TUTORIAL PARA SÍNTESE STANDARD-CELLS UTILIZANDO CADENCE (28 nm)

Atualizado em - **06/nov/2023**

Objetivo deste laboratório:

Compreender o fluxo de projeto *standard cells*
Simulação RTL, simulação com *netlist*, simulação com anotação de atraso de roteamento
Executar a síntese lógica e física

Após executar o laboratório, pesquise e responda (entregar as respostas no dia da segunda avaliação):

1. Qual a finalidade das três simulações executadas no fluxo *standard-cells* (RTL, *netlist*, com SDF)?

2. Qual a função dos arquivos com extensão **LEF**, e qual o significado desta sigla?
   Qual a função dos arquivos com extensão **LIB**, e qual o significado desta sigla?

3. Modifique a frequência utilizada na síntese lógica (`constraint/busca_padrao.sdc`) e preencha a tabela abaixo:

Frequência	N# de portas lógicas	Slack (ps)
250 MHz		
500 MHz	**753**	**74**
1 GHz		
1.5 GHz		

   Com o aumento da frequência, como se comporta o número de portas lógicas (aumenta ou diminui? Por quê?). Com o aumento da frequência, como se comporta o *slack time* (aumenta ou diminui? Por quê?).

4. A síntese lógica utiliza três comandos de síntese: `syn_generic` / `syn_map` / `syn_opt`. Qual o objetivo de cada comando? (dica para iniciar a resposta: `man syn_generic` ou `help syn_generic` no terminal do genus).

5. No arquivo *load.tcl* (síntese lógica) há o comando para ativar o modo PLE (*physical layout estimator*):

```
#Set PLE
set_db interconnect_mode ple
```

   Execute ao final da síntese lógica: *report_ple*. O que ele indica? Porque é importante ativar o modo PLE durante a síntese lógica?

6. Modifique o arquivo de síntese lógica, de forma que estes três comandos sejam como abaixo. Preencher a tabela abaixo e discutir os novos resultados.

```
syn_generic –create_floorplan –physical
syn_map –physical
syn_opt –incremental
```

Frequência	N# de portas lógicas	Slack (ps)
500 MHz		
1.0 GHz		

7. Porque na síntese física a geração do *floorplan* deve-se ter uma densidade inferior a 100%, como no comando utilizado neste laboratório?

```
##Generating square floorplan (1) with 85% of density (0.85) with 3um margins
create_floorplan –site CORE12T –core_density_size 1 0.85 3 3 3 3
```

8. Qual a função da etapa *clock tree synthesis* na síntese física (comando `ccopt_design`)? Por que esta etapa é importante no fluxo de projeto?

9. Qual a função das *filler cells* no projeto físico?

10. Qual a tendência do *slack time* após a síntese física? Explicar este comportamento.

Arquivos do projeto (detector de padrão)

- Conectar-se ao servidor **paxos:** ssh -X <usuário>@paxos.inf.pucrs.br
- Baixar o arquivo de distribuição:
  wget https://fgmoraes.github.io/testa_padrao28nm.tar
- Descompactar o arquivo: **tar -xvf testa_padrao28nm.tar**
- Ir para a raiz do projeto: **cd testa_padrao**

Abaixo está a estrutura da distribuição, a qual contém três diretórios principais:

```
.
├── rtl
│   └── busca_padrao.vhd                  código VHDL do circuito
├── sim
│   ├── rtl
│   │   └── file_list.f                   script de simulação RTL
│   ├── sdf
│   │   ├── file_list.f                   script de simulação com atraso de fios
│   │   └── sdf_cmd.cmd
│   ├── synth
│   │   └── file_list.f                   script de simulação com o netlist da síntese física
│   └── tb
│       └── tb_padrao.vhd
└── synthesis
    ├── comandos_genus.tcl                script de síntese lógica
    ├── comandos_innovus.tcl              script de síntese física
    ├── constraint
    │   ├── busca_padrao.sdc              restrições de temporização
    │   └── load.tcl                      parâmetros para a síntese lógica (tecnologia de 28 nm)
    └── physical                          scripts parciais para a síntese física
        ├── 1_init.tcl                    primeira linha aponta para o arquivo de configuração do netlist
        ├── 2_power_plan.tcl              definição do roteamento de alimentação e polarização das células
        ├── 3_pin_clock.tcl               posicionamento dos pinos de E/S e geração da árvore de clock
        ├── 4_nano_route.tcl              roteamento
        └── 5_fillers_reports.tcl         inserção das células de preenchimento
```

ETAPA 1 – Simulação RTL com o simulador *xrun*

Configurar o ambiente e ir para o diretório de simulação RTL:

```
source /soft64/source_gaph
module load xcelium
cd sim/rtl
```

Observar o script de simulação **file_list.f** :

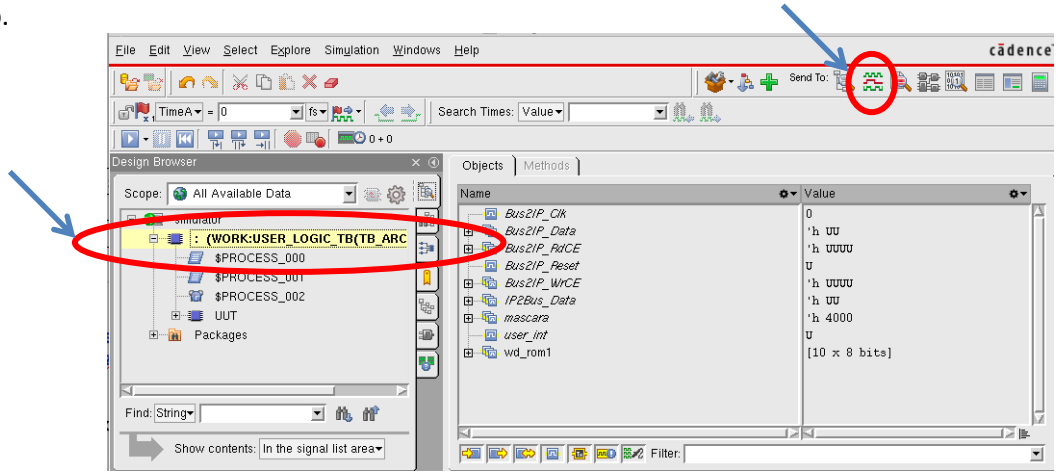
```
-smartorder -work work -V93 -top user_logic_tb -notimingchecks -gui -access +rw
../../rtl/busca_padrao.vhd
../tb/tb_padrao.vhd
```

onde:

- -smartorder – indica que o compilador deve reconhecer a ordem hierárquica das descrições fornecidas
- -work – define o nome da biblioteca onde serão armazenados os módulos compilados
- -V93 – habilita características do VHDL93, como evitar a declaração de componentes
- -top – topo da hierarquia do projeto (*user_logic_tb* – entidade do test_bench)
- -notimingchecks – desabilita verificações de timing
- -gui – habilita modo gráfico
- -access +rw – acesso aos sinais internos do circuito para exibição

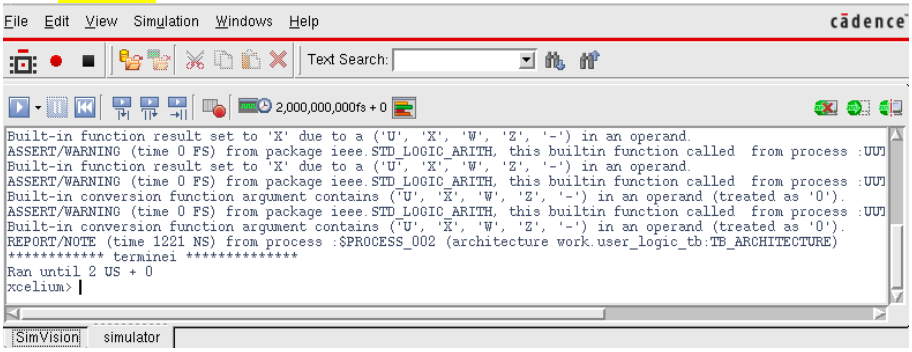
Executar o seguinte comando para inicializar a simulação: **xrun -f file_list.f**

A ferramenta *xrun* irá compilar e elaborar o projeto. A interface do simulador é aberta. Selecionando-se o *top* (USER_LOGIC_TB) tem-se os sinais da entidade, os quais podem ser enviados para uma *waveform*, clicando no local indicado.

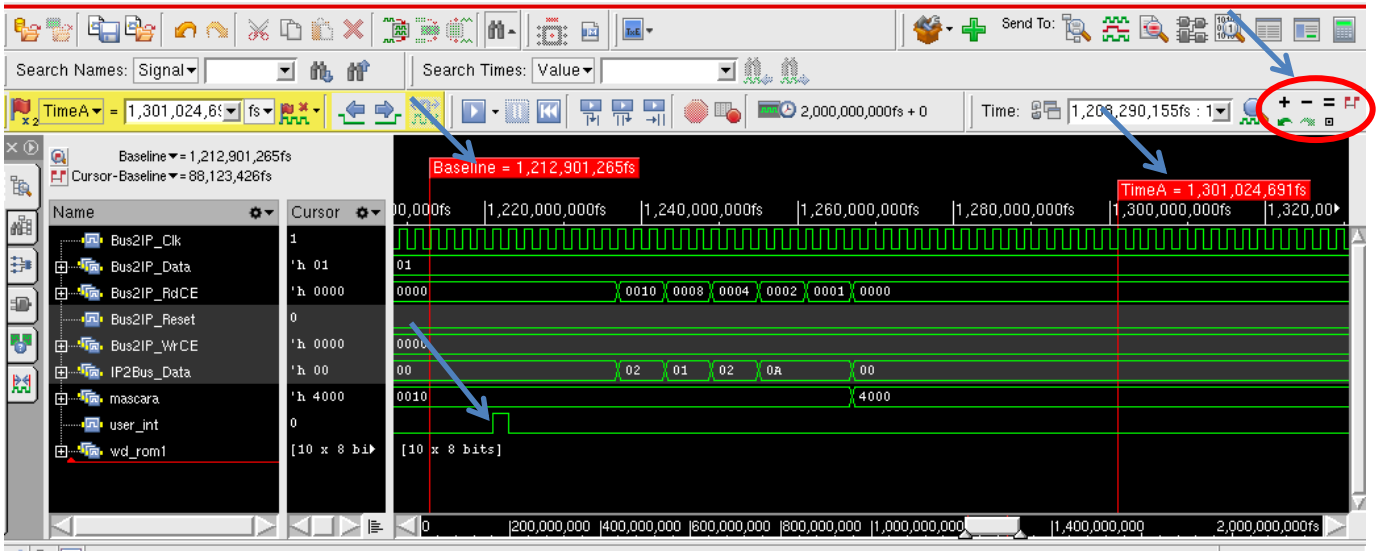


Inserir os sinais do **top** na *waveform* e executar por 2 µs, digitando no console do simulador:

reset
run 2 us



Clicar no sinal de "=" para fazer um zoom dos 2 µs de simulação, e depois com as barras verticais de "Baseline" e "TimeA" fazer um zoom entre 1,2 microssegundos e 1,3 microssegundos:



Para este circuito deve-se observar o sinal *user_int* (interrupção gerada pelo UUT), e depois os dados em *IP2bus_Data*. A interpretação destes dados são: 2 matches do padrão a ser pesquisado em uma dada imagem, nos endereços (1,2) e (A,A).

Para sair, menu *File → Exit SimVision*
Ao sair do simulador recomenda-se apagar arquivos temporários: **xrun -clean**

ETAPA 2 - Síntese Lógica

- Para a síntese lógica é utilizada a ferramenta *genus* da CADENCE. **Não** executar o *genus* com a opção '&', pois a ferramenta tem um *shell* interno. Configurar o ambiente e ir para o diretório de síntese:

```
cd ../../synthesis
module load genus
genus -gui
```

Na interface gráfica poderão ser acompanhados as respostas dos comandos inseridos no *shell* do *genus*. Os comandos necessários para a correta síntese do projeto estão disponíveis no arquivo *"comandos_genus.tcl"* e deverão ser inseridos sequencialmente no *shell* do *genus*.

A lista de comandos está dividida em **5 grupos distintos** (copie e cole os comandos no *shell* do *genus*).

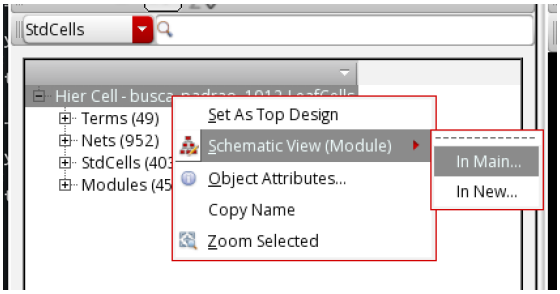
1. Configuração do ambiente de síntese e compilação do projeto

Abrir o arquivo *load.tcl*. Este é o arquivo que define os LIB e LEF files, e definições gerais para a síntese lógica. Usarem os a biblioteca 28 nm da STMicroelectronics.

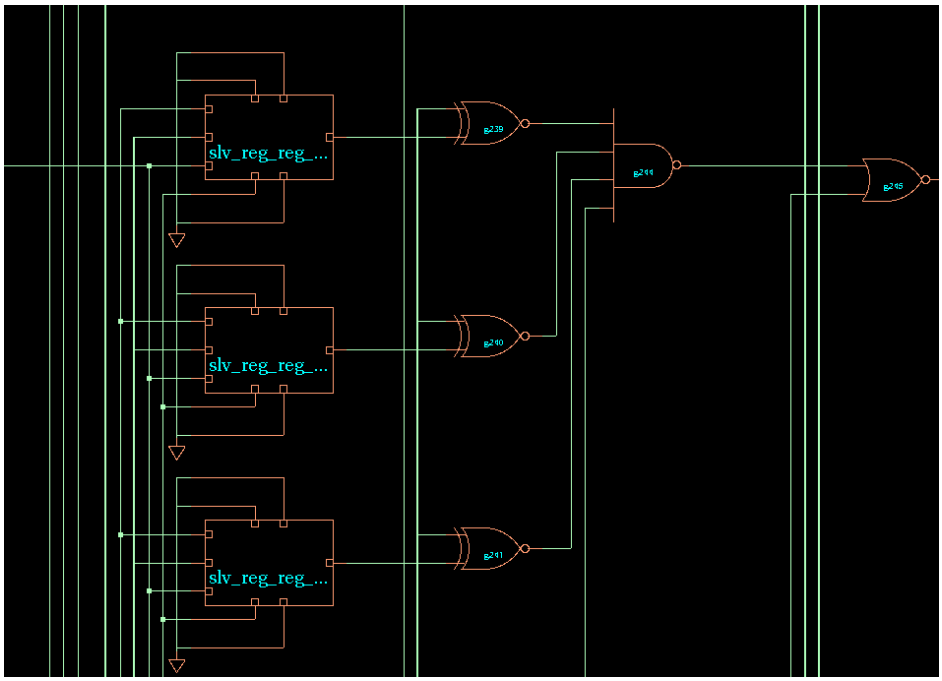
A *captable* define a condição operacional. A *captable* é um arquivo que contém valores de resistência e capacitância usadas para modelar as interconexões do projeto. Essa informação é usada quando a ferramenta de síntese extrair os fios de roteamento, para realizar análises de *timing* e *power*.

Digite os três abaixo no *shell* do *genus*:

```
include ./constraint/load.tcl
read_hdl -vhdl busca_padrao.vhd
elaborate busca_padrao
```



O resultado após executar esse bloco de comandos é dado na janela gráfica do *genus*. Navegar pelo visualizador de esquemáticos. Conforme pode ser observado, o projeto foi elaborado para funções definidas nas bibliotecas instanciadas na descrição, *ands*, *ors*, etc. Selecionar "*schematic view (Module)* → *in Main"*. **Faça um zoom em uma região para visualizar as portas lógicas.**



2. Restrições do projeto.

Abrir o arquivo de restrições "../constraint/busca_padrao.sdc" e observar seus comandos.

- Os dois primeiros comandos definem variáveis internas da ferramenta.
- O comando *create_clock* define quem é o clock do circuito e o período desejado (2.0 ns)
- O comando *set_false_path* evita que o reset seja utilizado na análise de atraso
- O comando *set_load* define a carga nas saídas do circuito

Com essas informações, a ferramenta de síntese pode escolher o ganho/tamanho das células que serão instanciadas no projeto. No *shell* do *genus* digite o segundo comando:

```
read_sdc  ./constraint/busca_padrao.sdc
```

Como resultado desse comando, deve-se obter a seguinte saída:

```
Statistics for commands executed by read_sdc:
 "all_inputs"              - successful      1 , failed      0 (runtime  0.00)
 "all_outputs"             - successful      2 , failed      0 (runtime  0.00)
 "create_clock"            - successful      1 , failed      0 (runtime  0.00)
 "get_ports"               - successful      2 , failed      0 (runtime  0.00)
 "set_false_path"          - successful      1 , failed      0 (runtime  0.00)
 "set_input_delay"         - successful      1 , failed      0 (runtime  0.00)
 "set_load"                - successful      1 , failed      0 (runtime  0.00)
 "set_output_delay"        - successful      1 , failed      0 (runtime  0.00)
 "set_units"               - successful      1 , failed      0 (runtime  0.00)
```

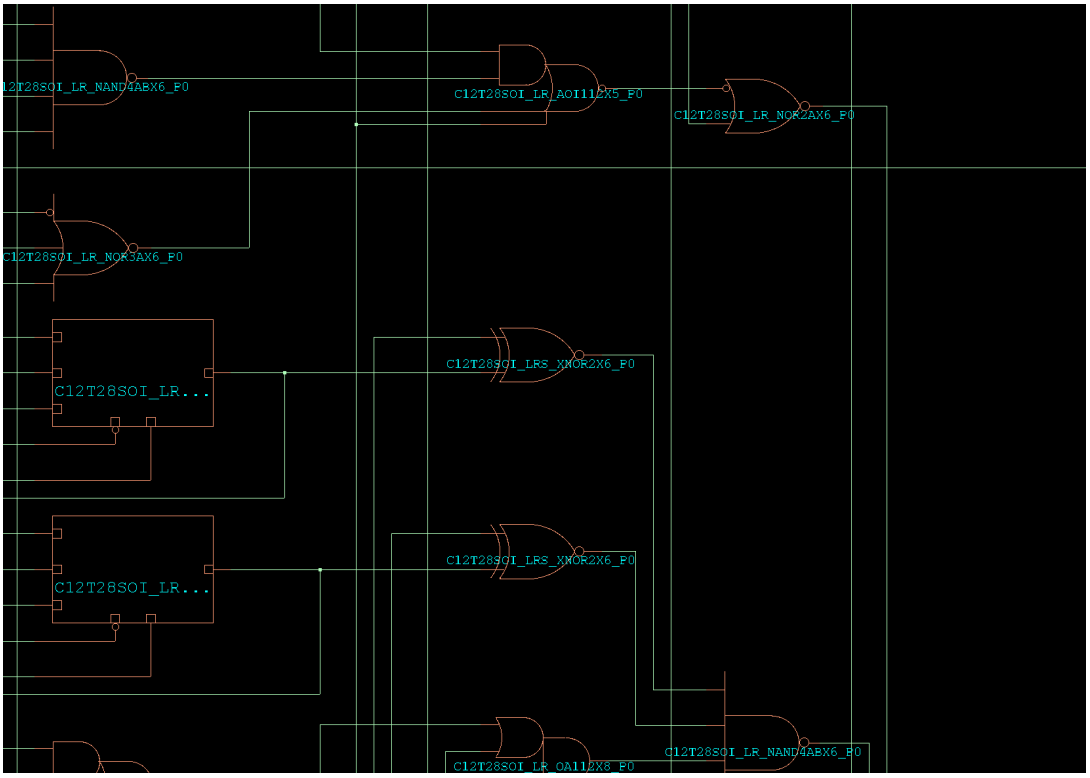
Que indica que as *constraints* foram geradas corretamente.

3. Síntese lógica otimizada

No *shell* do *genus* digite os comandos abaixo para a realização da síntese lógica:

```
syn_generic
syn_map
syn_opt
```

Esse passo realiza a síntese **optimizada** para o projeto, otimizando o projeto elaborado, para a biblioteca de células de 28 nm.



4. Relatórios

report_area: a coluna "*Instance*" indica o circuito, a coluna "xi" o número de células utilizadas, a coluna "*cell area*" a área dessas células e a coluna "*net area*" uma estimativa da área de fios que será necessária. Notar que neste circuito forma utilizadas **753** células (portas lógicas).

```
Instance    Module     Cell Count  Cell Area  Net Area   Total Area
-----------------------------------------------------------------
busca_padrao                753    962.717   457.369     1420.086
```

report_gates: apresenta todas as portas lógicas utilizadas no projeto.

```
   …
   C12T28SOI_LR_OAI222X3_P0           5    6.528    C28SOI_SC_12_CORE_LR
   C12T28SOI_LR_OAI22X5_P0            9    7.344    C28SOI_SC_12_CORE_LR
   C12T28SOI_LR_OR2X8_P0              6    3.917    C28SOI_SC_12_CORE_LR
   C12T28SOI_LR_SDFPRQX8_P0         100  440.640    C28SOI_SC_12_CORE_LR
   ---------------------------------------------------------------------
   total                            753  962.717

          Library       Instances   Area  Instances %
   ------------------------------------------------
   C28SOI_SC_12_CLK_LR          84  27.418        11.2
   C28SOI_SC_12_CORE_LR        669 935.299        88.8


        Type       Instances   Area  Area %
   ---------------------------------------
   sequential           108 469.526   48.8
   inverter              84  27.418    2.8
   logic                561 465.773   48.4
   physical_cells         0   0.000    0.0
   ---------------------------------------
   total                753 962.717  100.0
```

report_timing: informa o atraso de cada célula no caminho crítico, e principalmente se a síntese atendeu à restrição de timing. Dado que o *clock* é de 2000 ps, há uma sobra de 74 ps (***slack***) – destacado em amarelo.

```
=============================================================
  Generated by:           Genus(TM) Synthesis Solution 21.12-s068_1
  Generated on:           Nov 01 2022  03:55:35 pm
  Module:                 busca_padrao
  Operating conditions:   _nominal_
  Interconnect mode:      global
  Area mode:              physical library
=============================================================

Path 1: MET (74 ps) Setup Check with Pin EA_reg_3/CP->D
          Group: Bus2IP_Clk
     Startpoint: (R) EA_reg_2/CP
          Clock: (R) Bus2IP_Clk
       Endpoint: (F) EA_reg_3/D
          Clock: (R) Bus2IP_Clk

                     Capture       Launch
        Clock Edge:+    2000            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=    2000            0

             Setup:-      48
     Required Time:=    1952
      Launch Clock:-       0
         Data Path:-    1878
             Slack:=      74

#-------------------------------------------------------------------------------------------------------
# Timing Point   Flags   Arc   Edge           Cell            Fanout Load Trans Delay Arrival Instance
#                                                               (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------------------------
  EA_reg_2/CP    -       -     R     (arrival)                   108    -     0     0       0    (-,-)
  EA_reg_2/Q     -       CP->Q R     C12T28SOI_LR_DFPRQX17_P0      8 12.9    44    87      87    (-,-)
  ….
  g19228__2398/Z -       D->Z  F     C12T28SOI_LR_NOR4ABX6_P0      1  2.0    18    28    1877    (-,-)
  EA_reg_3/D     <<<     -     F     C12T28SOI_LR_DFPRQX17_P0      1    -     -     0    1878    (-,-)
#-------------------------------------------------------------------------------------------------------
```

report_power -unit mW

```
Power Unit: mW
PDB Frames: /stim#0/frame#0
  ------------------------------------------------------------------------
    Category          Leakage      Internal     Switching         Total     Row%
  ------------------------------------------------------------------------
       memory      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00    0.00%
     register      2.69864e-03   2.18846e-01   1.41636e-02   2.35708e-01   44.94%
        latch      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00    0.00%
        logic      3.93821e-03   7.79022e-02   1.67633e-01   2.49474e-01   47.56%
         bbox      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00    0.00%
        clock      0.00000e+00   0.00000e+00   3.93660e-02   3.93660e-02    7.50%
          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      6.63685e-03   2.96748e-01   2.21163e-01   5.24548e-01  100.00%
   Percentage           1.27%        56.57%        42.16%       100.00%  100.00%
 ------------------------------------------------------------------------
```

5. Exportação para a síntese física

write_netlist [current_design] > busca.v
write_snapshot -innovus -directory layout -tag logical [current_design]

Para sair do *genus* digite **exit**.

Para executar via script: **genus -f comandos_genus.tcl**

Os relatórios estão no arquivo: **rep_busca_padrao.txt**.

ETAPA 3 - Simulação pós síntese com o netlist busca.v

Deve-se garantir que o *netlist*, gerado no passo anterior, implementa a funcionalidade desejada. Para tanto, ir para o ambiente de simulação pós síntese:

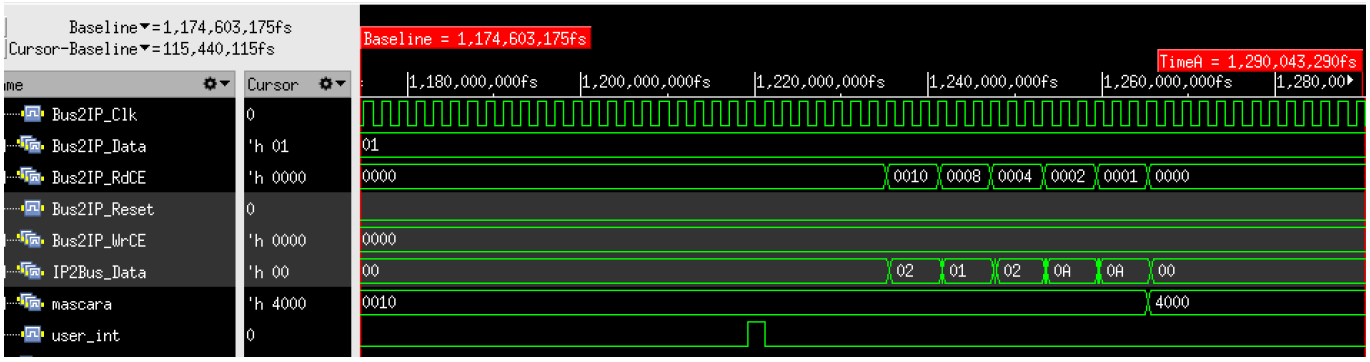
module purge
module load xcelium
cd ../sim/synth

Observar o *netlist* gerado: **more ../../synthesis/busca.v**. Este *netlist*, com 1637 linhas, corresponde ao circuito original, mapeado para as portas lógicas da tecnologia.

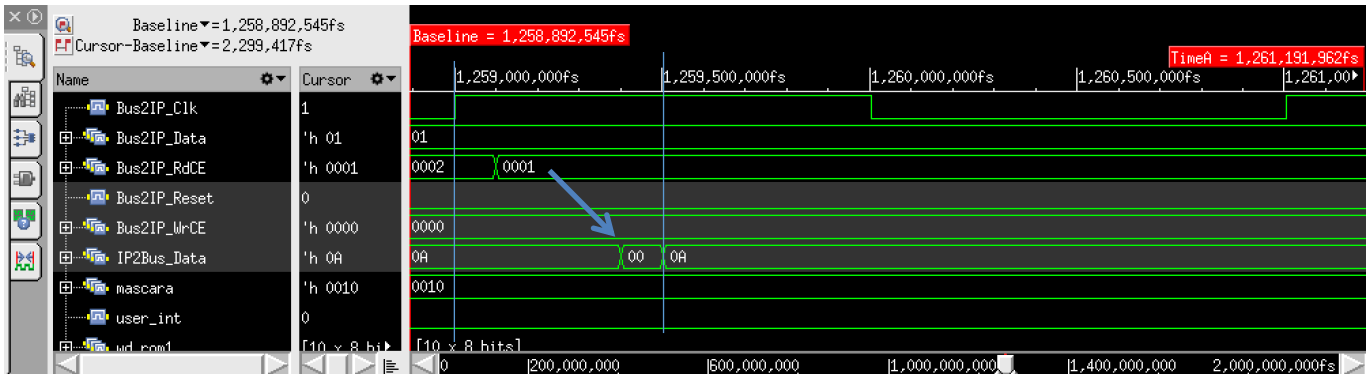
O *script* desse ambiente é similar ao de verificação RTL, porém agora também será compilada a descrição funcional das bibliotecas utilizadas. Isso é necessário pois o *netlist* contém as células específicas da tecnologia. Executar: **more file_list.f**

-smartorder -work work -V93 -top user_logic_tb -gui -access +rw
/soft64/design-kits/stm/28nm-cmos28fdsoi_25d/C28SOI_SC_12_CLK_LR@2.1@20130621.0/behaviour/verilog/C28SOI_SC_12_CLK_LR.v
/soft64/design-kits/stm/28nm-cmos28fdsoi_25d/C28SOI_SC_12_CORE_LR@2.0@20130411.0/behaviour/verilog/C28SOI_SC_12_CORE_LR.v
../../synthesis/busca.v
../tb/tb_padrao.vhd

Executar o comando **xrun -f file_list.f.** Enviar os sinais do top para uma *waveform* e simular o circuito por 2us (reset; run 2 μs).



Notar que fazendo um zoom no valor **0A** visualizamos o atraso no sinal *IP2Bus_Data*:



Este atraso é devido ao atraso das células. **A importância desta simulação é a demonstração que o VHDL originalmente escrito está correto, e o circuito opera como o esperado no nível de portas lógicas.**

Ao sair do simulador recomenda-se apagar arquivos temporários: **xrun -clean**

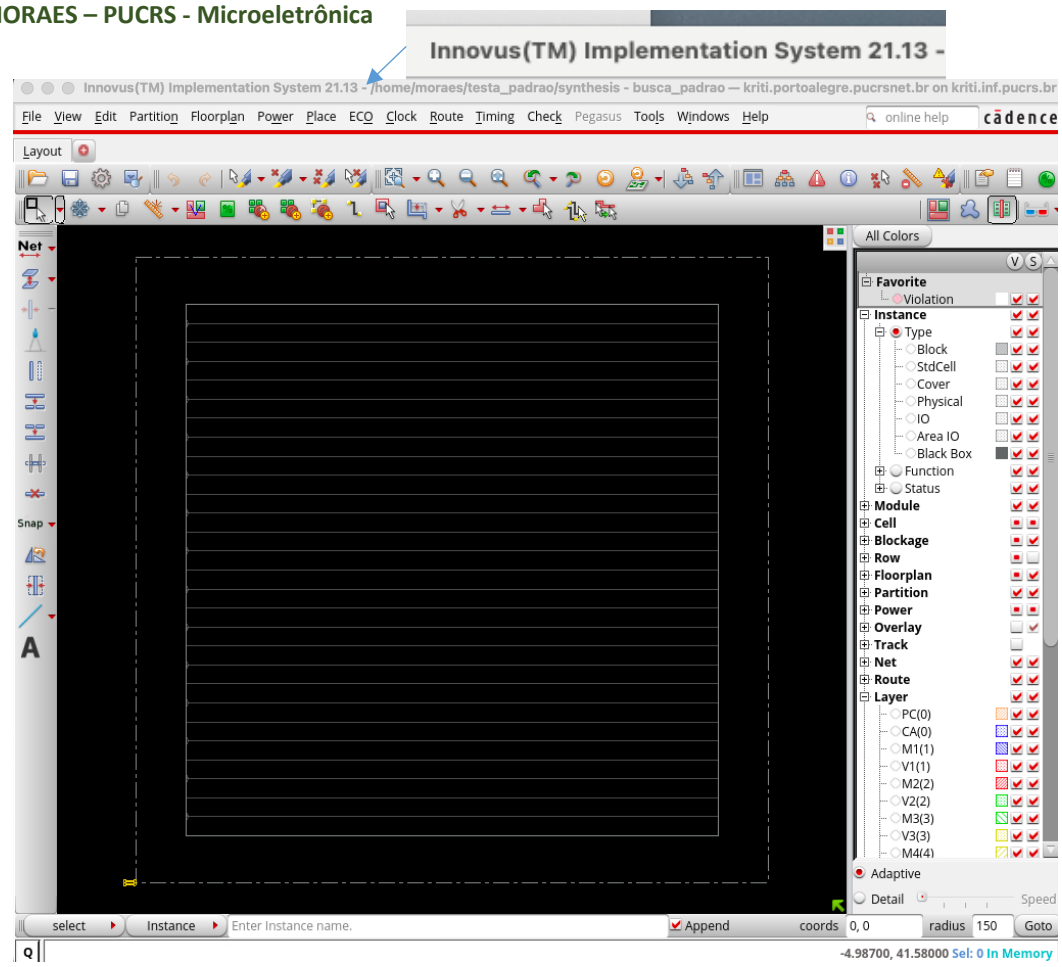
ETAPA 4 – SÍNTESE FÍSICA

Uma vez que a síntese lógica do projeto foi validada, deve ser feita a síntese física. Para isto iremos utilizar os arquivos gerados na ferramenta anterior (*genus*) e a ferramenta *innovus* da CADENCE:

```
module load genus innovus
cd ../../synthesis
innovus -common_ui
```

Carregar a configuração inicial da síntese física, digitando o seguinte comando no *shell* do *innovus*:

```
source physical/1_init.tcl
```

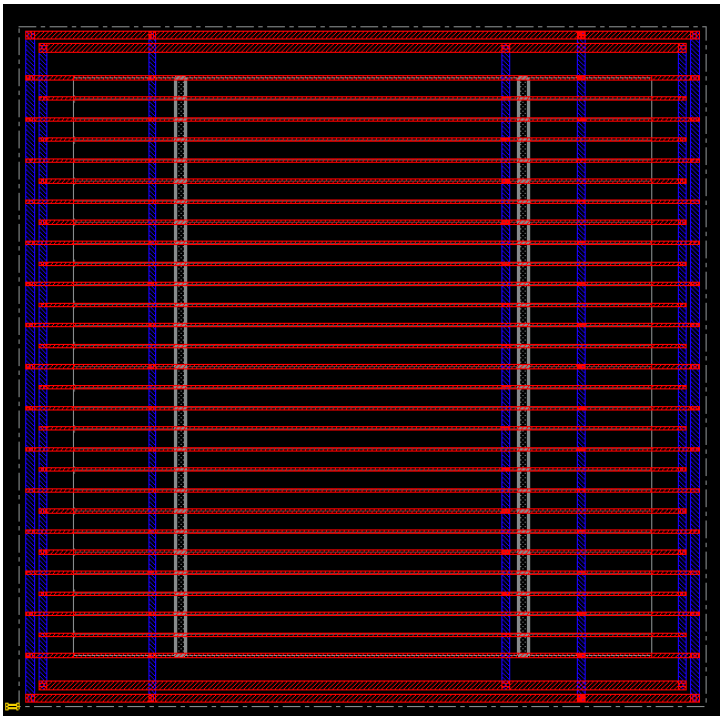


Abrir o arquivo *physical/1_init.tcl* e entender os comandos passados para o *innovus*. Dentre os comandos, os 2 principais sãs:

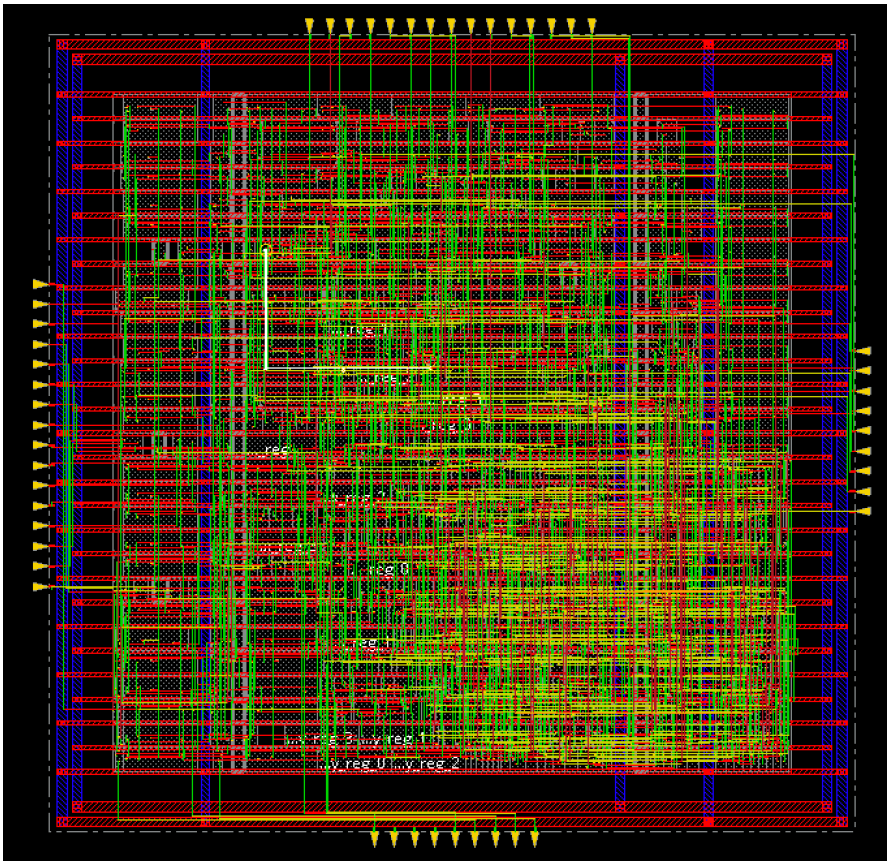
```
##Load the circuit configuration from genus
source layout/logical_busca_padrao.invs_setup.tcl

##Generating square floorplan (1) with 85% of density (0.85) with 3um margins
create_floorplan -site CORE12T -core_density_size 1 0.85 3 3 3 3
```

Carregar a configuração de power planning no shell do *innovus*: **source physical/2_power_plan.tcl**

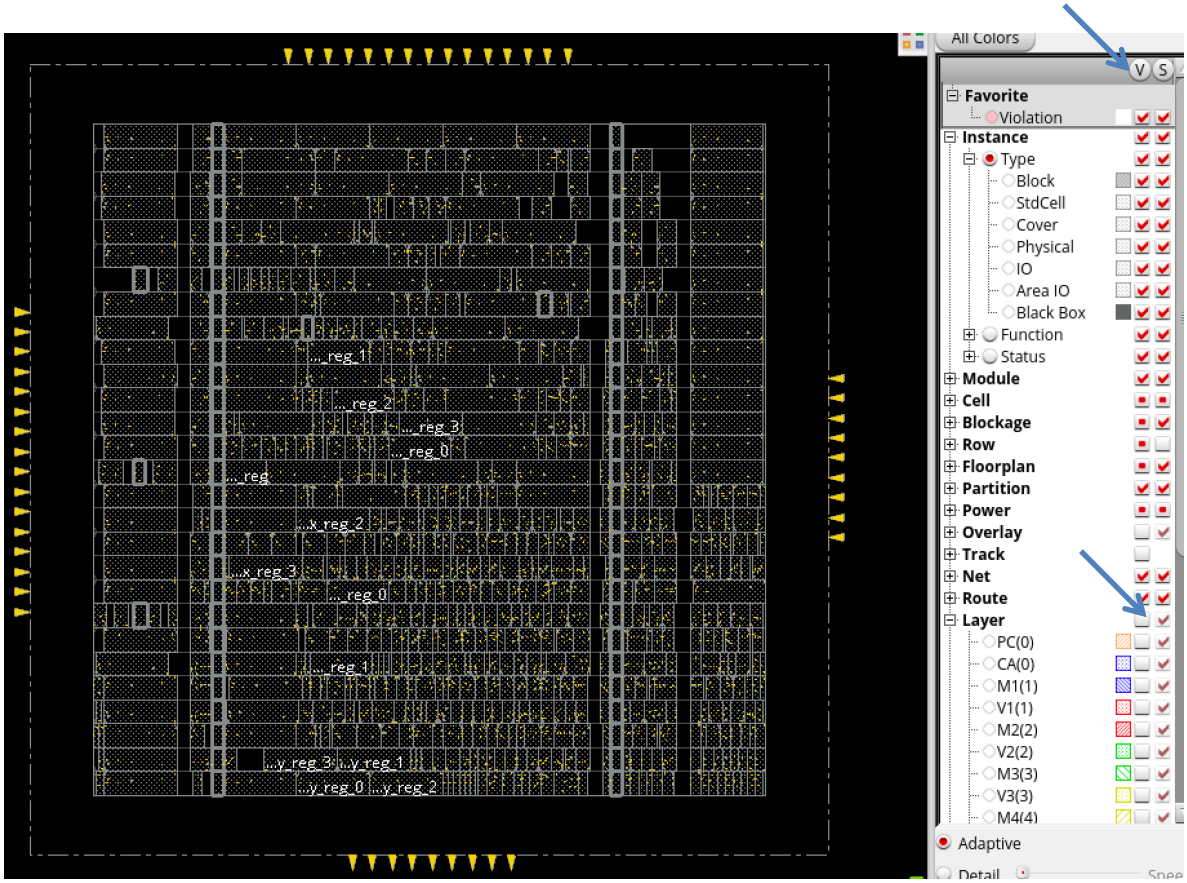
Abrir o arquivo ***2_ power planning.tcl***. Notar que foi gerado um anel e linhas de alimentação, que serão utilizadas para posicionar as células lado a lado (***add_rings***). A simetria dessas linhas (mesma altura) facilita o algoritmo de posicionamento e a instanciação das células físicas. As linhas de alimentação correspondem aos retângulos vermelhos (metal 2 - ***route_special***). Além disso, foram posicionadas colunas de *tap cells* (***add_well_taps***). Estas células garantem a polarização da difusão, já que para essa biblioteca, as células lógicas não possuem conexão com *bulk*. Essas células devem ser posicionadas no máximo 20 µm de distância uma da outra, para garantir polarização da difusão (informação obtida na documentação da biblioteca). Também foram inseridos reforços para a alimentação (***add_stripes***).



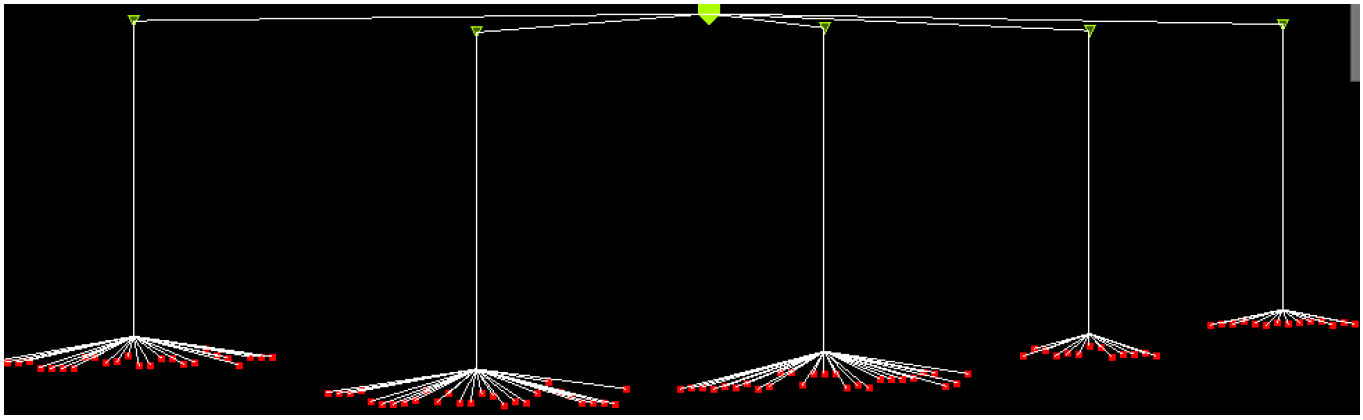
Instanciar as células físicas no projeto, posicionar os pinos na periferia do circuito, e realizar a árvore de *clock*. Executar o seguinte comando no *innovus*: **source physical/3_pin_clock.tcl** (*demora um pouco)*. O resultado deve ser:



➔ observar que há um menu à direita – se as camadas não aparecerem, selecionar como visível (V) as instâncias, fios, etc. Desmarcar "layer" para visualizar apenas as instâncias das células.

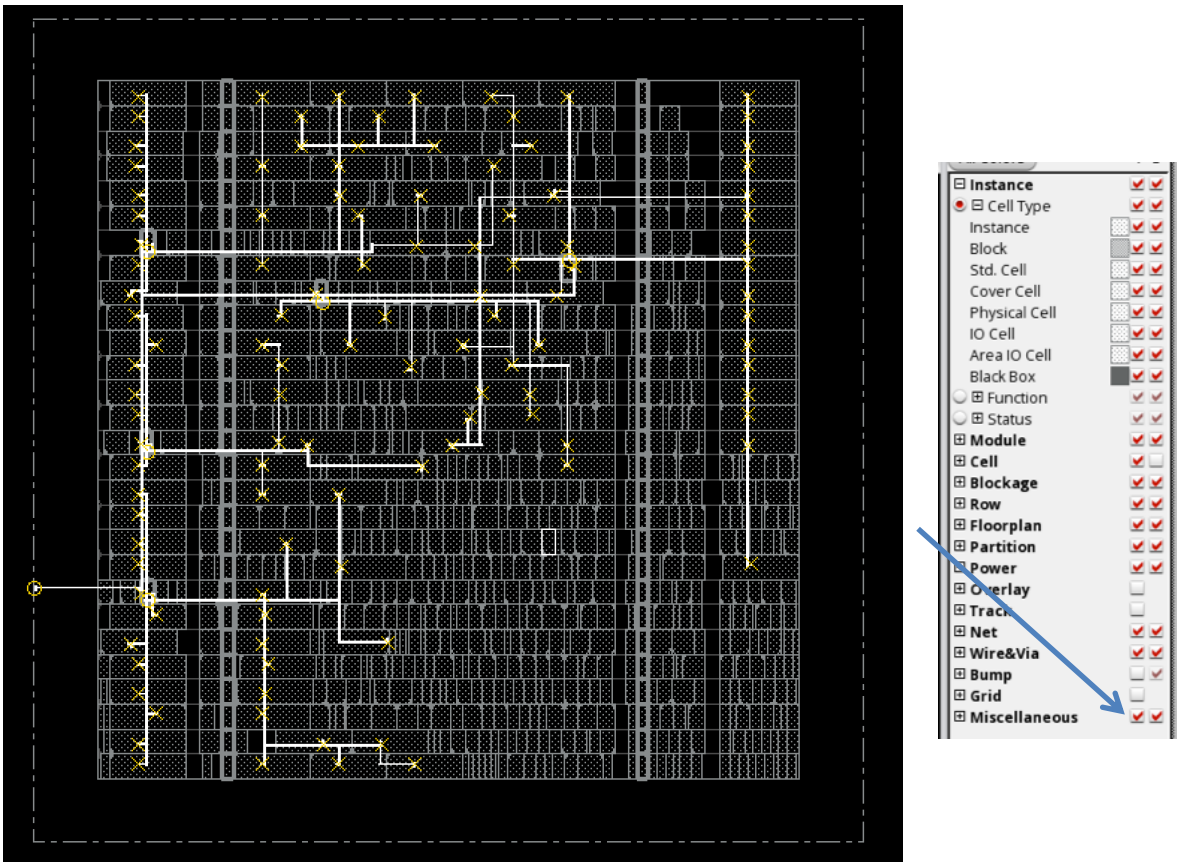


No topo da tela selecionar **clock → CCOpt Clock Debugger** e OK. Esta ação abre uma janela, com os buffers de *clock*:



Na figura acima selecionei os sinais de clock (shift-clicar) – desmarcar "micellaeous" na pallete.

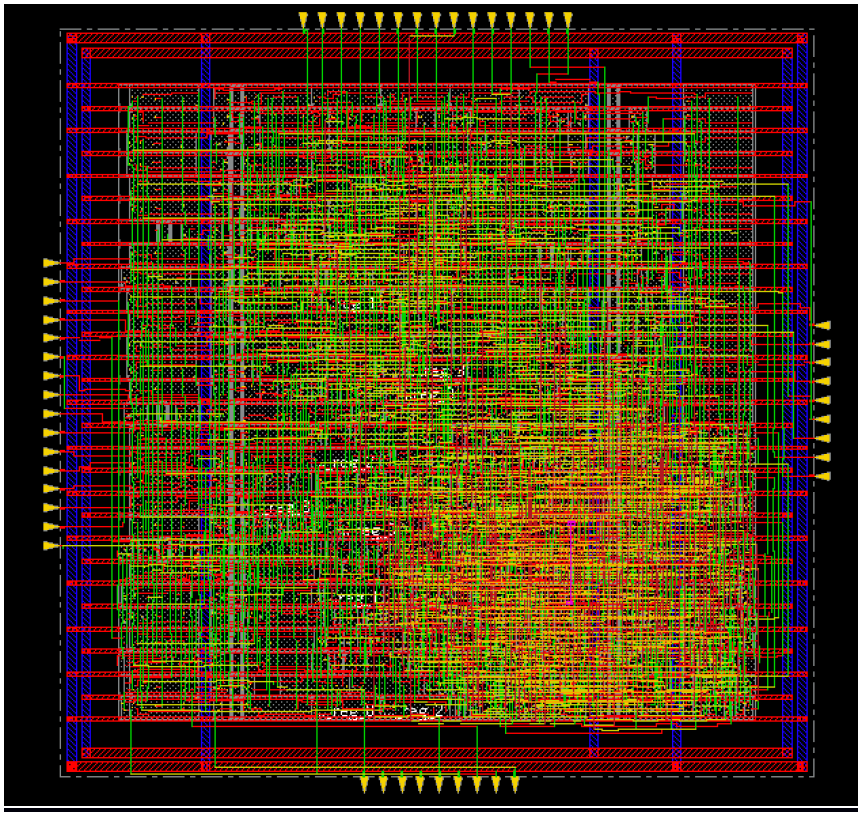
O resultado é a visualização da árvore de *clock*:



O próximo passo é executar o roteamento, ou seja conectar as células e os pinos de entrada/saída. Para isto executar (demora): **source physical/4_nano_route.tcl**

Este script contém 4 comandos:

```
route_design
route_design –global_detail –wire_opt
set_db timing_analysis_type ocv
opt_design –post_route
```



Para concluir a síntese física, temos o script: **source 5_fillers_reports.tcl**

```
##Add filler cells
set FILLER_CELLS {C12T28SOI_LR_FILLERNPW4 ….. }
add_fillers –base_cells $FILLER_CELLS –prefix FILLER

##verify design
check_drc

##Generate reports
report_summary –out_dir summaryReport

## Gera arquivo para simulação SDF
extract_rc
write_sdf buscaSYNTH.sdf
write_netlist buscaSYNTH.v

# Relatórios
report_area
report_gate_count
report_timing
```

- Notar que a figura acima possui “buracos” entre instâncias de células. Tal condição pode representar uma violação nas regras de manufatura, definidas pela *foundry*. Portanto, devem ser incluídas *filler cells*, que preencherão os espaços entre células e garantirão que o projeto não violará regras pelo motivo descrito → comando: **add_fillers**
- Utiliza-se o comando **check_drc** para verificar se não há erros no layout:

```
#–check_same_via_cell true              # bool, default=false, user setting
 *** Starting Verify DRC (MEM: 1853.5) ***

  VERIFY DRC ..... Starting Verification
  ...
  VERIFY DRC ..... Sub–Area: {0.000 0.000 39.984 39.600} 1 of 1
  VERIFY DRC ..... Sub–Area : 1 complete 0 Viols.

  Verification Complete : 0 Viols.

 *** End Verify DRC (CPU: 0:00:00.8  ELAPSED TIME: 0.00  MEM: 272.1M) ***
```

- Os comandos de extração de parasitas servem para gerar o verilog e o SDF para simulação com atraso de roteamento.

report_timing → observar agora que **o slack diminui** devido ao roteamento

```
Path 1: MET (0.302 ns) Setup Check with Pin EA_reg_3/CP->D
              View: default_emulate_view
             Group: Bus2IP_Clk
        Startpoint: (R) EA_reg_2/CP
             Clock: (R) Bus2IP_Clk
          Endpoint: (F) EA_reg_3/D
             Clock: (R) Bus2IP_Clk

                     Capture       Launch
       Clock Edge:+    2.000        0.000
      Src Latency:+    0.000       -0.040
      Net Latency:+    0.033 (P)    0.036 (P)
          Arrival:=    2.033       -0.004

            Setup:-    0.030
    Required Time:=    2.003
     Launch Clock:=   -0.004
        Data Path:+    1.706
            Slack:=    0.302
```

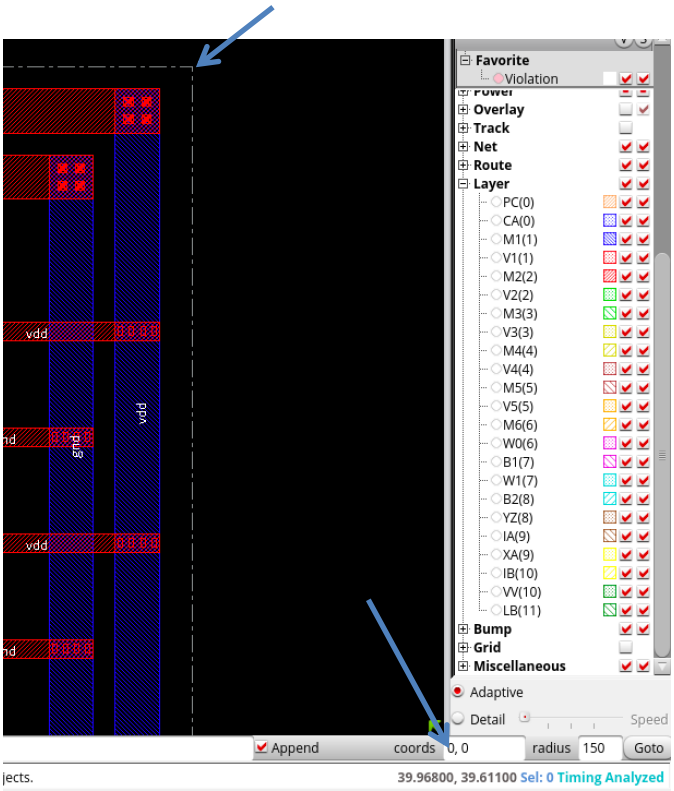
report_area

```
    Hinst Name       Module Name          Inst Count          Total Area
-------------------------------------------------------------------------
busca_padrao                                   757              966.470
```

report_gate_count

```
Gate area 0.4896 um^2
[0] busca_padrao Gates=1974 Cells=757 Area=966.5 um^2
```

Posicionar o mouse no canto superior do circuito como abaixo. O circuito tem uma área aproximada de 40 x 40 μm, o que resulta em 1.600 μm^2.



Observar que foi gerado um relatório do projeto, no diretório "*summaryReport*". Executar o seguinte na linha de comando (pode ser dentro do *innovus*):

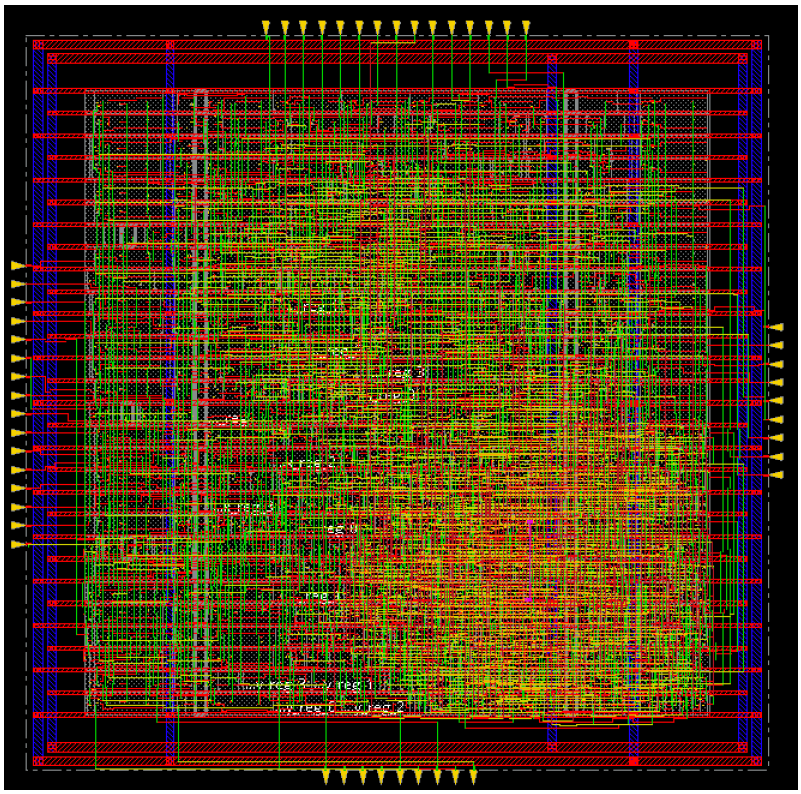
firefox summaryReport/busca_padrao.main.htm

Neste relatório temos por exemplo a área do *core* (área sem o anel de alimentação) e a área total do circuito, que confere com a medida realizada acima:

```
Total area of Standard cells                              1133.261 um^2
Total area of Standard cells(Subtracting Physical Cells)   966.470 um^2
Total area of Core                                        1133.261 um^2
Total area of Chip                                        1583.366 um^2
```

Explorar informação como, células instanciadas, área do core, comprimentos de fios, níveis de metal utilizados, etc.

Circuito final:



Para sair: exit

ETAPA 5 – SIMULAÇÃO COM ATRASO DE ROTEAMENTO

Neste passo iremos simular o circuito com atraso de portas e fios. O arquivo que contém estes atrasos é o *buscaSYNTH.*sdf. A descrição *sdf* é um formato de VHDL, e significa *standard delay format.*

Ir para o ambiente de simulação com atraso de roteamento: **cd ../sim/sdf**

O script desse ambiente é similar ao de verificação pós síntese, porém agora é dado um parâmetro a mais (-sdf_cmd_file), o script de configuração de atraso. Ver ***more sdf_cmd.cmd***:

```
SDF_FILE = "../../synthesis/buscaSYNTH.sdf",
LOG_FILE = "./sdf_log.log",
SCOPE = :UUT;
MTM_CONTROL = "MAXIMUM",
SCALE_FACTORS = "1.0:1.0:1.0",
SCALE_TYPE = "FROM_MAXIMUM";
```

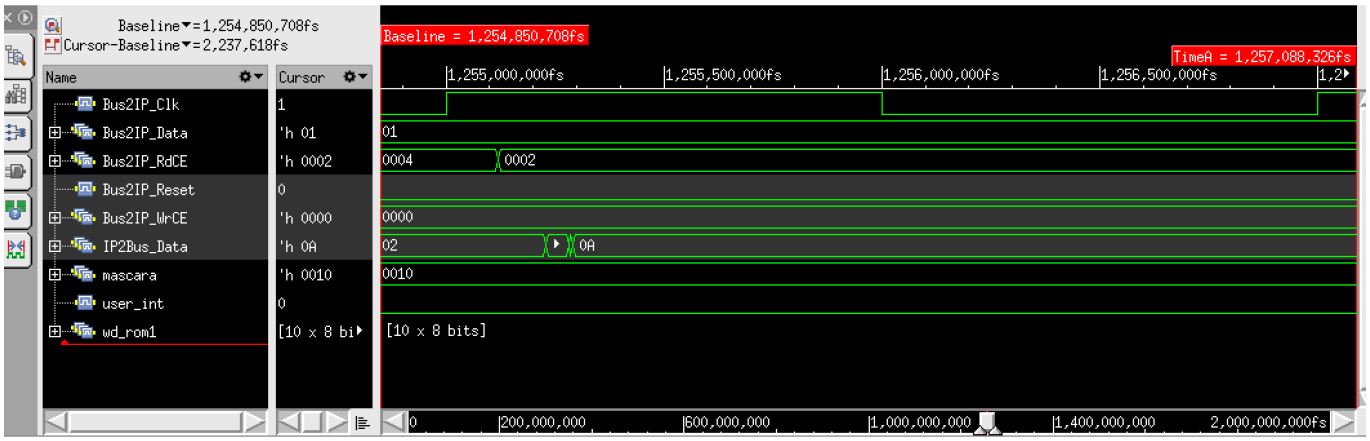
Arquivo: **file_list.f**

```
-smartorder -work work -V93 -top user_logic_tb -gui -access +rw -maxdelays -sdf_cmd_file sdf_cmd.cmd
/soft64/design-kits/stm/.../behaviour/verilog/C28SOI_SC_12_CLK_LR.v
/soft64/design-kits/stm/.../behaviour/verilog/C28SOI_SC_12_CORE_LR.v
../../synthesis/buscaSYNTH.v
../tb/tb_padrao.vhd
```

Executar

```
module purge
module load xcelium
cd ../sim/sdf
```

Enviar os sinais do top para uma *waveform* e simular o circuito por 2 µs:



Notar que agora o atraso é maior, em comparação com a simulação pós síntese lógica. O que ocorre é que agora esse atraso é um valor muito aproximado da realidade. Esse valor é baseado em modelos definidos pela fabricante. Havendo redes não anotas, o arquivo **sdf_log.log** as indica.

Observar o arquivo xrun.log (parte transcrita abaixo). Indica que todas os caminhos foram “anotados”.

```
Reading SDF file from location "../../synthesis/buscaSYNTH.sdf"
      Writing compiled SDF file to "buscaSYNTH.sdf.X".
xmelab: *W,FLFNOF: No SDF file specified, but annotation parameters were present, in SDF annotation
command file sdf_cmd.cmd, line 6.
      Annotating SDF timing data:
            Compiled SDF file:     buscaSYNTH.sdf.X
            Log file:              ./sdf_log.log
            Backannotation scope:  :UUT
            Configuration file:
            MTM control:
            Scale factors:
            Scale type:
      Annotation completed successfully...
      SDF statistics: No. of Pathdelays = 3817  Annotated = 100.00% -- No. of Tchecks = 1246  Annotated =
100.00%
                                 Total        Annotated             Percentage
               Path Delays            3817             3817            100.00
                    $width             523              523            100.00
                  $recrem              107              107            100.00
                 $setuphold            616              616            100.00
      Building instance overlay tables: .................... Done
```

FINAL DO TUTORIAL