UNIVERSIDADE FEDERAL DE SANTA CATARINA - UFSC
CENTRO TECNOLÓGICO - CTC
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA - DEEL

INE5422 - CIRCUITOS E SISTEMAS INTEGRADOS

RELATÓRIO - PROJETO FINAL
PROCESSAMENTO DE IMAGENS

ANDRÉ MARTINS PIO DE MATTOS

FLORIANÓPOLIS
DEZEMBRO, 2020

SUMÁRIO

INTRODUÇÃO

Neste trabalho é abordado o desenvolvimento de um sistema digital capaz de realizar a média aritmética de pixels de uma imagem. Esse tipo de operação é muito importante para algoritmos de compressão de imagem e outras transformações.

Para implementar esse circuito é usada uma abordagem aproximada que mantém um balanço entre desempenho e precisão, visto que com baixas taxas de erro ainda é possível executar esses tipos de algoritmos e obter resultados visivelmente equivalentes.

Por se tratar de um trabalho para síntese ASIC, também foram realizadas várias etapas de síntese e simulação para garantir o funcionamento do dispositivo implementado. A tecnologia alvo é de 180nm e direcionada para a implementação da IBM.

FERRAMENTAS

Para a execução do trabalho foram utilizadas diferentes ferramentas. Durante a fase inicial (desenvolvimento em VHDL do circuito) utilizou-se o GHDL e o GTKWave, dois projetos open-source para desenvolvimento, síntese, simulação e visualização. A partir dessas ferramentas foi possível averiguar o funcionamento do circuito e sua funcionalidade.

Após essa fase inicial, utilizou-se as ferramentas da Cadence (Gennus, Innovus, entre outras) para realizar a síntese lógica e física, além de realizar as simulações e relatórios de desempenho.

DESENVOLVIMENTO

O desenvolvimento se subdividiu em diversas etapas e contextos diferentes, mas para esse trabalho apenas os principais são abordados. Os próximos tópicos mostram etapas essenciais que foram realizadas e os resultados na próxima seção.

BLOCOS VHDL

Os blocos VHDL implementados consistem em circuitos assíncronos controlados por uma máquina de estados finita (Moore) e sincronizados com registradores (flip-flops tipo D). A Figura 1 ilustra esses arquivos e o tipo de implementação realizada.

Figura 1 - Lista de arquivos implementados

A Figura 2 mostra o diagrama de blocos do sistema. Nota-se que o circuito consiste de um somador sequencial que realiza a operação de soma durante vários ciclos de relógio (dependente do tamanho da imagem). Após esse período, o circuito controlado pela máquina de estados muda a função do adder a partir da mudança dos seletores dos muxes. Assim, é realizada uma operação de soma das divisões parciais realizadas por deslocamentos lógicos. Para mandar o sinal de que esse certo tempo de somas inicial foi atingido e produzir os endereços para a memória, o comparador sempre avalia o valor atual com o tamanho da imagem armazenada na memória (como pode-se notar na Figura 1, “IMAGE_SIZE”). Além disso, vale ressaltar a presença de alguns registradores para realizar a sincronização do sistema.

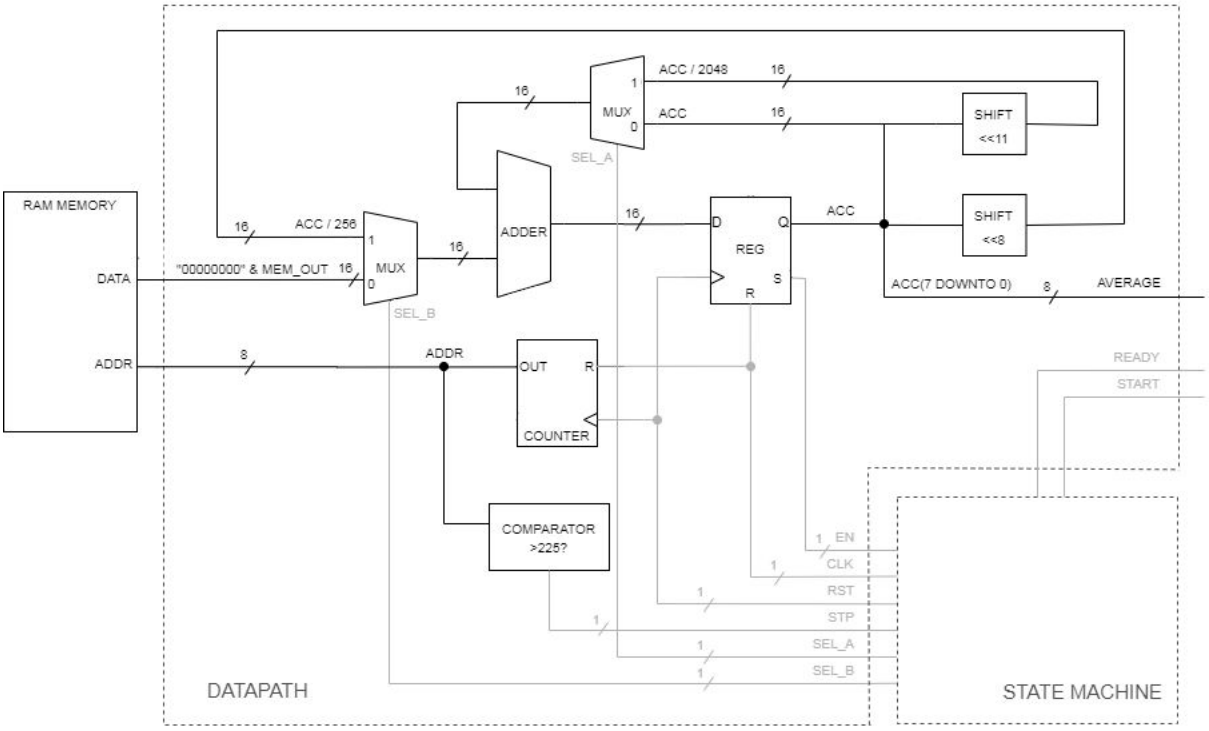

Figura 2 - Diagrama de blocos desenvolvido

A Figura 3 mostra o diagrama de estados do controle, além de mostrar as entradas e saídas do sistema. Nota-se que são utilizados apenas 4 estados: IDLE, período em que se permanece sem operação esperando uma operação e mostrando o resultado da anterior; RST, momento para resetar todos os circuitos síncronos para começar um novo ciclo; ACC, período de execução que acontece a leitura da memória, a soma de todos esses valores e a comparação de tamanho; DIV, quando a fase de acumulação acaba, entra a de divisão que gera a média final para o circuito.

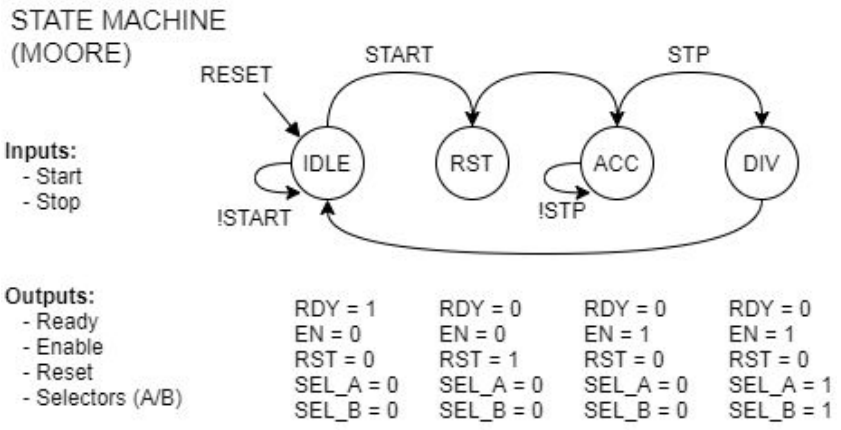

Figura 3 - Diagrama de estados desenvolvido

SÍNTESE

A partir dos arquivos VHDL implementados e os *testbenches*, foram realizadas as etapas de síntese lógica e física. A Figura 4 mostra a visualização RTL do circuito após a síntese. Como esperado, nota-se a presença de *flip-flops*, portas lógicas e *muxes* (e nenhum *latch* indesejado originado da síntese de *sources* ambíguos).

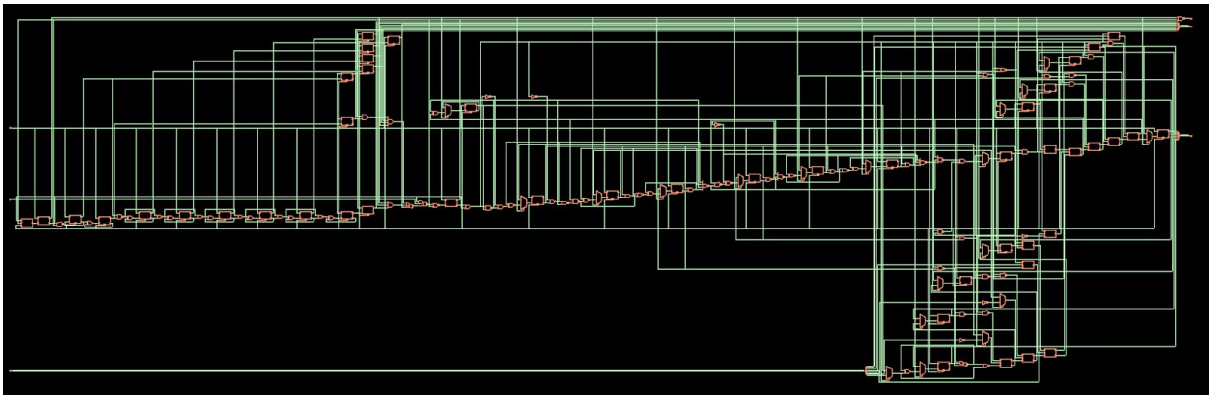

Figura 4 - Visualização RTL da síntese

Após algumas iterações foi possível se atingir o *clock* máximo de aproximadamente 333 MHz (ou um período de *clock* igual a 3 ns). Esse processo se deu a partir da avaliação do *slack time* e diferentes tentativas graduais de diminuição. Percebeu-se o comportamento de que à medida que se aumentava as otimizações para atingir *clocks* mais elevados, a área não apresentava grandes variações, mas a potência consumida aumentava em grandes percentuais. A Figura 5 mostra algumas imagens dos relatórios da ferramenta.

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 16.24-s065_1
  Generated on:           Dec 16 2020  06:35:32 pm
  Module:                 top
  Technology libraries:   PnomV180T025_STD_CELL_7RF
                          physical_cells
  Operating conditions:   _nominal_
  Interconnect mode:      global
  Area mode:              physical library
============================================================

                Leakage    Dynamic      Total
Instance Cells Power(nW)  Power(nW)   Power(nW)
------------------------------------------------------
top        138    63.278 3557226.665 3557289.943

legacy_genus:/>
```

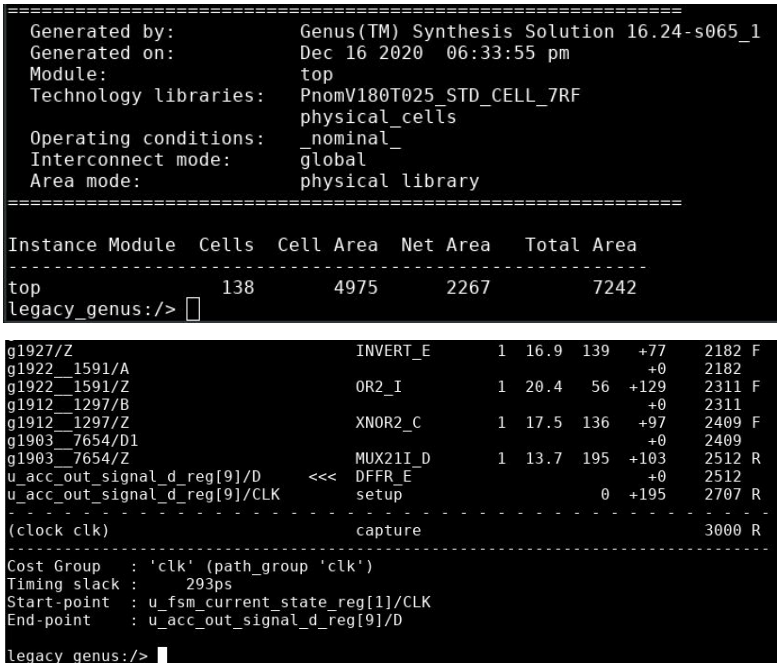

Figura 5 - Relatórios de *power*, *area*, *timing* da ferramenta (respectivamente)

Após essa fase de síntese RTL, foi executada a síntese física, a qual representa a implementação física do circuito. A Figura 6 mostra os diferentes relatórios de verificação. A Figura 7 mostra um *preview* do circuito sintetizado resultante.

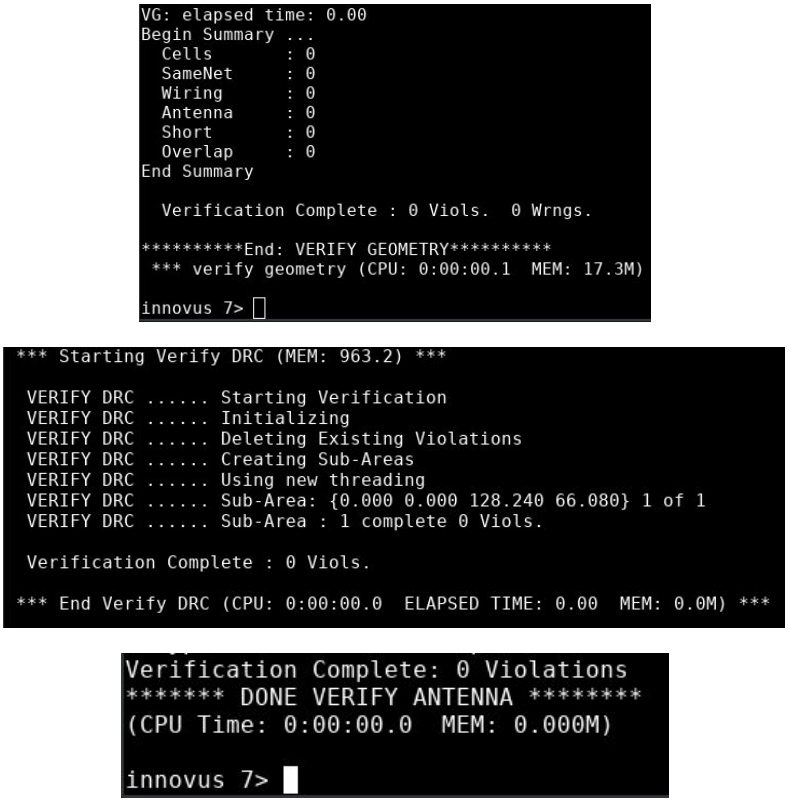

Figura 5 - Relatórios de *geometry*, *DRC*, *antenna* da ferramenta (respectivamente)

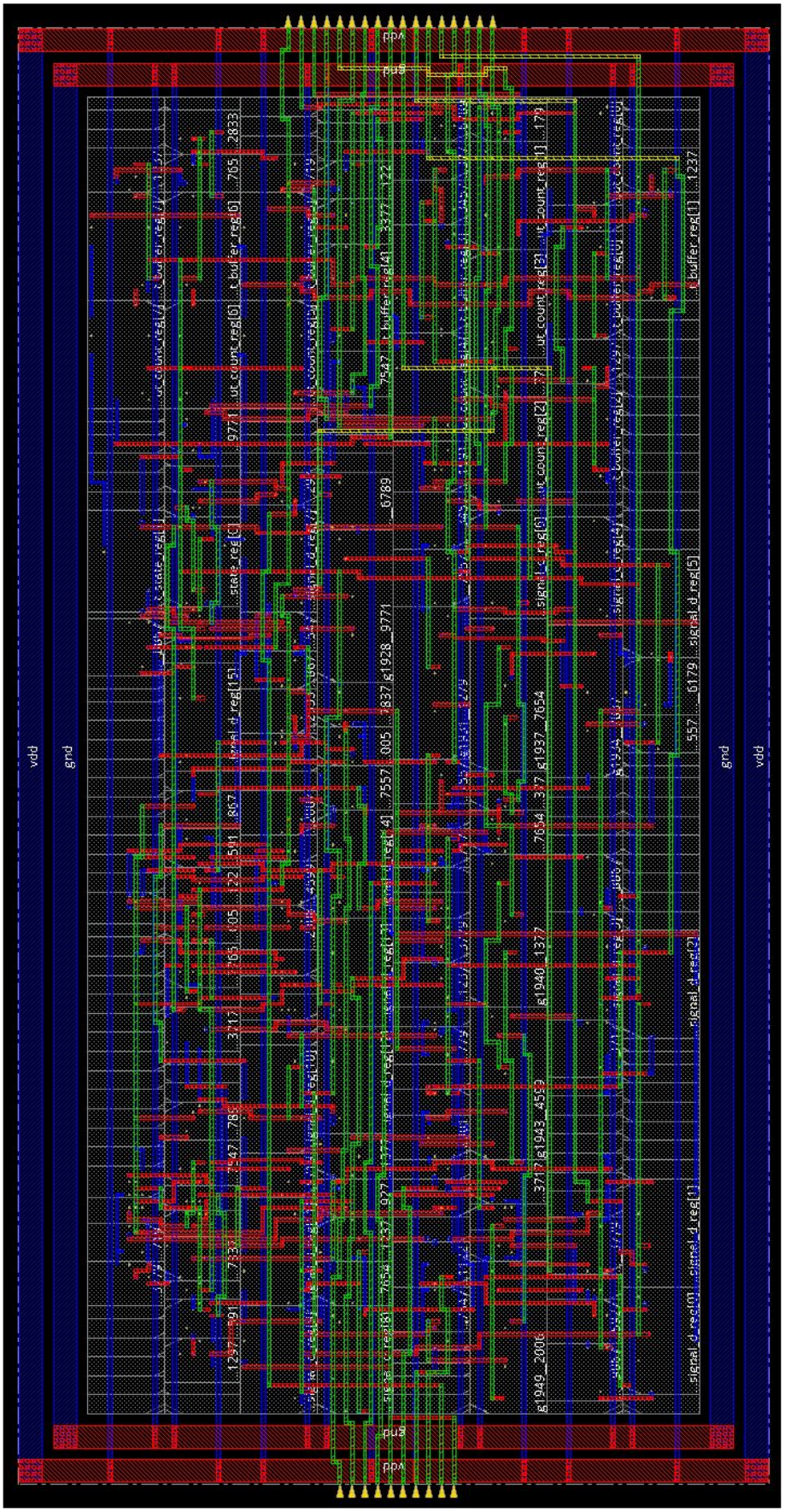

Figura 6 - Implementação em layout do circuito

SIMULAÇÕES

As simulações se subdividem em três momentos diferentes: simulação ideal, de atraso unitário e após a síntese física. Com a simulação ideal foram realizados e testados diferentes cenários para averiguar o desempenho e garantir a margem de erro máximo requisitada.

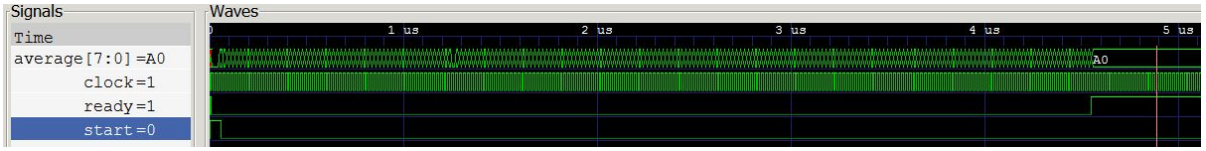

Figura 7 - Visualização da simulação ideal (GHDL e GTKWave)

Na Figura 7 é mostrado o caso de simulação que foi disponibilizado como exemplo. Nesse caso (ilustrado na Figura 8), a média aritmética ideal de todos os pixels é 163,5867. Inicialmente, como aproximação, já se faz necessário truncar o resultado por se tratar de uma implementação baseada em inteiros, obtendo-se 163 (ou A3 em hexadecimal).

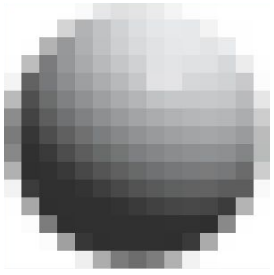

Figura 8 - Visualização do bitmap da imagem exemplo

A partir da equação disponibilizado para avaliar precisão, nesse exemplo temos:

Erro = 100 * |valor exato – resultado do circuito projetado| / 255

Assim, como foi obtido o resultado 160 (ou A0 em hexadecimal) para o exemplo, nota-se um erro de aproximadamente 1,407%. Testando para outros valores foi possível encontrar valores que não ultrapassam a margem requisitada de 8% (mesmo em casos limite). Além dessa simulação foram realizadas outras simulações nas ferramentas da Candence para averiguar o mesmo comportamento (Figura 9).

Figura 9 - Visualização da simulação na ferramenta da Cadence

RESULTADOS

Como resultado final, calcula-se a FoM comparativa proposta para esse trabalho pela seguinte expressão:

FoM = **área em layout** * **tempo para efetuar o cálculo** * (**1** + **erro no cálculo**)

Assim, utilizando uma erro médio aproximado de 3% (estimativa pessimista média de erro percentual encontrada para diferentes casos) e utilizando os resultados dos relatórios gerados pela ferramenta *Innovus* (Figura 10).

Floorplan/Placement Information

Total area of Standard cells	6231.859 um^2
Total area of Standard cells(Subtracting Physical Cells)	4974.950 um^2
Total area of Macros	0.000 um^2
Total area of Blockages	0.000 um^2
Total area of Pad cells	0.000 um^2
Total area of Core	6231.859 um^2
Total area of Chip	8474.099 um^2
Effective Utilization	1.0000e+00
Number of Cell Rows	8

# Format: clock	timeReq	slackR/slackF	setupR/setupF	instName/pinName	# cycle(s)
clk(R)->clk(R)	2.779	0.508/*	0.221/*	u_acc_out_signal_d_reg[9]/D	1
clk(R)->clk(R)	2.784	0.571/*	0.216/*	u_acc_out_signal_d_reg[14]/D	1
clk(R)->clk(R)	2.749	0.572/*	0.251/*	u_acc_out_signal_d_reg[15]/D	1
clk(R)->clk(R)	2.779	0.583/*	0.221/*	u_acc_out_signal_d_reg[13]/D	1
clk(R)->clk(R)	2.789	0.604/*	0.211/*	u_acc_out_signal_d_reg[11]/D	1
clk(R)->clk(R)	2.776	0.620/*	0.224/*	u_acc_out_signal_d_reg[10]/D	1
clk(R)->clk(R)	2.780	0.639/*	0.220/*	u_acc_out_signal_d_reg[12]/D	1
clk(R)->clk(R)	2.791	0.716/*	0.209/*	u_acc_out_signal_d_reg[8]/D	1
clk(R)->clk(R)	2.752	0.893/*	0.248/*	u_acc_out_signal_d_reg[7]/D	1
clk(R)->clk(R)	2.750	1.032/*	0.250/*	u_acc_out_signal_d_reg[6]/D	1
clk(R)->clk(R)	2.750	1.175/*	0.250/*	u_acc_out_signal_d_reg[5]/D	1
clk(R)->clk(R)	2.754	1.313/*	0.246/*	u_acc_out_signal_d_reg[4]/D	1
clk(R)->clk(R)	2.747	1.460/*	0.253/*	u_acc_out_signal_d_reg[3]/D	1
clk(R)->clk(R)	2.751	1.602/*	0.249/*	u_acc_out_signal_d_reg[2]/D	1
clk(R)->clk(R)	2.771	1.690/*	0.229/*	u_count_count_buffer_reg[7]/D	1
clk(R)->clk(R)	2.752	1.747/*	0.248/*	u_acc_out_signal_d_reg[1]/D	1
clk(R)->clk(R)	2.772	1.810/*	0.228/*	u_count_count_buffer_reg[6]/D	1
clk(R)->clk(R)	2.749	1.833/*	0.251/*	u_acc_out_signal_d_reg[0]/D	1
clk(R)->clk(R)	2.834	*/1.847	*/0.166	u_fsm_current_state_reg[1]/D	1
clk(R)->clk(R)	2.772	1.937/*	0.228/*	u_count_count_buffer_reg[5]/D	1

Figura 10 - Relatórios da de *area* e *timing* (respectivamente)

Realizando as contas, observa-se que a área total ocupada é de 8474,859 μm^2 e que o menor atraso para realizar uma operação dentro de um ciclo de relógio (3ns) tem um *slack time* positivo (uma pequena margem de segurança), então se considerarmos o número total mínimo de ciclos para uma execução (IDLE + RST + 255xACC + DIV), temos que o tempo total é de 774ns.

Portanto, temos que a **FoM** obtida para esse projeto é de aproximadamente:

FoM = 6756327,1 ns.μm^2 = 6756,3271 $\mu s.\mu m^2$

CONCLUSÃO

Neste trabalho foi abordado o desenvolvimento de um subsistema de processamento de imagem, mais especificamente para o cálculo de média aritmética de pixels de uma imagem. Com o projeto foi possível fixar os conceitos de implementação de sistemas síncronos e assíncronos, testes e simulações, síntese lógica e física e o uso de ferramentas de EDA.

Aponta-se que a implementação desenvolvida apresenta erro relativo aceitável, mas representa uma implementação muito simples para aplicações reais que fazem o uso de vários estágios de transformações. Assim, mesmo que com um erro pequeno, na cadeia de transformações é gerado um grande desvio. Por esse motivo, é necessário se investigar implementações mais precisas ou até mesmo que façam o uso de ponto flutuante.

Em resumo, os objetivos para esse trabalho foram obtidos, uma vez que foi possível fixar os conceitos vistos ao longo da matéria e obter resultados satisfatórios para um circuito de processamento de imagens.