Pontifícia Universidade Católica do Rio Grande do Sul

**Relatório Trabalho I – Etapa síntese física**

**Alunos:** Alexandre Carvalho / Daniel Vega

**Disciplina:** Projetos de Sistemas Integrados II

**Professora:** Letícia Pöhls

Conteúdo

[Etapas da síntese física 3](#_Toc485339777)

[Environment Configuration 3](#_Toc485339778)

[Floorplaning 3](#_Toc485339779)

[Pin Editor 3](#_Toc485339780)

[Power Planning 4](#_Toc485339781)

[Placement 5](#_Toc485339782)

[Optimization Post Placement 6](#_Toc485339783)

[Clock Tree Synthesis 6](#_Toc485339784)

[Optimization Post CTS 6](#_Toc485339785)

[Nano Route 6](#_Toc485339786)

[Verification 6](#_Toc485339787)

[Alterações para melhorar as métricas de code coverage 8](#_Toc485339788)

[Toggle 8](#_Toc485339789)

[Branch 9](#_Toc485339790)

[Síntese comportamental 10](#_Toc485339791)

[Resultados comentados dos relatórios gerados pela ferramenta 10](#_Toc485339792)

[Fanout caminho crítico 10](#_Toc485339793)

[Escolha do clock 10](#_Toc485339794)

[Gráficos dos resultados gerados pela ferramenta 11](#_Toc485339795)

[Timing 11](#_Toc485339796)

[Area 11](#_Toc485339797)

[Power 12](#_Toc485339798)

# Etapas da síntese física

## Environment Configuration

Nesta etapa definimos o nodo tecnológico de 65nm e também devemos informar os arquivos obtidos da síntese comportamental, no caso deste trabalho, serão os arquivos de output do *Genus* para a célula nos *corners typical, worst case* e *best case.*

## Floorplaning

A etapa de *floorplaning* consiste na definição de alguns parâmetros tais como altura e largura da célula, utilização do core (neste caso 70%) e também os valores da margem entre o *core* e as bordas da célula que para este trabalho foi de 20. Após a realização desta etapa o *design* ficou conforme a Figura 1.

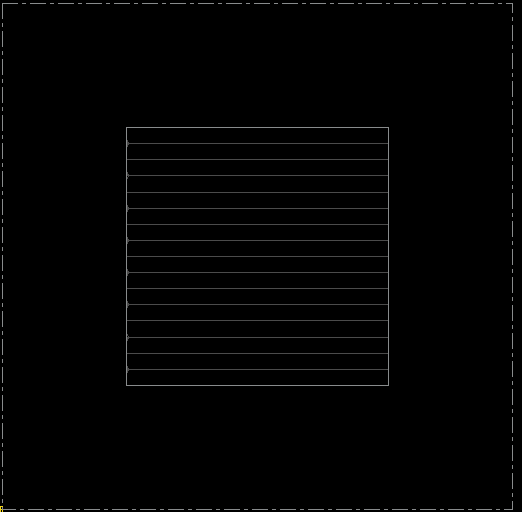


Figura – Design resultante após a etapa de floorplaning

## Pin Editor

Agora chegou a hora de configurar os pinos da célula que são ‘clk’, ‘clr’, ‘hlt’ e ‘q3[8]’. Todos os pinos são colocados no topo. O design resultante é mostrado na Figura 2.

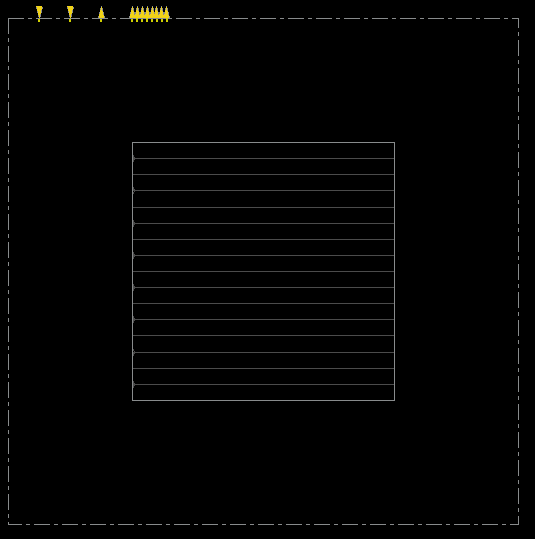


Figura 2 – Design resultante após a etapa de pin Editor

## Power Planning

Nesta etapa é realizada a configuração dos *Power rings*, assim, definimos as nets como gnd e vdd, tamanho da largura de 0.5, espaçamento de 0.4972 e offset de 1.5. Devido ao tamanho pequeno da célula não foi necessário pôr os Power stripes. A Figura 3 mostra o resultado do design após essa etapa.

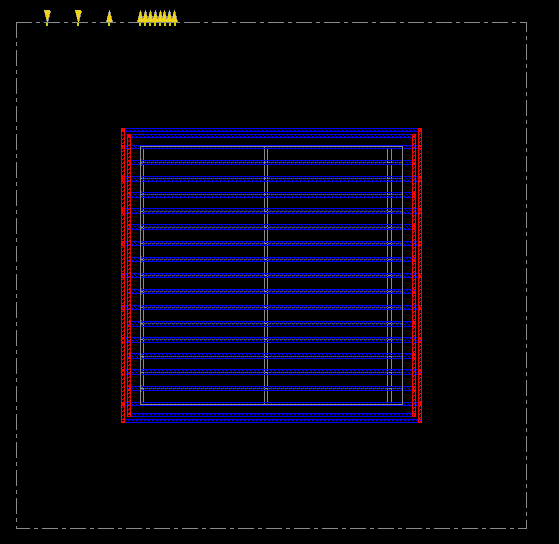


Figura 3 – Design resultante após a etapa de Power planning

## Placement

Nesta etapa inserimos os componentes do circuito. Na Physical view mostra-se a alocação dos componentes. Abaixo temos a figura 4 ilustrando a physical view do design e podemos observar que o projeto começa a tomar forma.

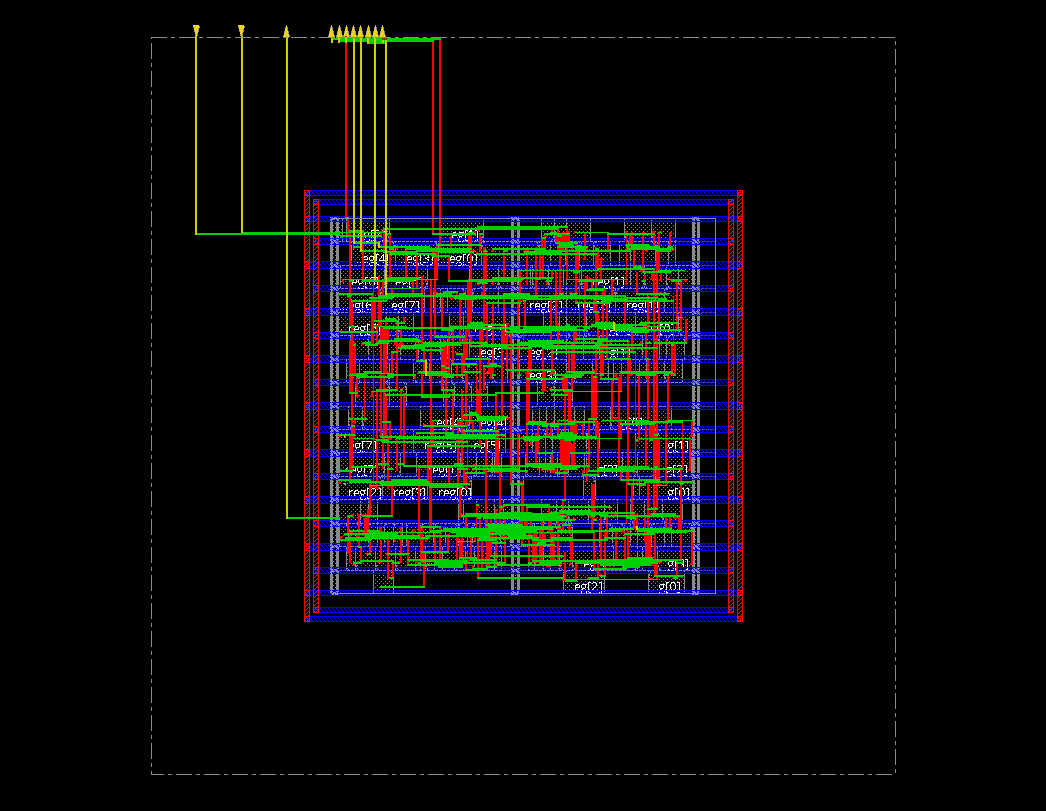


Figura 4 – Design resultante após a etapa de placement

## Optimization Post Placement

Nesta etapa do processo é feita uma primeira otimização e também é feita uma análise do tempo de setup, que indica se há algumas violações de tempos nos caminhos.

## Clock Tree Synthesis

Nesta etapa é feita a síntese da árvore de clock. Esta etapa é feita para garantir que o clock atenda o circuito de forma sincronizada.

## Optimization Post CTS

Esta etapa faz mais uma otimização de circuito a fim de melhorar os tempos de setup. Se houver violações no tempo, deve-se realizar a otimização novamente.

## Nano Route

Nesta etapa é feito o roteamento final do circuito, onde todos os fios do CI são distribuídos ao longo das células. É possível verificar se há erros de DRC (Design Rules Constraints). Caso haja algum erro, deve-se realizar o roteamento novamente ou realizar a otimização pós roteamento e verificar se o erro persiste. Caso o problema persista, deve-se diminuir a utilização o core (Core Utilization), realizada na etapa 2 (Floorplanning).

## Verification

Nesta etapa é executado o ‘check\_drc’ e ‘check\_design -all’ para tentar encontrar alguma violação no *design* ou algum erro.

## Output

Agora que o projeto está finalizado, deve-se salvar o projeto e exportar os arquivos do netlist, .DEF, .SDF e os arquivos de reports.

# Alterações para melhorar as métricas de code coverage

Na tabela abaixo é mostrado o resultado da execução de *code coverage* novamente, onde foi efetuada modificações na arquitetura para maximizar as métricas de *branches* e *toggle bins*.

A ideia principal por trás dessa tentativa de maximizar as métricas de *branches* e *toggle bins* foi de alterar o programa da ROM por um novo programa. O novo programa na ROM resultou em um aumento total de 1.5% na cobertura do código, este resultado foi devido ao aumento de *toggle bins* das entidades do projeto SAP, porém não foi possível realizar um aumento da métrica de *branches.*

Agora vamos analisar a razão na qual não foi possível maximizar as métricas de *toggle bins* e *branches* das entidades do projeto SAP:

## Toggle

MP\_STRUCT\_TB

* hlt: o sinal da entidade MP não alcança 100% de toggle porque este sinal é alterado quando o comando ‘HLT’ é executado. A instrução ‘HLT’ diz ao computador para finalizar o processamento de dados, ou seja, este comando marca o final do programa, portanto, só é utilizado uma vez fazendo o sinal ir de nível lógico ‘0’->’1’, sendo assim, o sinal alcança apenas 50% de toggle.
* q3: este sinal recebe o valor do registrador acumulador e transfere o valor para uma porta externa, porém isso acontece quando é executado uma instrução de ‘out’. No programa que armazenamos na ROM, os valores 15 e 0 são transferidos para a porta externa, portanto, apenas os bits de 0 a 3 deste sinal alcançaram 100% de *toggle* e os restantes não, devido ao fato deste sinal conter 8 bits.

ALU\_BEHAVE

* sum: este sinal é um vetor de 8 bits que recebe o resultado de uma operação realizada na entidade 'ALU'. Nessa versão do projeto SAP contém duas instruções de operação na 'ALU' que são uma operação de adição e outra de subtração. Nessas duas operações aritméticas as entradas são dois valores representados em 4 bits, portanto, realizando uma soma de dois valores de 4 bits o resultado máximo que podemos encontrar pode ser representado por 5 bits, sendo assim, o sinal de 'sum' da entidade 'ALU' que possui 8 bits não terá uma cobertura de toggle de 100% para os 3 bits restantes. No programa que foi posto na ROM para executar tem como objetivo ter uma cobertura de 100% de toggle para 5 bits do sinal 'sum', onde o objetivo foi tentar estimular o máximo de bits dos 8 que o sinal 'sum' possui.

IR\_DEC

* instruction: este sinal representa a instrução que será executada no próximo ciclo. Possui 6 bits para representar apenas 5 instruções que são:

“000001” -> instrução LDA

“000010” -> instrução ADD

“000100” -> instrução SUB

“001000” -> instrução OUTPUT

“100000” -> instrução HLT

Diante deste conjunto reduzido de instrução e mesmo utilizando todas elas no programa que colocamos na ROM não foi possível atingir *toggle* 100%, pois teríamos que ter mais uma instrução para ser a instrução de “010000”, portanto, apenas foi obtido métrica de 100% de *toggle* para os bits de 0 até 3 deste sinal.

PC\_BEHAVE:

* count: o sinal é um vetor de 4 posições e apenas os bits da posição 0, 1 e 2 alcança toggle de 100%, o bit 3 alcança apenas 50% de toggle, isto devido ao número de instruções executadas neste projeto, no caso 9 instruções. Devido a este número de instruções o ‘count’ começa a execução do programa apontando para o endereço de instrução ‘0x0’ e termina na instrução ‘0x9’, entretanto, quando executa a última instrução ele faz mais um incremento para o endereço ‘0xA’. Uma forma de aumentar a medida do toggle do bit 3 é realizar um ‘reset’ do programa para voltar para o endereço ‘0x0’, sendo assim, o bit 3 realiza a transição do nível lógico de ‘1’->’0’ que é a transição que falta para obter 100% de toggle deste sinal.

MP\_STRUCT

* d1: Recebe o resultado do bloco que realiza soma e subtração, sendo assim, segue a mesma lógica da explicação da entidade ALU\_BEHAVE sinal ‘sum’.
* q\_alu: sinal de entrada para o sinal ‘A’ da entidade ALU que vem do AC. O sinal AC também recebe o resultado de uma operação de ADD ou SUB, portanto, a explicação deste sinal não ter alcançado a métrica de *toggle* 100% é a mesma da entidade ALU\_BEHAVE do sinal ‘sum’.
* q\_w: este sinal de 4 bits recebe o endereço de dados de uma instrução. Uma instrução neste projeto é composta de 8 bits, os 4 primeiros bits identificam a instrução e os últimos 4 bits identifica o endereço do dado que a instrução vai ter que buscar o valor. No nosso programa na ROM estes endereços dos dados variam de 12 até 15, sendo assim, foi possível apenas conseguir métrica de 100% *de* *toggle* para os bits 0 e 1.
* q1: sinal de entrada para o sinal ‘B’ da entidade ALU que vem do registrador B. Sinal de 8 bits porém só foi possível alcançar medida de *toggle* de 100% para os bits de 0 a 3, pois variamos o seus valor de 0 para 15, depois para 1, depois para 14 e por fim para 1. O valor 15 é o maior número que podemos armazenar neste sinal.

## Branch

CONTROL UNIT

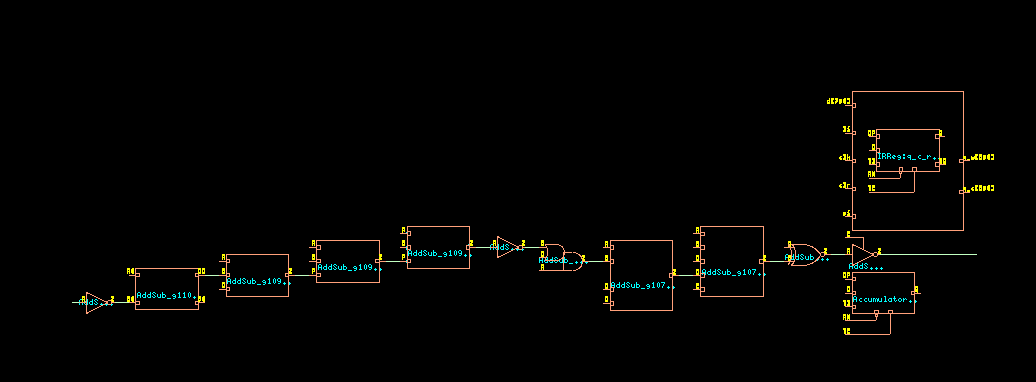
# Síntese comportamental

# Resultados comentados dos relatórios gerados pela ferramenta

* Power: A leakage power manteve-se praticamente o mesmo em todas as simulações, o que variou de fato foi a dynamic power. Na simulação low power, como esperado teve a menor dynamic power. Ficou visível que a potência total da low power ficou em aproximadamente 1/5 das outras simulações.
* Timing: Foi possível perceber que a cada simulação o caminho crítico foi alterado;
* Area: Foi possível perceber que na síntese low power, a quantidade células foi maior pois foi feito a adição de blocos de clock gating.

# Fanout caminho crítico

Para efetuar a identificação do caminho, a ferramenta gera todos os caminhos possíveis e seleciona aquele que possui o menor slack.



# Escolha do clock

Foi possível definir um clock de 2 GHz, fomos aumentando e gerávamos o relatório de timing até verificar que o slack ficou negativo.

# Gráficos dos resultados gerados pela ferramenta

## Timing

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | High View NOM | High View WC | High View BC | Medium View NOM | Medium View WC | Medium View BC | NOM Low Power |
| Slack (ps) | 17 | 27 | 23 | 8 | 26 | 22 | 43 |

## Area

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | High View NOM | High View WC | High View BC | Medium View NOM | Medium View WC | Medium View BC | NOM Low Power |
| Cells | 234 | 240 | 248 | 243 | 237 | 242 | 257 |
| Cell Area | 1231 | 1263 | 1281 | 1180 | 1227 | 1244 | 1213 |
| Net Area | 515 | 524 | 530 | 515 | 517 | 525 | 493 |
| Total Area | 1747 | 1787 | 1811 | 1695 | 1743 | 1769 | 1706 |

## Power

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | High View NOM | High View WC | High View BC | Medium View NOM | Medium View WC | Medium View BC | NOM Low Power |
| Leakage Power(nW) | 17.245 | 17.717 | 18.371 | 14.397 | 16.911 | 17.689 | 16.194 |
| Dynamic Power(nW) | 538.953 | 540.109 | 540.408 | 547.295 | 540.468 | 540.392 | 98.876 |
| Total Power(nW) | 556.199 | 557.826 | 558.779 | 561.692 | 557.378 | 558.081 | 115.070 |

Referências

<https://www.microsemi.com/document-portal/doc_view/131619-modelsim-user>

<http://vlsi.pro/code-coverage-fundamentals/>

<https://www.mathworks.com/help/hdlverifier/ug/add-a-value-change-dump-vcd-file.html>