UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

FÁBIO LEANDRO WALTER

Técnicas de Baixo Consumo para Módulos de Hardware de Codificação de Vídeo H.264

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Microeletrônica

Prof. Dr. Sergio Bampi
Orientador

Porto Alegre, agosto de 2011.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Walter, Fábio Leandro

Técnicas de Baixo Consumo para Módulos de Hardware de Codificação de Vídeo H.264 / Fábio Leandro Walter – Porto Alegre: Programa de Pós-Graduação em Microeletrônica, 2011.

83 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica. Porto Alegre, BR – RS, 2011. Orientador: Sergio Bampi;

1.Técnicas de Redução de Consumo de Potência em Sistemas Digitais CMOS 2. Núcleo de Processamento Aritmético para Estimação de Movimento em Vídeo 3. Decodificador Intra-Only em CMOS 65nm. I. Bampi, Sergio. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Aldo Bolten Lucion
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenador do PGMICRO: Prof. Ricardo Augusto da Luz Reis
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

Primeiramente, gostaria de agradecer aos meus pais, Aurea Maria do Nascimento Walter e Erwin Walter, pelo exemplo de valores, disciplina, pelo carinho dado durante toda a minha vida, pelo companheirismo nos momentos difíceis e, sobre tudo, pelo amor e respeito dados em todos os momentos de minha vida. Minha mãe pelas inúmeras conversas que eu tive em relação à minha vida, meu pai pelo incentivo a vencer os desafios da vida e enfrenta-los. Sem eles nada disso seria possível.

Um agradecimento, também, aos meus avós pela determinação em suas vidas e por terem criado filhos num ambiente de harmonia e felicidade. Todos eles - Oscar, Ester, Reinhold e Herta - guardo em meu coração.

Minha avó Herta, com quem tive contato mais próximo e foi a última a falecer, me dá o exemplo de sua religiosidade, moral e força com que enfrentou os desafios da vida, fugindo com sua família de um regime opressor e vindo semear os seus valores no sul do Brasil.

Também, gostaria de agradecer a irmã desta minha avó, a tia Vanda, que também foi um exemplo nesta vida. Agradecendo, assim, a todos os familiares desta minha tia, principalmente, a Marlene que viveu junto com esta seus últimos momentos de vida.

Gostaria de agradecer meus primos Paulo, Aline e Denise, tia Edith, tia Erna, tio Roberto e Plínio, todos eles estão em meu coração e me dão a doce lembrança da infância já ida.

Aos meus primos e tios mais distantes do Paraná, São Paulo, etc. Meus sinceros agradecimentos e amor. Parentes que eu não tenho um contato diário, mas pelo amor e carinho com que me recebem quando lá visito, me dão a dimensão do amor que tem por mim e por minha mãe, sua irmã.

Não poderia deixar de citar algum deles, como o tio Carlinhos que pela humildade e generosidade com que leva a vida, merece meu profundo apreço. Gostaria de lembrar também os seus filhos Diego e Renato e sua mulher Valdete, pessoas com as quais tive pouco contato, mas se forem tão bons quanto o meu tio Carlinhos merecem todo o meu amor e carinho.

A minha tia Leni, meus primos Marcela, Márcia e Marcelo, seus filhos, esposa e marido, a todos meus profundos agradecimentos pela receptividade com que me trataram. Principalmente, a Marcela, uma pequena e futura grande profissional, que tanto esforço e empenho serão recompensados.

A tia Luci e seus inúmeros parentes, tia Iracema, madrinha Sebastiana, Ednéia, Donizete, Ademar, Celso, esposas, marido e filhos. Ao tio Edgar, pelo apoio que dá aos

seus irmãos e pela disponibilidade em nos auxiliar, quando íamos ao Paraná, sua filha Gabriela e sua esposa Lê, e seus parentes, meus sinceros agradecimentos.

A tia Eunice e ao tio Alan, e aos primos Aline e André, minhas sinceras lembranças e carinho do tempo em que eles passaram vivendo em solo gaúcho.

Aos demais parentes de minha mãe, Manuel, esposa e filhos, a tia Aparecida, Rogério, Adriano e Patrícia, ... Meus sinceros agradecimentos.

Ao meu irmão Erwin, que foi um exemplo para mim, a sua esposa Inês e seus filhos Bianca e Guilherme, a minha irmã Josy, seu marido Vilson e seus filhos Mateus e Bruna, a minha irmã Jainete, com quem tenho convivido ultimamente e demonstrou ser uma pessoa verdadeira e leal, a minha irmã Jesseê, pelo carinho e lembranças da infância que tivemos, seu esposo João e filho João Pedro, a minha irmã Poliana, pela preocupação que tem para comigo, seu marido Marcus e filho Henrique, e parentes dos meus cunhados e cunhadas, meu profundo agradecimento e esperança de que sempre busquem a felicidade, o amor ao próximo e Deus.

A todos os meus familiares, meu eterno amor e carinho. Vocês estão em meu coração.

Ao colega na Elo Sistemas Eletrônicos, André Granero, meu parceiro em jogos do Grêmio, é muito bom contar com um amigo honesto, verdadeiro e de muito bom caráter.

Aos colegas de Datacom, destacando-se meu chefe imediato Rodrigo Trevisan, que me ensinou boa parte de meu conhecimento em descrição de *hardware*, Inácio, meu competente colega de trabalho, e aos demais colegas de trabalho, meus sinceros agradecimentos.

Aos bons e dedicados colegas de graduação minha saudade e estima.

Aos colegas de mestrado, que me acolheram num ambiente de respeito e profissionalismo, ambiente tal, que em sua grande maioria era formada de pessoas de grande honra e caráter, sendo tais:

Os colegas da sala 215, Cláudio, que me ajudou em inúmeras coisas, como a escrever um artigo, André, Cristiano, Kleber, Bruno Zatt, Eduarda, Bruno Vizotto, Guilherme, etc.

Os colegas de minha sala, Jerson Guex, uma pessoa simples e dedicada, Carolina Metzler, pela ajuda e apoio na confecção do texto, ao agora mestre Leandro Max, e aos inúmeros bolsistas e mestrandos que passaram por esta sala, destacando-se os bolsistas Luigi, Bruna e Davi.

A colega Gracieli, que apresentou um artigo meu no LASCAS, aos colegas Walter, Jimmy e Jorge, que mostraram que o Peru é feito de pessoas muito honradas e dignas.

E aos demais colegas de mestrado, aqui não citados, meus sinceros agradecimentos, e que as minhas palavras aos alunos de mestrado, também se refiram a eles.

Ao professor Junqueira, meu orientador de trabalho de conclusão, obrigado pela recomendação positiva, quanto ao pedido de mestrado.

Aos demais professores da graduação em Engenharia Elétrica na UFRGS e mestrado no Programa de Pós-Graduação em Microeletrônica da mesma, dos quais se destacam Carlos Eduardo, André Reis, Henry e Fernanda Lima.

Ao professor Eduardo Costa, do qual tive trabalhos em conjunto.

Aos membros da banca.

E, finalizando, ao meu professor orientador, Sergio Bampi, que pela paciência, caráter, dedicação e empenho, que me levou a finalizar este trabalho de maneira satisfatória. Meu muito obrigado.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 10

LISTA DE FIGURAS 11

LISTA DE TABELAS 12

RESUMO 13

ABSTRACT 14

1 INTRODUÇÃO 15

1.1 Algoritmo de SAD 15

1.2 Decodificador *Intra-Only* 16

1.3 Organização da Dissertação 16

2 TÉCNICAS DE REDUÇÃO DE CONSUMO DE POTÊNCIA EM SISTEMAS DIGITAIS CMOS 17

2.1 Consumo de Potência 17
2.1.1 Potência Estática 17
2.1.2 Potência Dinâmica 17
2.1.2.1 Potência de Chaveamento 17
2.1.2.2 Potência Interna 18

2.2 Técnicas de Minimização do Consumo de Potência 18
2.2.1 *Nível Lógico* 18
2.2.1.1 Clock Gating 18
2.2.1.2 Isolação de Operandos 19
2.2.1.3 Somadores Compressores 20
2.2.2 Nível Elétrico 21
2.2.2.1 Múltiplas Tensões de Alimentação 21
2.2.2.2 Multi-Threshold 22
2.2.3 Nível Arquitetural 22

2.3 Conclusão 24

3 NÚCLEO DE PROCESSAMENTO ARITMÉTICO PARA ESTIMAÇÃO DE MOVIMENTO EM VÍDEO 25

3.1 Algoritmo de SAD 26

3.2 Arquiteturas de SAD com Paralelismo e *Pipeline* 26

3.3 Método de Análise 29
3.3.1 Método de Análise de Máxima Taxa de Transferência 35
3.3.2 Método de Análise de Mínima Requerida *Isoperformance* 35

3.4 Resultados da Síntese Física de Arquiteturas para SAD em 180 nm 36
3.4.1 Máxima Taxa de Transferência em 180 nm 36
3.4.2 Mínima Requerida *Isoperformance* em 180 nm 38

3.5 Resultados de Síntese Física de Arquiteturas para SAD para 65 nm 40
3.5.1 Máxima Taxa de Transferência em 65 nm 40
3.5.2 Mínima Requerida *Isoperformance* em 65 nm 42

3.6 Análise dos Resultados Alcançados 44
3.6.1 Comparação de Potência Média por *Gates* Equivalentes e Área com relação aos dados deste trabalho 44
3.6.1.1 Potência Média por Gates Equivalentes 44
3.6.1.1.1. Potência Estática 44
3.6.1.1.2. Potência Dinâmica 45
3.6.1.2 Potência Média por Área 45
3.6.1.2.1. Potência Estática 45
3.6.1.2.2. Potência Dinâmica 46
3.6.2 Comparação de *Isoperformance* e Máxima Taxa de Transmissão com relação aos melhores casos deste trabalho e trabalhos de outros autores 46
3.6.2.1 Isoperformance 47
3.6.2.2 Máxima Taxa de Transferência 47

4 DECODIFICADOR *INTRA-ONLY* EM CMOS 65 NM 51

4.1 Arquitetura do Decodificador Intra-Only 51

4.2 Métodos de Análise 54

4.3 Resultados e Análise dos Resultados de Síntese 60
4.3.1 Comparação entre as análises somente deste trabalho 60
4.3.2 Comparação com as Análises de Outros Trabalhos 60

5 CONCLUSÃO 63

REFERÊNCIAS 65

APÊNDICE A < ARQUIVOS EM SYSTEM VERILOG PARA VERIFICAÇÃO FUNCIONAL > 67

APÊNDICE B < ARQUIVOS PARA SÍNTESE LÓGICA > 71

APÊNDICE C < MODIFICAÇÕES NA DESCRIÇÃO DE *HARDWARE* > 79

LISTA DE ABREVIATURAS E SIGLAS

ASIC	*Application Specific Integrated Circuit*
AVC	*Advanced Video Coding*
CAVLC	*Context Adaptive Variable Length Coding*
C_{CARGA}	Capacitância de Carga
CMOS	*Complementary Metal-Oxide Semiconductors*
C_{PARA}	Capacitância Parasita
DCT	Transformada de Cosseno Discreta
F	Frequência
FPGA	*Field-Programmable Gate Array*
Full-HD	*Full High Definition*
HDTV	*High Definition Television*
$I_{Curto\text{-}circuito}$	Corrente de curto-circuito
ME	*Motion Estimation*
NAL	*Network Abstraction Layer*
NMOS	Semicondutor Metal-Óxido Canal N
PMOS	Semicondutor Metal-Óxido Canal P
QCIF	*Quarter Common Intermediate Format*
RAM	*Random Access Memory*
SAD	*Sum of Absolute Differences*
SBTVD	Sistema Brasileiro de Televisão Digital
VDD	Tensão Dreno Dreno
VT	Tensão de *Threshold*
V	Tensão

LISTA DE FIGURAS

Figura 2.1 – (a) Circuito Sequencial (b) Versão com clock gating (BENINI, 1999) *18*
Figura 2.2 – Circuito sem Isolação de Operandos (MÜNCH, 2000)..... *19*
Figura 2.3 – Circuito com Isolação de Operandos (MÜNCH, 2000). *20*
Figura 2.4 – Somador Compressor de 4-2. ..... *20*
Figura 2.5 – Circuito Up Converter..... *21*
Figura 2.6 – Circuito Down Converter *21*
Figura 2.7 – Potência Estática Vs Atraso em 90 nm (BD03, 2008) *22*
Figura 3.1: Estimação de Movimento com Diamond Search e Sub-Amostragem(PORTO, 2009). *26*
Figura 3.2: Arquitetura de SAD com 4 amostras de entrada e diferentes profundidades de pipeline. (DINIZ, 2010) *27*
Figura 3.3: Arquitetura de SAD com 8 amostras de entrada e diferentes profundidades de pipeline. (DINIZ, 2010) *27*
Figura 3.4: Arquitetura de SAD com 16 amostras de entrada e diferentes profundidades de pipeline. (DINIZ, 2010) *28*
Figura 3.5: Fluxo para Análise de Potência das Arquiteturas de SAD. *29*
Figura 3.6: Modelo da Verificação Funcional..... *30*
Figura 3.7: Módulo de Checagem de Equivalência para Comparação de Hierarquia. *31*
Figura 3.8: Módulo de Checagem de Equivalência para Gerenciamento de Mapeamento...... *32*
Figura 3.9: Planejamento de VDD/GND no Encounter para o circuito do SAD em 180nm. *33*
Figura 3.10: Modelo das linhas de relógio do núcleo de SAD integrado no Encounter depois da síntese de Post-Route. *34*
Figura 3.11: Gráfico da Máxima Taxa de Transferência em 180 nm alcançada para cada análise na Síntese Física. MB/s = Macro-blocos/segundo *37*
Figura 3.12: Gráfico da Energia por Operação em 180 nm alcançada para cada análise em Isoperformance na Síntese Física..... *39*
Figura 3.13: Gráfico da Taxa de Transferência em 65 nm alcançada para cada análise na Síntese Física. *41*
Figura 3.14: Gráfico da Energia por Operação em 65 nm alcançada para cada análise na Síntese Física. *43*
Figura 4.1 : Diagrama de Blocos do Decodificador Intra-Only...... *51*
Figura 4.2 : Fases do Parser que determinam a ordem de execução do decodificador Intra-Only (DEPRÁ, 2009). *52*
Figura 4.3: Espaço de Subamostragem de Cor 4:2:0. ..... *53*
Figura 4.4 : Módulo de Predição Intra. *53*
Figura 4.5: Fluxo de Síntese no decodificador Intra-Only. *54*
Figura 4.6: Violações de espaçamento com linhas de potência em Metal 1. ..... *55*
Figura 4.7: Violações pela falta de linhas de potência em Metal 1 próximo aos blocos de memória. *56*
Figura 4.8: Violações de Geometria. *57*
Figura 4.9: Design com floorplan completo do decodificador Intra-Only. *58*
Figura 4.10: Design com das árvores de relógio do decodificador Intra-Only. *59*

LISTA DE TABELAS

Tabela 2.1: Fatores que diminuem a potência consumida (linha lisa) e aumentam (linha hachurada) quando usado o aumento de paralelismo e mantida a taxa de transferência (throughput). *23*
Tabela 2.2: Fatores que diminuem a potência consumida (linha lisa) e aumentam (linha hachurada) quando usado o aumento de estágios de pipeline e mantida a taxa de transferência (throughput). *23*
Tabela 3.1: Resultados de Máxima Taxa de Transferência em 180 nm na Síntese Física. *36*
Tabela 3.2 – Média de Consumo de Potência por gates equivalentes e área para as arquiteturas com Máxima Taxa de Transferência em 180 nm. *37*
Tabela 3.3: Resultados de Mínima Requerida Isoperformance em 180 nm na Síntese Física. *38*
Tabela 3.4 – Média de Consumo de Potência por gates equivalentes e área para as arquiteturas em Isoperformance de 180 nm. *39*
Tabela 3.5: Resultados de Máxima Taxa de Transferência em 65 nm na Síntese Física. *40*
Tabela 3.6 – Média de Consumo de Potência por gates equivalentes e área para as arquiteturas com Máxima Taxa de Transferência em 65 nm. *41*
Tabela 3.7: Resultados de Mínima Requerida Isoperformance em 65nm na Síntese Física. *42*
Tabela 3.8 – Média de Consumo de Potência por gates equivalentes e área para as arquiteturas em Isoperformance de 65 nm. *43*
Tabela 3.9 – Tabela comparativa de Potência Estática Média por Gate Equivalente a partir da síntese de 180 nm, 1,62 Volts e sem clock gating. *45*
Tabela 3.10 – Tabela comparativa de Potência Dinâmica Média por Gate Equivalente a partir da síntese de 180 nm, 1,62 Volts e sem clock gating. *45*
Tabela 3.11 – Tabela comparativa de Potência Estática Média por Área a partir da síntese de 180 nm, 1,62 Volts e sem clock gating. *46*
Tabela 3.12 – Tabela comparativa de Potência Dinâmica Média por Área a partir da síntese de 180 nm, 1,62 Volts e sem clock gating. *46*
Tabela 3.13 – Tabela comparativa de Isoperformance entre os melhores resultados deste trabalho e artigos correlatos. *47*
Tabela 3.14 – Tabela comparativa de Máxima Taxa de Transferência entre os melhores resultados deste trabalho e artigos correlatos. *48*
Tabela 4.1 – Tabela comparativa das análises de potência somente deste trabalho. *60*
Tabela 4.2 – Tabela comparativa das análises de potência deste trabalho com outros trabalhos. *61*

RESUMO

Este trabalho trata da aplicação de técnicas de minimização de consumo de potência para blocos digitais para o algoritmo de SAD e o decodificador H.264/AVC *Intra-Only*. Na descrição de hardware são acrescidas as técnicas de paralelismo e *pipeline*. Na síntese física e lógica, incluem-se as técnicas de inativação do relógio (*clock gating*), múltiplas tensões de *threshold*, diferentes tecnologias e diferentes tensões de alimentação.

A síntese é feita nas ferramentas da *Cadence*™ com exploração arquitetural e apresenta uma menor energia por operação, quando exigido desempenho equivalente (*isoperformance*) para SAD, em baixa frequência, alto paralelismo e, principalmente, com um estágio de *pipeline*. Além disso, tecnologias CMOS mais avançadas diminuem o consumo de potência dinâmica e, em alguns casos, também diminuem a potência estática por *gate* equivalente, se utilizadas células *High-VT* e tensão de alimentação a menor possível. Outro fator a ser destacado é o uso do *clock gating* que no caso das arquiteturas de SAD, em vez de diminuir, aumenta o consumo de potência dinâmica.

Neste trabalho foi realizada a síntese do decodificador *Intra-Only*. O decodificador com *clock gating* apresenta um menor consumo de potência, mostrando um caso em que esta técnica é benéfica. Além disso, a utilização de uma tecnologia CMOS 65 nm e, consequentemente, tensão de alimentação menor, levou a uma sensível diminuição no consumo de potência em relação a outros trabalhos similares.

Palavras-Chave: Arquiteturas VLSI, *low-power* CMOS, decodificador *Intra-Only*, *clock gating*.

ABSTRACT

This work presents low-power techniques applications to digital blocks in the SAD algorithm and in the Intra-Only H.264/AVC decoder. In the hardware description, we add parallelism and pipeline techniques. In the logical and physical synthesis exploration, includes the clock gating, multiple threshold voltage, different technologies and multiple supply voltage.

The synthesis are done in the Cadence™ tools and show a smaller energy per operation in isoperformance for SAD at low frequency, high parallelism and, mainly, with one pipeline stage. In addition to that, more advanced CMOS technologies decrease the dynamic power consumption and, also, decrease the static power for equivalent gates, if using High-VT cells and lowest possible power supply. Another factor is the clock gating use that in the SAD architecture, instead of decreasing, increases the dynamic power consumption.

In this work the design of an Intra-Only H.264/AVC Decoder was performed. This design with clock gating presents lower power consumption, showing a case in which this technique is beneficial in terms of dynamic power. Besides that, the 65 nm CMOS technology uses a lower power supply, resulting in lower power consumption in comparison to other related works.

Keywords: VLSI architecture, low-power CMOS, Intra-Only Decoder, clock gating.

1 INTRODUÇÃO

Com o uso de circuitos integrados que suportam o processamento de vídeo digital em aparelhos portáteis e o seu desenvolvimento em aparelhos com múltiplas tarefas como para música, transmissão de dados e voz, jogo eletrônico, etc, elevou-se o consumo de potência nestes aparelhos, levando a um menor tempo de recarga de suas baterias. Além disso, o avanço da tecnologia CMOS (*Complementary Metal-Oxide Semiconductor*) resultou na minimização dos transistores e na integração de um número cada vez maior de transistores em um mesmo circuito integrado. Isto resultou no aumento porcentual da potência estática em relação à potência total.

Um sistema de televisão digital usa um codificador/decodificador (*codec*) de vídeo padronizado para codificar o vídeo. O H.264/AVC é o padrão mais atual e é usado no SBTVD(Sistema Brasileiro de Televisão Digital). A codificação do vídeo é necessária para se comprimir o volume de dados para representar o vídeo. As técnicas aplicadas para esta compressão são efetivas porque muitos dos dados brutos em quadros sucessores contêm pouca informação nova em relação ao quadro anterior, exceto em situações de corte completo da cena de vídeo. Sendo muito semelhante o quadro atual e futuro, pois há mudanças na movimentação dos objetos e da câmera, a estimativa de movimento associada a cada bloco (p.ex., de 8 pixel por 8 pixel) da imagem é uma das técnicas utilizadas para a codificação de vídeo.

1.1 Algoritmo de SAD

O Algoritmo de SAD (Soma de Diferenças Absolutas) é um módulo de hardware inserido na ME(Estimação de Movimento), sendo um bloco aritmético de alta atividade de chaveamento. A ME faz a compactação de imagens semelhantes por movimento da câmera ou movimento dos objetos da imagem. Já, o SAD é usado para fazer o cálculo de similaridade entre blocos de quadros de imagem sucessivos.

Para minimizar o consumo de potência no módulo de SAD, foram inseridas várias técnicas que deram uma visão melhor do que pode se buscar para produzir módulos de baixo consumo. Na exploração arquitetural que se faz para o *design* deste módulo, são feitas análises de máxima taxa de transferência para saber o máximo desempenho alcançado e qual o consumo de potência estimado nesta arquitetura que foi sintetizada com otimização de "*timing*" para o máximo desempenho temporal.

Foram estudadas algumas soluções no meio acadêmico sobre as arquiteturas de SAD. (VANNE, 2006) propôs uma nova unidade de processamento de SAD e uma comparação de várias unidades de processamento de SAD em termos de área e atraso. (DINIZ, 2010) apresentou algumas diferentes arquiteturas e medições de potência e energia consumidas, mas não haviam sínteses dedicadas ao baixo consumo, além de suas medições serem feitas em *netlist* lógica pré-posicionamento e roteamento. Neste

trabalho, além de usar as técnicas arquiteturais de paralelismo e pipeline, vistas também em (DINIZ, 2010), fez-se as medições de potência em *netlist pos-route* e utilizando técnicas de baixo consumo como o *clock gating*, as múltiplas tensões de limiar, diferentes tensões de alimentação e tecnologias de 180 nm e 65 nm.

1.2 Decodificador *Intra-Only*

No Decodificador de Vídeo *Intra-Only*, isto é, um decodificador que trata apenas os quadros tipo I (*intra-frame*) e não utiliza blocos de predição inter-quadros, no padrão de vídeo digital QCIF (*Quarter Common Intermediate Format*), se fez algumas análises com técnicas de baixo consumo, buscando-se minimizar o consumo de potência.

Alguns trabalhos presentes na literatura foram comparados com a arquitetura do decodificador de vídeo que desenvolvemos nesta dissertação. Em (CHEN, 2006), é desenvolvido um decodificador de vídeo, com vários blocos específicos para o seu funcionamento, como: *Parser*, Quantização e Transformadas Inversas, Predição Intra, Predição Inter e o *Descrambler*, além das medições de frequência e consumo do circuito integrado em questão. (LIN, 2005) apresenta um circuito integrado de um decodificador de vídeo com os mesmos blocos especificados em (CHEN, 2006) e suas respectivas medições de frequência e potência. Em (SILVA, 2010), é utilizada a arquitetura de (PEREIRA, 2008) que é semelhante à de (CHEN, 2006) e (LIN, 2005), somente não contendo a Predição Inter, por isso, o nome de sua arquitetura de decodificador *Intra-Only*, que foi desenvolvida na tecnologia TSMC de 180 nm, com comparações com outros trabalhos em termos de área, *gates count*, tecnologia utilizada, formato de vídeo e potência consumida. Neste decodificador de vídeo, utilizou-se a arquitetura de (PEREIRA, 2008), que é a mesma utilizada por (SILVA, 2010), e, para tal, fez-se o uso de técnicas de baixo consumo, como o *clock gating* e as múltiplas tensões de *threshold*, além de utilizar uma tecnologia mais recente que tem um menor consumo de potência dinâmica.

1.3 Organização da Dissertação

Esta dissertação está organizada da seguinte forma. No capítulo 2, uma introdução as fontes de potência e uma explicação sobre as técnicas de baixo consumo utilizadas. No capítulo 3, faz-se um resumo sobre o núcleo de processamento aritmético para a estimação de movimento, dá-se uma introdução sobre o algoritmo de SAD e demonstra-se o método de análise, tendo por fim os resultados alcançados na síntese de 180 nm e 65 nm, para os diferentes métodos de análise, em termos de: frequência, potência estática e dinâmica, energia por operação, taxa de transferência, potência estática e dinâmica por *gates* equivalentes e potência estática e dinâmica por área. E, finalizando o capítulo 3, tem-se a análise dos resultados alcançados em comparações com os dados deste trabalho e com de outros trabalhos. No capítulo 4, tem–se uma introdução a arquitetura deste decodificador *Intra-Only*, logo a seguir, o método de análise com a descrição dos desafios encontrados em tal síntese de 65 nm, e, finalizando, a análise dos resultados alcançados, com comparações somente dos resultados deste trabalho ou com comparações de outros trabalhos.

2 TÉCNICAS DE REDUÇÃO DE CONSUMO DE POTÊNCIA EM SISTEMAS DIGITAIS CMOS

Neste capítulo, serão referenciados os tipos de consumo de potência a nível industrial em circuitos digitais CMOS e, também, as algumas técnicas de baixo consumo dentre as quais as utilizadas neste trabalho.

2.1 Consumo de Potência

Primeiramente, o consumo de potência em circuitos digitais CMOS é dividido em estático e dinâmico. Neste trabalho, ainda analisaremos o consumo de potência e algumas de suas subdivisões, constantes num tutorial da Synopsys(SYNOPSYS, 2006).

2.1.1 Potência Estática

A potência estática é a potência que independe da frequência e da quantidade de transições. Ela é medida colocando-se tensão nas fontes de alimentação do sistema e fixando em DC todas as entradas dos sinais, assim, a potência consumida é somente a potência estática. Esta potência estática pode variar em função do nível lógico das entradas de sinal.

Este tipo de consumo tem como principal fator a corrente de *sub-threshold leakage*, que é a causada pela reduzida tensão de *threshold*.

2.1.2 Potência Dinâmica

Potência Dinâmica é dissipada quando a tensão dos caminhos do circuito muda no tempo, devido a mudanças nos estímulos (sinais de entrada). Ela pode ser subdividida em duas formas de potência analisadas a seguir.

2.1.2.1Potência de Chaveamento

Potência de chaveamento é determinada pela carga ou descarga do capacitor de saída (C_{CARGA}), e pela frequência ($\boldsymbol{f}$) das transições lógicas neste nó com carga capacitiva. E tem como equação:

$$Potência\ de\ Chaveamento = \frac{1}{2} \times C_{CARGA} \times V^2 \times f \qquad (2.1)$$

2.1.2.2Potência Interna

Potência interna é causada pela carga e descarga das capacitâncias parasitas internas das portas lógicas (C_{PARA}), pela frequência(f) das transições lógicas e, também, pela dissipação causada intra-porta pela corrente de curto-circuito, isto é, quando os transistores NMOS e PMOS estão conduzindo ao mesmo tempo, durante transientes de tensão nas portas dos transistores.

$$Potência\ Interna = \left(\frac{1}{2} \times C_{PARA} \times V^2 \times f\right) + (V \times I_{Curto-circuito}) \qquad (2.2)$$

2.2 Técnicas de Minimização do Consumo de Potência

Para minimizar o consumo de potência existem várias técnicas, mas somente algumas delas foram aplicadas neste trabalho, tais estão explicadas nas seções posteriores deste capítulo.

2.2.1 *Nível Lógico*

2.2.1.1Clock Gating

Clock Gating é uma das primeiras técnicas de diminuição do consumo de potência. Sendo basicamente colocada com um controle sobre determinado bloco ou circuito, que identifica se os sinais e estados daquele módulo não estão sofrendo modificação, em caso afirmativo, é desligado o relógio do módulo pela utilização do *clock gating*.

Um circuito *clock gating* é obtido pela modificação do circuito descrito na figura 2.1(a). Sendo o circuito sequencial com lógica de *clock gating* mostrado na figura 2.1 (b).

Em que se define a chamada função de ativação (Fa) igual a 1, quando o bloco lógico não modifica o nível lógico dos estados ou das entrada, selecionando-se assim a parada do relógio local do sistema. O bloco “L” representa um *latch*, transparente para o relógio em zero lógico. E o último estágio é a AND que determina a passagem ou não do relógio. (BENINI, 1999)

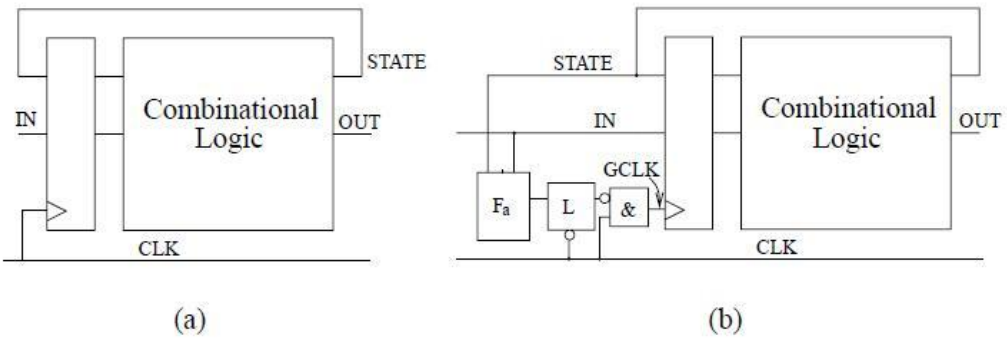

Figura 2.1 – (a) Circuito Sequencial (b) Versão com *clock gating* (BENINI, 1999)

A técnica de *clock gating* tem como principais características: i) A possibilidade de redução da potência dinâmica, mas somente em *latches* em que não há alta atividade de chaveamento, pois, caso contrário, o acréscimo de circuito de ativação/desativação do relógio acarretará num aumento no consumo de potência dinâmica. A função de ativação deve ser válida por uma fração significativa dos ciclos de relógio, para obter a diminuição desta potência pelo uso desta técnica; ii) quanto à potência estática, ela sofre um acréscimo devido ao aumento de área e transistores adicionais na lógica de ativação do "*gating*".

2.2.1.2Isolação de Operandos

A Isolação de Operandos é uma técnica para reduzir o consumo de potência devido às operações redundantes. Esta técnica ocorre dentro do bloco combinacional, isolando-se o circuito redundante e se prevenindo a atividade de chaveamento.

A isolação é feita com um *latch*. E tem como resultados negativos o acréscimo de área e atraso, e, como resultado positivo a diminuição da potência. Na figura 2.2, é mostrado um circuito sem isolação de operandos, e, na figura 2.3, é mostrado o circuito com isolação de operandos.

Sendo utilizando *latches* no bloco anterior ao somador de A e B, em que o *latch* é desativado, evitando se a propagação de sinal, se A e B, quando modificados, gerarem um valor igual ao anterior a modificação em C. Tendo-se $AS_{a0}f$ (figura 2.3), em valor '0', quando se tem computação redundante.

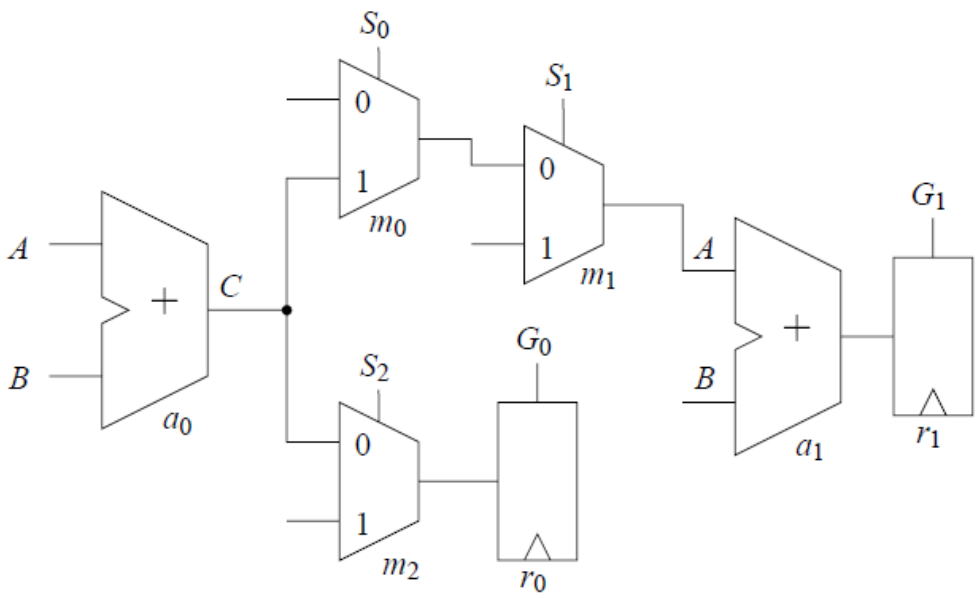

Figura 2.2 – Circuito sem Isolação de Operandos (MÜNCH, 2000).

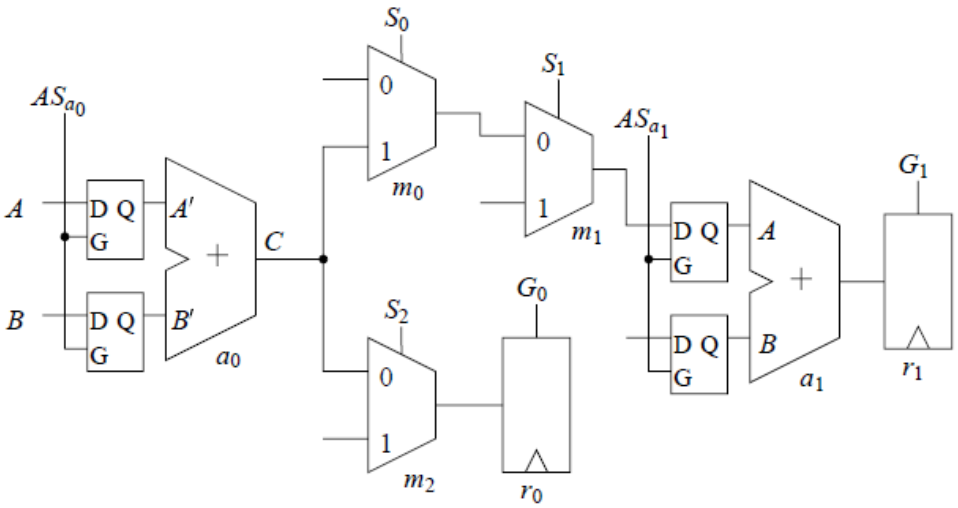

Figura 2.3 – Circuito com Isolação de Operandos (MÜNCH, 2000).

2.2.1.3Somadores Compressores

Um exemplo de técnica de minimização de consumo de potência são os somadores compressores. Nestes casos, existem algoritmos que diminuem as transições no bloco, sendo assim, tem-se um menor consumo de potência.

Um exemplo é o somador compressor 4-2, mostrado na figura 2.4, em que, por usar XORs e MUXs, é mais indicado a sua implementação com transistores de passagem. Diminuindo, além do número de transições do circuito, a área ocupada pelo somador.

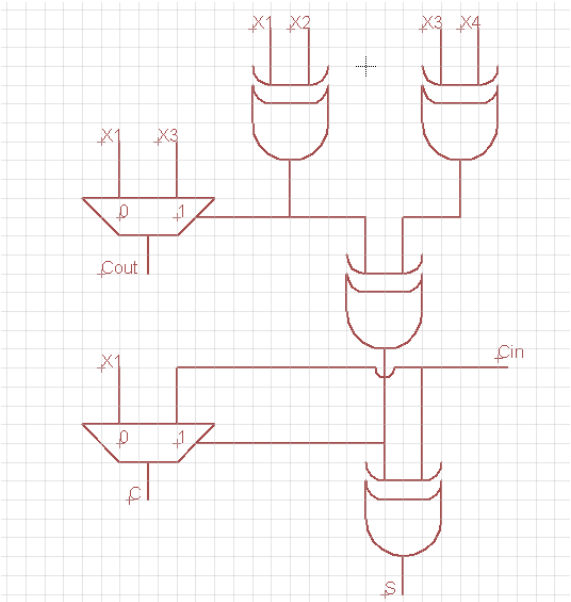

Figura 2.4 – Somador Compressor de 4-2.

Estes Somadores compressores, muitas vezes, utilizam transistores de passagem como no somador compressor 4-2 de (RADHA, 2000) e, também, somente transistores CMOS como no artigo de (VEERA, 2007).

2.2.2 Nível Elétrico

2.2.2.1Múltiplas Tensões de Alimentação

A técnica de múltiplas tensões de alimentação é muito utilizada, pois diminui a potência dinâmica e estática de um circuito sem apresentar perdas significativas em atraso, pois a tensão de alimentação menor pode ser usada somente nos blocos que exijam pouco desempenho em termos de atraso.

Quando se tem determinados blocos, como memórias, processadores e blocos de células padrão em um mesmo circuito integrado, geralmente, eles utilizam tensões diferenciadas, para isso, é necessário um modificador de nível, que são células específicas para fazer a passagem de uma tensão para outra.

Nesta seção, são mostrados os blocos do *Up Converter* (figura 2.5) e *Down Converter* (figura 2.6), sendo o *Up Converter*, um DCVS(*Differencial Cascode Voltage Switch Logic*) utilizado para a passagem de sinais de um domínio de tensão menor para uma tensão maior, e o *Down Converter*, um inversor utilizado para a passagem de uma tensão maior para uma tensão menor. O inversor da fig. 2.6 é dimensionado com transistores que nominalmente suportem tensões de *gate* V_{DD_H}.

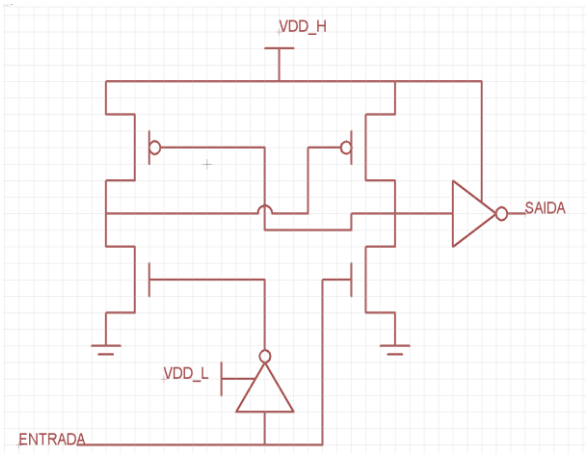

Figura 2.5 – Circuito *Up Converter*

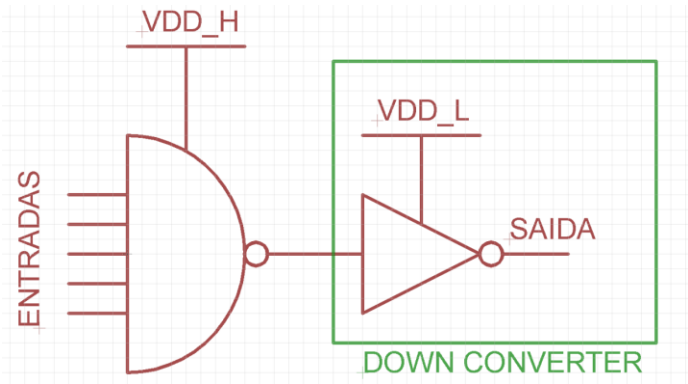

Figura 2.6 – Circuito *Down Converter*

2.2.2.2Multi-Threshold

Tensão de *Multi-Threshold* é uma das novas técnicas de minimização do consumo de potência em que se permite reduzir a corrente estática em transistores de geometrias de (130 nm, 90 nm, ...). Hoje em dia, as bibliotecas de células oferecem duas ou três versões de suas células: *Low VT*, *Standard VT* e *High VT*.

A implementação destas bibliotecas pode ter vantagens para diminuir o atraso em tempo e a potência simultaneamente, como demonstra a figura 2.7. Estas células podem ser implementadas para alta performance, usando células *Low VT* nos caminhos críticos, e, nos caminhos não críticos, podendo definir células *Standard VT* ou *High VT*, tendo assim uma diminuição da potência nos caminhos não críticos, tipicamente. Estas tensões típicas de limiar diferem em cerca de 100 mV entre *Low VT* e *Standard VT*, ou *Standard VT* e *High VT*.

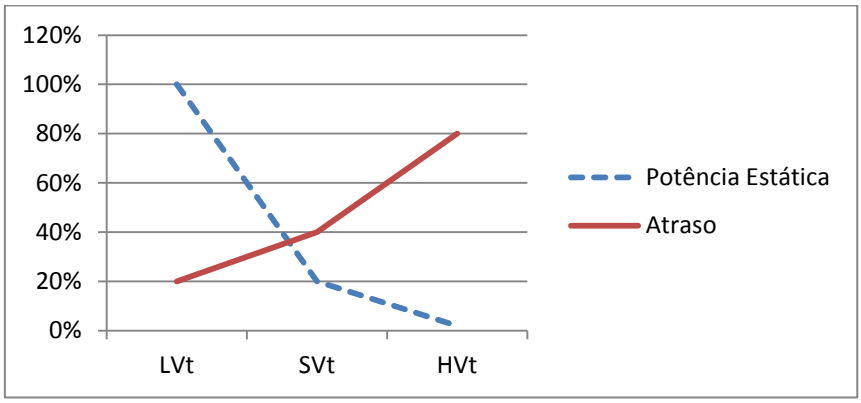

Figura 2.7 – Potência Estática Vs Atraso em 90 nm (BD03, 2008)

Também, pode-se optar por um sistema com a mínima potência estática, definindo-se principalmente o mapeamento de células *High VT* e tendo-se uma sensível diminuição na frequência do sistema.

2.2.3 Nível Arquitetural

No nível arquitetural, têm-se entre as várias técnicas duas para testar a diminuição ou não do consumo de potência, são elas: o aumento do paralelismo e o aumento dos estágios de *pipeline*.

Os fatores que diminuem o consumo de potência com o aumento do paralelismo e mantida a mesma taxa de transferência são mostrados na tabela 2.1. A esquerda da seta com sentido para baixo e linha lisa identifica-se os fatores que diminuirão o consumo de potência. A esquerda da seta com sentido para cima e linha hachurada está o fator que aumenta o consumo de potência.

Quando se aumenta o paralelismo, tem de se diminuir a frequência na mesma proporção para manter a mesma taxa de transferência. Pelo fato de se diminuir a frequência, pode-se usar células mais lentas, que consomem menos potência estática e dinâmica, células *High-VT*, quando existir esta opção na biblioteca, que consome menos

potência estática e, também, usar uma tensão de alimentação menor que abaixa o consumo de potência dinâmica e estática em uma proporção muito grande.

Porém, o aumento de paralelismo não somente diminui o consumo de potência, ele também tem um fator que amplia o consumo de potência que é o aumento de área.

Tabela 2.1: Fatores que diminuem a potência consumida (linha lisa) e aumentam (linha hachurada) quando usado o aumento de paralelismo e mantida a taxa de transferência (*throughput*).

▪ Diminui a frequência; ▪ Utiliza células mais lentas; ▪ Utiliza células *High-VT*; ▪ Utiliza menor tensão e mantém as células lentas;	▪ Aumenta a área;

Com o aumento dos estágios de *pipeline* e mantida a mesma taxa de transferência, os fatores que aumentam e diminuem o consumo de potência são mostrados na tabela 2.2 da mesma maneira que a tabela 2.1.

Sendo que com o aumento dos estágios de *pipeline* em cada barreira temporal, se tem menos células para vencer o tempo de um estágio de pipeline. Assim, se usa células mais lentas, *High-VT* e com tensão de alimentação menor, pois elas são mais lentas e consomem menos potência.

Porém, com o aumento dos estágios de *pipeline*, se utiliza mais *flip-flops* que são as células que têm o maior consumo de potência nas bibliotecas. Não se sabendo, por isso, quando esta técnica arquitetural diminui ou aumenta o consumo de potência, isto é, dependente do caso a ser analisado.

Tabela 2.2: Fatores que diminuem a potência consumida (linha lisa) e aumentam (linha hachurada) quando usado o aumento de estágios de pipeline e mantida a taxa de transferência (*throughput*).

▪ Utiliza células mais lentas; ▪ Utiliza células *High-VT*; ▪ Utiliza menor tensão e mantém as células lentas;	▪ Aumenta o número de *flip-flops*;

2.3 Conclusão

Este capítulo revisou as principais técnicas de economia de potência em circuitos digitais CMOS. No trabalho a seguir, que trata do projeto de um Núcleo de Processamento Aritmético no Capítulo 3, foram utilizadas as seguintes técnicas:

- De exploração, no nível arquitetural, do paralelismo e *pipeline*;
- No nível elétrico, a utilização de células lógicas de múltiplos *thresholds;*
- No nível lógico, técnicas de *clock gating;*
- Além disto, neste projeto de um Núcleo de cálculo de SAD (soma de diferenças absolutas), o trabalha faz análises em diferentes tecnologias CMOS e utiliza diferentes tensões de alimentação para os circuitos.

No Capítulo 4 desta dissertação é descrito o projeto físico de um Decodificador de Vídeo H.264 *Intra-Only*. Na síntese lógica e na síntese física deste decodificador foi utilizada uma tecnologia CMOS 65nm e sua respectiva biblioteca de células. Naquele projeto foram aplicadas as técnicas:

- No nível lógico, técnicas de *clock gating;*
- No nível elétrico, a utilização de células lógicas de múltiplos *thresholds.*

3 NÚCLEO DE PROCESSAMENTO ARITMÉTICO PARA ESTIMAÇÃO DE MOVIMENTO EM VÍDEO

Com a análise temporal de um vídeo, nota-se que as imagens são muito semelhantes entre si. As principais variações entre as imagens são devido ao movimento da câmera ou de objetos na cena.

Há um ativo campo de pesquisa em estimação de movimento (PORTO, 2008), que visa identificar a redundância temporal entre as imagens vizinhas e mapeá-las através de vetores de movimento, de forma a reduzir o conjunto de informações a ser enviado para o decodificador de vídeo, o que resulta na compressão do vídeo.

Existem vários algoritmos de busca para estimação de movimento, sendo o *Full Search* o que pesquisa em todas as amostras vizinhas do quadro de referência, necessitando, por isso, um maior número de cálculos para colocar a amostra estimada na saída do circuito de estimação de movimento.

Sendo assim, existem outros algoritmos de busca, que minimizam o consumo de potência, área e quantidade de cálculos necessários para se gerar a amostra estimada. No trabalho de (PORTO, 2009), se procurou minimizar o consumo de potência aplicando sub-amostragem de pixels e um algoritmo *Diamond Search*, que pesquisa numa área em forma de losango no quadro de referência e tendo como centro a mesma região da amostra em questão do quadro atual.

A figura 3.1 mostra a arquitetura do trabalho de (PORTO, 2009), em que estão incluídas 9 unidades de processamento, cada uma podendo processar 8 amostras em paralelo, sendo este processamento feito com o algoritmo de cálculo de diferenças.

A maneira como estas diferenças entre as regiões comparadas são avaliadas, se chama critério de similaridade. Dentre os critérios possíveis de serem utilizados se destacam; erro quadrático (do inglês, *Mean Square Error*, MSE), menor erro absoluto(do inglês, *Mean Absolute Error*, MAE) e soma de diferenças absolutas(do inglês, *Sum of Absolute Difference*, SAD).

O critério de similaridade mais utilizado em implementação de codificadores de vídeo é o SAD, devido à sua simplicidade de implementação frente aos outros métodos (PORTO, 2008). Esta métrica foi adotada para ser utilizada neste trabalho.

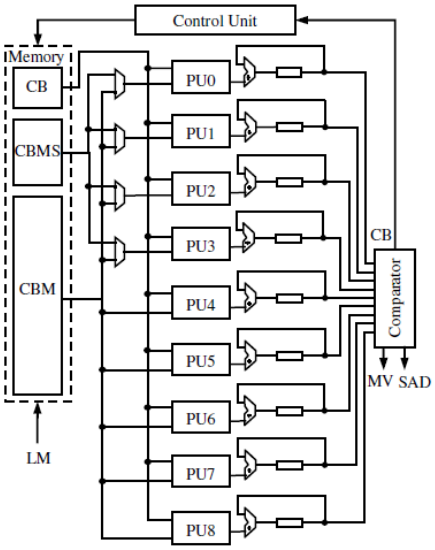

Figura 3.1: Estimação de Movimento com Diamond Search e Sub-Amostragem(PORTO, 2009).

3.1 Algoritmo de SAD

SAD é calculada para estimação de movimento (predição-inter). O cálculo de SAD é feito com a comparação entre as amostras do bloco predito, que é a amostra do bloco escolhido do quadro de referência, com as amostras do bloco atual. As operações de SAD são descritas pela equação abaixo.

$$SAD = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left| P_{i,j} - O_{i,j} \right| \qquad (3.1)$$

Nesta equação (3.1), P é um bloco candidato predito, O é o bloco original e *m* e *n* são as dimensões do bloco em número de amostras.

3.2 Arquiteturas de SAD com Paralelismo e *Pipeline*

No trabalho de (DINIZ, 2010) foram apresentadas algumas alternativas arquiteturais para o módulo de SAD. Estes módulos de SAD compreendem três níveis de paralelismo (com 4, 8 ou 16 amostras de 8-bits de entrada), e quatro níveis de *pipeline* (1, 3, 5 ou 6 estágios de pipeline). As Figuras 3.2, 3.3 e 3.4 mostram os três níveis de paralelismo juntamente com as profundidades de *pipeline* aplicadas a cada versão (DINIZ, 2010). Sendo a quantidade de Orig (bloco de amostra original) e Pred (bloco de amostra predita) em pares, o número de amostras de entrada. Os tracejados são barreiras temporais, sendo o tracejado em preto e o desenho do flip-flop são os *flip-flops* usados em todas as versões de arquiteturas, o tracejado em vermelho se refere à barreira temporal que é utilizada somente para os *pipelines* com 3, 5 e 6 estágios, e o tracejado em azul é referente aos *flip-flops* usados somente nos *pipelines* com 5 e 6 estágios.

Tal projeto de SAD, foi feito para um Bloco 4X4 (que é um bloco de quatro amostras verticais por quatro amostras horizontais), por isso, o uso da realimentação é

utilizada nas arquiteturas com 4 e 8 amostras, pois estas não compreendem um bloco 4X4 em uma iteração. Já nas arquiteturas com 16 amostras não é utilizada a realimentação, pois esta arquitetura já compreende o bloco 4X4 completo.

Pode-se destacar também que foi utilizado o reset assíncrono nas barreiras temporais. Devido ao fato de que, sem a ação de reset, os dados de saída ficam sempre em alta impedância.

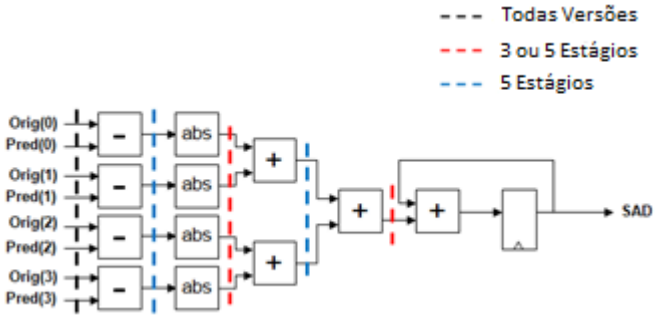

Figura 3.2: Arquitetura de SAD com 4 amostras de entrada e diferentes profundidades de *pipeline*. (DINIZ, 2010)

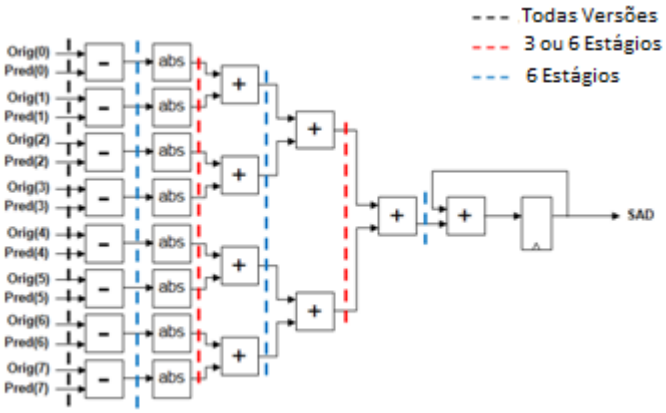

Figura 3.3: Arquitetura de SAD com 8 amostras de entrada e diferentes profundidades de *pipeline*. (DINIZ, 2010)

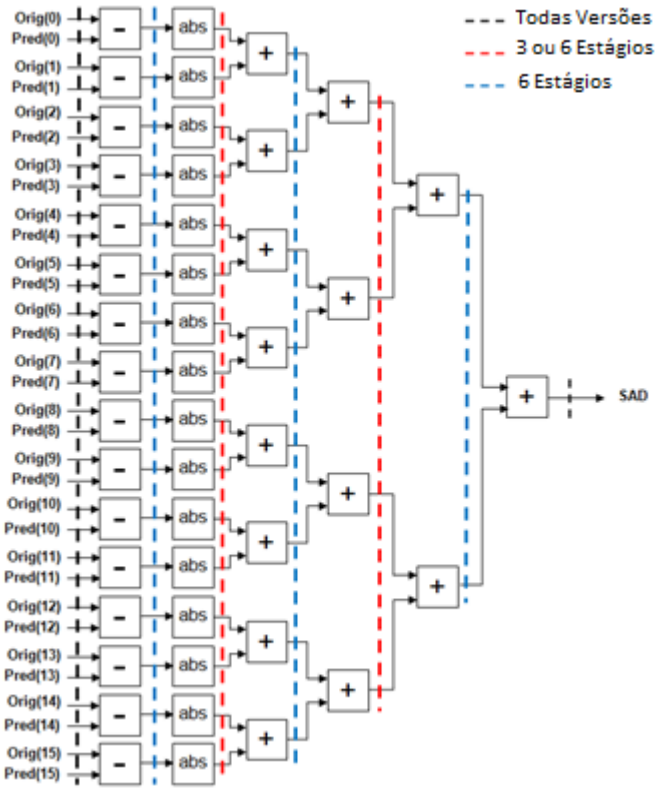

Figura 3.4: Arquitetura de SAD com 16 amostras de entrada e diferentes profundidades de *pipeline*. (DINIZ, 2010)

3.3 Método de Análise

O fluxo de *Front-End* e de *Back-End* utilizado para desenvolver as arquiteturas de SAD é mostrado na Figura 3.5. As arquiteturas são sintetizadas na tecnologia de 180 nm, com a biblioteca da TSMC, e, em 65 nm, com a biblioteca da *Synopsys* para o processo da IBM *Microelectronics*.

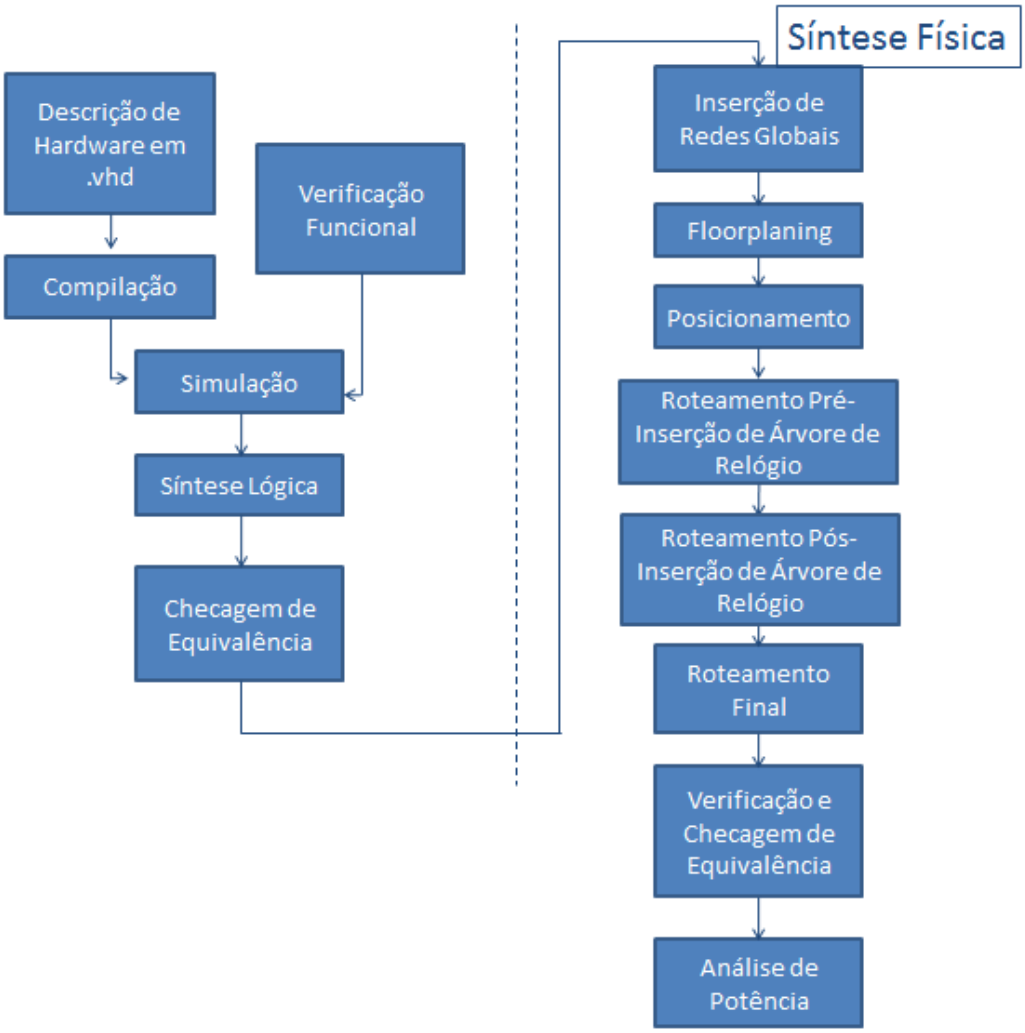

Figura 3.5: Fluxo para Análise de Potência das Arquiteturas de SAD.

A verificação funcional é separada em três blocos que são mostrados na Figura 3.6.

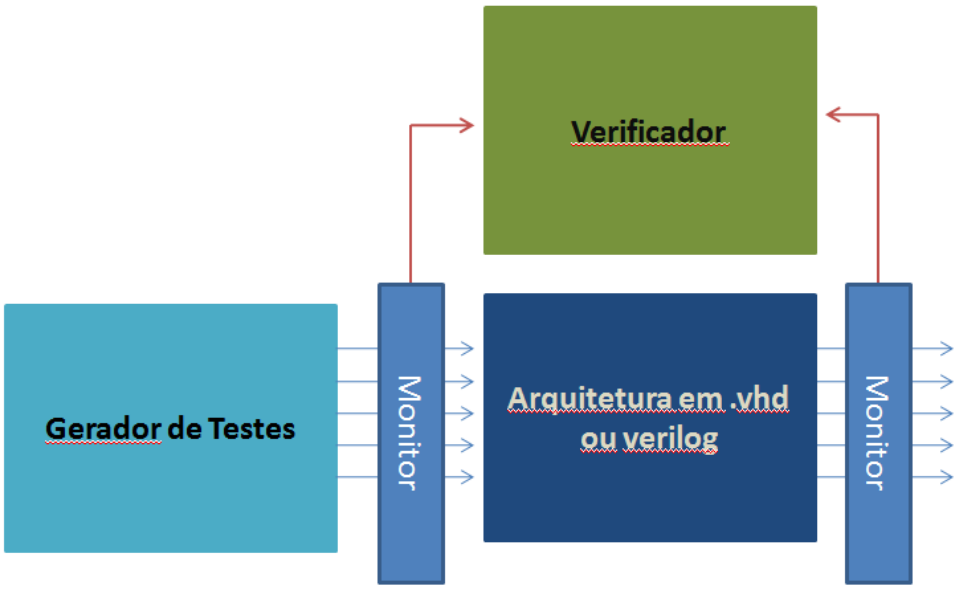

Figura 3.6: Modelo da Verificação Funcional

Os três blocos da figura 3.6 são: Gerador de Testes, Monitor e Verificador (como pode ser verificado na descrição em System Verilog no apêndice A) (SPEAR, 2008). O Gerador de Testes é um gerador com valores randômicos para a arquitetura descrita em .vhd. O Monitor, na Figura 3.6, é o bloco que monitora as entradas e as saídas da arquitetura, enviando tais dados ao Verificador. O Verificador é o bloco que verifica se os dados que estão saindo da Arquitetura estão de acordo com as entradas geradas pelo Gerador de Testes. Estando corretas, continua a verificação por um tempo de 1 ms, e, estando erradas, mostra as entradas que geraram tal saída, o valor correto para a saída e o valor que a simulação da arquitetura apresentou.

Na verificação funcional foram encontradas falhas nos dados de saída que estavam sempre como alta impedância, devido ao fato de não terem sinal de reset. Erro este corrigido acrescentando-se reset assíncrono em todos os *flip-flops* da arquitetura de SAD.

O próximo passo é a síntese lógica em 180 nm que é configurada de duas maneiras com e sem inclusão de *clock gating*. Já a biblioteca de 65 nm é configurada somente sem *clock gating*, devido a não obtenção de bons resultados nas análises de energia por operação em *isoperformance* e nas análises de taxa de transferência para máxima taxa de transferência para esta arquitetura utilizando *clock gating*. Nesta etapa, são incluídas as bibliotecas de células que são as .lib e .lef que na tecnologia de 180 nm, apresenta somente a tensão de 1,8 Volts como típica. Já na tecnologia de 65 nm, a tensão típica é de 1,2 e 1,0 Volts, além de apresentar múltiplas tensões de *Threshold*. Também, é necessário um arquivo de controle dos tempos que é .sdf, em que a principal função é definir o tempo do relógio de sistema. Todos estes arquivos e ainda a descrição de hardware são incluídos nos *scripts* .tcl para fazer esta síntese lógica.

Os principais destes arquivos da síntese lógica estão descritos no apêndice B.

Logo após, é feita a checagem de equivalência entre a descrição de hardware original e a *netlist*. Neste, são mostradas as figuras de Checagem de Equivalência para Comparação de Hierarquia, Figura 3.7, e para Gerenciamento de Mapeamento, Figura 3.8. Os *stripes* (linhas de reforço de alimentação) verticais e horizontais de reforço de alimentação estão separados de 100um, como linhas de reforço de tensão com uma espessura maior, evitando-se densidades de corrente em excesso nas linhas de potência com largura menor. Assim, dando maior robustez ao circuito e consumindo uma menor quantidade de potência.

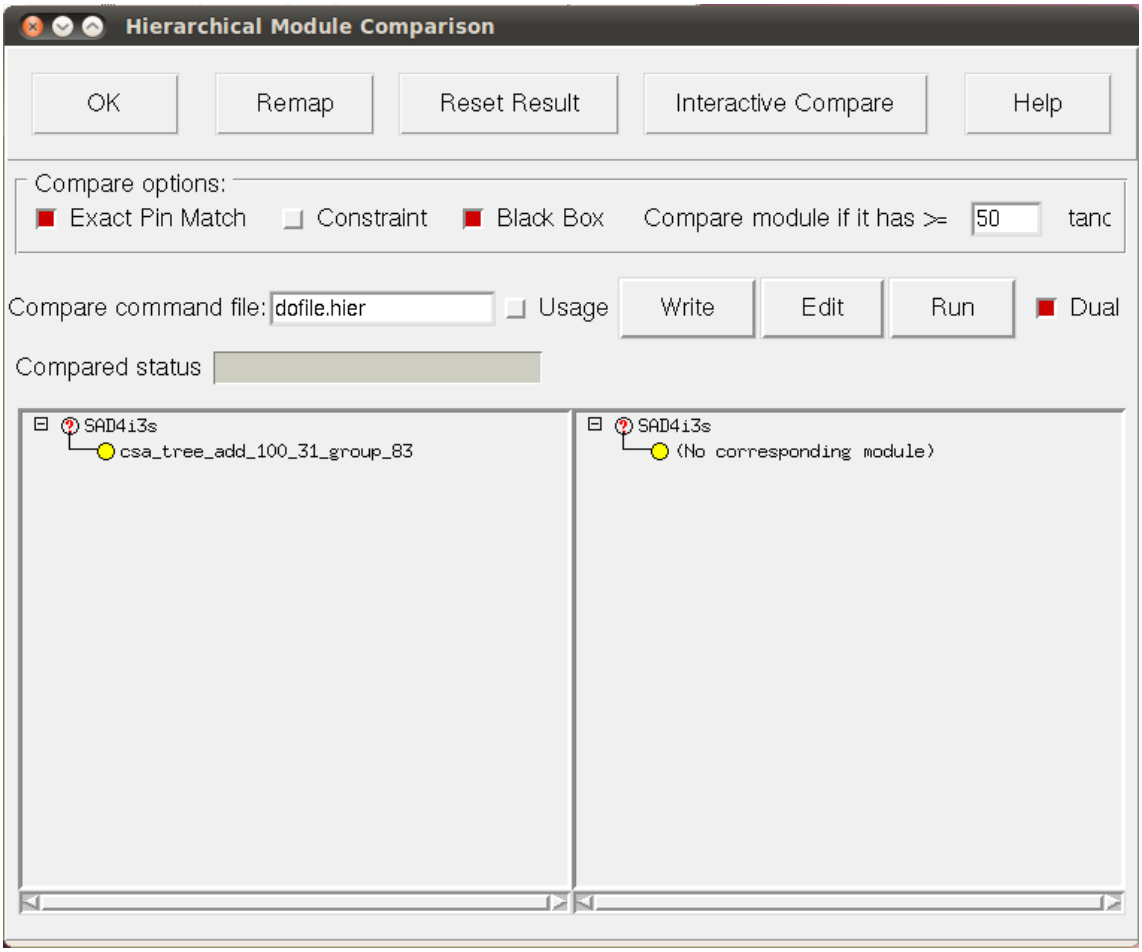

Figura 3.7: Módulo de Checagem de Equivalência para Comparação de Hierarquia.

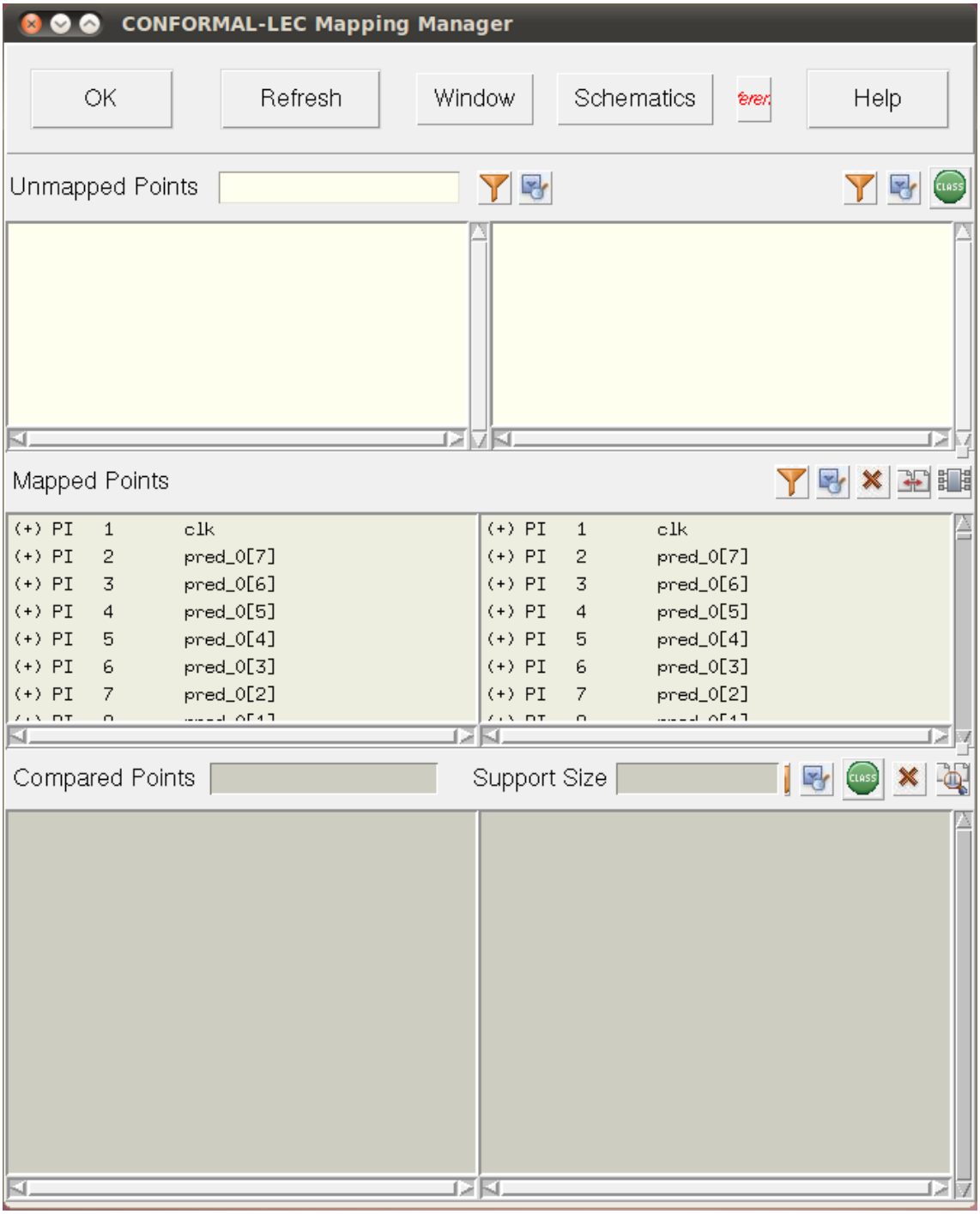

Figura 3.8: Módulo de Checagem de Equivalência para Gerenciamento de Mapeamento.

Tem-se na figura 3.9 as linhas de potência implementadas com as funções da ferramenta da *Cadence*(*Encounter*): *ADD RINGS*, *ADD STRIPES* e *SROUTE* para o Metal 1 em 180 nm e Metal 2 em 65nm. Sendo que o Metal 2 teve-se que fazer a correção de algumas falhas da ferramenta manualmente. Os *stripes* verticais e horizontais de reforço de alimentação estão separados de 100um nos circuitos de 180 nm e 50um em 65nm, neste caso com o VDD e GND também espaçados de 50um. Este espaçamento é definido nos documentos da Virage Logic(VIRAGE LOGIC, 2009) para a biblioteca de 65nm utilizada.

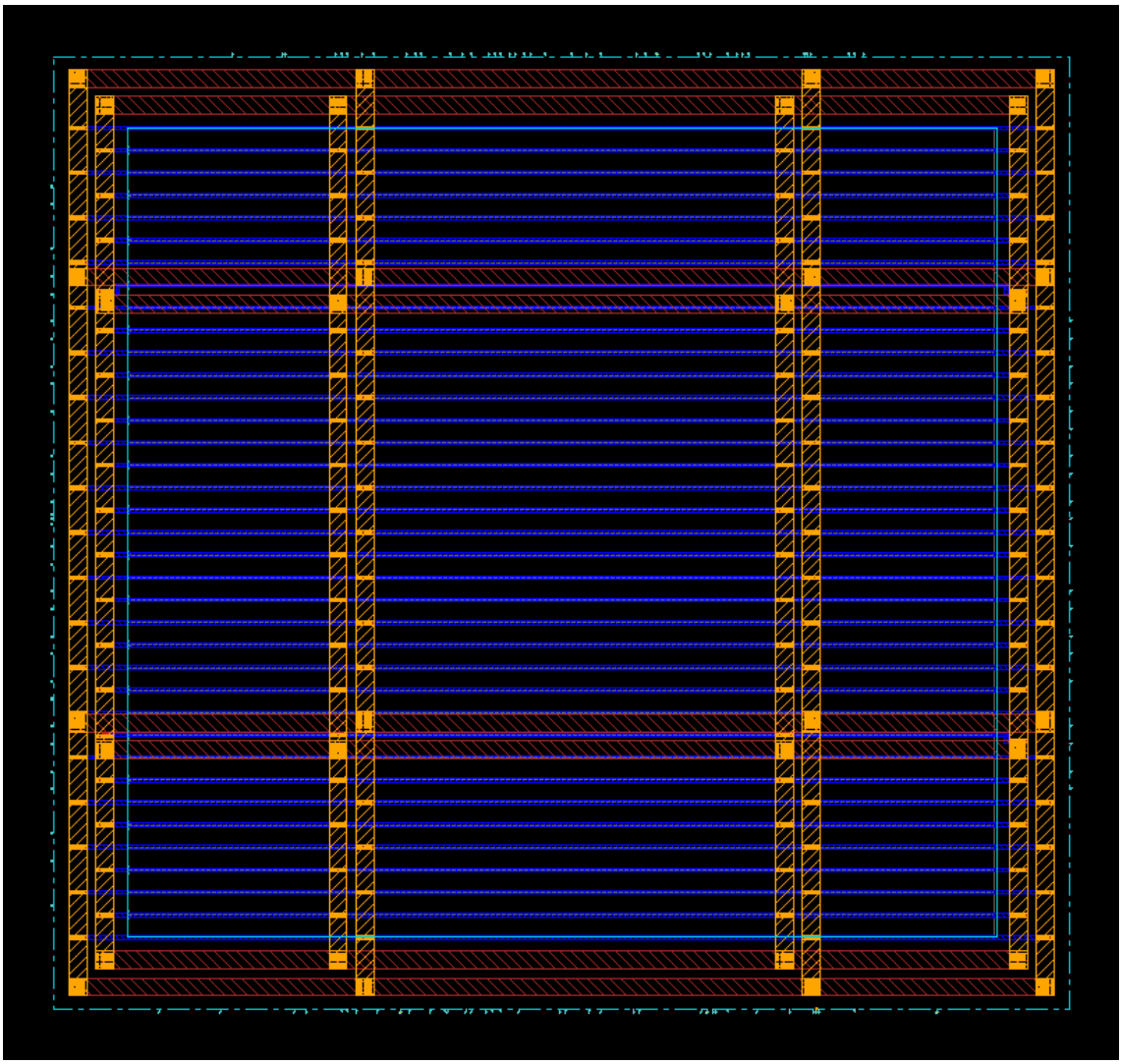

Figura 3.9: Planejamento de VDD/GND no *Encounter* para o circuito do SAD em 180nm.

Na figura 3.10, está mostrado como é feita a distribuição de relógio do circuito após a síntese de roteamento final, que está no formato de árvore H. Esta árvore H tem como principais características manter igual atraso de propagação em todo o circuito e, também, minimizar o *skew* do relógio. (BD03, 2008)

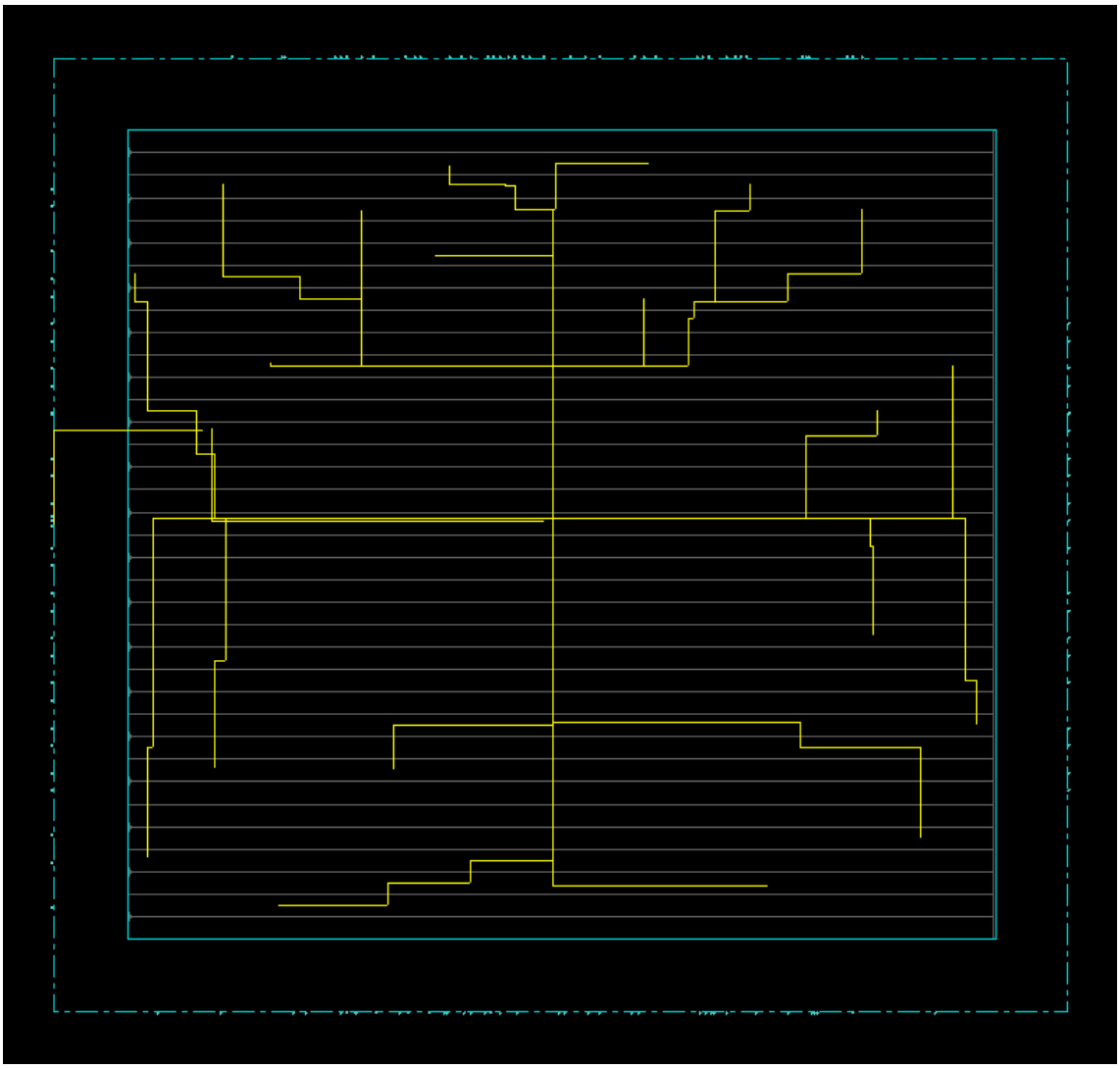

Figura 3.10: Modelo das linhas de relógio do núcleo de SAD integrado no *Encounter* depois da síntese de *Post-Route*.

Finalizando são desenvolvidas as fases de verificação de Geometria e de Conectividade. Logo após, é feita a análise de potência estática com o *Power Analysis* e a análise de potência dinâmica.

3.3.1 Método de Análise de Máxima Taxa de Transferência

As análises de máxima taxa de transferência, medida em macro-blocos por segundo na análise em questão, foram feitas colocando-se um período de relógio alvo que a ferramenta de síntese não possa alcançar e verificando-se até que ponto pode-se diminuir o período sem ocasionar erro na análise de tempo. A seguir este período mínimo é especificado no arquivo .sdc que utilizou-se para a síntese lógica e física.

3.3.2 Método de Análise de Mínima Requerida *Isoperformance*

As análises de mínima requerida i*soperformance* foram feitas buscando-se um desempenho de 1 milhão de macro-blocos processados por segundo, o que já é bem maior que a mínima requerida performance do padrão HDTV de 244.800 macro-blocos processados por segundo.

3.4 Resultados da Síntese Física de Arquiteturas para SAD em 180 nm

3.4.1 Máxima Taxa de Transferência em 180 nm

Na tabela 3.1, é mostrada a análise de potência na síntese física para máxima taxa de transferência em 180 nm, com a potência sendo medida no pior caso para o desempenho temporal (*timing*), isto é, 1,62 Volts e 125ºC. A potência estática e dinâmica são menores na análise com 4 amostras de entrada e com um estágio de *pipeline*, pois sabe-se que os valores de potência é menor devido menor paralelismo. Dobrando-se o paralelismo dobra-se a área do circuito, aumentando-se a potência estática. Além disto, uma menor profundidade de *pipeline* requer menos *flip-flops*, que são o principal fator no consumo de potência, além de resultar em menor frequência máxima.

Nas análises de 16 amostras de entrada não se conseguiu fazer a análise com *clock gating*, pois a ferramenta utilizada não conseguiu aplicar tal técnica de redução de consumo de potência.

Tabela 3.1: Resultados de Máxima Taxa de Transferência em 180 nm na Síntese Física.

# Amos-tras de Entrada	*# Estági-os de Pipeli-ne*	*Clock Gatin g*	*Freqüên-cia (MHz)*	*Potência Estática (nW)*	*Potên-cia Interna (mW)*	*Potência de Chaveamen-to (mW)*	*Potên-cia Total (mW)*
4	1	Não	204	889,8	5,31	2,754	8,065
4	1	Sim	204	1092	5,715	2,846	8,562
4	3	Não	370	1469	18,65	7,392	26,04
4	3	Sim	357	1453	17,67	6,352	24,02
4	5	Não	476	1502	32,9	10,68	43,58
4	5	Sim	400	1200	21,23	5,843	27,08
8	1	Não	185	1836	9,482	5,536	15,02
8	1	Sim	188	2062	10,16	5,726	15,89
8	3	Não	344	2106	26,63	9,643	36,28
8	3	Sim	333	2079	24,51	9,16	33,67
8	6	Não	476	2797	62,38	19,6	81,98
8	6	Sim	384	2251	37,41	10,05	47,46
16	1	Não	192	3536	17,87	9,515	27,38
16	1	Sim					
16	3	Não	357	4813	57,9	22,61	80,51
16	3	Sim					
16	6	Não	476	5261	118	34,15	152,2
16	6	Sim					

No gráfico da figura 3.11, é mostrada a taxa de transferência (*throughput*) alcançada por cada uma destas análises. Sendo a maior a que tem maior paralelismo e a com mais estágios de *pipeline*. Devido ao fato de quando se dobra o paralelismo se dobra a taxa de transferência e, com mais estágios de *pipeline*, se pode aumentar a frequência da arquitetura em questão. Também nota-se que os circuitos com *clock gating* apresentam uma menor taxa de transferência, pois está limitando a passagem de informações pela seleção do relógio em situações favoráveis.

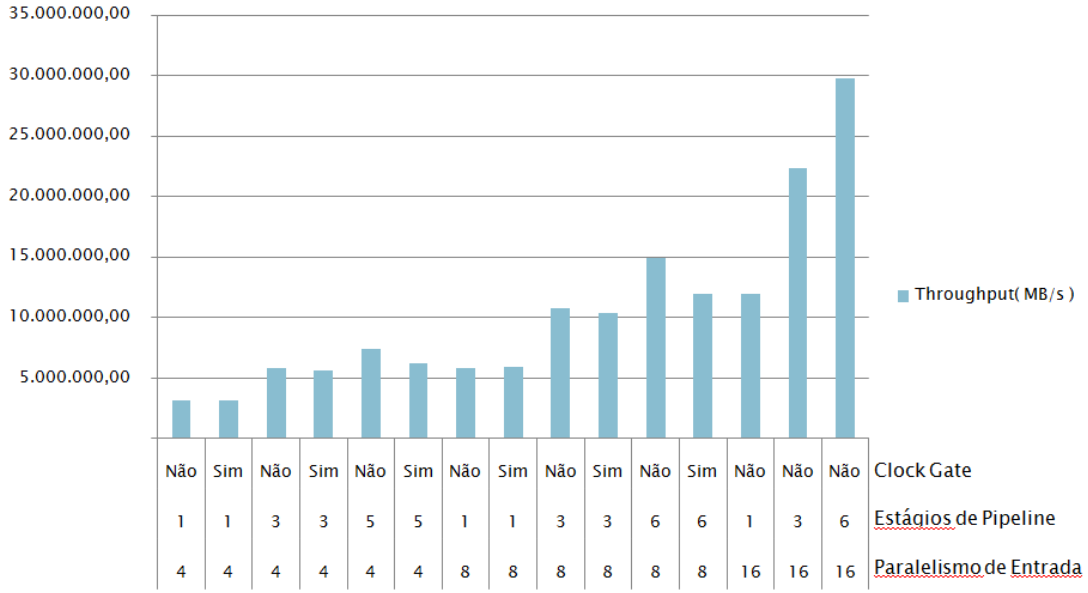

Figura 3.11: Gráfico da Máxima Taxa de Transferência em 180 nm alcançada para cada análise na Síntese Física. MB/s = Macro-blocos/segundo

Além disto, é mostrado na tabela 3.2 a média das potências estáticas e dinâmicas por *gates* equivalentes e por área com células padrão em $mm^{2.}$

Tabela 3.2 – Média de Consumo de Potência por *gates* equivalentes e área para as arquiteturas com Máxima Taxa de Transferência em 180 nm.

	Potência Estática por *gates* equivalentes	Potência Dinâmica por *gates* equivalentes	Potência Estática por área	Potência Dinâmica por área
Média sem *Clock Gating*	426,39 pW/*gate*	7,60 uW/*gate*	$42{,}73 \times 10^{-6}$ W/mm^2	$761{,}97 \times 10^{-3}$ W/mm^2
Média com *Clock Gating*	417,65 pW/*gate*	6,25 uW/*gate*	$41{,}85 \times 10^{-6}$ W/mm^2	$626{,}92 \times 10^{-3}$ W/mm^2

3.4.2 Mínima Requerida *Isoperformance* em 180 nm

Na tabela 3.3, é mostrada a análise de potência na síntese física para a mínima requerida *isoperformance* em 180 nm, em que o desempenho das arquiteturas da é de 1 milhão de MB/s. Notando-se que o consumo de potência dinâmica nos circuitos com *clock gating* aumenta na maioria dos casos, sendo isso pelo fato das altas atividades do caminho não garantirem que o uso de *clock gating* diminui o consumo de potência dinâmica.

Nota-se que com o aumento de paralelismo se aumentou a potência estática pelo circuito ocupar uma área maior e, também, que com o aumento do paralelismo a potência dinâmica foi reduzida, devido à redução da frequência.

Tabela 3.3: Resultados de Mínima Requerida *Isoperformance* em 180 nm na Síntese Física.

# Amos-tras de Entrada	*# Estági-os de Pipeli-ne*	*Clock Gatin g*	*Freqüên-cia (MHz)*	*Potência Estática (nW)*	*Potên-cia Interna (mW)*	*Potência de Chaveamen-to (mW)*	*Potên-cia Total (mW)*
4	1	Não	67	478,5	0,9119	0,3636	1,276
4	1	Sim	67	661,5	1,015	0,4198	1,435
4	3	Não	67	709	1,46	0,4314	1,892
4	3	Sim	67	704,8	1,478	0,4566	1,935
4	5	Não	67	911,6	2,196	0,5579	2,754
4	5	Sim	67	1100	2,307	0,6576	2,965
8	1	Não	33	1072	0,8533	0,3493	1,204
8	1	Sim	33	1243	0,8886	0,3799	1,27
8	3	Não	33	1370	1,342	0,4145	1,758
8	3	Sim	33	1334	1,335	0,4031	1,739
8	6	Não	33	1574	2,103	0,5355	2,64
8	6	Sim	33	1706	2,163	0,5884	2,753
16	1	Não	17	1939	0,7584	0,3275	1,088
16	1	Sim	17				
16	3	Não	17	2484	1,236	0,324	1,601
16	3	Sim	17				
16	6	Não	17	3059	2,009	0,5057	2,518
16	6	Sim	17				

No gráfico da figura 3.12 é mostrada a energia por operação alcançada por cada uma destas análises. Sendo o melhor caso o com mínima energia por operação, que ocorre quando se tem o menor número de estágios de *pipeline*, já que utiliza menos *flip-flops*, e com a maior taxa de paralelismo, que para manter a mesma taxa de transferência utiliza a metade da frequência. Reduzida a frequência alvo na síntese, pois há um "slack" maior combinacional, sabe-se que diminui proporcionalmente a potência dinâmica nas capacitâncias parasitas e de carga, e que células mais lentas são utilizadas no mapeamento, que têm um menor consumo de potência.

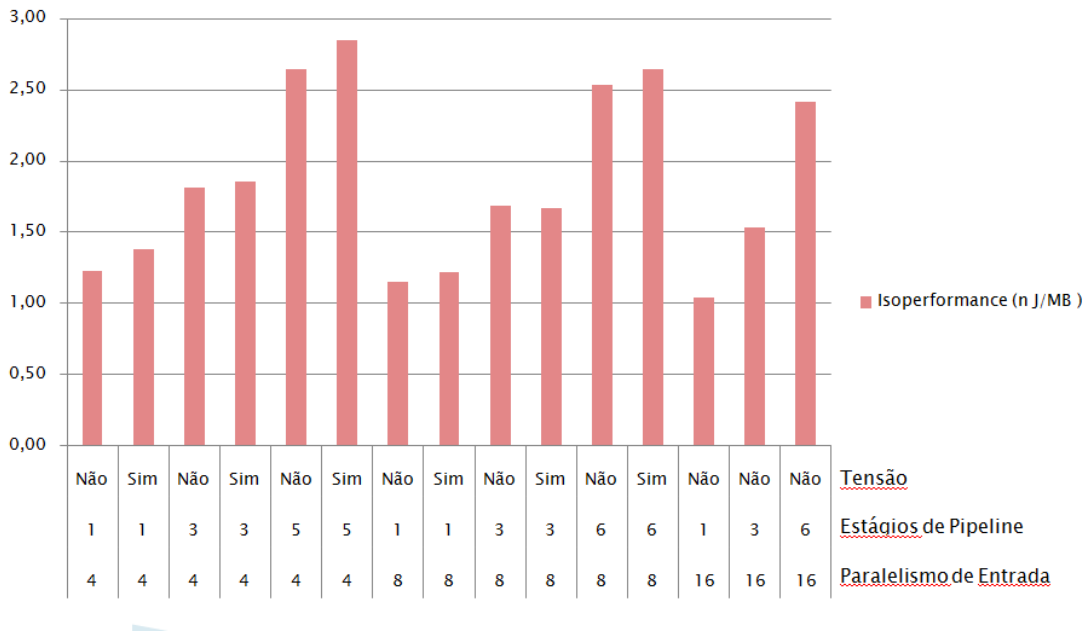

Figura 3.12: Gráfico da Energia por Operação em 180 nm alcançada para cada análise em *Isoperformance* na Síntese Física.

As medidas de potência estática e dinâmica por *gates* equivalentes e por área com células padrão em mm^2 são mostradas na tabela 3.4.

Tabela 3.4 – Média de Consumo de Potência por *gates* equivalentes e área para as arquiteturas em *Isoperformance* de 180 nm.

	Potência Estática por *gates* equivalentes	Potência Dinâmica por *gates* equivalentes	Potência Estática por área	Potência Dinâmica por Área
Média sem *Clock Gating*	363,74 pW/*gate*	581,37 nW/*gate*	$36,45 \times 10^{-6}$ W/mm^2	$58,25 \times 10^{-3}$ W/mm^2
Média com *Clock Gating*	370,90 pW/*gate*	710,14 nW/*gate*	$37,16 \times 10^{-6}$ W/mm^2	$71,15 \times 10^{-3}$ W/mm^2

3.5 Resultados de Síntese Física de Arquiteturas para SAD para 65 nm

Na biblioteca de 65 nm, foram feitos *designs* somente para circuitos sem *clock gating*, devido a não ser possível inserir *clock gating* nas análises com 16 amostras de paralelismo. Já na seção anterior restou comprovado que o uso deste "*gating*" não diminui sensivelmente o consumo de potência nas análises de *isoperformance*. Tem-se, também, que as bibliotecas CMOS 65nm utilizam tanto tensões típicas/nominais de 1,0 Volt e 1,2 Volts. Nas análises de pior caso a serem apresentadas, utilizam-se VDD=0,9 Volts VDD=1,08 Volts, i. e. 10% menos que o nominal

3.5.1 Máxima Taxa de Transferência em 65 nm

Com a tabela 3.5, são mostrados os resultados de potência e frequência alcançada pelos designs que geraram a máxima taxa de transferência nos SADs em CMOS 65 nm. A maior freqüência de operação, como se espera, é obtida quanto maior o número de estágios de pipeline e para a tensão de alimentação mais alta (1,08V). Além disso, notou-se que nestas análises aumentou-se o consumo de potência estática, devido ao uso em grande quantidade de células *Low-VT*, que possuem um consumo estático de potência maior que as células de VT padrão.

Tabela 3.5: Resultados de Máxima Taxa de Transferência em 65 nm na Síntese Física.

# Amos-tras de Entrada	*# Estági-os de Pipeli-ne*	*Tensão*	*Freqüên-cia (MHz)*	*Potência Estática (uW)*	*Potên-cia Interna (mW)*	*Potência de Chaveamen-to (mW)*	*Potên-cia Total (mW)*
4	1	0,9	416	13,780	0,734	1,083	1,832
4	3	0,9	714	11,000	1,133	1,477	2,622
4	5	0,9	833	10,740	1,489	1,565	3,065
8	1	0,9	384	23,490	1,145	1,715	2,884
8	3	0,9	625	16,440	1,664	1,777	3,457
8	6	0,9	667	18,900	2,243	2,418	4,680
16	1	0,9	384	40,130	1,913	2,943	4,896
16	3	0,9	555	20,220	2,242	2,341	4,603
16	6	0,9	625	27,220	3,608	3,277	6,913
4	1	1,08	625	23,410	1,308	2,052	3,384
4	3	1,08	833	13,140	1,497	1,393	2,904
4	5	1,08	909	16,790	2,730	2,987	5,734
8	1	1,08	555	41,010	2,155	3,275	5,471
8	3	1,08	667	16,640	2,085	1,907	4,009
8	6	1,08	714	21,510	4,213	3,907	8,141
16	1	1,08	555	65,430	3,567	5,437	9,069
16	3	1,08	625	24,190	4,035	3,959	8,018
16	6	1,08	667	29,150	7,681	5,709	13,42

A Figura 3.13 mostra a máxima taxa de transferência em 65 nm para os diferentes designs e se tem como melhores resultados os com máximo paralelismo e maior quantidade de estágios de *pipeline*, como em 180 nm. Além disso, com maior tensão, as células conseguem desenvolver um fluxo com menor atraso, por isso, os dados de mesmo paralelismo e *pipeline*, mas com tensão maior, apresentam uma maior taxa de transferência.

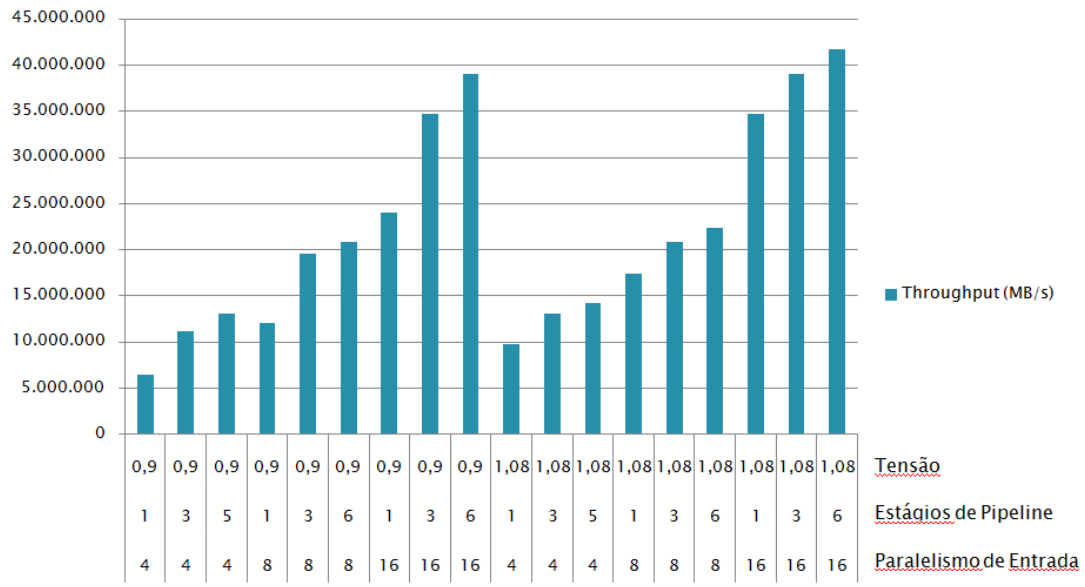

Figura 3.13: Gráfico da Taxa de Transferência em 65 nm alcançada para cada análise na Síntese Física.

Na Tabela 3.6, são mostradas as médias de consumo de potência por *gates* equivalentes e área para as diferentes tensões de alimentação em máxima taxa de transferência. Sua comparação com a biblioteca de 180 nm será feita na seção de Resultados Finais.

Tabela 3.6 – Média de Consumo de Potência por *gates* equivalentes e área para as arquiteturas com Máxima Taxa de Transferência em 65 nm.

	Potência Estática por *Gates* Equivalentes	Potência Dinâmica por *Gates* Equivalentes	Potência Estática por Área	Potência Dinâmica por Área
Média em 0,9 Volts	2501,13 pW/*gates*	493,49 nW/*gates*	$2{,}31 \times 10^{-3}$ W/mm^2	$456{,}92 \times 10^{-3}$ W/mm^2
Média em 1,08 Volts	3782,29 pW/*gates*	871,06 nW/*gates*	$3{,}50 \times 10^{-3}$ W/mm^2	$806{,}48 \times 10^{-3}$ W/mm^2

3.5.2 Mínima Requerida *Isoperformance* em 65 nm

Para as análises da Tabela 3.7 de potência e frequência em mínima requerida *isoperformance* de 65nm, com menor paralelismo, estágios de *pipeline* e tensão de alimentação levam a um menor consumo de potência, mas, sabendo-se que para manter a mesma performance tem de se diminuir a frequência à metade quando se dobra o paralelismo, o caso com maior paralelismo atrelado a menor frequência leva a um consumo menor de potência, pois estes circuitos usam células mais lentas e que consomem menos potência.

Tabela 3.7: Resultados de Mínima Requerida *Isoperformance* em 65nm na Síntese Física.

# Amos-tras de Entrada	*# Estági-os de Pipeli-ne*	*Tensão*	*Freqüên-cia (MHz)*	*Potência Estática (nW)*	*Potên-cia Interna (μW)*	*Potência de Chaveamen-to (μW)*	*Potên-cia Total (μW)*
4	1	0,9	66	426,9	65,94	41,67	108
4	3	0,9	66	358,8	99,51	52,52	152,4
4	5	0,9	66	468,3	149,4	73,8	223,7
8	1	0,9	33	758,3	56,97	40,46	98,19
8	3	0,9	33	627,7	92,87	51,85	145,3
8	6	0,9	33	1333	136	71,58	208,9
16	1	0,9	16	960	50,44	37,63	89,03
16	3	0,9	16	1069	79,03	47,05	127,2
16	6	0,9	16	2618	124,6	65,49	192,7
4	1	1,08	66	700,6	88,01	59,89	148,6
4	3	1,08	66	978,7	133,3	76,22	210,5
4	5	1,08	66	1323	201	105,1	307,4
8	1	1,08	33	1748	78,89	57,97	138,6
8	3	1,08	33	2255	121,3	73,37	197
8	6	1,08	33	2462	195,6	105,4	303,4
16	1	1,08	16	1495	69,19	53,39	124,1
16	3	1,08	16	1746	108,1	65,04	174,9
16	6	1,08	16	3285	178,4	95,96	277,7

Assim, o caso que consome menos Energia por Operação é o com menor tensão (0,9 Volts), menor número de estágios de *pipeline* (1 estágio) e maior paralelismo (16 amostras de entrada), como mostrado na Figura 3.14 onde o consumo de potência é mostrado no eixo vertical em pJ/macrobloco.

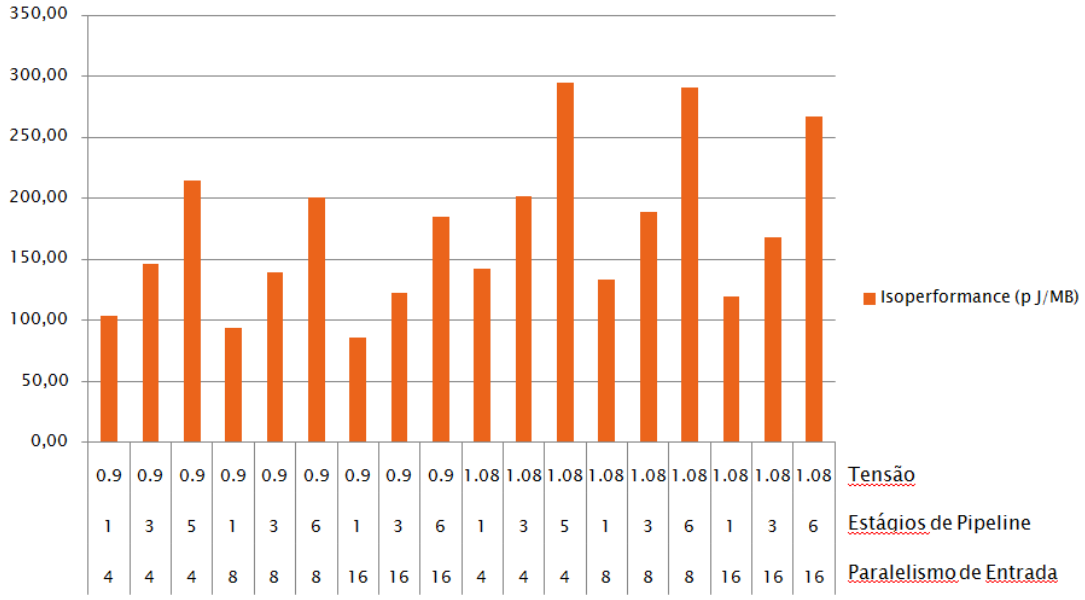

Figura 3.14: Gráfico da Energia por Operação em 65 nm alcançada para cada análise na Síntese Física.

Na Tabela 3.8, são mostradas as médias de consumo de potência por *gates* equivalentes e área para as diferentes tensões de alimentação em *isoperformance*. E sua comparação com a biblioteca de 180 nm será feita na seção de Resultados Finais.

Tabela 3.8 – Média de Consumo de Potência por *gates* equivalentes e área para as arquiteturas em *Isoperformance* de 65 nm.

	Potência Estática por *gates* equivalentes	Potência Dinâmica por *gates* equivalentes	Potência Estática por área	Potência Dinâmica por Área
Média em 0,9 Volts	171,02 pW/*gate*	35,04 nW/*gate*	$158{,}34 \times 10^{-6}$ W/mm^2	$32{,}44 \times 10^{-3}$ W/mm^2
Média 1,08 em Volts	367,10 pW/*gate*	50,51 nW/*gate*	$339{,}89 \times 10^{-6}$ W/mm^2	$46{,}77 \times 10^{-3}$ W/mm^2

3.6 Análise dos Resultados Alcançados

As análises dos resultados estão separadas em dois blocos principais. Estes blocos são:

- Comparação de Potência Média por *Gates* Equivalentes e Área com relação aos dados deste trabalho;
- Comparação de *Isoperformance* e Máxima Taxa de Transmissão com relação aos melhores casos deste trabalho e trabalhos de outros autores;

3.6.1 Comparação de Potência Média por *Gates* Equivalentes e Área com relação aos dados deste trabalho

Este bloco de análise está subdividido em potência por *gates* equivalente e potência por área. Tendo sido apresentados os resultados de *Isoperformance* e Máxima Taxa de Transmissão.

Nestas análises, são comparadas as médias das potências consumidas sem a utilização de *clock gating* para as bibliotecas de 180 nm e 65 nm. As análises de *Isoperformance* em 180 nm foram comparadas com a mesma análise em 65 nm e, da mesma forma, as análises de Máxima Taxa de Transferência em 180 nm comparadas com a mesma análise em 65 nm.

3.6.1.1Potência Média por Gates Equivalentes

Tem-se como forma de cálculo da potência média por *Gate* Equivalente, pela fórmula apresentada abaixo.

Sendo P_{MGE} a potência média por *gate* equivalente, P_M a potência média, A_{MT} a área média total ocupada por células e A_{GE} a área do *gate* equivalente, que é a área da menor NAND2 presente na biblioteca de células CMOS utilizada neste trabalho.

$$P_{MGE} = \frac{P_M}{\frac{A_{MT}}{A_{GE}}} \qquad (3.2)$$

As comparações são feitas para as duas potências médias: estáticas e dinâmicas.

3.6.1.1.1. Potência Estática

Na tabela 3.9, nota-se que as comparações com a síntese de 180 nm, 1,62 Volts e sem *clock gating* apresentam uma grande diminuição na análise de *Isoperformance* em 65nm e 0,9 Volts, isto é devido a tensão de alimentação e, também, ao uso de células *High-VT* que tem um consumo de potência estática bem reduzido.

Na comparação de *Isoperformance* de 65 nm e 1,08 Volts, tem-se um leve acréscimo na potência estática utilizada. Isto é devido ao uso de uma tensão um pouco maior, se comparada com a síntese de 65 nm e 0,9 Volts.

Estes dados de *Isoperformance* levam a conclusão de que tecnologias mais novas têm um diminuição de consumo de potência estática, quando se usa baixas tensões de alimentação e se necessita de baixa performance, isto é, utiliza-se células *High-VT*.

Nas comparações de máxima taxa de transferência, notou-se que houve um aumento significativo de consumo médio de potência estática. Devido a máxima taxa de transferência necessitar de uma alta performance, o que leva ao uso de células *Low-VT*, que consomem muita potência estática. Ocasionando assim, um aumento significativo no consumo de potência estática por *gates* equivalentes nas análises em questão.

Tabela 3.9 – Tabela comparativa de Potência Estática Média por Gate Equivalente a partir da síntese de 180 nm, 1,62 Volts e sem *clock gating*.

Performance	65 nm – 0,9 Volts	65 nm – 1,08 Volts
Isoperformance	Diminuição de 53%	Aumento de 0,9%
Máxima Taxa de Transferência	Aumento de 486%	Aumento de 787%

3.6.1.1.2. Potência Dinâmica

Na tabela 3.10, têm-se comparações de potência dinâmica por *gate* equivalente dos circuitos de 65 nm em relação à síntese de 180 nm, 1,62 Volts e sem *clock gating*, mostram uma redução acentuada no consumo de potência dinâmica nesta tecnologia mais nova. Esta diminuição ocorre tanto nas análises de *Isoperformance* quanto nas de Máxima Taxa de Transferência, mas são mais acentuadas nas análises de *Isoperformance*, pois as análises de Máxima Taxa de Transferência apresentam uma frequência maior que as de 180 nm, consumindo, assim, mais potência dinâmica.

Tabela 3.10 – Tabela comparativa de Potência Dinâmica Média por Gate Equivalente a partir da síntese de 180 nm, 1,62 Volts e sem clock gating.

Performance	65 nm – 0,9 Volts	65 nm – 1,08 Volts
Isoperformance	Diminuição de 94%	Diminuição de 91%
Máxima Taxa de Transferência	Diminuição de 93%	Diminuição de 88%

3.6.1.2Potência Média por Área

O cálculo da Potência Média por Área é feito se calculando a média das potências estáticas ou dinâmicas pela área ocupada por células.

3.6.1.2.1. Potência Estática

Na tabela 3.11, são mostrados dados comparativos de potência estática média por área entre a síntese em 180 nm, 1,62 Volts e sem *clock gating* e as sínteses em 65 nm. Nota-se um acréscimo muito grande no consumo de potência estática na tecnologia

mais reduzida (65nm) em *isoperformance*, e, muito mais, em máxima taxa de transferência que utiliza células *Low-VT*, que consomem mais potência estática que as outras células. Isto é devido aos transistores de 65nm serem menores e a sua corrente de sub-limiar por micrometro de largura de canal de transistor ser muito maior que na tecnologia 180nm. Além disto, há uma maior quantidade de transistores por área nas tecnologias reduzidas, o que aumenta até 80 vezes a potência estática consumida por unidade de área do circuito 65nm.

Tabela 3.11 – Tabela comparativa de Potência Estática Média por Área a partir da síntese de 180 nm, 1,62 Volts e sem *clock gating*.

Performance	65 nm – 0,9 Volts	65 nm – 1,08 Volts
Isoperformance	Aumento de 334,4 %	Aumento de 832 %
Máxima Taxa de Transferência	Aumento de 53X	Aumento de 80X

3.6.1.2.2. Potência Dinâmica

Na tabela 3.12, mostra-se uma diminuição significativa no consumo de potência dinâmica por área nas análises de 65 nm – 0,9 Volts. Mas há um aumento de 6% no consumo de potência dinâmica por área, em relação à síntese de 180 nm a 1,62V, para circuitos de 65nm que foram sintetizados para um desempenho máximo (Máxima Taxa de Transferência), e sem *clock gating.* Mesmo sendo 1,08V a alimentação na análise de 65 nm – para a máxima taxa de transferência. Isto ocorre porque a síntese é feita com o objetivo de obter-se a maior frequência possível, gerando um consumo de potência dinâmica maior, mesmo sendo a tensão de operação a 65nm menor do que na análise da tecnologia CMOS 180 nm.

Tabela 3.12 – Tabela comparativa de Potência Dinâmica Média por Área a partir da síntese de 180 nm, 1,62 Volts e sem *clock gating*.

Performance	65 nm – 0,9 Volts	65 nm – 1,08 Volts
Isoperformance	Diminuição de 44%	Diminuição de 19%
Máxima Taxa de Transferência	Diminuição de 40%	Aumento de quase 6%

3.6.2 Comparação de *Isoperformance* e Máxima Taxa de Transmissão com relação aos melhores casos deste trabalho e trabalhos de outros autores

As comparações foram feitas retirando-se os melhores casos deste trabalho em suas várias tecnologias e tensões de alimentação e se comparando com outros artigos publicados para assim verificar qual é o melhor dentre eles e o porquê de seus diferentes valores.

3.6.2.1Isoperformance

Conforme a tabela 3.13, o melhor caso deste trabalho em 180 nm teve uma dissipação de energia por operação 53 % maior que o trabalho de (DINIZ, 2010). Isto é devido à análise de potência deste trabalho ser mais acurada, pois apresenta o resultado depois do *post-route netlist*. Porém, o trabalho de (DINIZ, 2010), não faz uma estimativa de potência no nível de síntese física, isto é feito na *netlist* lógica pré posicionamento e roteamento.

Na comparação dos dados deste trabalho, que tem a mesma forma de análise, a melhor síntese de 180 nm e 1,62 Volts apresenta 8,57 X mais energia por operação do que a melhor síntese em 65 nm e 1,08 Volts, isto é devido em grande parte à diminuição do VDD, que gera uma diferença de 2,25X neste caso. Os outros fatores que geram uma diminuição no consumo de potência em 65 nm é a área ocupada por esta célula, que é muito menor, além da utilização de outros materiais e do processo desta tecnologia, o que pode diminuir a potência consumida.

A análise de 65 nm e 0,9 volts apresenta uma diminuição de quase 29% no consumo de potência em relação à análise de 65 nm e 1,08 volts. Se fosse comparada somente a variação da tensão desta análise em relação aquela, a variação seria de mais de 30%. Esta diferença maior da variação de tensão das análises do que das análises em si é devido a tensão de 1,08 volts fazer as células funcionarem mais rápido, sendo assim, usa-se células menores que consomem menos potência.

Tabela 3.13 – Tabela comparativa de *Isoperformance* entre os melhores resultados deste trabalho e artigos correlatos.

	(Diniz, 2010)	Este Trabalho		
Tecnologia CMOS	TSMC 180 nm	TSMC 180 nm	Synopsys 65 nm	Synopsys 65 nm
Tensão	1,62 Volts	1,62 Volts	0,9 Volts	1,08 Volts
Amostras de Entrada	8	16	16	16
Estágios de *Pipeline*	1	1	1	1
Clock Gating	Não	Não	Não	Não
Taxa de Transferência (MB/s)	$3{,}12x10^{6}$	$1{,}06x10^{6}$	$1{,}06x10^{6}$	$1{,}06x10^{6}$
Energia por Operação (Joule/MB)	$4{,}6x10^{-10}$	$1{,}02x10^{-9}$	$8{,}54x10^{-11}$	$1{,}19x10^{-10}$

3.6.2.2Máxima Taxa de Transferência

Na tabela 3.14, vê-se alguns trabalhos de outros autores e as melhores análises deste trabalho em diferentes tecnologias e tensões de alimentação. Estes têm as melhores

análises no máximo de paralelismo e no máximo de estágios de *pipeline*, isto porque com o dobro de paralelismo se dobra a taxa de transferência e, com mais estágios de *pipeline*, aumenta-se a frequência do sistema, aumentando, assim, a taxa de transferência.

Esta análise em 180 nm e 1,62 volts apresenta uma taxa de transferência menor em 4%, se comparada ao trabalho de (DINIZ, 2010). Sendo as arquiteturas as mesmas, a diferença é devido ao trabalho de (DINIZ, 2010) fazer a análise usando *Mentor Graphics Leonardo Spectrum* e as análises deste trabalho serem sintetizadas com as ferramentas *Cadence*(para síntese lógica e física).

A análise de (VANNE, 2006) tem uma queda de quase 80% na taxa de transferência se comparada com a análise de 180 nm e 1,62 volts deste trabalho. Isto é devido ao trabalho de (VANNE, 2006) utilizar menos estágios de *pipeline* e um menor paralelismo.

Entre as comparações deste trabalho, a tecnologia de 65 nm e 0,9 volts apresenta um aumento em 31% na taxa de transferência se comparada a de 180 nm e 1,62 volts deste mesmo trabalho. Isto mostra que a tecnologia, mesmo em tensão mais baixa, apresenta uma velocidade de células superior, mesmo assim, consumindo bem menos energia por operação.

A comparação entre as tecnologias de 65 nm mostra que se aumentando a tensão se aumenta a taxa de transferência em um pouco mais de 6% nas análises de 1,08 volts em comparação com as de 0,9 volts.

Tabela 3.14 – Tabela comparativa de Máxima Taxa de Transferência entre os melhores resultados deste trabalho e artigos correlatos.

	(Diniz, 2010)	(Vanne, 2006)	Este Trabalho		
Tecnologia CMOS	TSMC 180 nm	TSMC 180 nm	TSMC 180 nm	Synopsys 65 nm	Synopsys 65 nm
Tensão	1,62 Volts	1,62 Volts	1,62 Volts	0,9 Volts	1,08 Volts
Amostras de Entrada	16	2	16	16	16
Estágios de *Pipeline*	6	3	6	6	6
Clock Gating	Não	Não	Não	Não	Não
Taxa de Transferência (MB/s)	$3{,}1x10^{7}$	$6x10^{6}$	$2{,}975x10^{7}$	$3{,}906x10^{7}$	$4{,}168x10^{7}$
Energia por Operação (Joule/MB)	$1{,}9x10^{-9}$	---	$5{,}11x10^{-9}$	$1{,}769x10^{-10}$	$3{,}219x10^{-10}$

4 DECODIFICADOR *INTRA-ONLY* EM CMOS 65 NM

O decodificador *Intra-Only* foi primeiramente projetado para FPGA por (PEREIRA, 2008), utilizando alguns módulos já prontos feitos por (STAEHLER, 2006) e (AGOSTINI, 2007) dos quais se modificou minimamente para se encaixar na estrutura projetada por (PEREIRA, 2008).

O passo seguinte a este foi a sua implementação em ASIC 180 nm por (SILVA, 2010), do qual se necessitou algumas mudanças também para se encaixar no padrão ASIC e dos blocos de RAM utilizados.

Sendo assim, este tópico da dissertação versa sobre a implementação deste decodificador *Intra-Only* utilizando técnicas de baixo consumo na biblioteca de 65 nm da Synopsys, que foi escolhida por apresentar técnicas de baixo consumo, como *Multi-Threshold*.

4.1 Arquitetura do Decodificador Intra-Only

A arquitetura em questão já foi apresentada por (PEREIRA, 2008) e (SILVA, 2010), estrutura tal que suporta decodificação de entropia com o uso de CAVLC (*Context-Based Adaptative Binary Arithmetic Coding*), quantização inversa, transformada inversa e algoritmos de predição-intra.

O diagrama de blocos do decodificador *Intra-Only* é mostrado na figura 4.1.Este decodificador em questão tem o nome de *Intra-Only* por só apresentar a Predição Intra e não conter em suas descrições de *hardware* a Predição Inter, que é necessária para um decodificador completo.

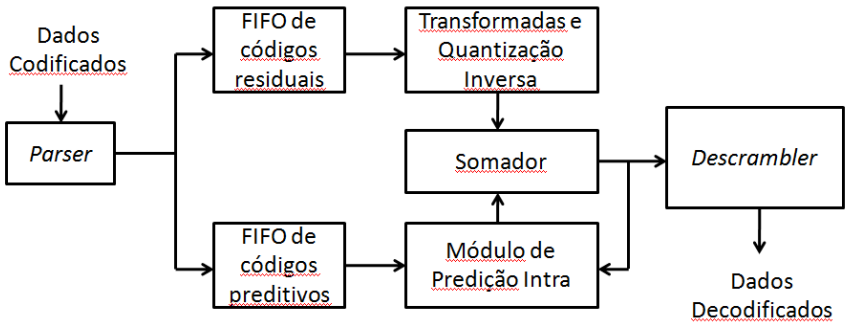

Figura 4.1 : Diagrama de Blocos do Decodificador *Intra-Only*.

O *Parser*, o primeiro módulo, é o módulo que recebe os dados codificados e gera informação predita e residual para os próximos módulos. O processo de decodificação feito pelo *Parser* é desenvolvido em 4 fases, mostrada na figura 4.2, sendo elas: os parâmetros de inicialização, a inicialização de contexto, a decodificação dos macro-blocos e o encerramento do *slice*.

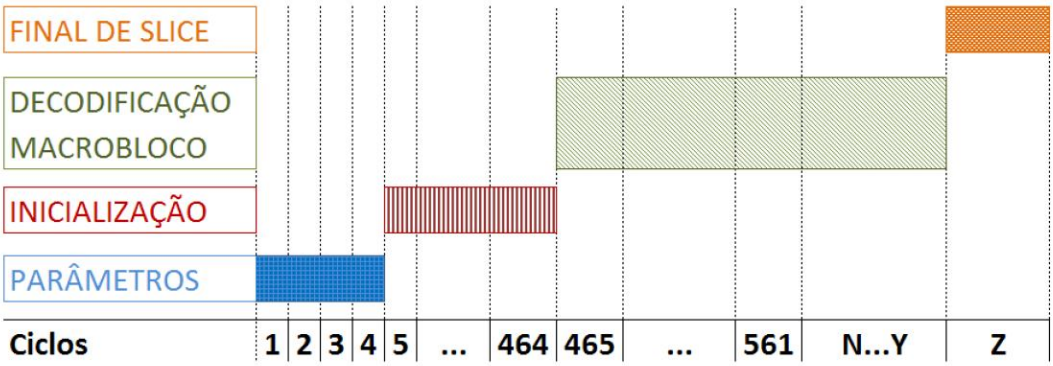

Figura 4.2 : Fases do *Parser* que determinam a ordem de execução do decodificador *Intra-Only* (DEPRÁ, 2009).

Estas fases desenvolvidas pelo *Parser* determinam a ordem de execução do decodificador *Intra-Only*. Podendo variar o número de macro-blocos processados por um determinado tempo de acordo com o padrão utilizado no H.264/AVC.

Na fase de inicialização de contexto, a memória de contexto, que pode ser fixa ou adaptativa, é inicializada com os valores gerados no parâmetro de inicialização.

Na fase de decodificação dos macro-blocos, são processados os elementos sintáticos de cada um dos macro-blocos presentes no *slice*. E, finalizando, o encerramento do *slice*, onde se esvazia o *buffer* de *bitstream* e se elimina os *bits* excedentes. (DEPRÁ, 2009)

As transformadas e quantizações inversas, que foram primeiramente desenvolvidas por (AGOSTINI, 2007), utilizam as informações residuais e aplicam a quantização inversa e transformada inversa sobre esta matriz de resíduos.

Sendo a transformada inversa (T^{-1}) separada em duas partes, a primeira envolve as transformadas 2x2 e 4x4 inversas, que são aplicadas sobre os resultados da quantização direta, e com seus resultados passados para a quantização inversa(Q^{-1}). E os resultados da quantização inversa (Q^{-1}) são levados novamente as transformadas inversas(T^{-1}) para aplicação da Transfornada de Cosseno Discreta 2-D inversa (DCT 2-D inversa).

Estas transformadas não serão explicadas neste breve resumo de seu funcionamento, mas estão referenciadas em (AGOSTINI, 2007).

E o módulo de Predição Intra, que para explicá-lo terá de se voltar a alguns padrões do H.264. Em que o espaço de amostragem utilizado é o YCbCr, onde Y é a informação de luminância e Cb e Cr são componentes de crominância. Tendo como padrão *Main* no H.264 o espaço 4:2:0, Figura 4.3, que gera uma amostra de crominância para quatro de luminância, sendo a (O) referente a crominância e (X), a luminância. Este padrão é o utilizado, pois a nossa visão é mais sensível ao brilho do que a cor, sendo assim,

necessita-se mais de amostras de luminância do que de crominância. (STAEHLER, 2006).

X	X	X	X	X	X
O		O		O	
X	X	X	X	X	X
X	X	X	X	X	X
O		O		O	
X	X	X	X	X	X
X	X	X	X	X	X
O		O		O	
X	X	X	X	X	X

Figura 4.3: Espaço de Subamostragem de Cor 4:2:0.

Assim, de acordo com este padrão, são calculadas as predições de amostras de luminância em blocos 4x4 e 16x16 e as predições das amostras de crominância, tendo-se assim a unidade GP, Gerador de Predição, inserida na figura 4.4. Esta unidade precisa de uma referência para comparações com as predições futuras, e isto é feito com o Buffer de Amostras Vizinhas (BAV). Além destes, existe o Módulo de Elementos Sintáticos (DES), que fornecem informações para o sistema, como o modo de predição escolhido (STAEHLER, 2006).

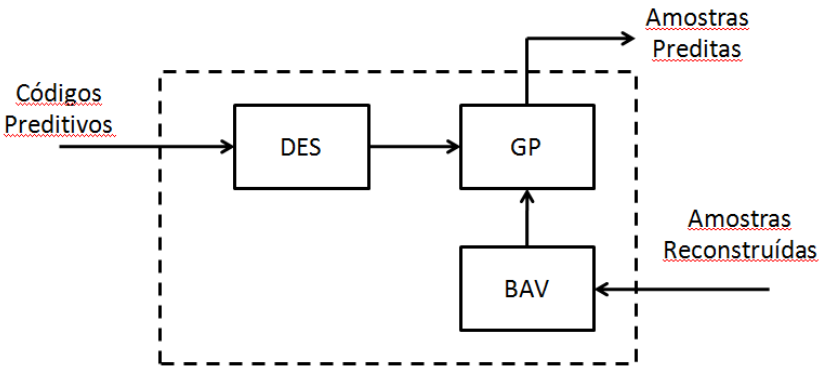

Figura 4.4 : Módulo de Predição Intra.

4.2 Métodos de Análise

O fluxo da síntese do decodificador *Intra-Only* é mostrado na figura 4.5 e tem como primeiro estágio a modificação das memórias utilizadas por (SILVA, 2010) pelas geradas no gerador de memória da tecnologia de 65 nm. Depois de inseridas as novas memórias, fizeram-se a compilação e, logo após, a simulação que em sua verificação é gerado um arquivo de saída dos dados da descrição de *hardware*. Tendo o arquivo prévio dos dados corretos de saída e o novo arquivo de saída com as novas memórias, compararam-se estes dois arquivos e se tratando de arquivos com dados iguais, significa que está passando na verificação.

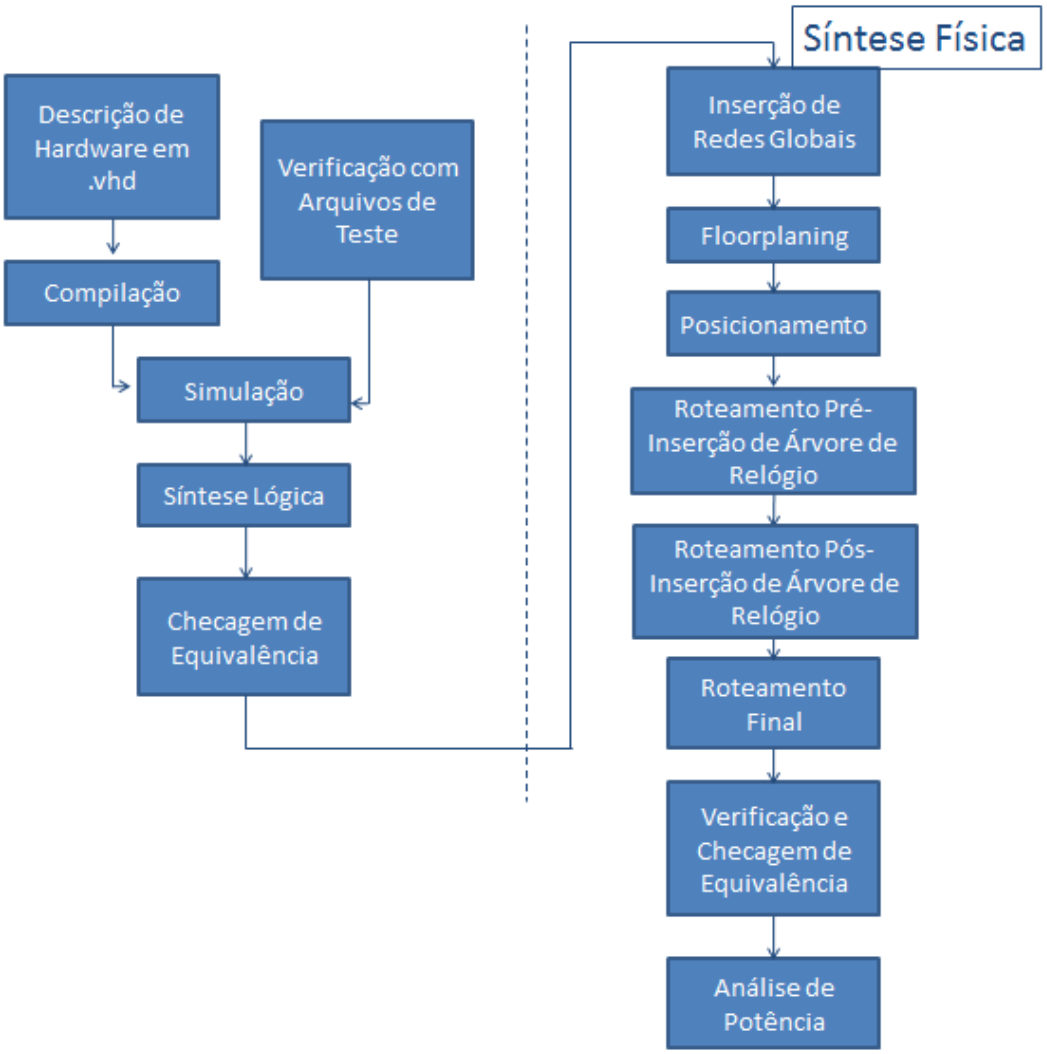

Figura 4.5: Fluxo de Síntese no decodificador *Intra-Only*.

Posteriormente, incluiu-se como topo da descrição de hardware um arquivo *verilog* com os *pads* já instanciados. Fez-se a síntese lógica, com a inclusão de células *multi-threshold*, estágio em que se determina se o circuito terá *clock gating* ou não. Posteriormente, a checagem de equivalência e se passou para a síntese física. Nesta etapa, não foi possível passar nas violações de tempo e o erro constatado era no sinal de *reset* da descrição de *hardware*. Sendo assim, fez-se um teste substituindo o *reset* assíncrono pelo síncrono, constando do apêndice C um exemplo de modificação de assíncrono para síncrono. Feita a alteração, novamente fez-se a compilação, simulação, síntese lógica, checagem de equivalência e, a seguir, a síntese física. Assim, cessaram as violações de tempo.

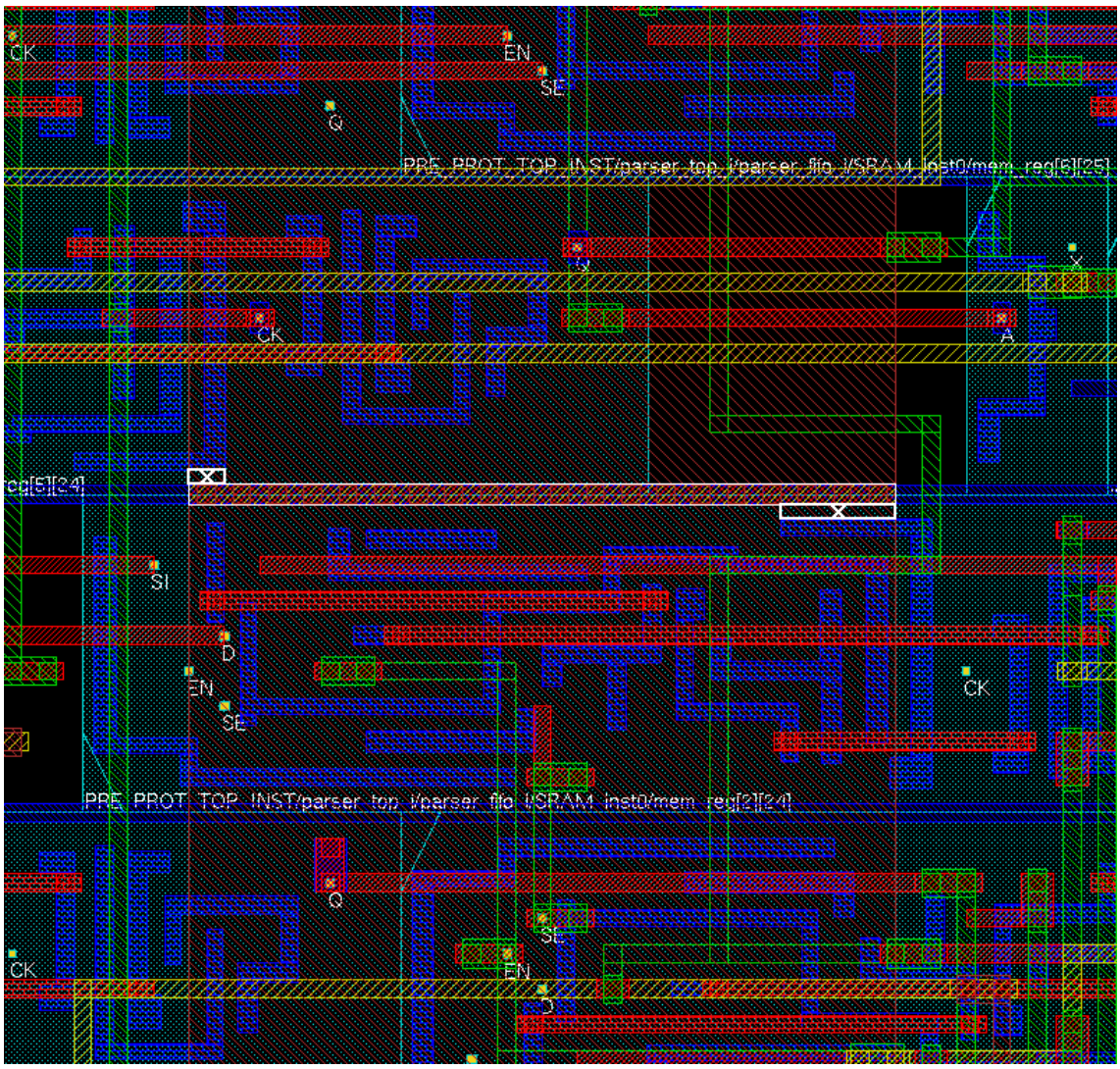

Figura 4.6: Violações de espaçamento com linhas de potência em Metal 1.

No próximo passo, carregou-se um arquivo de *pads* de entrada/saída (I/O), que informa o local correto dos *pads* no circuito. Optamos por inserir os pads na síntese lógica, e não após. Tal arquivo de *pads* consta do apêndice D. A seguir, inseriu-se os anéis de potência em volta das memórias RAMs e do core do circuito com o *ADD Rings* e as linhas de potência com o *SRoute* em Metal 1 apresentando violações de

espaçamento depois do posicionamento das células padrão, mostrada tais violações na figura 4.6. Estas violações foram resolvidas colocando as linhas de potência em Metal 2.

Também, inseriu-se, antes do posicionamento das células, um bloqueio de células padrão na área ao redor dos blocos de RAM, para não ter o perigo de inserir uma célula onde não há linhas de potência, mostrada a violação que ocorre quando não se faz este bloqueio de células padrão na Figura 4.7.

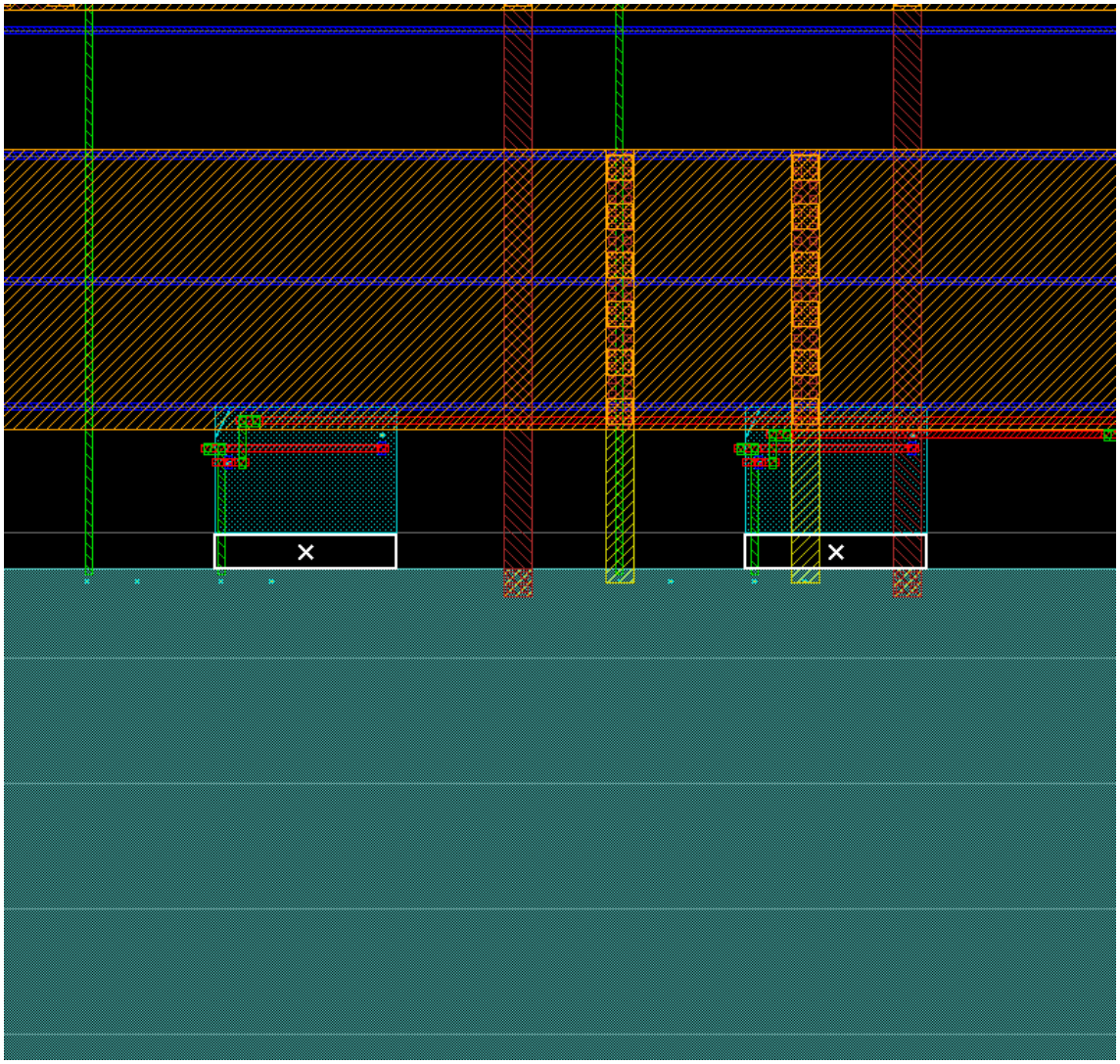

Figura 4.7: Violações pela falta de linhas de potência em Metal 1 próximo aos blocos de memória.

Logo após, inseriu-se os *Stripes* sendo o VDD e o GND separados de 50 um, este tipo de separação é definida como padrão da biblioteca de 65 nm, e não passando por dentro dos blocos de memória RAM. Finalizando o passo anterior ao posicionamento das células padrão, corrigiram-se pequenos defeitos de geometria, como os de mínimo espaçamento, mostrados na Figura 4.8.

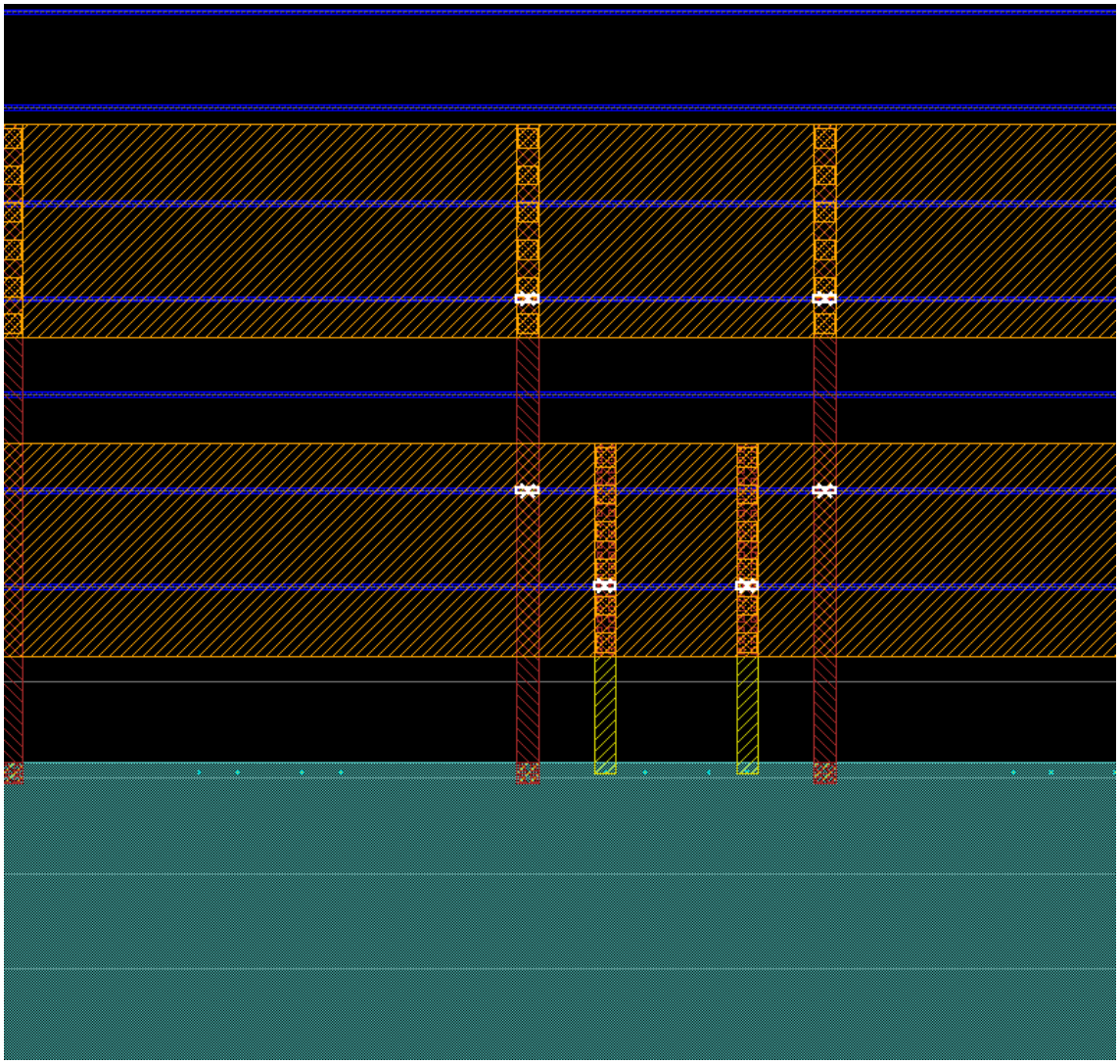

Figura 4.8: Violações de Geometria.

Assim, teve-se o término da etapa o *floorplan*, que é a estruturação dos locais dos *PAD's*, blocos dedicados e sinais de potência, como mostrado na figura 4.9.

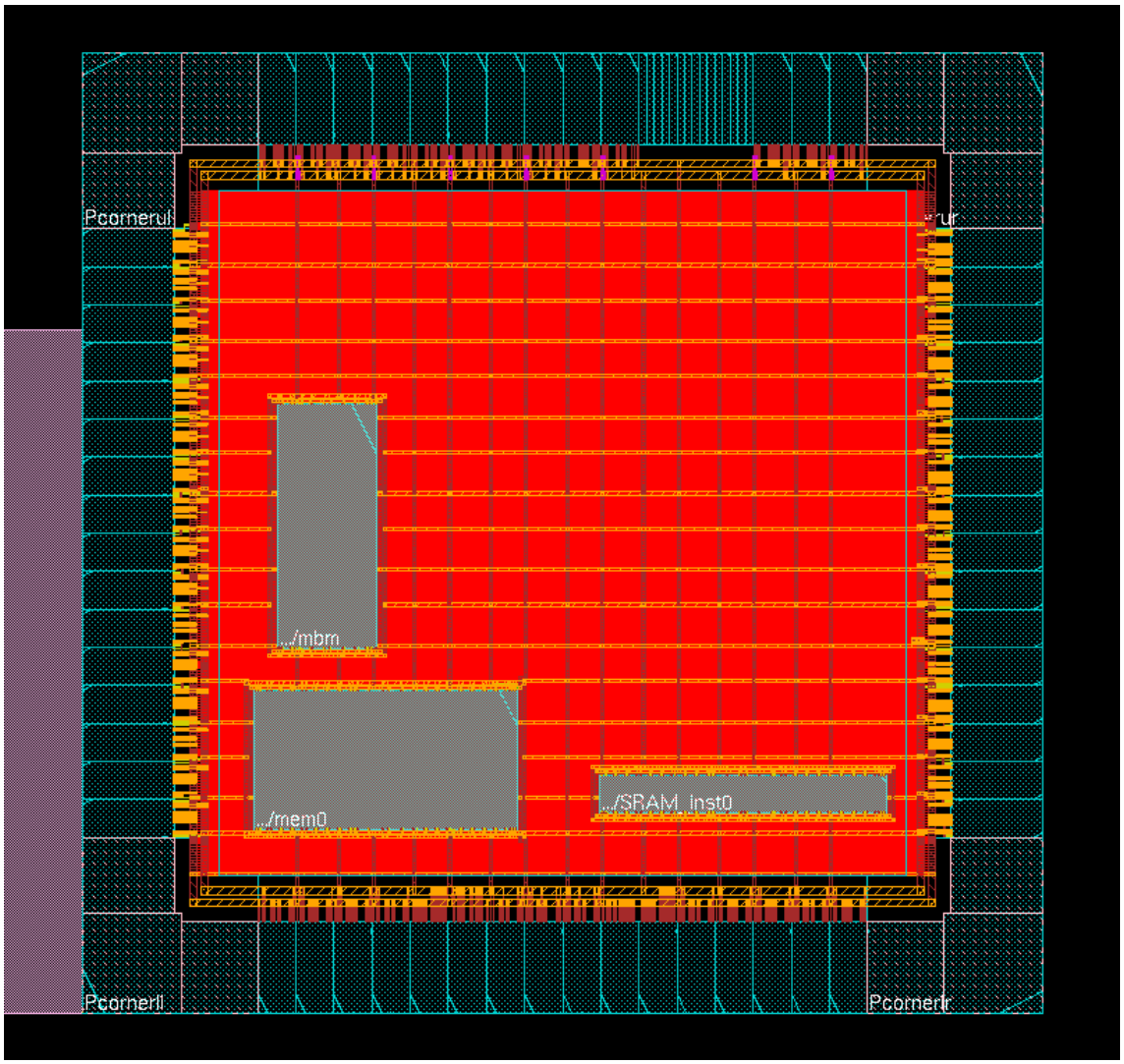

Figura 4.9: Design com *floorplan* completo do decodificador *Intra-Only*.

A partir do *floorplan*, fez-se o posicionamento das células padrão, o roteamento anterior à inserção da árvore de relógio, o roteamento após a inserção da árvore de relógio, resultando como mostrado na figura 4.10 após a inserção da árvore de relógio. As linhas brancas mostram os caminhos percorridos por estes sinais de relógio do Decodificador *Intra-Only*. Na figura 4.10 está mostrado o roteamento final com a ferramenta *Nanoroute*. As verificações de timing, com a ferramenta de STA, foram aplicadas pós-roteamento final, verificando-se o correto timing, sem violações de tempos de *setup* e *hold*, na frequência de operação de 50 MHz. O design passou pela verificação de timing e correção em todas as etapas. Logo após, como verificação final, fez-se a checagem final de equivalência e a verificação de possíveis violações de geometria, de regras de antena e de regras de conectividade.

E, por último, fez-se a análise de potência, gerando-se um arquivo .tcf, lendo este arquivo e fazendo-se sua análise de potência.

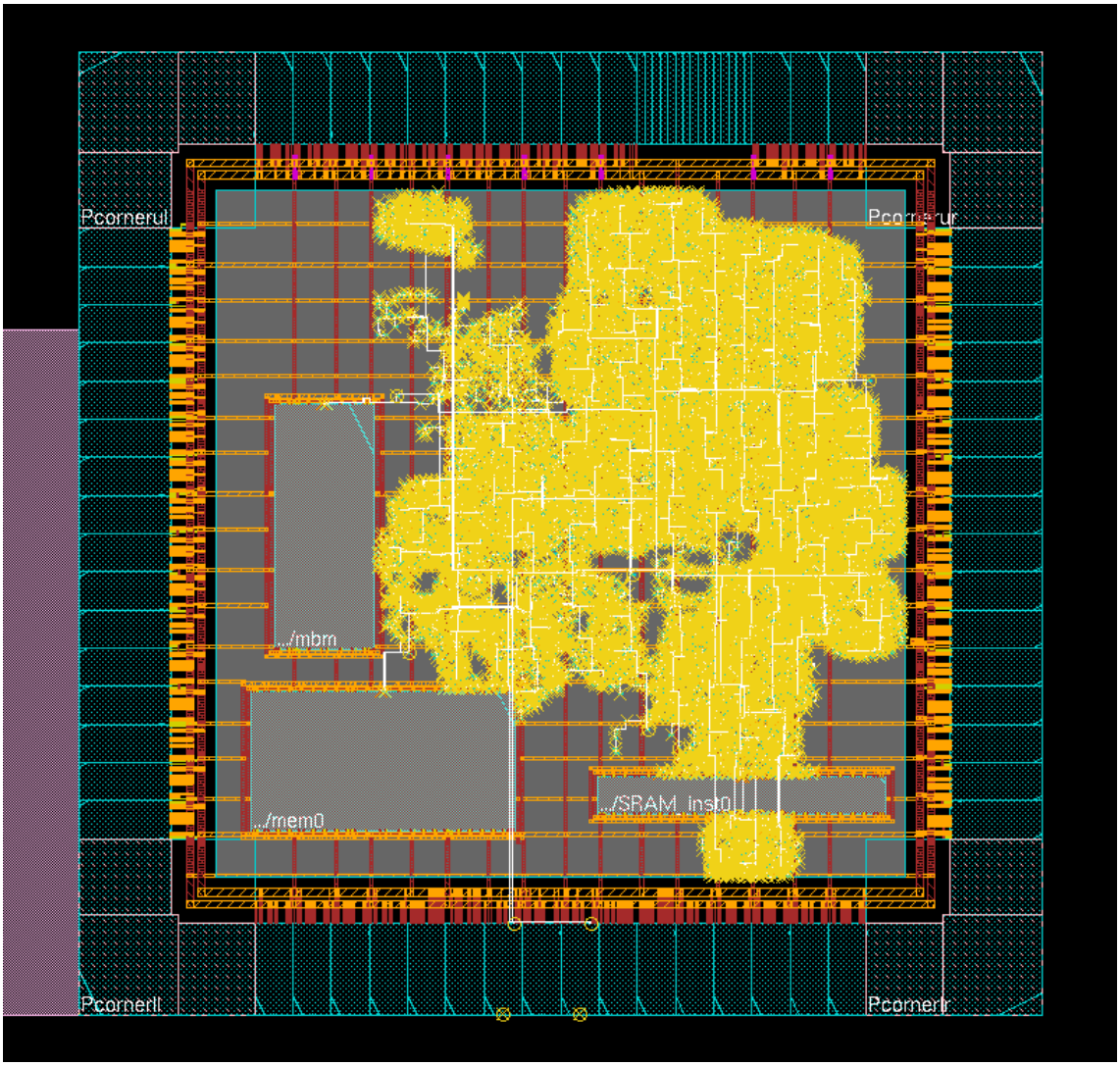

Figura 4.10: *Design* com das árvores de relógio do decodificador *Intra-Only*.

4.3 Resultados e Análise dos Resultados de Síntese

Os resultados de potência deste trabalho foram feitos para análises sem *clock gating* e com *clock gating*, na tecnologia 65 nm, com tensão padrão de 1,2 volts tanto para os blocos de RAM, quanto para as células padrão. Isto porque o bloco de RAM é gerado por ferramenta automática de SRAM da Synopsys Inc., e estes blocos são caracterizados para operar somente a 1,2 volts e não se teve a ferramenta apropriada para se fazer Multi-VDD, i.e. 1,2 V e 1,0 V, e tendo ambos os relógios do circuito funcionando a 50 MHz.

4.3.1 Comparação entre as análises somente deste trabalho

Na comparação entre os circuitos deste trabalho, resultados mostrados na Tabela 4.1, para a mesma frequência e alimentado por 1,2 volts, se diferenciam pela existência ou não do *clock gating*, o circuito com *clock gating* apresentando um maior consumo de potência estática em 23% se comparado com o circuito sem *clock gating*. Isto é devido ao *clock gating* acrescentar células de controle ao sistema gerando assim uma maior área e consumo de potência estática.

Porém, no caso da potência dinâmica, o circuito com *clock gating* tem uma economia de potência dinâmica da ordem de 7,6% se comparado ao circuito sem *clock gating*, o que representa uma economia de potência de 2,27 mW. Este resultado mostra que o *clock gating* é uma boa técnica de redução de potência para este circuito do Decodificador *Intra-Only* acarretando um aumento na potência estática, mas uma diminuição muito maior no consumo de potência dinâmica, pois o circuito incumbido de fazer o desligamento do relógio consome menos potência dinâmica do que a diminuição que ele gera devido ao não chaveamento do relógio,.

Tabela 4.1 – Tabela comparativa das análises de potência somente deste trabalho.

Único VDD	*Clock Gating*	Potência Estática	Potência de Chaveamento	Potência Interna	Potência Total
1,2 Volts	Não	96,78 uW	7,157 mW	22,36 mW	29,61 mW
1,2 Volts	Sim	119,5 uW	6,126 mW	21,09 mW	27,34 mW

4.3.2 Comparação com as Análises de Outros Trabalhos

Nas comparações com outros decodificadores de H.264, tabela 4.2, este trabalho apresentava uma nova tecnologia de 65 nm que, consequentemente, acarretou em uma menor *core area*, além disso, apresentava uma menor tensão de 1,2 volts em comparação com 1,8 volts da tecnologia de 180 nm, todos estes fatores deveriam levar a um menor consumo de potência.

Na comparação do trabalho de (CHEN, 2006) com o melhor caso deste trabalho, o circuito de (CHEN, 2006) consome 5,8X mais potência que este trabalho. Isto é devido aquele utilizar uma frequência 2,4X maior, o que aumenta quase na mesma proporção o consumo de potência. Outro fator é a diminuição da tensão de alimentação, que leva a uma diminuição de 2,25X na potência do circuito. Além disso, acrescenta-se que este trabalho não apresenta o módulo de Predição Inter, o que diminui ainda mais o consumo de potência.

No outro artigo apresentado por (LIN, 2005), também, tem-se as mesmas razões para o seu consumo de potência maior, da ordem de 10,7X. Estas razões são a frequência e a tensão, além da qual se pode adicionar a *core area* que é muito maior em (LIN, 2005) e a arquitetura apresenta também a Predição Inter.

Na última comparação com os trabalhos desenvolvidos por outros autores, (SILVA, 2010) fez a síntese do decodificador *Intra-Only*, com a mesma descrição de *hardware* de (PEREIRA, 2008), que este decodificador *Intra-Only* desenvolvido neste trabalho. Os seus resultados de consumo de potência são 2,52X maiores do que o deste trabalho sem *clock gating*. Já, na comparação deste trabalho com *clock gating*, esta diferença ficou em 2,72X.

Estas diminuições do consumo de potência deste trabalho são devidas em 2,25X a tensão de alimentação e o resto a menor área utilizada e a utilização de células *multi-threshold*.

Tabela 4.2 – Tabela comparativa das análises de potência deste trabalho com outros trabalhos.

	(CHEN, 2006)	(LIN, 2005)	Intra-Only*	Intra-Only **	
Formato de Vídeo	2Kx1K 30 fps	1080p HD 30 fps	QCIF (176x144) 30 fps	QCIF (176x144) 30 fps	
Tecnologia	180 nm 1.8 volts	180 nm 1.8 volts	180 nm 1.8 volts	65 nm 1.2 volts	
Core Area	2,19x2,19	2,9x2,9	2,13x2,13	0,9x0,9	
Gates Count	217 K	160 K	150 K	379 K	323 K
Memória Interna	9,98 KB	4,5 KB	5 KB	5 KB	
Frequência	120 MHz	120 MHz	50 MHz	50 MHz	
Clock Gating	N. I.	N. I.	Sim	Não	Sim
Potência Total	186,4 mW	320 mW	74,6 mW	29,61 mW	27,34 mW

N. I. – Não Informado

* Síntese de (SILVA, 2010), Descrição de *Hardware* de (PEREIRA, 2008)

** Síntese deste trabalho, Descrição de *Hardware* de (PEREIRA, 2008)

5 CONCLUSÃO

As respostas dos dados analisados no núcleo de processamento aritmético são bastante importantes para o estudo das técnicas de baixo consumo. Se destacando a potência estática por *gates* equivalentes foi reduzida e otimizada bastante em 65 nm. Em baixo VDD, conseguiu-se reduzir drasticamente a potência estática, da ordem de 53% comparando-se as médias desta tecnologia 65 nm com a de 180 nm. Fato que não é muito comum, mas pelo circuito em *isoperformance* apresentar a característica de utilizar uma baixa frequência, fez-se o ambiente para o uso de várias células *High-VT*.

Sendo assim, um ambiente que permite o uso de células *High-VT* em tecnologias menores e menor tensão de alimentação é melhor em termos de potência estática do que em tecnologias maiores como a CMOS 180 nm.

Outro ponto a se destacar, é a diminuição da potência dinâmica por *gates* equivalentes nas novas tecnologias até mesmo quando elas apresentam uma frequência muito maior em seus circuitos, isto é, mesmo tendo circuitos muito mais rápidos.

Nas comparações de *isoperformance*, se destaca que as novas tecnologias diminuem sensivelmente a energia por operação, principalmente, quando utilizadas tensões mais baixas. Tendo a nova tecnologia, como fatores de diminuição do consumo de potência: *multi-threshold*, área menor para a mesma função e diferentes materiais e técnicas de fabricação, são responsáveis por mais de 73% da diminuição de energia por operação. Os outros fatores responsáveis pela diminuição da energia por operação é a diminuição da tensão de alimentação, o menor número de estágios de *pipeline* e o maior paralelismo combinado com a diminuição da frequência.

Nas comparações de máxima taxa de transferência, as novas tecnologias têm uma maior máxima taxa de transferência, mesmo em tensões mais baixas, de 0,9 volts. Com tensões mais altas, de 1,1 volts, e taxas de transferência muito maiores, a tecnologia de 65 nm tem um consumo, mesmo assim, de energia por operação muito menor, da ordem de 10 vezes em comparação com 180 nm. Outros fatores arquiteturais para uma maior taxa de transferência são o aumento do paralelismo e do número de estágios de *pipeline*.

Nas comparações com o decodificador *Intra-Only*, mostra-se, em relação a (CHEN, 2006) e (LIN, 2005), os benefícios de utilizar uma nova tecnologia com a técnica de *multi-threshold* e, também, de utilizar *clock gating*, o que ocasionou uma diminuição significativa no consumo de potência em relação a estes artigos citados.

Além disso, um fator a se destacar é que com *clock gating*, nas análises de SAD, a energia por operação do sistema deveria diminuir, um fato que não ocorre devido à alta atividade de chaveamento dos caminhos, o que não recomenda o uso de *clock gating*. Fator este que não ocorre no decodificador *Intra-Only*, onde o *clock gating* diminui o consumo do sistema em questão.

As comparações realizadas neste trabalho com outros em 180 nm demonstraram a importância da utilização criteriosa de todas as alternativas de design *low-power*, seja no nível arquitetural, seja no nível lógico e elétrico, como a escolha dos *VT's* das células lógicas.

Futuros trabalhos recomendados para o decodificador *Intra-Only* a partir desta dissertação são os seguintes:

- Na descrição de hardware, seria necessária a inserção do bloco de predição Inter para o decodificador H.264/AVC ficar completo, paralelo a isto, se adicionaria as modificações necessárias para quando fizer um design em *Full-HD*, que é o padrão utilizado no Sistema Brasileiro de Televisão Digital (ITU-T, 2010);
- Na parte de teste lógico, é necessário um teste lógico que contemple as novas funcionalidades;
- Na parte de teste físico a melhor escolha é a inserção de BIST, pelo baixo custo, e, a seguir, a inserção das *scan-chains* na síntese lógica;
- Ainda na síntese lógica, seria de grande valia a implementação de Multi-VDD para a redução ainda maior na potência consumida;
- Na síntese física, seria necessários arquivos das tecnologias para gerar os testes de integridade de sinal e teste de violação de tempo no *Sign-Off*, além disso a inserção de *Metal-Fill* para a extração do GDSII e futura fabricação do circuito integrado.

Já para a arquitetura de SAD, têm-se como recomendação os seguintes pontos:

- Utilização de somadores compressores para a síntese em Células Padrão com as ferramentas de síntese digital;
- Utilização de somadores compressores para a síntese em *Full-Custom* com transistores de passagem.

REFERÊNCIAS

AGOSTINI, L. **Desenvolvimento de Arquiteturas de Alta Performance Dedicadas à Compressão de Vídeo Segundo o Padrão H.264.** 2007. 173f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

BD03 – Digital Physical Design. Cadence Student Lecture Material. 2008.

BENINI, L. ; MICHELI, G. DE ; MACII, E. ; PONCINO M. ; SCARSI, R. . Symbolic Synthesis of Clock-Gating Logic for Power Optimization of Synchronous Controllers. In: ACM TRANSACTIONS ON DESIGN AUTOMATION OF ELECTRONIC SYSTEMS. 10, 1999. **Proceedings...** [S.I.: s.n.] 1999.

DEPRÁ, D. A. **Algoritmos e Desenvolvimento de Arquitetura para a Codificação Binária Adaptativa ao Contexto para o Decodificador H.264/AVC.** 2009. 153f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

CHEN, T.; LIAN, C.; CHEN, L. Hardware Architecture Design of an H.264/AVC Video Codec. In: ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE. 2006. **Procedings…** [S.I.: s.n.] : 2006.

DINIZ, C. ; CORRÊA, G.; SUSIN, A. and BAMPI S. Comparative Analysis of Parallel SAD Calculation Hardware Architectures for H.264/AVC Video Coding. In: IEEE LASCAS, 2010. **Proceedings…** [S.I.: s.n.]: 2010.

ITU-T Recommendation H.264/AVC (03/10): **advanced video coding for generic audiovisual services**, 2010.

LIN, C. et al. A Memory_Efficient Deblocking Filter for H.264/AVC Video Coding. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUIT AND SYSTEMS, 2005. **Proceedings…** [S.I.: s.n.]: 2005.

MOHANTY, S. P.;VADLAMUDI, S. T.; KOUGIANOS, E. A universal voltage level converter for multi-Vdd based low-power nano-CMOS systems-on-chips(SoCs). In: NASA SYMPOSIUM ON VLSI DESIGN. 2.2, 13, 2007. **Proceedings…** [S.I.: s.n.]: 2007.

MÜNCH, M.; WURTH, B.; MEHRA, R.; SPROCH, J.; WEHN, N. Automating RT-Level Operand Isolation to Minimize Power Consumption in Datapaths. In: IEEE DESIGN AUTOMATION AND TEST IN EUROPE. 03, 2000. **Proceedings…** [S.I.: s.n.]: 2000.

PEREIRA, F. **Hardware Implementation of a High Performance Minimalist H.264 Video Decoder.** 2008. 58f. Master’s thesis – Information Technology / Multimedia Communications, Helsink Metropolia University of Applied Sciences, Helsinki.

PORTO, E. C. **Desenvolvimento Arquitetural para Estimação de Movimento de Blocos de Tamanhos Variáveis Segundo o Padrão H.264/AVC de Compressão de Vídeo Digital.** 2008. 96 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

RADHA KRISHNAN, D.; PREETHY, A. P. Low Power CMOS Pass Logic 4-2 Compressor for High-Speed Multiplication. In: THE 23RD IEEE MIDWEST SYMP. ON CIRCUITS AND SYSTEMS. 08, 2000. **Proceedings...** [S.I.: s.n.] 2000.

SILVA, L. M. L. **Implementação Física de Arquiteturas de Hardware para a Decodificação de Vídeo Digital Segundo o Padrão H.264/AVC.** 2010. 136 f. Dissertação (Mestrado em Ciências da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SPEAR, C. **System Verilog for Verification: A Guide to Learning the Testbench Language Features.** 2nd ed. USA: Springer, 2008.

STAEHLER, W. T. **Projeto de Sistemas Digitais Complexos: uma Aplicação ao Decodificador H.264.** 2006. 108f. Dissertação (Mestrado em Ciências da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SYNOPSYS. **Expanding the Synopsys PrimeTime® Solution with Power Analysis.** 7 f. June, 2006.

VANNE, J. et al. A High-Performance Sum of Absolute Difference Implementation for Motion Estimation. In: IEEE TCSVT. 07, 2006. **Proceedings…** [S.I.: s.n.] 2006.

VEERA MACHANENI, S. et al. Novel Architectures for High-Speed and Low-Power 3-2, 4-2 and 5-2 Compressors. In: THE 20TH IEEE INTERNATIONAL CONFERENCE ON VLSI DESIGN, 2007. **Proceedings…** [S.I.: s.n.] 2007.

VIRAGE LOGIC. **SiWare™ High-Density Tapless Standard Cell Logic Library for Common Platform 65nm LPe LowK Standard Vt Process.** 50 f. March, 2009.

APÊNDICE A < ARQUIVOS EM SYSTEM VERILOG PARA VERIFICAÇÃO FUNCIONAL >

```
// Módulo topo

module top;

    bit clk;
    bit rst;
    always #5 clk = ~clk;

    initial
    begin
      rst = 1'b1;
      #1
      rst = 1'b0;
      #1
      rst = 1'b1;
    end

    arb_if arbif(clk,rst);
    arb a1(arbif);
    test t1(arbif);
    monitor m1(arbif);

endmodule : top

// Módulo que liga os blocos da verificação Funcional
interface arb_if(input bit clk, rst );
    logic [7:0] pred_0, pred_1, pred_2, pred_3;
    logic [7:0] orig_0, orig_1, orig_2, orig_3;
    logic first_line, scan_in, scan_en;
    logic scan_out;
    logic [12:0] SAD;

    modport TESTE ( output pred_0, pred_1, pred_2, pred_3, orig_0,
orig_1, orig_2, orig_3, first_line, scan_in, scan_en,
    input clk, rst, SAD );

    modport DUT ( input clk, rst, pred_0, pred_1, pred_2, pred_3,
orig_0, orig_1, orig_2, orig_3, first_line, scan_in, scan_en,
    output scan_out, SAD);

    modport MONITOR ( input clk, rst, pred_0, pred_1, pred_2, pred_3,
orig_0, orig_1, orig_2, orig_3, first_line, scan_in, scan_en,
scan_out, SAD);

endinterface

// Módulo que liga a Verificação Funcional com o Design em Verilog ou
// .vhd

module arb (arb_if.DUT arbif);

   SAD4i_Comp42 m1(
            .clk(arbif.clk),
```

```
            .rst(arbif.rst),
            .pred_0(arbif.pred_0),
            .pred_1(arbif.pred_1),
            .pred_2(arbif.pred_2),
            .pred_3(arbif.pred_3),
            .orig_0(arbif.orig_0),
            .orig_1(arbif.orig_1),
            .orig_2(arbif.orig_2),
            .orig_3(arbif.orig_3),
            .first_line(arbif.first_line),
            .scan_in(1'b0),
            .scan_en(1'b0),
            .scan_out(arbif.scan_out),
            .SAD(arbif.SAD)

);
endmodule

// Módulo com as entradas randômicas de teste
module test (arb_if.TESTE arbif);

    class Packet;
    rand bit [7:0]  rand_pred_0, rand_pred_1, rand_pred_2,
rand_pred_3;
    rand bit [7:0]  rand_orig_0, rand_orig_1, rand_orig_2,
rand_orig_3;
    endclass

    int i;
    logic  [1:0]  count;
    Packet p;

    initial
    begin
      #4
      p = new();
      for(i=0; i<900; i++)
      begin
      assert( p.randomize() );
        arbif.pred_0 = p.rand_pred_0;
        arbif.pred_1 = p.rand_pred_1;
        arbif.pred_2 = p.rand_pred_2;
        arbif.pred_3 = p.rand_pred_3;
        arbif.orig_0 = p.rand_orig_0;
        arbif.orig_1 = p.rand_orig_1;
        arbif.orig_2 = p.rand_orig_2;
        arbif.orig_3 = p.rand_orig_3;
        #10;
      end
    end



    always_ff @(posedge arbif.clk or negedge arbif.rst)
      if (!arbif.rst)
        count = 2'b00;
      else
      begin
```

```
        count = count + 1;
        if ( !count )
          arbif.first_line = 1'b1;
        else
          arbif.first_line = 1'b0;
      end

endmodule

// Módulo Monitor e checa a funcionalidade do Verilog ou .vhd
module monitor (arb_if.MONITOR arbif);

      logic [8:0] rpred_0, rpred_1, rpred_2, rpred_3;
      logic [8:0] rorig_0, rorig_1, rorig_2, rorig_3;
      logic fline;
      logic [8:0] sub_0, sub_1, sub_2, sub_3;
      logic [8:0] abs0, abs1, abs2, abs3;
      logic [9:0] tree_0_1, tree_2_3;
      logic [10:0] tree_0_3;
      logic [12:0] SAD_out, SAD_parcial, SAD_0;

    always_comb
    begin

      sub_0 = rorig_0 - rpred_0;
      sub_1 = rorig_1 - rpred_1;
      sub_2 = rorig_2 - rpred_2;
      sub_3 = rorig_3 - rpred_3;


      if ( sub_0[8] )
        abs0 = ~(sub_0) + 9'b1;
      else
      abs0 = sub_0;

      if ( sub_1[8] )
        abs1 = ~(sub_1) + 9'b1;
      else
      abs1 = sub_1;

      if ( sub_2[8] )
        abs2 = ~(sub_2) + 9'b1;
      else
      abs2 = sub_2;


      if ( sub_3[8] )
        abs3 = ~(sub_3) + 9'b1;
      else
      abs3 = sub_3;

      tree_0_1 = abs0 + abs1;
      tree_2_3 = abs2 + abs3;

      tree_0_3 = tree_0_1 + tree_2_3;

      if ( fline )
        SAD_parcial = tree_0_3;
      else
        SAD_parcial = tree_0_3 + SAD_0;
```

```
    end


    always_ff @( posedge arbif.clk or negedge arbif.rst )
    if ( !arbif.rst )
    begin
        rpred_0 <= 9'b0;
        rpred_1 <= 9'b0;
        rpred_2 <= 9'b0;
        rpred_3 <= 9'b0;
        rorig_0 <= 9'b0;
        rorig_1 <= 9'b0;
        rorig_2 <= 9'b0;
        rorig_3 <= 9'b0;
        fline   <= 1'b0;
      SAD_0   <= 13'b0;
    end
    else
    begin
        rpred_0 <= arbif.pred_0;
        rpred_1 <= arbif.pred_1;
        rpred_2 <= arbif.pred_2;
        rpred_3 <= arbif.pred_3;
        rorig_0 <= arbif.orig_0;
        rorig_1 <= arbif.orig_1;
        rorig_2 <= arbif.orig_2;
        rorig_3 <= arbif.orig_3;
        fline   <= arbif.first_line;

        SAD_0   <= SAD_parcial;


      if ( SAD_0 == arbif.SAD )
        $display("Value %b", SAD_0);
      else
      begin
        $display("Erro no SAD. %b   %b", SAD_0, arbif.SAD);
        $stop;
      end
    end

endmodule
```

APÊNDICE B < ARQUIVOS PARA SÍNTESE LÓGICA >

```
#######################################################
## Arquivo top_synth.tcl
#######################################################

### Template Script for RTL->Gate-Level Flow (generated from RC v07.20-s021_1)
#######################################################
## Include TCL utility scripts..
#######################################################
include load_etc.tcl
include top_setup.tcl
#######################################################
## Preset global variables and attributes
#######################################################
#set_attribute super_thread_servers {simulacao2 simulacao2 microel2 microel2
microel7 microel7 microel8 microel8}
set SYN_EFF high
set MAP_EFF high
set DATE [clock format [clock seconds] -format "%b%d-%T"]
## The following variables are used for diagnostic purposes only.
## They donot have any effect on the results of synthesis.
## Setting 'map_fancy_names' to 1 tells the tool to name the combinational
## cells based on the criteria that was used when selecting them.
set map_fancy_names 1

## Setting 'iopt_stats' to 1 prints out statistics during incremental
## optimization.
set iopt_stats 1

set _OUTPUTS_PATH outputs_${DATE}
set _LOG_PATH logs_${DATE}
set _REPORTS_PATH reports_${DATE}
set _LEC_PATH ${_OUTPUTS_PATH}/lec/
set _LEC_LOG_PATH ${_LOG_PATH}/lec/
set_attribute lib_search_path ${LIBRARY_PATH} /
set_attribute script_search_path ${SCRIPT_PATH} /
set_attribute hdl_search_path ${HDL_PATH} /
#set_attribute wireload_mode <value> /
set_attribute information_level 9 /
##set_attribute retime_reg_naming_suffix __retimed_reg /
#######################################################
## Library setup
#######################################################
set_attribute library $LIBRARY /

## PLE
set_attribute lef_library $LEFS /
#set_attribute lef_stop_on_error true /
#set_attribute cap_table_file $CAP_TABLE /
#set_attribute interconnect_mode ple /
##generates <signal>_reg[<bit_width>] format
#set_attribute hdl_array_naming_style %s\[%d\] /

## Turn on TNS, affects global and incr opto
set_attribute tns_opto true /

## Power root attributes
set_attribute lp_insert_clock_gating true /
#set_attribute lp_clock_gating_prefix <string> /
#set_attribute lp_insert_operand_isolation true /
#set_attribute lp_operand_isolation_prefix <string> /
set_attribute lp_power_analysis_effort high /
```

```
#set_attribute lp_power_unit mW /
#set_attribute lp_toggle_rate_unit /ns /
set_attribute hdl_track_filename_row_col true /
set_attribute lp_multi_vt_optimization_effort high /
#set_attribute retime_async_reset true /
#######################################################
## Load Design
#######################################################
read_hdl -v2001 $RTL_VLOG
read_hdl -vhdl $RTL
elaborate $DESIGN
puts "Runtime & Memory after 'read_hdl'"
timestat Elaboration
check_design -unresolved -all
#######################################################
## Constraints Setup
#######################################################
#sdc_set_dont_touch_network 1

read_sdc -stop_on_errors $SDC_CONSTRAINT_FILE
puts "The number of exceptions is [llength [find /designs/$DESIGN -exception
*]]"

#report power -rtl
#set_attribute force_wireload <wireload name> "/designs/$DESIGN"

if {![file exists ${_LOG_PATH}]} {
  file mkdir ${_LOG_PATH}
  puts "Creating directory ${_LOG_PATH}"
}

if {![file exists ${_OUTPUTS_PATH}]} {
  file mkdir ${_OUTPUTS_PATH}
  puts "Creating directory ${_OUTPUTS_PATH}"
}

if {![file exists ${_REPORTS_PATH}]} {
  file mkdir ${_REPORTS_PATH}
  puts "Creating directory ${_REPORTS_PATH}"
}

if {![file exists ${_OUTPUTS_PATH}/lec]} {
  file mkdir ${_OUTPUTS_PATH}/lec
  puts "Creating directory ${_OUTPUTS_PATH}/lec"
}
if {![file exists ${_LOG_PATH}/lec]} {
  file mkdir ${_LOG_PATH}/lec/h264_chip/PRE_PROT_TOP_INST/scan_out
  puts "Creating directory ${_LOG_PATH}/lec"
}
report timing -lint
#######################################################
## Define cost groups (clock-clock, clock-output, input-clock, input-output)
#######################################################
## Uncomment to remove already existing costgroups before creating new ones.
rm [find /designs/* -cost_group *]

if {[llength [all::all_seqs]] > 0} {
  define_cost_group -name I2C -design $DESIGN
  define_cost_group -name C2O -design $DESIGN
  define_cost_group -name C2C -design $DESIGN
  path_group -from [all::all_seqs] -to [all::all_seqs] -group C2C -name C2C
  path_group -from [all::all_seqs] -to [all::all_outs] -group C2O -name C2O
  path_group -from [all::all_inps]  -to [all::all_seqs] -group I2C -name I2C
}

#define_cost_group -name I2O -design $DESIGN
#path_group -from [all::all_inps]  -to [all::all_outs] -group I2O -name I2O
```

```
foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] >> $_REPORTS_PATH/${DESIGN}_pretim.rpt
}

#######################################################
## DFT Setup
#######################################################
define_dft test_clock -name scan_clk1 -domain clockI
/h264_chip/PRE_PROT_TOP_INST/clk_i
define_dft test_clock -name scan_clk2 -domain clockUPCI
/h264_chip/PRE_PROT_TOP_INST/upc_clk_i
define_dft shift_enable -name scan_enable -active high
/h264_chip/PRE_PROT_TOP_INST/scan_en
set_attribute dft_mix_clock_edges_in_scan_chain true /h264_chip
#define_dft test_mode -name scan_input -active high scan_in
set_attribute dft_scan_map_mode tdrc_pass "/designs/$DESIGN"
set_attribute dft_connect_shift_enable_during_mapping tie_off
"/designs/$DESIGN"
set_attribute dft_connect_scan_data_pins_during_mapping loopback
"/designs/$DESIGN"
#set_attribute dft_dont_scan true <instance or subdesign>
#set_attribute dft_controllable "<from pin> <inverting|non_inverting>" <to
pin>

define_dft scan_chain -name scan_test01 -sdi
/h264_chip/PRE_PROT_TOP_INST/scan_in  -sdo
/h264_chip/PRE_PROT_TOP_INST/scan_out  -shift_enable scan_enable -domain
clockI -edge rise
define_dft scan_chain -name scan_test02 -sdi
/h264_chip/PRE_PROT_TOP_INST/scan_in_upc  -sdo
/h264_chip/PRE_PROT_TOP_INST/scan_out_upc  -shift_enable scan_enable -domain
clockUPCI -edge rise

check_dft_rules > ${_REPORTS_PATH}/${DESIGN}-tdrcs
## insert_dft shadow_logic -around <instance> -mode <string> -test_mode
<test_signal>
## insert_dft test_point -location <port|pin> -test_mode <test_signal> -
test_clock_pin <port|pin> -node <port|pin> -type <string>
## check_dft_rules >> ${DESIGN}-tdrcs
report dft_registers > ${_REPORTS_PATH}/${DESIGN}-DFTregs
report dft_setup > ${_REPORTS_PATH}/${DESIGN}-DFTsetup_tdrc
## set numDFTviolations [check_dft_rules]
## if {$numDFTviolations > "0"} {
##   fix_dft_violations -async_set -async_reset [-clock] -test_mode
<shiftEnableObjectOrTestModeObject>
## }

#### Retime
#set_attribute retime 1 "/designs/$DESIGN"
#retime wdt -effort high
#######################################################
## Leakage/Dynamic power/Clock Gating setup.
#######################################################

#set_attribute lp_clock_gating_cell [find /lib* -libcell <cg_libcell_name>]
"/designs/$DESIGN"
set_attribute max_leakage_power 0.0 "/designs/$DESIGN"
set_attribute max_dynamic_power 0.0 "/designs/$DESIGN"
#set_attribute lp_power_optimization_weight <value from 0 to 1>
"/designs/$DESIGN"
#set_attribute max_dynamic_power <number> "/designs/$DESIGN"
set_attr lp_optimize_dynamic_power_first true "/designs/$DESIGN"
set_attribute lp_clock_gating_test_signal scan_enable "/designs/$DESIGN"
#set_attribute lp_map_to_srpg_cells true "/designs/$DESIGN"
#set_attribute lp_srpg_pg_driver <driver> "/designs/$DESIGN"
## read_tcf <TCF file name>
## read_saif <SAIF file name>
```

```
#read_vcd -static -vcd_module i_wdt $VCD_FILE
#######################################################
## Synthesizing to generic
#######################################################
synthesize -to_generic -eff $SYN_EFF
puts "Runtime & Memory after 'synthesize -to_generic'"
timestat GENERIC
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_generic.rpt
## ungroup -threshold <value>
#######################################################
## Synthesizing to gates
#######################################################
synthesize -to_mapped -eff $MAP_EFF -no_incr
puts "Runtime & Memory after 'synthesize -to_map -no_incr'"
timestat MAPPED
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_map.rpt

##Post global map netlist for LEC verification..
write_hdl  > ${_OUTPUTS_PATH}/${DESIGN}_global_mapped.v
write_do_lec -no_exit -revised_design
${_OUTPUTS_PATH}/${DESIGN}_global_mapped.v -logfile
${_LEC_LOG_PATH}/rtl2globalmap.lec.log > ${_LEC_PATH}/rtl2globalmap.lec.do
#######################################################
## Retime
#######################################################
## if {[llength [all::all_seqs]] > 0} {
    ##set_attribute dont_retime true [all::all_seqs -clock <clock_name>] #for
multiple clock designs
## }
## retime -min_delay #manual retiming for minimum delay
## or
## retime -min_area #manual retiming for minimum area

foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] > $_REPORTS_PATH/${DESIGN}_[basename
$cg]_post_map.rpt
}
#######################################################
## Optional additional DFT commands.
#######################################################
connect_scan_chains
report dft_chains > ${DESIGN}-DFTchains
report dft_setup > ${DESIGN}-DFTsetup
## write_scandef > ${DESIGN}-scanDEF
## write_atpg [-stil_or_mentor_or_cadence] > ${DESIGN}-ATPG
## write_dft_abstract_model > ${DESIGN}-scanAbstract
## write_hdl -abstract > ${DESIGN}-logicAbstract
#######################################################
## Incremental Synthesis
#######################################################
synthesize -to_mapped -eff $MAP_EFF -incr
puts "Runtime & Memory after incremental synthesis"
timestat INCREMENTAL

foreach cg [find / -cost_group -null_ok *] {
  report timing -cost_group [list $cg] > $_REPORTS_PATH/${DESIGN}_[basename
$cg]_post_incr.rpt
}

write_sdf > ${_OUTPUTS_PATH}/${DESIGN}.sdf
#write_spef > ${_OUTPUTS_PATH}/${DESIGN}.spef
#write_set_load filter > ${_OUTPUTS_PATH}/${DESIGN}.setload
#######################################################
## Remove assigns from the netlist..
#######################################################
## remove_assigns
## Low effort Incremental Synthesis after 'remove_assigns'
```

```
## synthesize -to_mapped -eff low -incr
#######################################################
## write Encounter file set (verilog, SDC, config, etc.)
#######################################################
##write_encounter design -basename <path & base filename> -lef <lef_file(s)>
report summary > $_REPORTS_PATH/${DESIGN}_summary.rpt
report timing > $_REPORTS_PATH/${DESIGN}_timing.rpt
report power > $_REPORTS_PATH/${DESIGN}_power2.rpt
report clock_gating > $_REPORTS_PATH/${DESIGN}_clockgating.rpt
report power -depth 0 > $_REPORTS_PATH/${DESIGN}_power.rpt
report gates -power > $_REPORTS_PATH/${DESIGN}_gates_power.rpt
report area > $_REPORTS_PATH/${DESIGN}_area.rpt
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_incr.rpt
write_hdl > ${_OUTPUTS_PATH}/${DESIGN}_m.hvsyn
write_sdc > ${_OUTPUTS_PATH}/${DESIGN}_m.sdc
write_script > ${_OUTPUTS_PATH}/${DESIGN}_m.script
#######################################################
### write_do_lec
#######################################################
write_do_lec -no_exit -golden_design
${_OUTPUTS_PATH}/${DESIGN}_global_mapped.v -revised_design
${_OUTPUTS_PATH}/${DESIGN}_m.hvsyn -logfile
${_LEC_LOG_PATH}/globalmap2final.lec.log > ${_LEC_PATH}/globalmap2final.lec.do
##Uncomment if the RTL is to be compared with the final netlist..
write_do_lec -no_exit -revised_design ${_OUTPUTS_PATH}/${DESIGN}_m.hvsyn -
logfile ${_LEC_LOG_PATH}/rtl2final.lec.log > ${_LEC_PATH}/rtl2final.lec.do
#######################################################
### write_do_lec
#######################################################
write_encounter design $DESIGN -basename ../fe/top
puts "Final Runtime & Memory."
timestat FINAL
puts "============================"
puts "Synthesis Finished ........."
puts "============================"
file copy [get_attr stdout_log /] ${_LOG_PATH}/.
##quit
```

```
#######################################################
## Arquivo top_setup.tcl
#######################################################

set DESIGN h264_chip
set SDC_CONSTRAINT_FILE ../constraints/top.sdc
set LIBRARY_PATH { ./ }
set SCRIPT_PATH { . }
set HDL_PATH {../rtl ../rtl/CAVLD ../rtl/parser ../rtl/itiq ../rtl/intra}
set LIBRARY {
/ferramentas/design_kits/ibm_65nm/stdcells/lvt/cp65nm_HighDensity/liberty/si/c
p65npkldst_ss1p08v125c.lib
/ferramentas/design_kits/ibm_65nm/stdcells/svt/cp65nm_HighDensity/liberty/si/c
p65npksdst_ss1p08v125c.lib
/ferramentas/design_kits/ibm_65nm/stdcells/hvt/cp65nm_HighDensity/liberty/si/c
p65npkhdst_ss1p08v125c.lib
../../Memories/compout/views/sadslnpky2p1024x32cm4sw0bk1ltlc1/ss1p08v125c/sads
lnpky2p1024x32cm4sw0bk1ltlc1.lib
../../Memories/compout/views/sadslnpky2p16x36cm4sw0bk1ltlc1/ss1p08v125c/sadsln
pky2p16x36cm4sw0bk1ltlc1.lib
../../Memories/compout/views/sadslnpky2p2048x4cm4sw0bk1ltlc1/ss1p08v125c/sadsl
npky2p2048x4cm4sw0bk1ltlc1.lib
/ferramentas/design_kits/ibm_65nm/pad/FE_rgo_csm65_25v33_lpe_50c_rev_1_3/synop
sys/1_80/rgo_csm65_25v33_lpe_50c_ss_108_162_125.lib
}

set LEFS
{/ferramentas/design_kits/ibm_65nm/stdcells/lvt/cp65nm_HighDensity/lef/5.6/cp6
5npkldst_m06f0f1.lef
/ferramentas/design_kits/ibm_65nm/stdcells/lvt/cp65nm_HighDensity/lef/5.6/cp65
npkldst.lef
/ferramentas/design_kits/ibm_65nm/stdcells/svt/cp65nm_HighDensity/lef/5.6/cp65
npksdst.lef
/ferramentas/design_kits/ibm_65nm/stdcells/hvt/cp65nm_HighDensity/lef/5.6/cp65
npkhdst.lef
../../Memories/compout/views/sadslnpky2p1024x32cm4sw0bk1ltlc1/sadslnpky2p1024x
32cm4sw0bk1ltlc1.plef
../../Memories/compout/views/sadslnpky2p16x36cm4sw0bk1ltlc1/sadslnpky2p16x36cm
4sw0bk1ltlc1.plef
../../Memories/compout/views/sadslnpky2p2048x4cm4sw0bk1ltlc1/sadslnpky2p2048x4
cm4sw0bk1ltlc1.plef
/ferramentas/design_kits/ibm_65nm/pad/FE_rgo_csm65_25v33_lpe_50c_rev_1_3/lef/r
go_csm65_25v33_lpe_50c_5_00_01_00.lef
}

set RTL_VLOG {h264_chip.v \
}

set RTL {pack_intra.vhd \
buf1to4.vhd \
buf4to16.vhd \
buf16to16.vhd \
bufctx.vhd \
bufvars.vhd \
controle.vhd \
dec_contexto.vhd \
GP4x4.vhd \
GP.vhd \
gp_luma16.vhd \
intra.vhd \
mem_viz.vhd \
mux2to1.vhd \
mux4to1.vhd \
mux16x1.vhd \
partecon.vhd \
parteoper.vhd \
PE.vhd \
```

```
pred4.vhd \
reg_amostra.vhd \
rom_controle.vhd \
v4x4.vhd \
intra4.vhd \
pred_fifo_extract.vhd \
BlocoTIcont.vhd \
bufCbac.vhd \
bufCdc.vhd \
bufCrac.vhd \
bufYac.vhd \
conIHad2.vhd \
fifo_ti.vhd \
IBS.vhd \
idct4.vhd \
idct4cont.vhd \
IHad2.vhd \
ihad4.vhd \
ihad4cont.vhd \
IQuant.vhd \
MSDgenerator.vhd \
Mux6E.vhd \
Mux8a.vhd \
Mux8ah.vhd \
Mux8b.vhd \
Mux8bh.vhd \
Mux8c.vhd \
Mux8ch.vhd \
Mux8s.vhd \
Mux8sh.vhd \
mux16x1.vhd \
ping4.vhd \
ping4B.vhd \
ping16a.vhd \
ping16ah.vhd \
ping16b.vhd \
ping16bh.vhd \
ping16c.vhd \
ping16ch.vhd \
ping16x.vhd \
ping16xh.vhd \
BlocoTIvhd.vhd \
pars2itiq_fifo_itf.vhd \
res_merge.vhd \
SRAM_itiq.vhd \
itiq_fifo.vhd \
ITIQ_top.vhd \
pack_cavld.vhd \
coeff4_1bit.vhd \
coeff5_1bit.vhd \
coeff3_1bit.vhd \
coeff2_1bit.vhd \
coeff1_1bit.vhd \
coeff0_1bit.vhd \
coeff_token.vhd \
t1.vhd \
ctokenPlusT1.vhd \
remount_sync.vhd \
run_before.vhd \
levels.vhd \
total_zeros_4x4_1bit.vhd \
total_zeros_2x4_1bit.vhd \
total_zeros_2x2_1bit.vhd \
total_zeros.vhd \
cavlc_decoder_control.vhd \
cavlc_uCore.vhd \
cavlc_wsBuff.vhd \
```

```
mem.vhd \
SRAM_parser.vhd \
parser_fifo.vhd \
NAL_delimiter.vhd \
NAL_unit.vhd \
Serializer.vhd \
slice_header.vhd \
slice_data.vhd \
nc.vhd \
expgolomb.vhd \
parser_control.vhd \
cbp_dec.vhd \
resToFifoItf.vhd \
predToFifoItf.vhd \
parser_top.vhd \
sum_sel.vhd \
SRAM_p2itiq.vhd \
parser_itiq_fifo.vhd \
SRAM_p2intra.vhd \
parser_intra_fifo.vhd \
pre_prot_top.vhd
}
```

APÊNDICE C < MODIFICAÇÕES NA DESCRIÇÃO DE *HARDWARE* >

```
---------------------------
-- Assíncrono
---------------------------
pREGi:process(intra_clk_i,rst_i)
    begin
      if rst_i = '1' then
        sum_inv_reg_r <= (others=>'0');
        intra_reg_r <= (others=>'0');
      elsif intra_clk_i'event and intra_clk_i='1' then
         sum_inv_reg_r <= sum_inv_w;
         intra_reg_r <= intra_w;
      end if;
    end process;

---------------------------
-- Síncrono
---------------------------
pREGi:process(intra_clk_i)
    begin
      if intra_clk_i'event and intra_clk_i='1' then
         if rst_i = '1' then
           sum_inv_reg_r <= (others=>'0');
           intra_reg_r <= (others=>'0');
         else
            sum_inv_reg_r <= sum_inv_w;
            intra_reg_r <= intra_w;
         end if;
      end if;
    end process;
```

APÊNDICE D < ARQUIVO DE IO >

```
#######################################################
#                                     #
#  Cadence Design Systems, Inc.                    #
#  FirstEncounter Data file for I/O Placement
#                                     #
#######################################################
# Created by First Encounter v08.10-s207_1 on Tue May 17 13:04:34 2011
(globals
   version = 3
   io_order = default
)
(iopad
   (topright
         (inst name="Pcornerur"         )
   )
   (left
         (inst  name="PAD_lop_out_o05"     )
         (inst  name="PAD_lop_out_o04"     )
         (inst  name="PAD_lop_out_o03"     )
         (inst  name="PAD_lop_out_o02"     )
         (inst  name="PAD_lop_out_o01"     )
         (inst  name="PAD_lop_out_o00"     )
         (inst  name="PAD_lop_en_i"        )
         (inst  name="PAD_config_ip08"     )
         (inst  name="PAD_config_ip07"     )
         (inst  name="PAD_config_ip06"     )
         (inst  name="PAD_config_ip05"     )
         (inst  name="PAD_config_ip04"     )
         (inst  name="PAD_config_ip03"     )
         (inst  name="PAD_config_ip02"     )
         (inst  name="PAD_config_ip01"     )
```

```
        (inst  name="PAD_config_ip00"      )
)
(topleft
        (inst name="Pcornerul"         )
)
(bottom
        (inst  name="PAD_lop_out_o28"     )
        (inst  name="PAD_lop_out_o27"     )
        (inst  name="PAD_lop_out_o26"     )
        (inst  name="PAD_lop_out_o25"     )
        (inst  name="PAD_vdd_io"   )
        (inst  name="PAD_vdd_core1"       )
        (inst  name="PAD_clk_i"      )
        (inst  name="PAD_rst_i"      )
        (inst  name="PAD_upc_clk_i"       )
        (inst  name="PAD_vss_core1"       )
        (inst  name="PAD_vss_io"    )
        (inst  name="PAD_lop_out_o24"     )
        (inst  name="PAD_lop_out_o23"     )
        (inst  name="PAD_lop_out_o22"     )
        (inst  name="PAD_lop_out_o21"     )
        (inst  name="PAD_lop_out_o20"     )
)
(bottomleft
        (inst name="Pcornerll"         )
)
(right
        (inst  name="PAD_lop_out_o19"     )
        (inst  name="PAD_lop_out_o18"     )
        (inst  name="PAD_lop_out_o17"     )
        (inst  name="PAD_lop_out_o16"     )
        (inst  name="PAD_lop_out_o15"     )
        (inst  name="PAD_lop_out_o14"     )
        (inst  name="PAD_lop_out_o13"     )
        (inst  name="PAD_lop_out_o12"     )
```

```
        (inst  name="PAD_lop_out_o11"     )
        (inst  name="PAD_lop_out_o10"     )
        (inst  name="PAD_lop_out_o09"     )
        (inst  name="PAD_lop_out_o29"     )
        (inst  name="PAD_lop_out_o30"     )
        (inst  name="PAD_lop_out_o31"     )
        (inst  name="PAD_lop_vld_o"       )
        (inst  name="PAD_scan_en" )
)
(bottomright
        (inst name="Pcornerlr"          )
)
(top
        (inst  name="PAD_upc_dat_i00"     )
        (inst  name="PAD_upc_dat_i01"     )
        (inst  name="PAD_upc_dat_i02"     )
        (inst  name="PAD_upc_dat_i03"     )
        (inst  name="PAD_upc_dat_i04"     )
        (inst  name="PAD_upc_dat_i05"     )
        (inst  name="PAD_upc_dat_i06"     )
        (inst  name="PAD_upc_dat_i07"     )
        (inst  name="PAD_upc_prog_full_o"          )
        (inst  name="PAD_upc_vld_i"       )
        (inst  name="Spare_Cell01"  )
        (inst  name="Spare_Cell02"  )
        (inst  name="Spare_Cell03"  )
        (inst  name="Spare_Cell04"  )
        (inst  name="Spare_Cell05"  )
        (inst  name="Spare_Cell06"  )
        (inst  name="Spare_Cell07"  )
        (inst  name="Spare_Cell08"  )
        (inst  name="Spare_Cell09"  )
        (inst  name="Spare_Cell10"  )
        (inst  name="Spare_Cell11"  )
        (inst  name="Spare_Cell12"  )
```

```
        (inst  name="Spare_Cell13"  )
        (inst  name="Spare_Cell14"  )
        (inst  name="Spare_Cell15"  )
        (inst  name="PAD_lop_out_o08"     )
        (inst  name="PAD_lop_out_o07"     )
        (inst  name="PAD_lop_out_o06"     )
    )
)
```