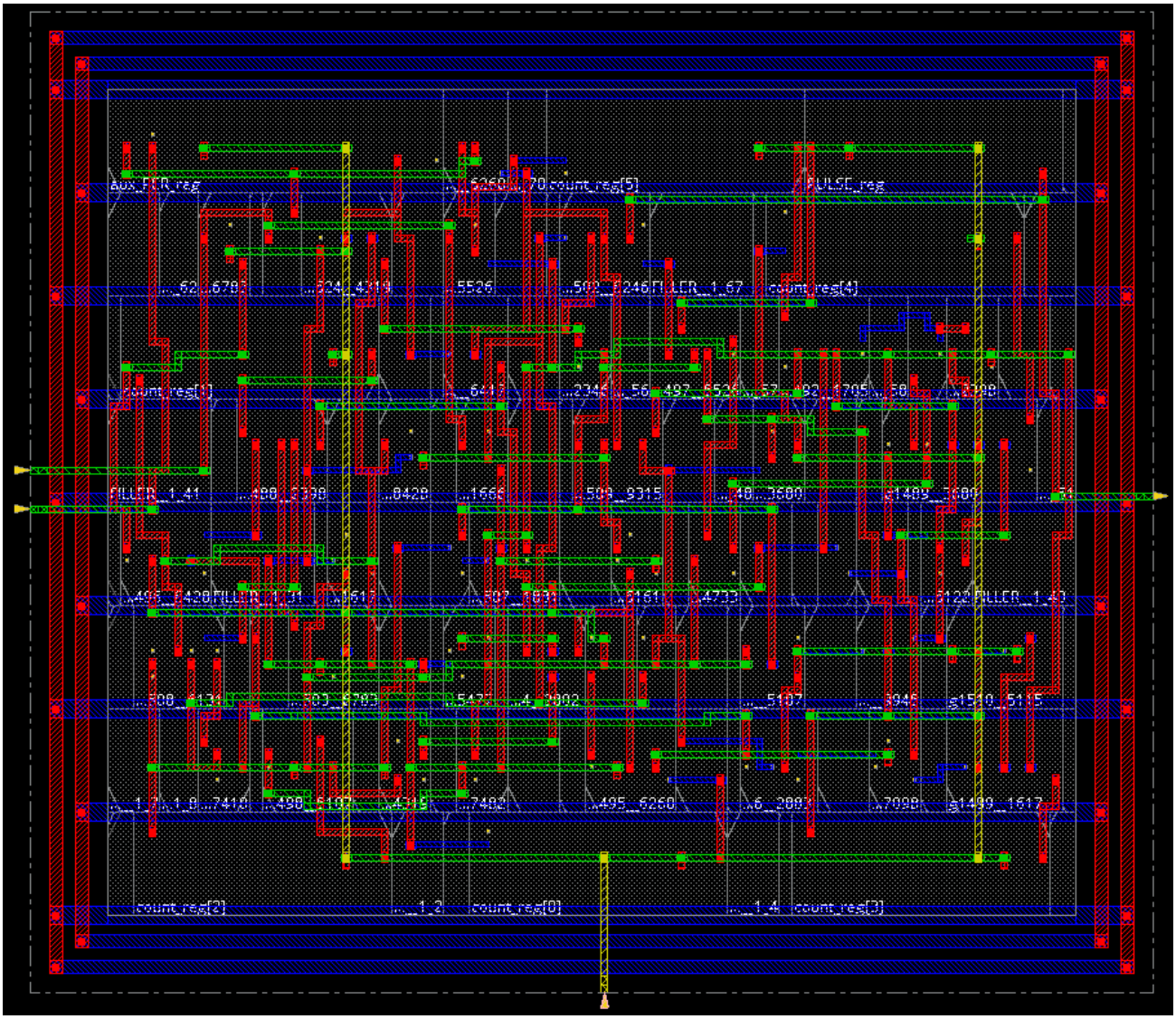

Figure 14: Final layout obtained from the P&R design flow.

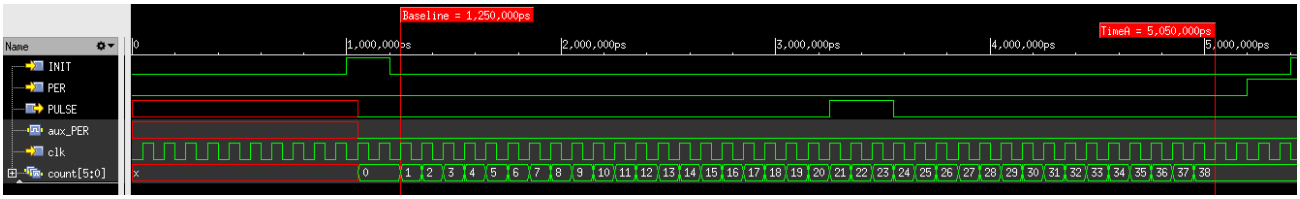

Figure 15: Simulation results after P&R - first single pulse waveform.

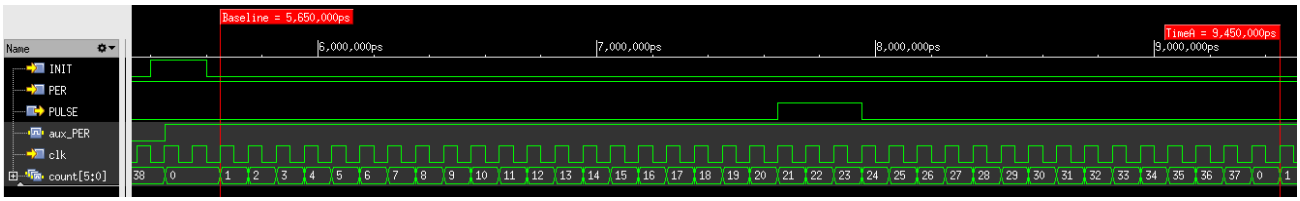

Figure 16: Simulation results after P&R - first pulse in the period signal, after the single pulse.

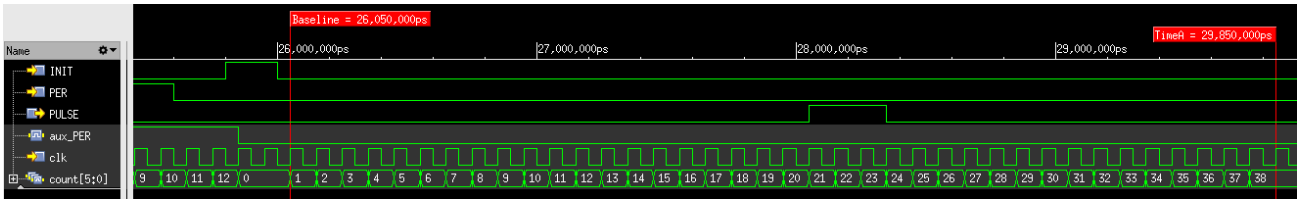

Figure 17: Simulation results after P&R - change from periodic signal to single pulse.

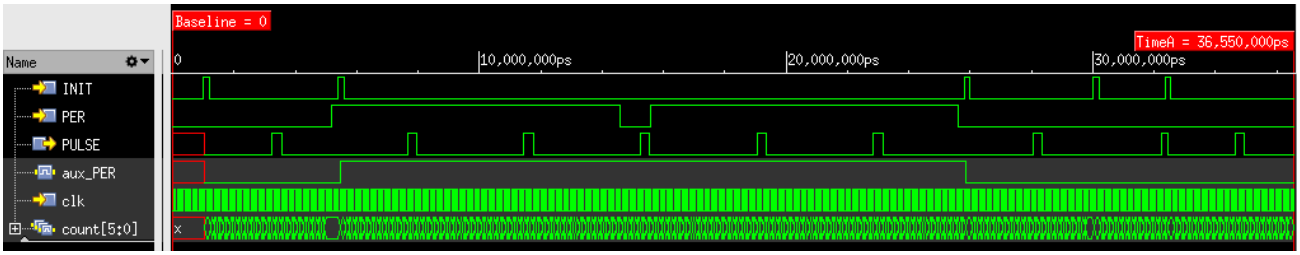

Figure 18: Simulation results after P&R - full waveform.

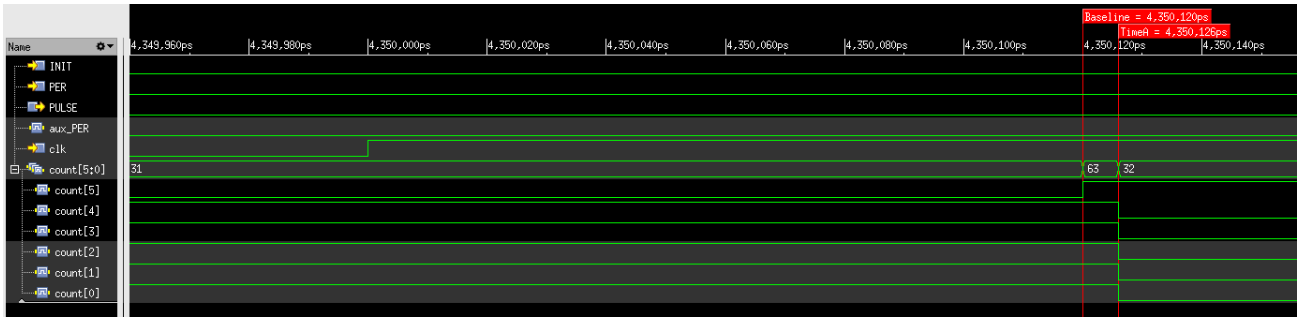

Figure 19: Simulation results after P&R - detail of the delays between changes in the signals.

5 Conclusion

In this project, a pulse generator was successfully implemented by performing all the flow steps necessary to transform the initial digital circuit gate level (RTL) description into a GDSII layout representation ready for fabrication submission, using Xcelium (an HDL functional simulator) to test the operation of the digital integrated circuit. Initially, the functional description implemented in Verilog RTL code consisted of a design with a single output `PULSE`, where a single pulse waveform with imposed specifications is generated when the input `PER` is L. On the other hand, `PER` being H leads to a periodic signal in the output. The value of this input is registered whenever the other input (`INIT`) is H. Using a testbench module, this behaviour was then tested and the desired results were obtained. After implementing the pulse generator with logic gates in the synthesis phase, using the tool `genus`, the same desired behaviour was observed in the output, but some consequences from the use of actual technology standard cells to implement the circuit were apparent. Finally, during the place and route phase, the tool `innovus` was used to produce a physical layout with a floorplan, PG rings, pins, standard cells, clock tree, routing and filler cells.