

Universidade Federal do Rio Grande do Sul
Programa de Pós Graduação em Microeletrônica
MIC61 - Projeto de um Sistema VLSI Digital

From RTL to GDS - Síntese do core AES128

Karina Kerne
Setembro
2023

1 Introdução

Neste documento é apresentada a estruturação do trabalho de síntese lógica e física de um core de criptografia de 128 bits, o AES128. Para tal, é utilizado um [RTL](#) disponível no site [Open Cores](#) e ferramentas de síntese da empresa *Cadence*.

2 Metodologia

2.1 Descrição do Core

O AES128 implementa o Padrão Avançado de Criptografia (Rijndael Algorithm) de acordo com o padrão NIST. Três membros da família Rijndael são especificados neste Padrão: AES-128, AES-192 e AES-256. Cada um deles transforma dados em blocos de 128 bits e o sufixo numérico indica o comprimento de bits das chaves criptográficas associadas. O tamanho da chave usado para uma cifra AES especifica o número de rodadas de transformação que convertem a entrada (chamada de texto simples) na saída final (chamada de texto cifrado). O número de rodadas é o seguinte:

- 10 rodadas para chaves de 128 bits.
- 12 rodadas para chaves de 192 bits.
- 14 rodadas para chaves de 256 bits.

O núcleo AES utilizado é desenvolvido para um tamanho de chave de 128 bits e opera no modo ECB (Electronic Codebook). O projeto contém um Register gate level (RTL) sintetizável junto com um Test Bench (TB) configurado para verificar o core com vetores de teste conforme descrito no documento FIPS (Federal Information Processing Standards).

Este é um core de alta velocidade e de baixa latência, comparado à seus similares. O RTL e o Test Bench disponibilizados no site [Open Cores](#) são apresentado em Verilog e já foram implementados/provados em FPGA.

2.2 Modo de operação AES-128

O AES segue um procedimento para criptografia que consiste em substituição de bytes, shift de colunas, mix de colunas e adição de chave da rodada. Nas Figuras 1 e 2 são mostrados diagramas com o funcionamento do circuito[2].

- KeyExpansion: As chaves da rodada são derivadas da chave cifrada
- Initial round key: Cada byte do estado é combinado com um byte da chave de rodada usando xor bit a bit.
- 9 iterações:
 - SubBytes: Usa SBox Look-up Table (LUT) para substituir cada byte nos dados de texto simples de 128 bits.

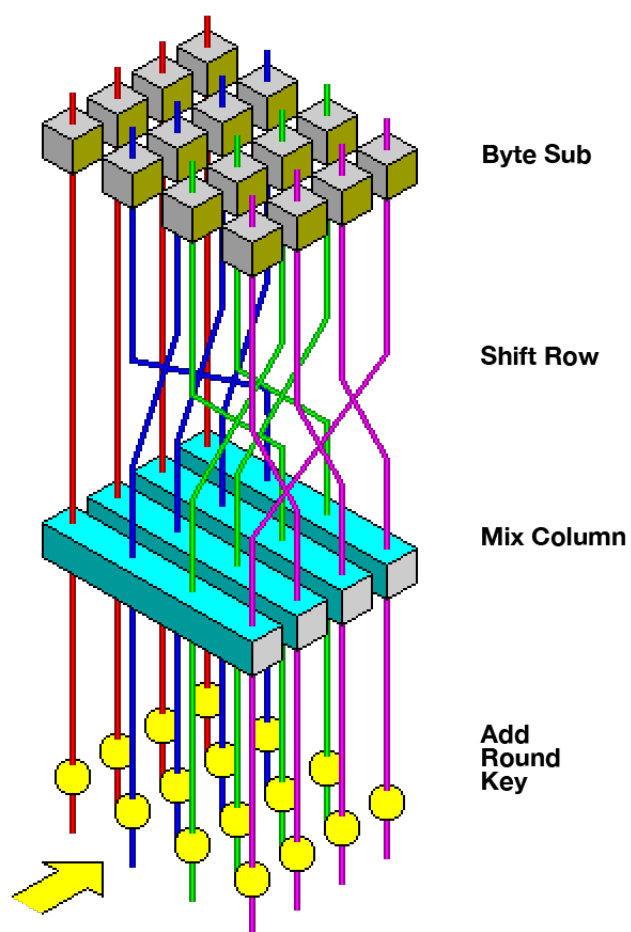


Figura 1: Demonstração de funcionamento do algoritmo [1]

- Shift rows: Organiza os dados na matriz de estado e mudar as linhas dessa matriz.
- Mix columns: Mistura as colunas de dados.
- Add Round key: Cada byte do estado é combinado com um byte da chave de rodada usando xor bit a bit.
- Última rodada:
 - SubBytes
 - ShiftRows
 - AddRoundKey

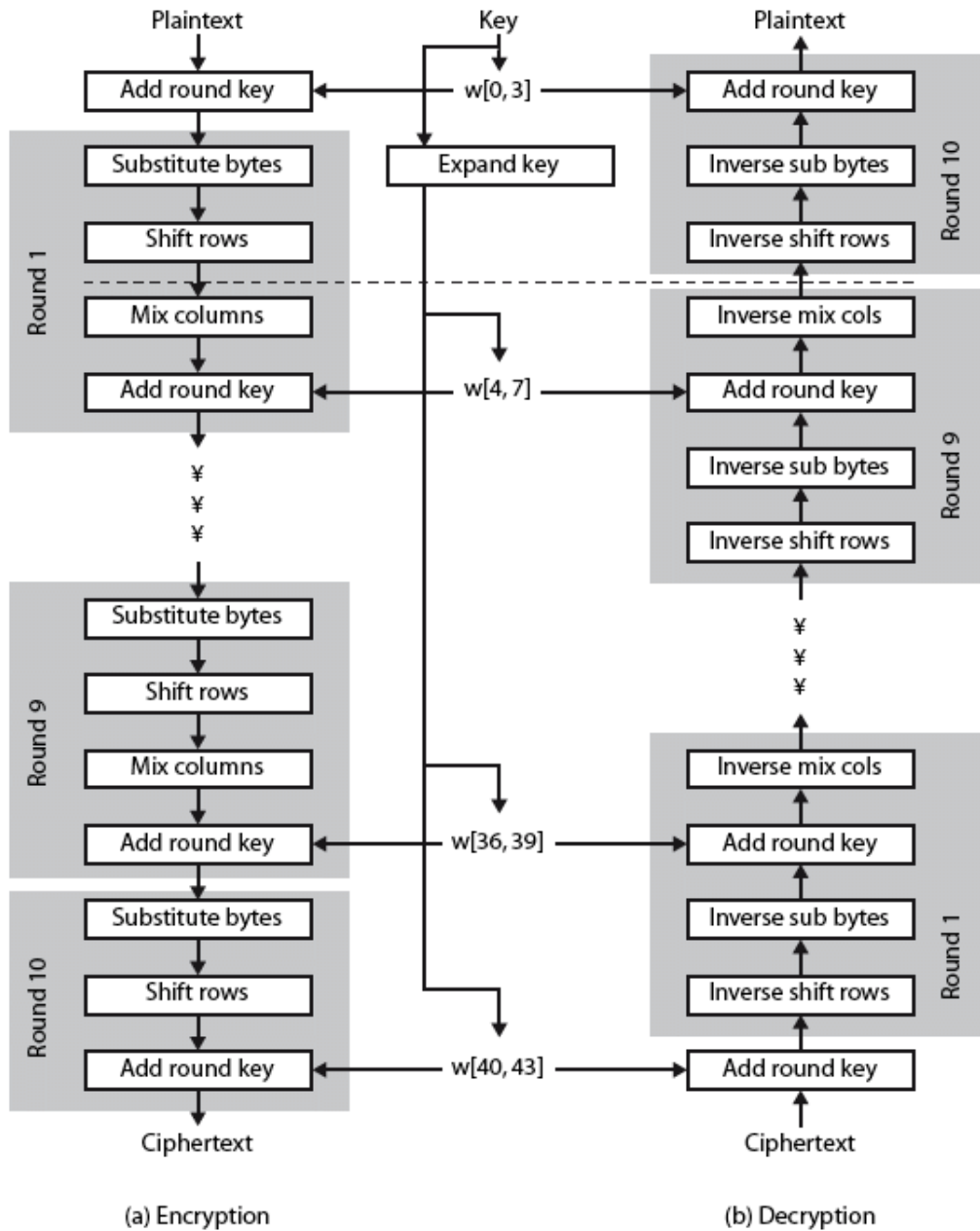


Figura 2: Demonstração de funcionamento do algoritmo [2]

2.3 Síntese lógica

A síntese lógica é uma etapa crítica no design digital que traduz uma descrição de hardware de alto nível em uma netlist em nível de porta. Nessa fase é feito um mapeamento independente da tecnologia utilizada, algumas otimizações que dependem da tecnologia e a inserção da cadeia de *Scan Chain*.

Foi feita uma análise de tempo que tem por objetivo verificar se o design respeita os requisitos

do projeto. Essa análise determina o quão rápido o circuito pode funcionar sem violar restrições de tempo. Para iniciar a síntese lógica, estima-se um período grande (em ns) no arquivo de *Constraints* de forma que não sejam violadas as restrições de tempo. Executa-se então a síntese e verifica-se se o *slack* ainda se mantém positivo. De forma iterativa, altera-se o período nas *Constraints* até que seja encontrado o menor valor que não viole as restrições de tempo.

Quando falamos de *clock*, é importante definir alguns conceitos como:

- Insertion Delay: atraso conhecido da *Clock tree* em qualquer ponto
- Clock uncertainty: soma do *clock skew* + *clock jitter* (Figura 3) é uma variação desconhecida que é estimada pelo projetista

No momento Pre-CTS (antes da *Clock tree synthesis*) clocks ideais com uma incerteza adicionada são usados. Neste projeto, utilizou-se 0.3 (ps) como *clock uncertainty*.

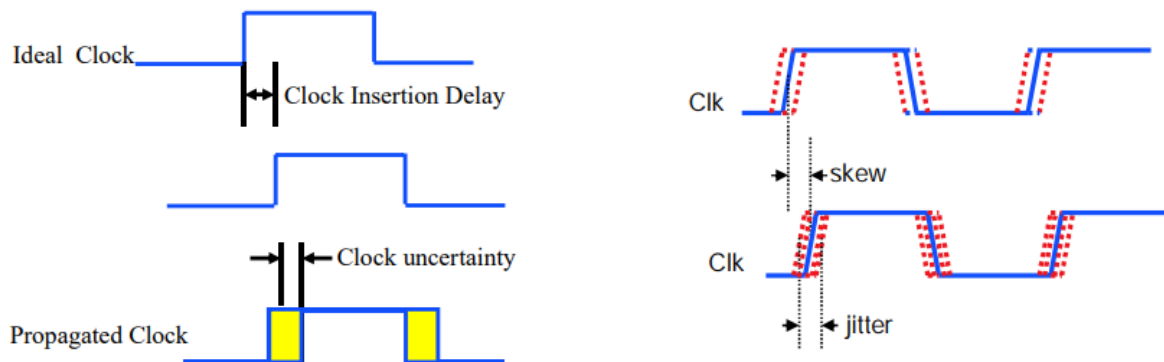


Figura 3: Definições de *Clock*

A performance dos circuitos integrados depende das características PVT (*Process, Voltage, Temperature*). Cada lote de wafers é fabricado com pequenas variações que podem provocar alterações nas características de processo. Assim, avalia-se os parâmetros atingidos em 3 situações, típica, melhor e pior.

Com a síntese lógica analisa-se os valores obtidos no relatório de timing. Uma violação de *setup time* é quando um sinal chega tarde demais e perde o tempo que deveria avançar (negativo no relatório). Uma violação do *hold time* é quando um sinal chega muito cedo e avança um ciclo do clock antes do que deveria (positivo no relatório), como é ilustrado na Figura 4.

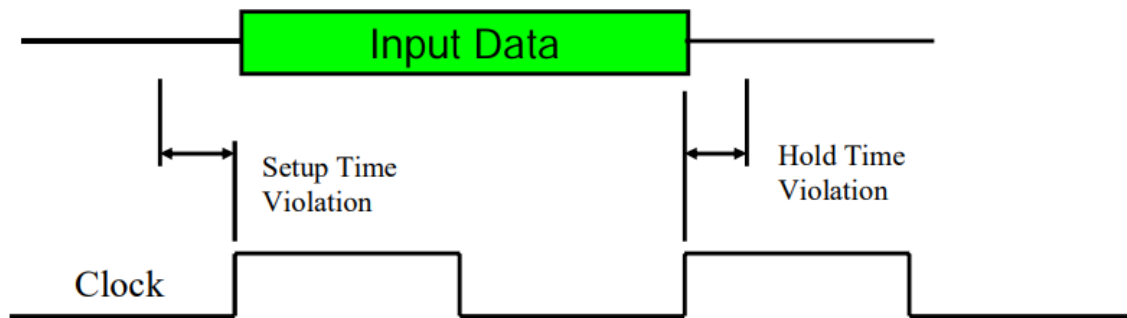


Figura 4: Definições de *Setup time* e *Hold time*

A inserção da cadeia de teste do tipo *Scan-chain* foi feita adicionando no arquivo RTL do módulo AES128 o input *scan_enable* conforme é mostrado na Figura 5.

```

karina.santos@pgmicro02:~/MIC61_karina/rtl
File Edit View Search Terminal Help
GNU nano 2.3.1 File: aes_128.v

* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

module aes_128(clk, scan_enable, state, key, out);
    input      clk, scan_enable;
    input  [127:0] state, key;
    output [127:0] out;
    reg      [127:0] s0, k0;
    wire     [127:0] s1, s2, s3, s4, s5, s6, s7, s8, s9,
                    k1, k2, k3, k4, k5, k6, k7, k8, k9,
                    k0b, k1b, k2b, k3b, k4b, k5b, k6b, k7b, k8b, k9b;

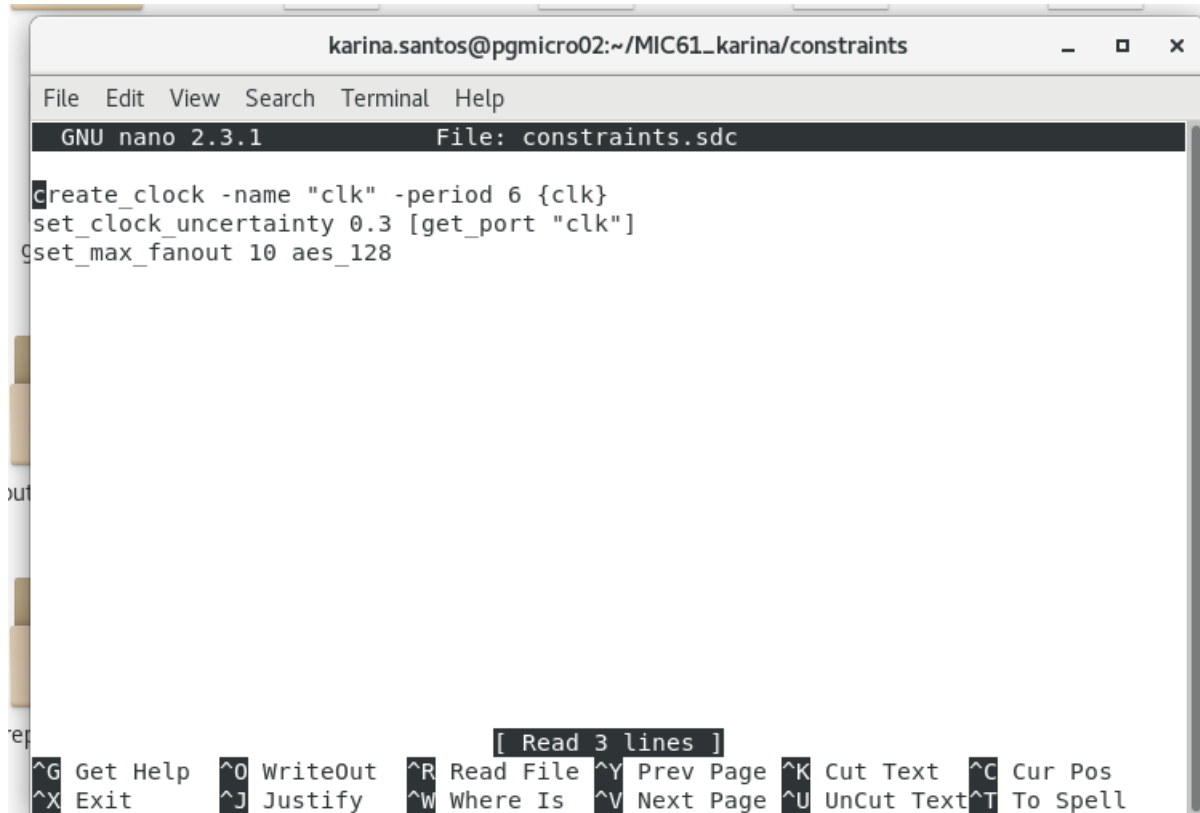
    always @ (posedge clk)
    begin
        s0 <= state ^ key;
        k0 <= key;
    end

```

[^]G Get Help [^]O WriteOut [^]R Read File [^]Y Prev Page [^]K Cut Text [^]C Cur Pos
[^]X Exit [^]J Justify [^]W Where Is [^]V Next Page [^]U UnCut Text [^]T To Spell

Figura 5: Arquivo de RTL

O arquivo de restrições utilizado é mostrado na Figura 6. Inicialmente definiu-se um período de 10 e com essas definições, inicia-se a execução do script dentro da ferramenta *Genus*.



```
karina.santos@pgmicro02:~/MIC61_karina/constraints
File Edit View Search Terminal Help
GNU nano 2.3.1 File: constraints.sdc

create_clock -name "clk" -period 6 {clk}
set_clock_uncertainty 0.3 [get_port "clk"]
set_max_fanout 10 aes_128

[ Read 3 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

Figura 6: Arquivo de constraints

2.4 Síntese física

Na etapa de síntese física é feita a integração da síntese lógica com o posicionamento. Utiliza-se o fluxo com predição PLE e QoS.

O primeiro passo para iniciar a síntese física foi a adição dos PADS ao arquivo datapath.v que é gerado na pasta layout após a síntese lógica, conforme é mostrado na Figura 7.


```

module aes_128_top(clk_pad, scan_enable_pad, state_pad, key_pad, out_pad, clk_pad, scan_enable_pad, state_pad, key_pad, out_pad);
    input clk_pad, scan_enable_pad;
    input [127:0] state_pad, key_pad;
    output [127:0] out_pad;

    wire clk, scan_enable;
    wire [127:0] state, key;
    wire [127:0] out;

    aes_128 bloco(.);

    ICP PAD_clk(.PAD(clk_pad), .Y(clk));
    ICP PAD_scan_enable(.PAD(scan_enable_pad), .Y(scan_enable));

    ICP PAD_state_pad0(.PAD(state_pad[0]), .Y(state[0]));

    ICP PAD_key_pad0(.PAD(key_pad[0]), .Y(key[0]));

    BT8SP PAD_out0(.PAD(out_pad[0]), .A(out[0]));

endmodule

```

Figura 7: Inserção de pads no arquivo datapath.v

Em seguida inicia-se a criação do arquivo IO através do Innovus. Para isso, faz-se a importação do design. Ao fazer a importação do design abrem-se as opções de importação e neste momento adicionam-se os arquivos .lef. Esses são os arquivos que carregam as informações das bibliotecas que serão utilizadas, em referência às camadas de metais, células do core e modelos dos PADS.

No arquivo .io é possível observar que cada instância dos PADS apresenta valores de offset gerados automaticamente e que o arquivo está dividido em blocos (left, top, bottom e right). Estes blocos definem o posicionamento dos PADS em cada lado no layout seguindo a nomenclatura dos blocos [3].

Para a execução da síntese física, definem-se alguns parâmetros, conforme Figura 8, no arquivo physical.syn.tcl que é o arquivo que contém o script para a execução da síntese física no Innovus. Esses parâmetros fazem a limitação de espaço do circuito e com essas condições a ferramenta executa o melhor posicionamento possível.

```

##Generating square floorplan (1) with 70% of density (0.7) with 3um margins (3 3 3 3)
create_floorplan -site core -match_to_site -core_density_size 1 0.7 1000 1000 1000 1000
#resize_floorplan -x_size 2 -y_size 20
gui_fit
puts "done"

```

Figura 8: Definições necessárias para a execução da síntese física

Após essas configurações, inicia-se a execução do script dentro da ferramenta *Innovus*.

2.5 LEC - Logic equivalence check

A Verificação de Equivalência Lógica (LEC) é uma técnica de verificação formal usada em design digital e verificação para garantir que duas representações diferentes de um design, normalmente um design RTL e uma netlist em nível de porta, sejam funcionalmente equivalentes. O

objetivo principal do LEC é verificar se não existem diferenças funcionais entre estas duas representações. LEC é uma etapa essencial nos fluxos de projeto ASIC e FPGA para detectar erros de projeto e garantir que o processo de síntese lógica não introduziu alterações indesejadas.

Na Figura 9 é mostrado o comando utilizado para abrir a ferramenta *Conformal* e na Figura 10 a visualização da *GUI* da ferramenta.

```
[karina.santos@pgmicro02 lec]$ lec -xl
CONFORMAL (R)
Version 17.20-s300 (10-Feb-2018) (64 bit executable)
Copyright (c) Cadence Design Systems, Inc., 1997-2018. All Rights Reserved

This program is proprietary and confidential information belonging to
Cadence Design Systems, Inc., and may be used and disclosed only as authorized
in a license agreement controlling such use and disclosure.

// Warning: This version is 2027 days old. You can download the latest version f
rom http://downloads.cadence.com.

// Check out Conformal_Ultra 17.2 license
// Check out Conformal_Asic 17.2 license
```

Figura 9: Inicialização do *Conformal*

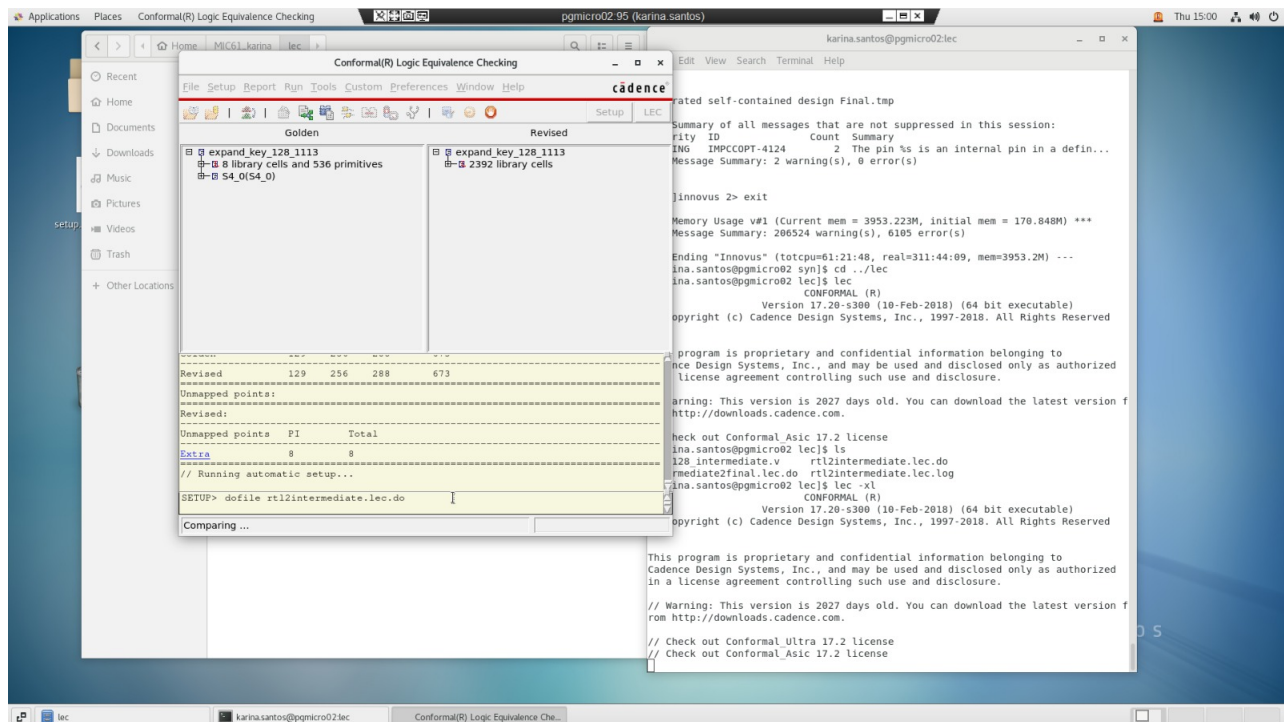


Figura 10: GUI *Conformal*

No diagrama da Figura 11 fica ilustrado os arquivos .v de entrada e saída para as etapas de mapeamento. Utiliza-se a ferramenta LEC duas vezes, uma comparando o arquivo .v do RTL e o

arquivo .v intermediário e outra comparando o arquivo .v intermediário e o arquivo .v já mapeado para biblioteca.

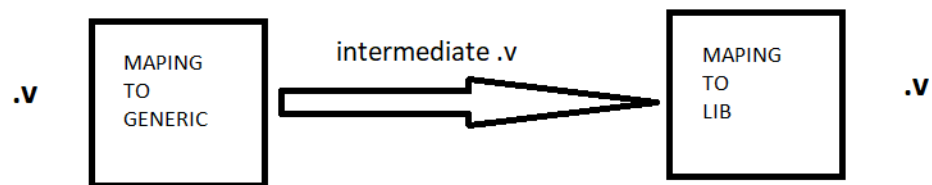


Figura 11: Diagrama que mostra arquivos .v de entrada e saída para etapas de mapeamento

3 Resultados e Discussões

3.1 Síntese para FPGA

No próprio site do [OpenCores](#) é disponibilizado os resultados obtidos ao sintetizar o AES128 para FPGA. Tais dados são mostrados na Figura 12. Nas próximas semanas será feita a síntese desse RTL novamente e documentada a experiência pessoal em tal feito.

Synthesis result

Xilinx XC6VLX240T-1FF1156
(by Xilinx ISE version 14.2)

	AES-128	AES-192	AES-256
Number of Slice Registers:	3,968	5,280	6,848
Number of Slice LUTs:	3,536	4,264	6,503
Number of bonded IOBs:	385	449	513
Number of Block RAM/FIFO:	86	100	121
Number of BUFG/BUFGCTRLs:	1	1	1

Figura 12: Resultados para a síntese para FPGA do Core AES128

3.2 Síntese lógica

Iniciou-se a síntese utilizando um período de 10ns. Como o *slack* estava muito alto, baixou-se o período até que chegasse a um número próximo de zero e assim pode-se definir a frequência máxima de operação do circuito.

O cabeçalho do relatório de tempo é mostrado na Figura 13. A frequência máxima de operação encontrada foi de 166,66 MHz, obtendo *timing closure*, isto é, respeitando as restrições de tempo.

```

=====
Generated by:      Genus(TM) Synthesis Solution GENUS15.22 - 15.20-s024_1
Generated on:      Jul 22 2023  05:56:17 pm
Module:           aes_128
Interconnect mode: global
Area mode:        physical library
=====

Path 1: MET (13 ps) Setup Check with Pin r2/t1_t0/s0_out_reg[0]/C->D
View: worst_view
Group: C2C
Startpoint: (R) r1/state_out_reg[92]/C
Clock: (R) clk
Endpoint: (F) r2/t1_t0/s0_out_reg[0]/D
Clock: (R) clk

      Capture      Launch
Clock Edge:+      6000      0
Src Latency:+      0      0
Net Latency:+      0 (I)    0 (I)
Arrival:=         6000      0

      Setup:-      496
      Uncertainty:- 300
Required Time:=    5204
Launch Clock:-     0
Data Path:-        5191
Slack:=           13

#-----
# Timing Point      Flags  Arc  Edge  Cell      Fanout  Load Trans Delay Arrival
#                  (fF)  (ps) (ps) (ps)
#-----
r1/state_out_reg[92]/C -    -    R    (arrival)  6105    -    0    -    0
r1/state_out_reg[92]/Q -    C->Q R    SDFRQX2    26 120.7  755  787  787
r2/t1_t0/g17283/Q     -    A->Q F    INX1       18  74.1  547  460 1246
r2/t1_t0/g17235/Q     -    B->Q F    AND2X1     15  78.2  516  595 1842
r2/t1_t0/g17220/Q     -    A->Q R    INX1       12  45.8  399  331 2172
r2/t1_t0/g17203/Q     -    A->Q F    NO2X1       6  25.5  298  195 2367
r2/t1_t0/g17077/Q     -    C->Q F    A021X0      1  4.6  210  445 2812
r2/t1_t0/g16833/Q     -    B->Q F    A0222X0     1  4.5  281  831 3643
r2/t1_t0/g16755/Q     -    D->Q F    A0211X0     1  4.8  304  668 4311
r2/t1_t0/g16733/Q     -    D->Q R    AN211X0     1  5.1  621  400 4711
r2/t1_t0/g16706/Q     -    D->Q F    NA6I3X1     1  5.5  105  479 5191
r2/t1_t0/s0_out_reg[0]/D <<< -    F    SDFRQX0     1  -    -    0  5191
#-----

```

Figura 13: Worst Time para T=6ns

Com essa frequência, o *Slack* obtido foi de 13ps. Testou-se também utilizar um período de 5ns, o que resultaria em uma frequência de 200MHz e com essa definição teria-se um *Slack* de 0. Como precaução, optou-se escolher o período de 6ns para garantir funcionalidade na síntese física.

Na Tabela 1 são mostrados os períodos testados e slacks correspondentes obtidos.

Tabela 1: Slacks obtidos na síntese lógica do AES-128 para diferentes períodos

Período (ns)	Worst Slack	Typical Slack	Best Slack
2	-449	304	694
4	0	1328	2007
5	0	1711	2569
6	13	2077	3121
10	13698	-	-

O caminho crítico do projeto, para esta frequência de operação é dado pelo *r1 state_out_reg[92]* *C* até o *r2 t1_t0 s0_out_reg[0]* *D*, conforme Figura 13.

A quantidade de instâncias, em número de flip-flops e latches é 6105, conforme pode-se ver no relatório de worst gates na Figura 14.

Type	Instances	Area	Area %
sequential	6105	404065.318	19.8
inverter	19490	119960.014	5.9
buffer	8	98.381	0.0
logic	101432	1519325.438	74.4
physical_cells	0	0.000	0.0
total	127035	2043449.150	100.0

Figura 14: Worst Gates para T=6ns

A área estimada, para o pior caso, do design, sem pads ficou em $3,895mm^2$, conforme mostra o relatório da Figura 15.

=====					
Generated by: Genus(TM) Synthesis Solution GENUS15.22 - 15.20-s024_1					
Generated on: Jul 22 2023 05:56:10 pm					
Module: aes_128					
Library domain: worst_timing_cond					
Domain index: 0					
Technology libraries: D_CELLS_M0SST_slow_1_62V_125C V 2.1.0					
IO_CELLS_3V_M0S3ST_slow_1_62V_3_30V_125C 1.1.0					
Operating conditions: slow_1_62V_125C					
Interconnect mode: global					
Area mode: physical library					
=====					
Instance	Domain	Cells	Cell Area	Net Area	Total Area

aes_128	worst_timing_cond	127035	2043449	1851368	3894817
r1	worst_timing_cond	10906	165317	148605	313921
t0_t0	worst_timing_cond	923	13088	12504	25591
t1_t3	worst_timing_cond	922	13103	12401	25504
t3_t2	worst_timing_cond	921	13032	12423	25455
t2_t1	worst_timing_cond	599	8664	8282	16946
t1_t1	worst_timing_cond	594	8651	8259	16911
t2_t0	worst_timing_cond	611	8562	8186	16748
t2_t3	worst_timing_cond	602	8525	8122	16648
t1_t0	worst_timing_cond	599	8430	8034	16464
t3_t1	worst_timing_cond	597	8439	8016	16455
t1_t2	worst_timing_cond	597	8430	8012	16442
t0_t3	worst_timing_cond	601	8418	8020	16438
t3_t0	worst_timing_cond	600	8418	8002	16419
t0_t2	worst_timing_cond	593	8439	7978	16418
t2_t2	worst_timing_cond	600	8408	7998	16406
t3_t3	worst_timing_cond	591	8390	7956	16346
t0_t1	worst_timing_cond	572	8190	7774	15964
r8	worst_timing_cond	10846	164766	148225	312991
t1_t3	worst_timing_cond	914	13008	12341	25349
t0_t0	worst_timing_cond	914	13008	12341	25349

Figura 15: Worst Area para T=6ns

O circuito possui um alto número de PADs, o que o caracteriza como PAD-limited, ou seja, sua área total será delimitada pelos PADs.

Os demais arquivos referentes à síntese lógica estão em anexo neste documento.

3.3 Síntese Física

Na Figura 16 observamos como o chip se apresentou após a inserção dos pads. Em seguida, na Figura 17 pode-se observar a etapa de floorplan, que é o momento em que se limita a área que irá conter as células lógicas. No primeiro momento escolheu-se uma margem de 1000 e notou-se que o circuito ficou muito espalhado pelo módulo, então escolheu-se aumentar a margem para 1400, compactando mais o circuito.

Na Figura 18 observamos a inserção dos power rings e power trunks que são utilizados para alimentar o core. Na Figura 19 pode-se observar o placement executado. Na próxima etapa insere-se a árvore de clock (Figura 20) e então executa-se o roteamento (Figura 21) e a inserção de *fillers* (Figura 22) como etapa final.

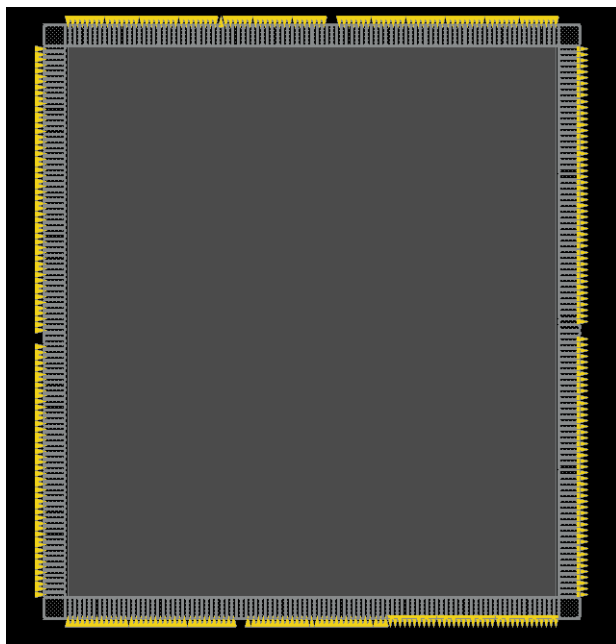


Figura 16: Primeira etapa da síntese física - inserção dos pads

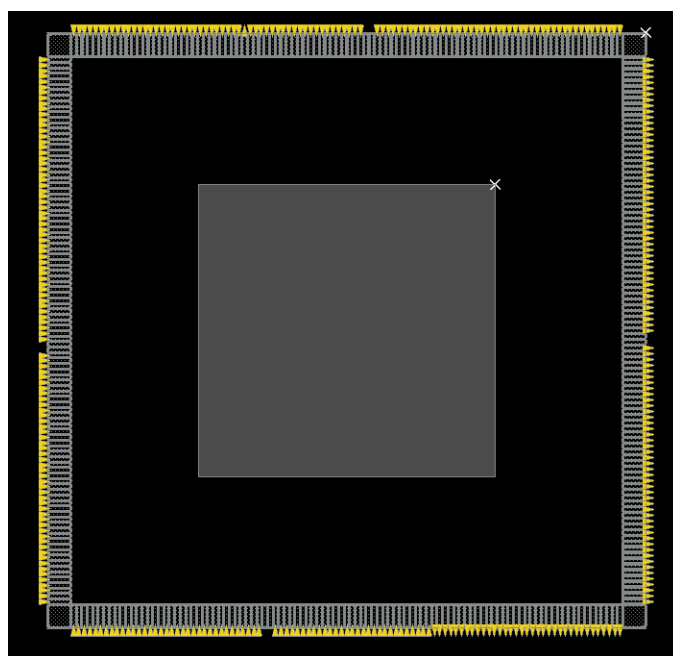


Figura 17: Segunda etapa da síntese física - Floorplan

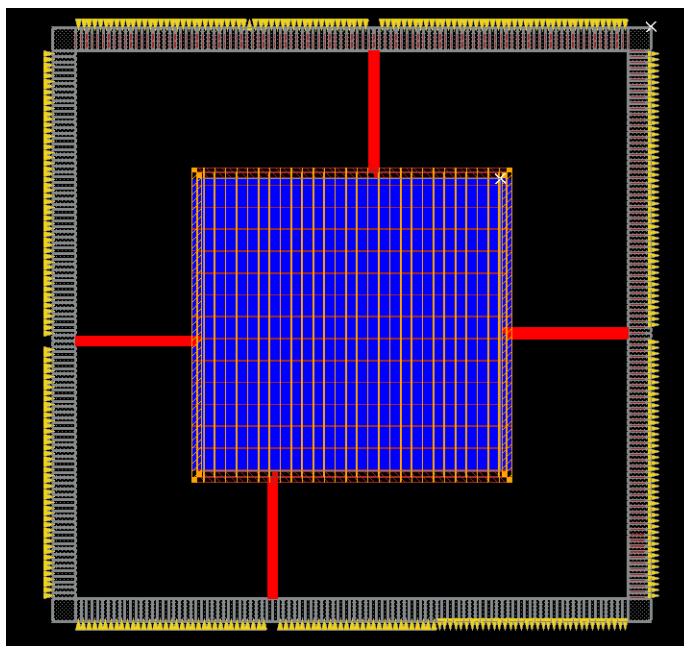


Figura 18: Terceira etapa da síntese física - Power deliver network

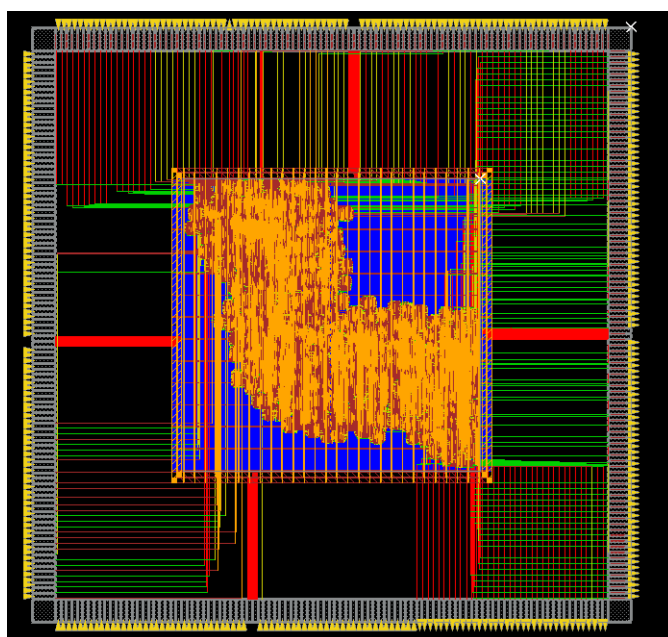


Figura 19: Quarta etapa da síntese física - Placement

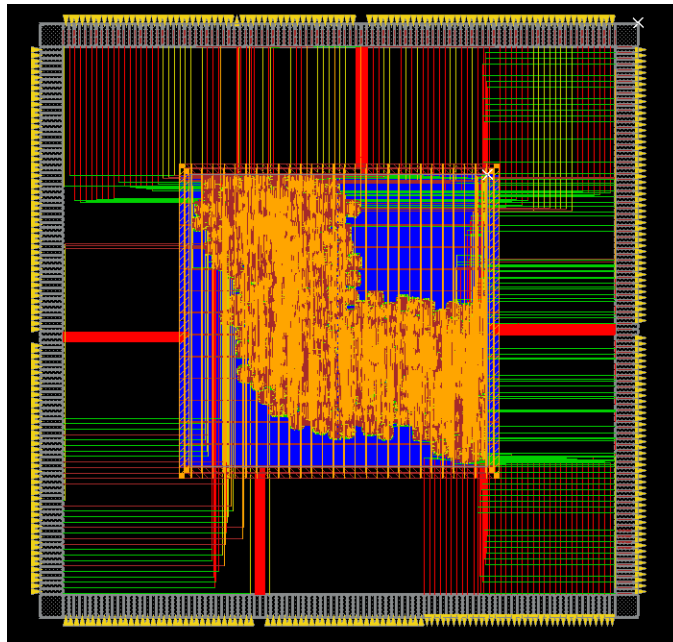


Figura 20: Quinta etapa da síntese física - Clock tree synthesis

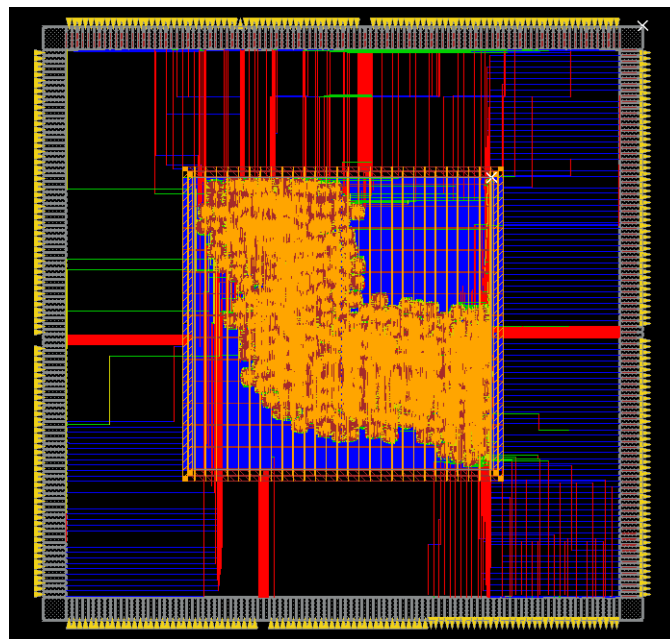


Figura 21: Sexta etapa da síntese física - Routing

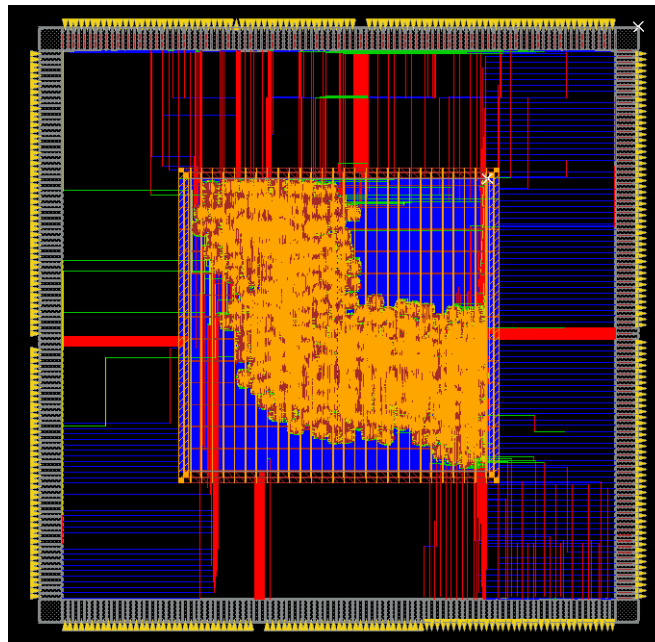


Figura 22: Sétima etapa da síntese física - Inserção de fillers

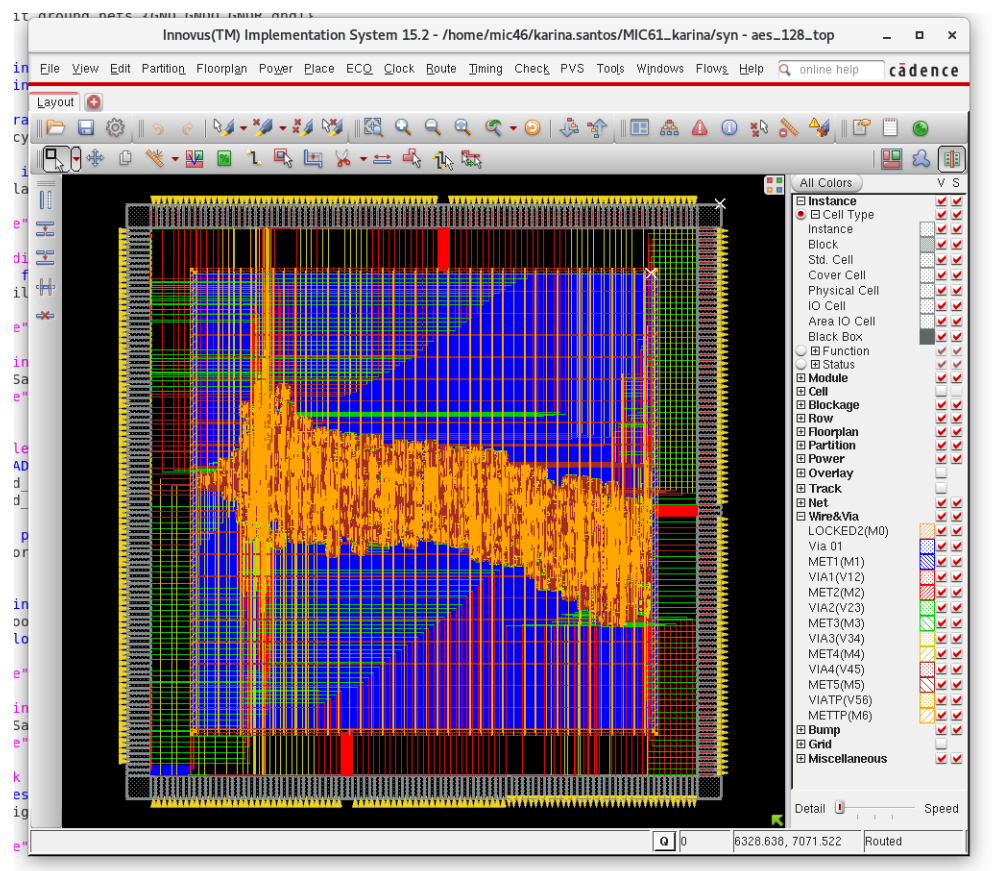


Figura 23: Resultado da primeira síntese física executada com margem de 1000

Na Figura 24 pode-se verificar que não foram encontrados erros de DRC.

```
Verification Complete : 0 Viols.  
*** End Verify DRC (CPU: 0:01:47  ELAPSED TIME: 16.00  MEM: 139.3M) ***  
RC Out has the following PVT Info:  
  RC:default_emulate_rc_corner, Operating temperature 25 C  
rcOut completed:: 39 % █
```

Figura 24: Comprovação de ausência de violação de DRC

Na Figura 25 podemos observar a árvore de buffers do clock.

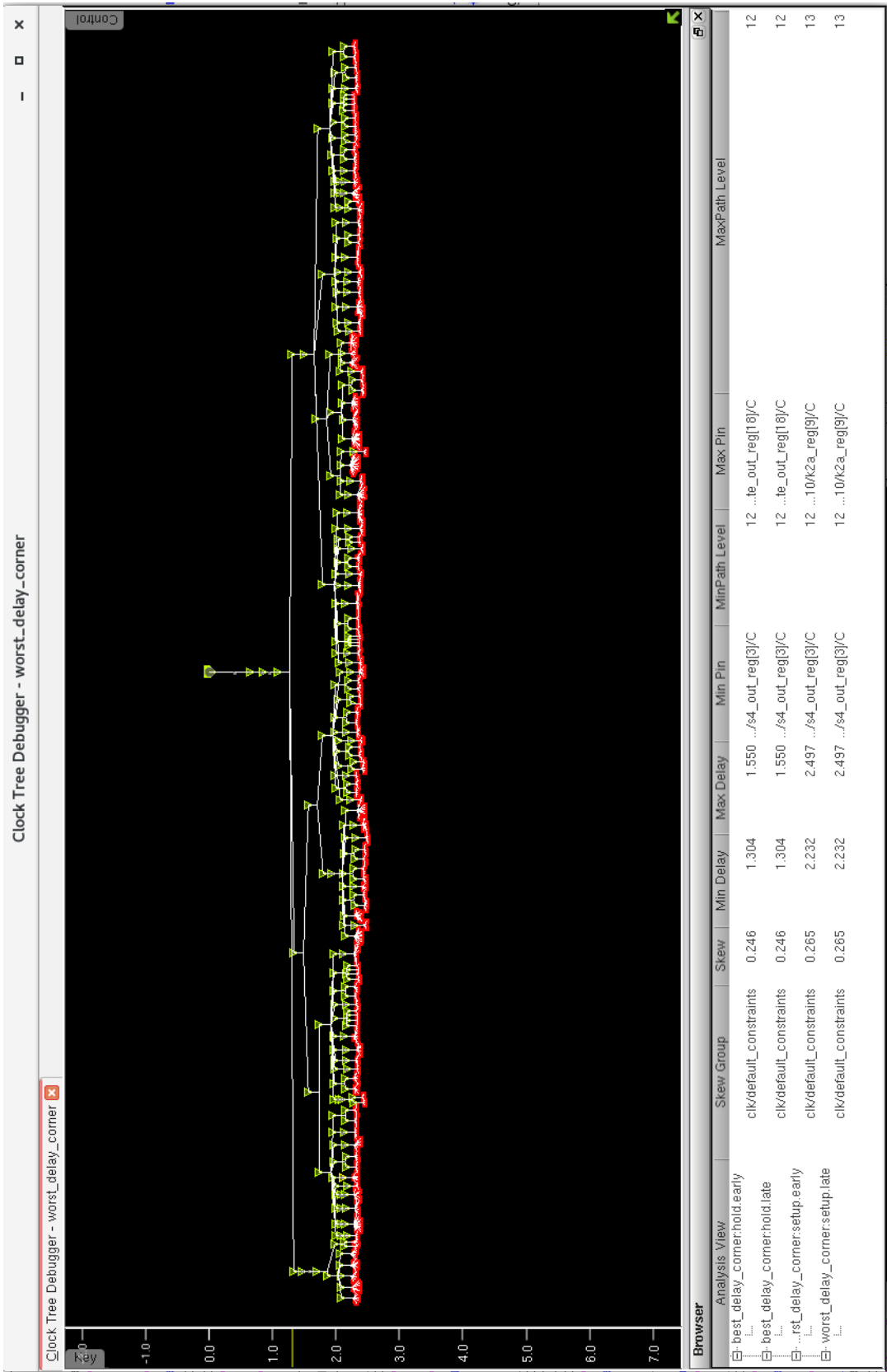


Figura 25: Geração da árvore de buffers de clock

```
#####
# Generated by:      Cadence Innovus 15.20-p005_1
# OS:               Linux x86_64(Host ID pgmicro02)
# Generated on:      Fri Aug 18 16:21:45 2023
# Design:           aes_128_top
# Command:          report_timing > pos_route_timing_report.txt
#####
Path 1: MET (0.059 ns) Setup Check with Pin bloco/r2/state_out_reg[7]/C->D
    View:worst_view
    Group:clk
    Startpoint:(R) bloco/a2/k3a_reg[7]/C
    Clock:(R) clk
    Endpoint:(R) bloco/r2/state_out_reg[7]/D
    Clock:(R) clk

    Capture          Launch
    Clock Edge:++    6.000      0.000
    Src Latency:++   0.000     -2.086
    Net Latency:++   2.165 (P)   2.204 (P)
    Arrival:=        8.165      0.118

    Setup:-          0.448
    Uncertainty:-     0.300
    Required Time:=   7.417
    Launch Clock:-    0.118
    Data Path:-       7.240
    Slack:=           0.059
```

Figura 27: Relatório de tempos pós roteamento para AES128

A área total com PADs ficou em $8,521mm^2$, a potência máxima estimada 426,8mW (Figura 26) e a frequência máxima de operação ficou em 136MHz, conforme Figura 27. O número de instâncias total ficou em 131555. Em anexo à este relatório são adicionados os *reports* relevantes à esta síntese.

Total Power		

Total Internal Power:	275.77902734	64.6143%
Total Switching Power:	150.85075167	35.3439%
Total Leakage Power:	0.17859140	0.0418%
Total Power:	426.80837615	

Figura 26: Consumo de potência do core AES128

3.4 LEC - Logic Equivalence Check

O resultado da LEC para o AES128 feita com o *Conformal* para o arquivo .v RTL para o .v intermediário é mostrado na Figura 28.

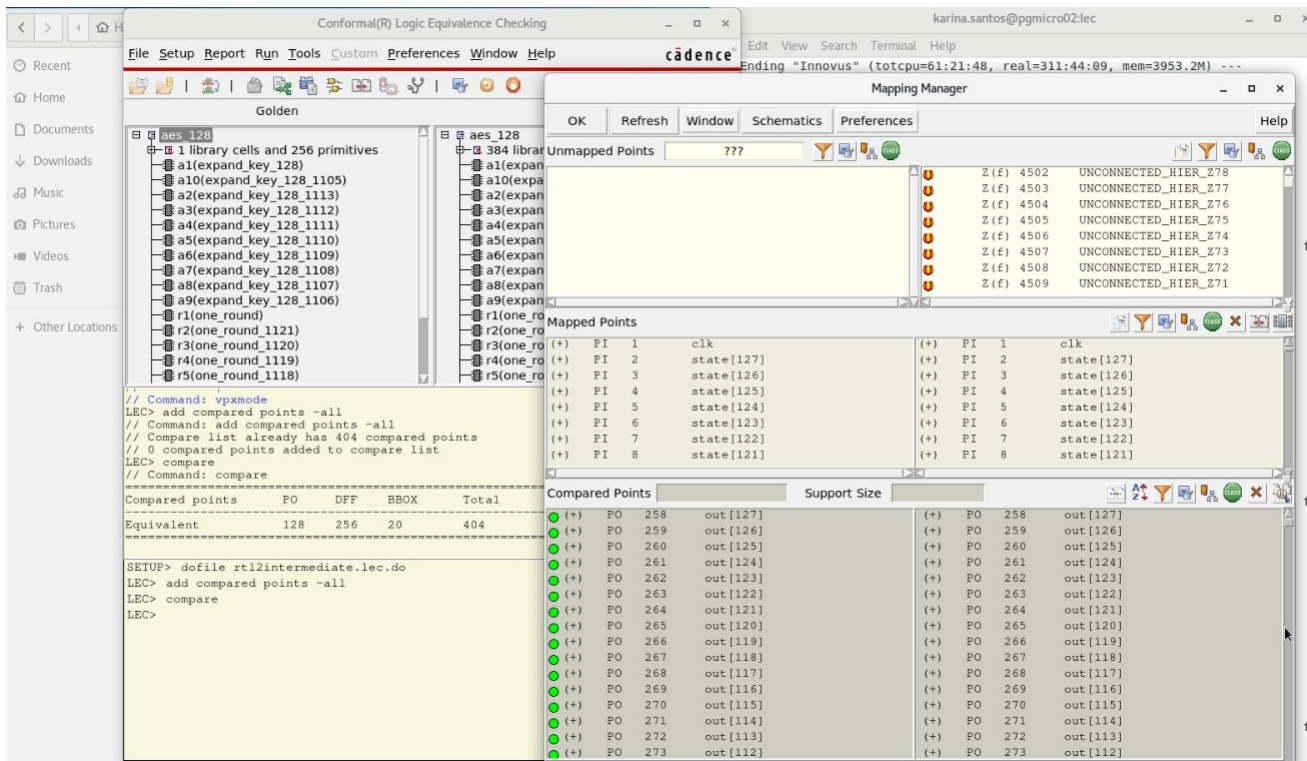


Figura 28: Comprovação de execução de LEC para arquivo .v RTL e arquivo .v intermediário

Para a comparação entre o arquivo intermediário com o final, o resultado é o mostrado na Figura 29.

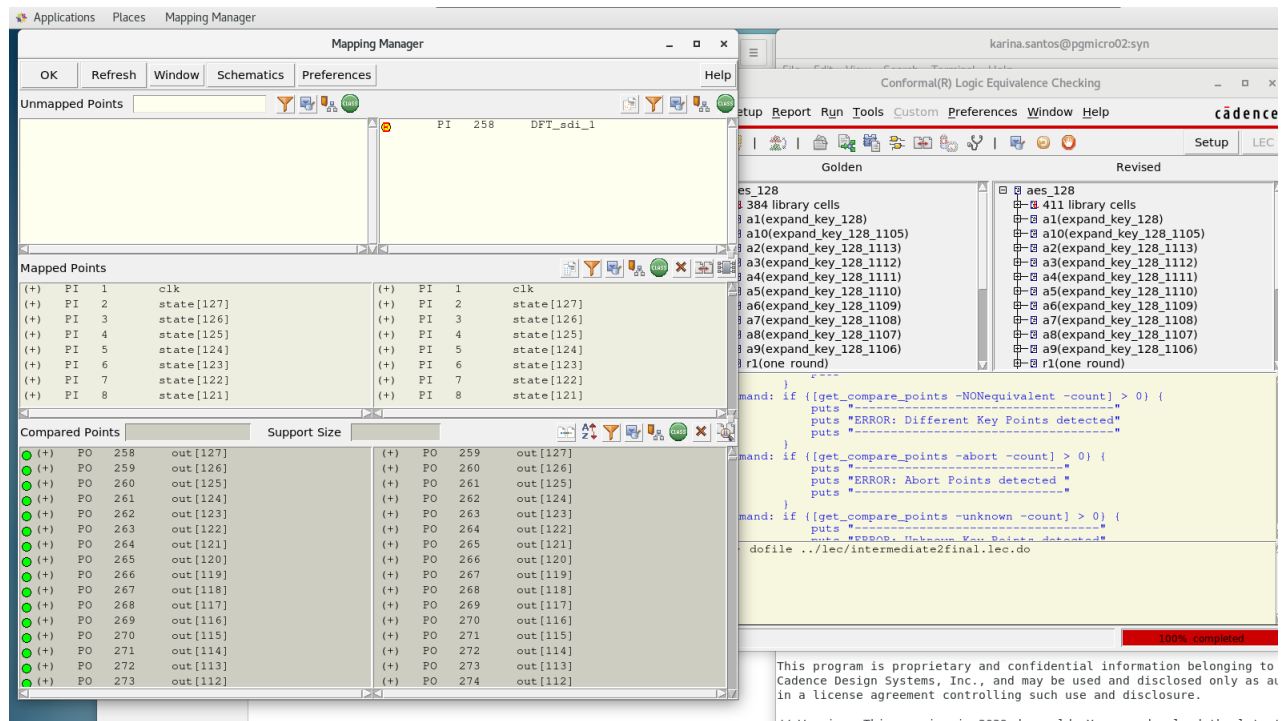


Figura 29: Comprovação de execução de LEC para arquivo .v intermediário e arquivo .v final

4 Conclusão

Foi apresentada a execução do fluxo de digital design para o módulo AES128 utilizando softwares da Cadence. Partiu-se de uma descrição RTL e chegou-se ao arquivo GDS passando por etapas de síntese lógica, síntese física e verificação.

O módulo sintetizado apresentou uma área total de $3,895mm^2$ sem pads e de $8,521mm^2$ com pads. A potência máxima estimada foi de 426,8mW e a frequência máxima de operação ficou em 136MHz.

O fluxo de design digital é um processo altamente iterativo e cada etapa envolve uma combinação de atividades de design, verificação e validação. A execução do fluxo completo traz uma aprendizagem de como transformar projetos em nível de código (descrição high-level, RTL) em circuitos integrados, acompanhando todas as etapas necessárias para criar um hardware ASIC.

Durante a execução do fluxo, são necessárias diferentes ferramentas de software, cada uma desempenhando um papel específico em uma etapa particular. O processo é altamente especializado e requer conhecimento técnico para otimizar os projetos da melhor forma possível. A colaboração entre engenheiros de hardware e software é essencial para assegurar o funcionamento adequado do produto final.

Referências

- [1] [https://commons.wikimedia.org/wiki/file:aes\(rijndael\)_roundfunction.png](https://commons.wikimedia.org/wiki/file:aes(rijndael)_roundfunction.png). 2023.
- [2] Devyani Madhukar Mirajkar. Design and verification of a pipelined advanced encryption standard (aes) encryption algorithm with a 256-bit cipher key using the uvm methodology. *Rochester Institute of Technology*, 2018.
- [3] Pedro Tauã L. Pereira. Tutorial de inserção de pads. Outubro de 2021.