ANDERSON CASTELLAR

Proposta de metodologia para utilização em hardware reconfigurável para aplicações aeroespaciais.

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Processamento de sinais e instrumentação.

Orientador: Prof. Dr. Evandro Luís Linhari Rodrigues

SÃO CARLOS
2008

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

C348p

Castellar, Anderson
Proposta de metodologia para utilização em *hardware* reconfigurável para aplicações aeroespaciais / Anderson Castellar ; orientador Evandro Luís Linhari Rodrigues. -- São Carlos, 2008.

Dissertação (Mestrado-Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Processamento de Sinais e Instrumentação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2008.

1. Sensoriamento remoto. 2. *Hardware* reconfigurável. 3. Metodologia de desenvolvimento. 4. Engenharia de *software*. 5. Engenharia aeroespacial. 6. Satélites. 7. Verificação de *software*. 8. Validação de *software*. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **ANDERSON CASTELLAR**

Dissertação defendida e julgada em 19/09/2008 perante a Comissão Julgadora:

APROVADO

Prof. Dr. **EVANDRO LUIS LINHARI RODRIGUES (Orientador)**
(Escola de Engenharia de São Carlos/USP)

APROVADO

Prof. Dr. **MARCELO ANDRADE DA COSTA VIEIRA**
(Escola de Engenharia de São Carlos/USP)

APROVADO

Prof. Associado **NORIAN MARRANGHELLO**
(Universidade Estadual Paulista "Júlio de Mesquita Filho" /UNESP/Campus de São José do Rio Preto).

Prof. Associado **GERALDO ROBERTO MARTINS DA COSTA**
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica e
Presidente da Comissão de Pós-Graduação

DEDICATÓRIA

À minha companheira e amiga Bruna pela paciência e pelo apoio incondicional durante a realização deste trabalho.

Aos meus pais Oswaldo e Lúcia que me ensinaram os verdadeiros valores da vida.

Ao meu irmão Alexandre pelo companheirismo sempre presente.

AGRADECIMENTOS

Primeiramente a Deus pela oportunidade de participar desta grande lição que é a vida.

Ao meu orientador Prof. Dr. Evandro Luís Linhari Rodrigues pela confiança depositada, pela paciência e principalmente pelas observações valiosas a este trabalho.

À Universidade de São Paulo por me ensinar engenharia.

À empresa Opto Eletrônica S/A, representada pelo seu presidente Prof. Dr. Jarbas Caiado de Castro Neto, por permitir o desenvolvimento deste trabalho.

Ao Dr. Mário Stefani, diretor do departamento de pesquisa e desenvolvimento da empresa Opto Eletrônica S/A pelo apoio no desenvolvimento deste trabalho.

Aos amigos pesquisadores da empresa Opto Eletrônica S/A que, de forma direta ou indireta, contribuíram com a realização deste trabalho.

“Change is the law of life and who look only to the past or present are certain to miss the future.”

John Fitzgerald Kennedy

RESUMO

CASTELLAR, A. **Proposta de metodologia para utilização em hardware reconfigurável para aplicações aeroespaciais.** 2008. 191f. Dissertação (Mestrado em Engenharia Elétrica) - Departamento de Engenharia Elétrica – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos 2008.

O programa CBERS é uma parceria entre o governo Brasileiro e o governo Chinês para desenvolvimento de satélites para sensoriamento remoto. A metodologia proposta será aplicada na Câmera Multi Espectral (MUXCAM) dos satélites CBERS-3 e 4, a primeira deste gênero a ser totalmente produzida no Brasil. Devido à alta confiabilidade exigida, principalmente devido ao custo elevado, as aplicações aeroespaciais que envolvem hardware reconfigurável devem possuir uma metodologia de desenvolvimento, desde a definição dos requisitos até o processo de verificação e validação. A utilização da linguagem VHDL e da ferramenta de síntese, processo este chamado de metodologia clássica, produzem um circuito final não otimizado, eliminando redundâncias e alterando a arquitetura proposta. Este trabalho propõe uma metodologia que busca garantir a utilização de uma única arquitetura desde o início do ciclo de desenvolvimento até sua finalização. Esta metodologia torna o processo de desenvolvimento mais confiável e determinístico.

Palavras chave: hardware reconfigurável, engenharia aeroespacial, satélites, sensoriamento remoto, metodologia de desenvolvimento, engenharia de software, verificação de software, validação de software.

ABSTRACT

CASTELLAR, A. **Proposal methodology for use in reprogrammable hardware in aerospace applications.** 2008. 191f. Thesis (Master Degree) - Departamento de Engenharia Elétrica –Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos 2008.

The CBERS program is a partnership between Brazil and China to produce satellites for remote sensing, producing images of the Earth for studies in several areas, mainly the ones related to the sustainable exploitation of natural resources. The methodology proposed in this work will be applied on the satellite CBERS-3 e 4's Multispectral Camera (MUXCAM), the first of it's gender fully produced in Brazil. Because the high reliability involved in aerospace applications, a methodology is necessary from software specification until the verification and validation process to guarantee the high reliability. The use of the synthesis tool and VHDL produce a poor circuit, eliminating redundance and making architectural changes. This work propose a methodology to keep the architectural the same during all the development cycle, making the development process more trustful for aerospace applications.

Keyworks: reprogrammable hardware, aerospace engineering, satellite, remote sensing, development methodology, software engineering, software verification, software validation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVEATURAS E SIGLAS

ASIC	*Application Specific Integrated Circuits*
CBERS	*China Brazil Earth Resourses Satellite*
CCD	*Charge-Coupled Device*
CLBI	Centro de Lançamento da Barreira do Inferno
CMOS	*Complementary Mmetal Oxide Semiconductor*
CNAE	Comissão Nacional de Atividades Espaciais
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
COBAE	Comissão Brasileira de Atividades Espaciais
CQFP	*Ceramic Quad Flat Pack*
CVS	*Concurrent Version System*
DDR	*Digital Data Recorder*
ELCK	Placa eletrônica de geração de clocks
ELVD	Placa eletrônica de processamento de vídeo digital
ERTS	*Earth Resources Technology Satellite*
ESP	*Embedded Standard Products*
FPGA	*Field Programmable Gate Array*
GOCNAE	Grupo de Organização da Comissão Nacional de Atividades Espaciais
GPS	*Global Positioning System*
GTL	*Gunning Transceiver Logic*
HRC	*High Resolution Camera*
HRCCD	*High Resolution CCD Camera*

IEEE	*Institute of Electrical and Eletronic Engineers*
INPE	Instituto Nacional de Pesquisas Espaciais
IRMSS	Imageador por Varredura de Média Resolução – CBERS1/2
IRSCAM	Imageador por Varredura de Média Resolução – CBERS3/4
ISS	*International Space Station*
ITAR	*International Traffic in Arms Regulations*
LIT	Laboratório de Integração e Testes
LVCMOS	*Low Voltage Complementary Metal Oxide Semiconductor*
LVTTL	*Low Voltage Transistor Transistor Logic*
MECB	Missão Espacial Completa Brasileira
MUXCAM	Câmera Multi Espectral
OTP	*One Time Programable*
PANMUX	Câmera PanMux
PCI	*Peripheral Component Interconnect*
PLL	*Phase Locked Loop*
RADHARD	*Radiation Hardened*
RBN	Câmera Multi Espectral - INPE
RBNA	Unidade ótica da Câmera Multi Espectral
RBNB	Unidade de controle térmico, calibração e ajuste de foco da Câmera Multi Espectral
RBNC	Unidade de processamento de vídeo da Câmera Multi Espectral
SAA	*South Atlantic Anomaly*
SAFO	Sondagem Aeronômica de Foguetes
SCD	Satélite de Coleta de Dados

SEB	*Single Event Burnout*
SEE	*Single Event Efects*
SEGR	*Single Event Gate Rupture*
SEL	*Single Event Latch-up*
SERE	Sensoriamento Remoto
SSTL2	*Stub Series Terminated Logic for 2,5v*
SSTL3	*Stub Series Terminated Logic for 3,3V*
STM	*Structural Thermal Model*
TID	*Total Ionizing Dose*
TMR	*Triplicate Modular Redundancy*
VLSI	*Very Large Scale Integration*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*
WFI	Câmera Imageadora de Amplo Campo de Visada – CBERS1/2
WFICAM	Câmera Imageadora de Amplo Campo de Visada – CBERS3/4

SUMÁRIO

1 Introdução

Em 2004 a empresa Opto Eletrônica S/A foi escolhida, através de uma licitação pública, para realizar o desenvolvimento de uma câmera para sensoriamento remoto, a Câmera Multi Espectral (MUXCAM) para Instituto Nacional de Pesquisas Espaciais (INPE). Esta câmera fará parte dos satélites de sensoriamento remoto CBERS-3 e 4 e é responsável pela captação de imagens da terra com resolução espacial de 20 metros. O objetivo principal é de monitorar todo território nacional, consolidando assim a autonomia no segmento de sensoriamento remoto. O custo estimado pelo INPE para o desenvolvimento desta câmera é de 45 milhões de reais.

A Câmera Multi Espectral (MUXCAM) é composta por elementos ópticos e por diversas placas eletrônicas, responsáveis pelo processamento analógico e digital de sinais. Todo o processamento digital de sinais desta câmera é realizado em *hardware* reconfigurável do tipo *Field Programmable Gate Array* (FPGA). *Hardware* reconfigurável são dispositivos lógicos programáveis que estão sendo amplamente utilizados em projetos aeroespaciais, devido às vantagens apresentadas em relação ao desenvolvimento convencional. Estes dispositivos eletrônicos são sensíveis, assim como outros dispositivos, a efeitos de radiação. Estes efeitos podem gerar alterações de parâmetros internos, causando falhas de funcionamento, caso estas alterações não estejam dentro das margens de segurança do projeto. Devido ao elevado custo de desenvolvimento da câmera MUXCAM, o *software* utilizado no *hardware* reconfigurável deve garantir, por projeto, que variações de parâmetros internos, principalmente atrasos nos sinais digitais, não afetem o seu funcionamento durante sua vida útil.

1.1 Objetivo do trabalho

O objetivo principal deste trabalho é propor uma nova metodologia de desenvolvimento para utilização em hardware reconfigurável do tipo FPGA, visando garantir seu pleno funcionamento em órbita, garantindo por projeto margens de segurança e redundâncias.

A metodologia clássica limita a implementação de redundâncias, devido as características da ferramenta de síntese, que otimiza o circuito gerado e elimina qualquer redundância implementada.

A nova metodologia proposta será aplicada nas FPGAs presentes nas placas de processamento de vídeo (RBNC-ELVD) e de geração de *clocks* do CCD (RBNC-ELCK) da Câmera Multi Espectral (MUXCAM).

1.2 Relevância do trabalho

O Brasil, através do desenvolvimento dos satélites CBERS-3 e 4, inicia um processo de capacitação tecnológica de empresas nacionais, visando aplicações aeroespaciais. Esta é a primeira oportunidade que empresas nacionais, por intermédio do INPE, colocarão em órbita hardware reconfigurável do tipo FPGA cujo *software* foi totalmente desenvolvido, verificado e validado em território nacional. Neste cenário, o trabalho se torna relevante, por propor uma metodologia específica para aplicações aeroespaciais, que poderá ser utilizada como base inicial para futuros desenvolvimentos de *software* para hardware reconfigurável do tipo FPGA em aplicações aeroespaciais no território nacional.

1.3 Contribuições do trabalho

A primeira contribuição deste trabalho é a análise crítica da ferramenta de desenvolvimento QuickWorks™ fornecida pela empresa Quicklogic (2008c) para o desenvolvimento de aplicações aeroespaciais. A segunda contribuição do trabalho é a proposição de uma metodologia de desenvolvimento de *software* para utilização em hardware reconfigurável, visando garantir redundâncias e margens de segurança por projeto, eliminando os principais problemas apresentados pela ferramenta de síntese. Assim, há um aumento da confiabilidade e uma redução do tempo do processo de verificação e validação para aplicações aeroespaciais.

1.4 Estrutura do trabalho

O primeiro capítulo que termina aqui apresenta uma introdução ao trabalho.

O segundo capítulo apresenta uma introdução ao cenário aeroespacial brasileiro, apresentando também detalhes do projeto da Câmera Multi Espectral dos satélites CBERS-3 e 4.

O terceiro capítulo apresenta as principais características do hardware reconfigurável utilizado durante o desenvolvimento do projeto da Câmera Multi Espectral dos satélites CBERS-3 e 4. Ele apresenta também um histórico e as principais características, vantagens e desvantagens da linguagem de descrição de hardware VHDL e um estudo dos efeitos de radiação em componentes eletrônicos, visando seus efeitos principalmente em hardware reconfigurável, assim como os possíveis meios de mitigação.

O quarto capítulo apresenta um estudo da ferramenta de desenvolvimento QuickWorks™ (2008) fornecida pela empresa Quicklogic (2008). Neste estudo

verificamos a possibilidade de utilização desta plataforma para aplicações aeroespaciais de alta confiabilidade.

O quinto capítulo apresenta as principais metodologias de desenvolvimento de software, apresentando o modelo Cascata e a metodologia clássica.

O sexto capítulo apresenta a metodologia proposta.

O sétimo capítulo apresenta os resultados e as discussões do trabalho.

O oitavo capítulo apresenta as conclusões finais do trabalho.

O nono capítulo apresenta as propostas para trabalhos futuros.

O décimo capítulo apresenta as referências bibliográficas.

2 Programa Espacial Brasileiro

O programa espacial brasileiro teve início em 1961 através de um decreto presidencial criando o Grupo de Organização da Comissão Nacional de Atividades Espaciais (GOCNAE) para a realização de estudos no campo das ciências espaciais e atmosféricas. Em 1963, o GOCNAE torna-se Comissão Nacional de Atividades Espaciais (CNAE).

Em 1965, foram realizadas as primeiras campanhas de lançamento de foguetes de sondagem, com carga útil, a partir do Centro de Lançamento da Barreira do Inferno (CLBI) na cidade de Natal no Rio Grande do Norte. Durante os anos seguintes até 1970, foram lançados cerca de 230 foguetes nacionais e estrangeiros dentro do projeto de Sondagem Aeronômica de Foguetes (SAFO).

Em 1969, com a criação do projeto SERE, a CNAE deu início às atividades de sensoriamento remoto, envolvendo treinamento de pessoal nos Estados Unidos, para realizações de missões de mapeamento dos recursos naturais do território brasileiro utilizando fotos aéreas e de recepção de dados do *Earth Resources Technology Satellite* (ERTS), que deu origem à série de satélites *Landsat*.

Com a extinção da CNAE em 1971, ocorre a criação oficial do Instituto Nacional de Pesquisas Espaciais (INPE), subordinado diretamente ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). O decreto de criação do INPE define o Instituto como o principal órgão de execução civil para o desenvolvimento das pesquisas espaciais, sob a orientação da Comissão Brasileira de Atividades Espaciais (COBAE), órgão de assessoramento da Presidência da República.

Em 1979 ocorre a aprovação da Missão Espacial Completa Brasileira (MECB) cujos objetivos iniciais eram “[...] o desenvolvimento de quatro satélites

e de um veículo lançador e a construção da infra-estrutura para as operações de lançamento” (INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS, 2007a).

A MECB permitiu a construção do Laboratório de Integração e Testes (LIT), inaugurado em 1987, sendo responsável pela montagem e integração de satélites brasileiros e estrangeiros, além da prestação de serviços de teste, verificação e calibração em vários ramos da indústria nacional. A Figura 2-1 apresenta as instalações do LIT durante a montagem final do satélite CBERS 2.

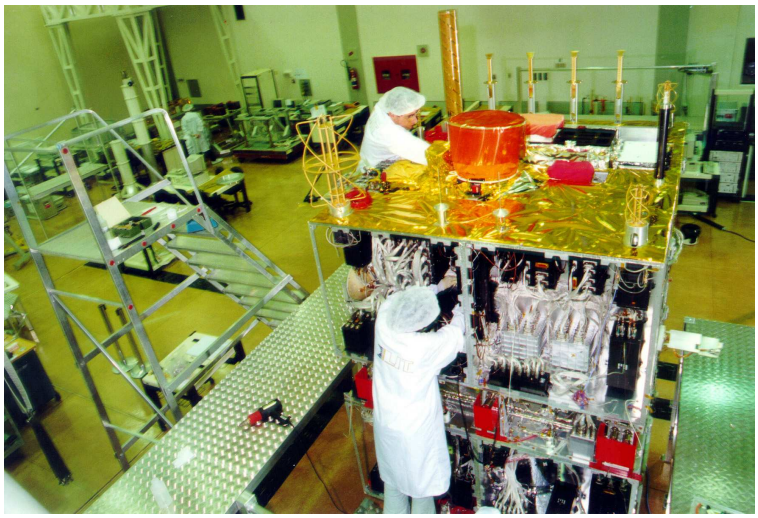


Figura 2-1. LIT-INPE – Montagem Final CBERS 2.
Fonte: INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (2007a).

Em 1988 Brasil e China assinam um acordo de cooperação visando o desenvolvimento de satélites de recursos terrestres.

O lançamento do SCD-1, primeiro satélite brasileiro de coleta de dados, totalmente desenvolvido pelo INPE, é realizado com sucesso em 1993 através da base de Cabo Canaveral, na Flórida (EUA). O lançamento do SCD-2 ocorre em 1998 com sucesso, também pela base americana de Cabo Canaveral, na Flórida (EUA).

Em 1999, é lançado com sucesso o satélite CBERS-1, a partir da base de Taiyuan, na China. Devido ao sucesso do CBERS-1, em 2002 Brasil e China

assinam um novo acordo de cooperação para o desenvolvimento de mais dois satélites, os CBERS-3 e 4, tendo uma participação brasileira de 50%. Em 2003 é lançado com sucesso o satélite CBERS-2, também da base chinesa de Taiyuan.

Após passar por problemas funcionais, operando 2 anos além de sua vida útil estimada, o satélite CBERS-2 é substituído pelo satélite CBERS-2B, uma cópia do satélite CBERS-2 com algumas modificações. Este foi lançado com sucesso em 19 de setembro de 2007, a partir da base de Taiyuan, na China. O lançamento dos satélites CBERS 3 e 4 está previsto para ocorrer em 2009 e 2011 respectivamente.

2.1 Satélites CBERS

O programa CBERS nasceu com a assinatura de um acordo bilateral entre Brasil e China, em 06 de julho de 1988, com o objetivo de desenvolver dois satélites avançados de sensoriamento remoto de médio porte, batizado de Satélite Sino-Brasileiro de Recursos Terrestres (*China Brazil Earth Resourses Satellite* - CBERS).

O resultado direto deste acordo foi o desenvolvimento de 2 satélites, o CBERS-1 e 2, lançados com sucesso em 14 de outubro de 1999 e 21 de outubro de 2003 respectivamente. A participação do Brasil foi de 30% dos custos totais do projeto, incluindo os custos de lançamento.

Os satélites CBERS-1 e 2 são compostos por dois módulos: o módulo de carga útil e o módulo de serviço. O módulo de carga útil acomoda os sistemas ópticos (Câmera Imageadora de Alta Resolução - HRCCD, Imageador por Varredura de Média Resolução - IRMSS e Câmera Imageadora de Amplo Campo de Visada - WFI) utilizadas para observação da Terra e um repetidor para o Sistema Brasileiro de Coleta de Dados Ambientais. Já o módulo de serviço

contém os equipamentos que asseguram o suprimento de energia, os controles, as telecomunicações e demais funções necessárias à operação do satélite. O satélite CBERS-2B é muito semelhante aos CBERS-1 e 2, mas o IRMSS foi substituído pela Câmera Pancromática de Alta Resolução (HRC). A Figura 2-2 apresenta uma ilustração do satélite CBERS 2B.

Figura 2-2. Ilustração do satélite CBERS 2B.
Fonte: INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (2007a).

Para cumprir os rigorosos requisitos de apontamento das câmeras necessários à obtenção de imagens de alta resolução, o satélite dispõe de um preciso sistema de controle de atitude. No caso do CBERS-2B houve uma melhoria significativa pela instalação de um receptor de Posicionamento Global (*Global Positioning System* - GPS) e de um sensor de estrelas para assistir os mecanismos de controle de atitude. Esse sistema é complementado por um conjunto de propulsores a hidrazina que também auxilia nas eventuais manobras de correção da órbita nominal do satélite.

Os dados internos para monitoramento do estado de funcionamento do satélite são coletados e processados por um sistema distribuído de computadores antes de serem transmitidos à Terra. Um sistema de controle térmico ativo e

passivo provê o ambiente apropriado para o funcionamento dos sofisticados equipamentos do satélite.

As características mais relevantes dos sistemas ópticos utilizados são apresentadas na Tabela 1.

Tabela 1. Características das câmeras do CBERS-1 e 2

Câmera	Comprimento de onda	Região	Campo de visada	Resolução espacial	Largura de faixa imageada	Apontamento do espelho	Tempo de revisita
WFI	0,63-0,69μm 0,77-0,89μm	R NIR	60°	260x260m	890Km	Não permitido	5 dias
HRCCD	0,51-0,73 μm 0,45-0,52 μm 0,52-0,59 μm 0,63-0,69 μm 0,77-0,89 μm	PAN B G R NIR	8,3°	20x20m	113Km	±32°	26 dias (3 dias – visada lateral)
IRMSS	0,50-1,10 μm 1,55-1,75 μm 2,08-2,35 μm 10,40-12,50 μm	PAN MIR SWIR TH	8,8°	80x80m (160x160m - TH)	120Km	Não permitido	26 dias

PAN-Pancromática; B-Azul; G-Verde; R-Vermelho; NIR-Infravermelho próximo; MIR-Infravermelho médio; SWIR-Infravermelho de ondas curtas; TH-Infravermelho termal.
Fonte: Epiphanio (2005).

No programa CBERS-3 e 4 o Brasil assumirá 50% dos custos totais do projeto, incluindo os custos de lançamento. A Figura 2-3 apresenta uma ilustração do satélite CBERS-3 e 4.

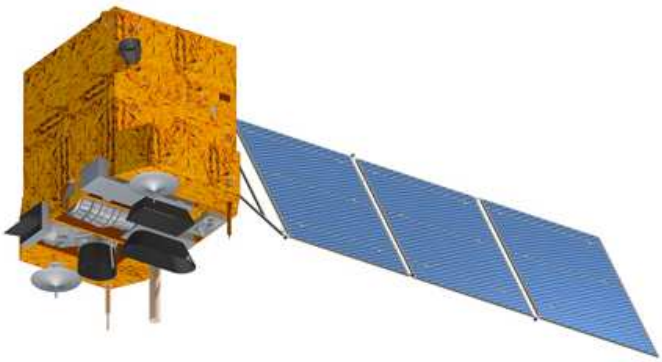


Figura 2-3. Ilustração do satélite CBERS-3 e 4
Fonte: INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (2007a).

Os satélites CBERS-3 e 4 representam uma evolução dos satélites CBERS-1 e 2. Para os satélites CBERS-3 e 4, serão utilizadas no módulo de carga útil 4 câmeras (Câmera PanMux - PANMUX, Câmera Multi Espectral - MUXCAM,

Imageador por Varredura de Média Resolução – IRSCAM, e Câmera Imageadora de Amplo Campo de Visada – WFICAM) com desempenhos geométricos e radiométricos melhorados.

A órbita de todos os satélites do programa CBERS é a mesma. Sua órbita é heliossíncrona a uma altitude de 778 km, perfazendo aproximadamente 14 revoluções por dia. Nesta órbita, o satélite cruza a linha do Equador sempre na mesma hora local, 10h30min da manhã, permitindo assim que se tenham sempre as mesmas condições de iluminação solar para a comparação de imagens tomadas em dias diferentes.

2.2 Câmera Multi Espectral (MUXCAM)

A Câmera Multi Espectral (MUXCAM) será uma atualização da Câmera de Alta Resolução (HRCCD) dos satélites CBERS-1 e 2. As principais características da MUXCAM são apresentadas na Tabela 2.

Tabela 2. Características Câmera Multi Espectral (MUXCAM)

Bandas Espectrais	B05: 0,45 - 0,52 µm; B06: 0,52 - 0,59 µm B07: 0,63 - 0,69 µm B08: 0,77 - 0,89 µm
Largura da faixa imageada	120Km
Resolução espacial	20m
Visada lateral de espelho	32º
Taxa bruta de dados	65MHz

Fonte: INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (2007c).

A Câmera Imageadora de Alta Resolução (HRCCD) dos satélites CBERS-1 e 2, foi totalmente fabricada na China e por questões estratégicas, o INPE decidiu que sua substituta, a MUXCAM seria totalmente fabricada no Brasil, um feito inédito. Através de uma licitação pública, a empresa Opto Eletrônica S/A foi escolhida para o desenvolvimento da MUXCAM no Brasil.

A MUXCAM é dividida em três equipamentos com funções específicas:

- Equipamento RBNA: é composto pelo conjunto óptico, sistema de ajuste de foco, sistema de calibração, dispositivo de carga acoplada (*Charge Coupled Device* – CCD), eletrônica de acionamento e controle térmico;
- Equipamento RBNB: é responsável pelo controle térmico de todo o equipamento RBNA, pelo controle do sistema de calibração e sistema de ajuste de foco;
- Equipamento RBNC: é responsável pela geração dos sinais necessários para o funcionamento do CCD e pela formatação dos dados de saída, enviados ao Gravador de Dados Digital (*Digital Data Recorder* - DDR).

O desenho em corte e uma foto do equipamento RBNA é apresentada na Figura 2-4.

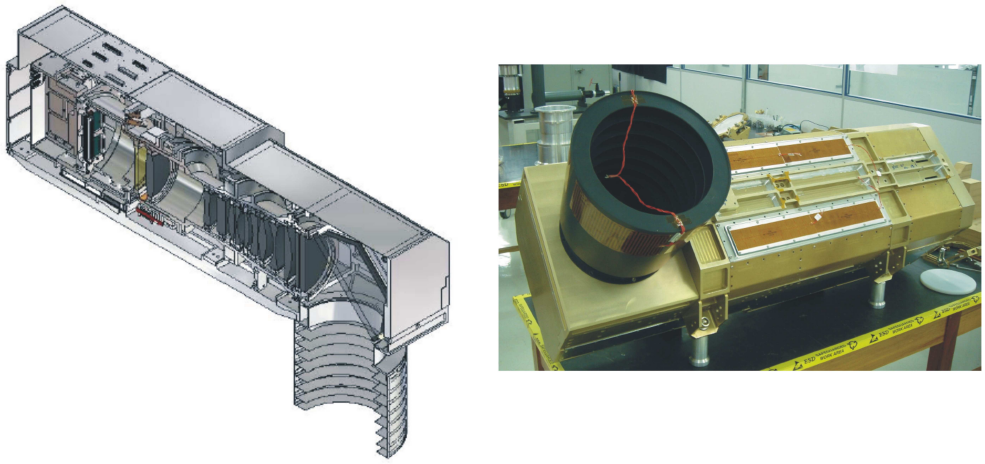


Figura 2-4. Desenho em corte e foto da Câmera Multi-Espectral MUXCAM.
Fonte: OPTO ELETRÔNICA S/A (2007a).

O desenho do equipamento RBNB é apresentado na Figura 2-5 e o desenho do equipamento RBNC é apresentado na Figura 2-6.

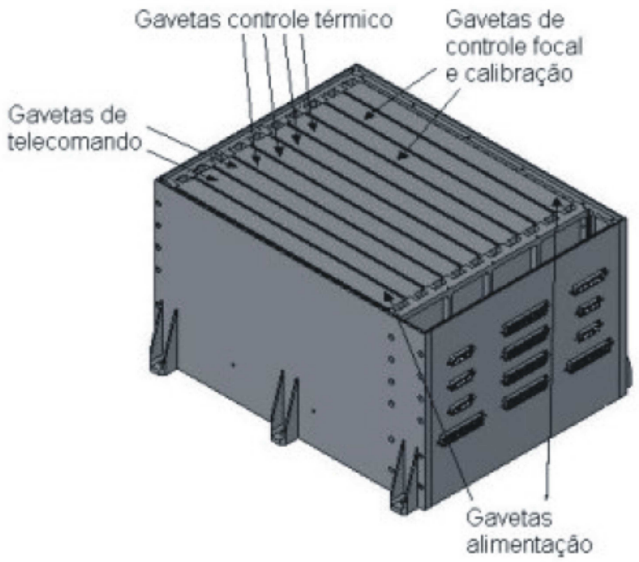


Figura 2-5. Desenho do equipamento RBNB.
Fonte: OPTO ELETRÔNICA S/A (2007a).

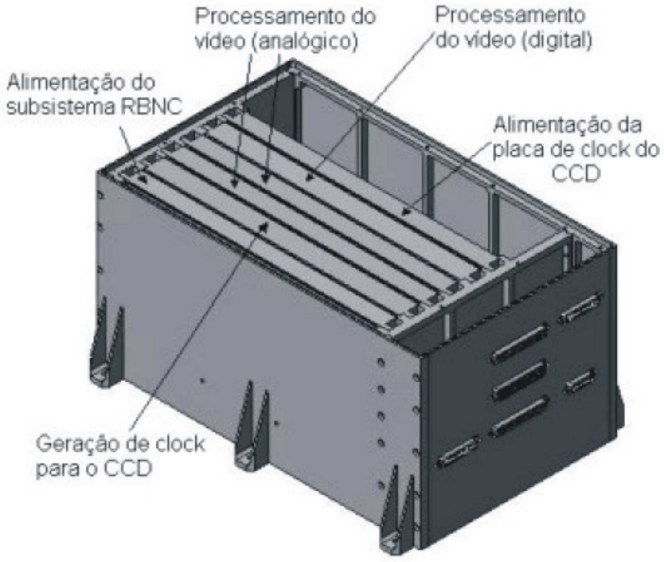


Figura 2-6. Desenho do equipamento RBNC.
Fonte: OPTO ELETRÔNICA S/A (2007a).

Na MUXCAM, o CCD é responsável pela captação da imagem, sendo composto por 4 linhas paralelas com 6034 *pixels*, sendo apenas 6000 *pixels* ativos para obtenção de imagem, com a dimensão de 13µm. Os *pixels* restantes não são ópticos e sim referências elétricas de nível de preto. A rotação terrestre completa o

processo de captação da imagem, realizando a varredura no sentido perpendicular as linhas do CCD.

Todo o sistema de geração de *clocks* do CCD e o processamento de vídeo estão divididos em placas eletrônicas distintas. A placa de geração de *clocks* do CCD é chamada de RBNC-ELCK e a placa responsável pelo processamento de vídeo é chamada de RBNC-ELVD. A formação das siglas utilizadas por cada equipamento é descrita pela árvore de produto, cujo nível um é apresentado na Figura 2-7. A MUXCAM é tratada pelo INPE como um subsistema do satélite, e recebe a sigla RBN. Como a MUXCAM é composta por três equipamentos, a Opto Eletrônica S/A adotou a nomenclatura incremental alfabética, designando os equipamentos de RBNA, RBNB e RBNC, sempre iniciando com a sigla RBN.

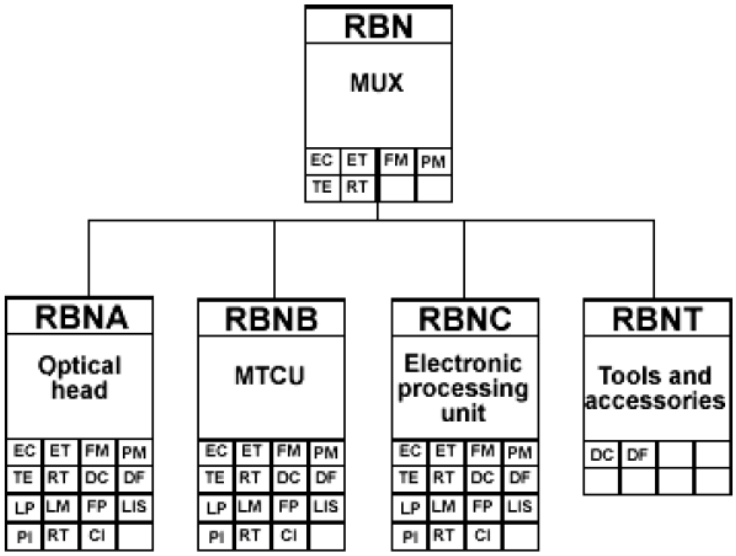


Figura 2-7. Nível um da árvore de produto da MUXCAM.
Fonte: OPTO ELETRÔNICA S/A (2007a).

Todos os equipamentos da MUXCAM possuem telemetrias e telecomandos com o intuito de permitir ações e verificações dos equipamentos durante o vôo orbital.

Todo o processamento de vídeo, a geração de sinais e algumas telemetrias e telecomandos são realizados em hardware reconfigurável do tipo *Field Programmable Gate Array* (FPGA).

3 *Hardware* Reconfigurável

Hardware reconfigurável, também chamado de FPGA, é um semicondutor que possui circuitos lógicos programáveis, conhecidos como blocos lógicos ou células lógicas, e interconexões programáveis. As arquiteturas dos blocos lógicos e as tecnologias de interconexão diferem bastante entre fabricantes.

A utilização de FPGAs se justifica devido ao baixo custo de prototipação, alta flexibilidade e pequeno tempo de desenvolvimento (ROSE, 1993). A FPGA foi introduzido em 1985 pela empresa norte americana Xilinx Inc, fundada em 1984. Logo em seguida, surgiram diversos tipos de FPGA lançadas por várias empresas do mesmo ramo, tais como Actel, Altera, AMD, Quicklogic, Algotronix, Atmel, entre outras (Virtual Computer Corporation, 1997).

FPGAs vem sendo utilizadas em espaçonaves há 10 anos. Até hoje, cerca de 100 lançadores e 300 satélites utilizam FPGA, como por exemplo, o satélite *Echo Star*, a ISS, o telescópio *Hubble*, o robô *Spirit*, entre outros. No projeto de espaçonaves modernas são utilizadas cerca de 40 FPGAs embarcadas (NASA, 2005). No Brasil, a aplicação de FPGAs em satélites desenvolvidos em território nacional é inédita.

Em aplicações espaciais as FPGAs são utilizadas em sistemas de navegação, câmeras, telecomunicações, controle de motores e em sistemas de pouso (NASA, 2005). Devido ao alto custo de desenvolvimento de Circuitos Integrados de Aplicação Específica (*Application Specific Integrated Circuits* – ASIC*)*, a utilização de dispositivos reconfiguráveis está se perpetuando (NASA, 2005).

A seguir, descrevemos as estruturas básicas das FPGAs fabricadas pelas empresas Quicklogic (2007a) e Aeroflex (2007b), e que estão sendo utilizadas no ciclo de desenvolvimento da MUXCAM.

3.1 Estrutura básica QuickLogic

A empresa Quicklogic é fabricante de FPGAs para uso comercial e militar, não possuindo componentes para uso espacial, cuja principal característica é a presença de proteção contra radiação. A empresa Quicklogic, licenciou juntamente com a Aeroflex a fabricação de FPGAs para uso espacial através de uma licença do tipo *Embedded Standard Products* (ESP) (AEROFLEX, 2008). Através desta licença, a empresa Aeroflex pode utilizar um núcleo comercial ou militar da empresa Quicklogic para a fabricação de componentes espaciais *Radiation Hardened* (*RadHard*). Com isto, os componentes comerciais e militares da Quicklogic passam a ser as ferramentas mais adequadas para a fase de desenvolvimento, pois possuem o mesmo núcleo dos componentes de vôo, com um custo muito menor.

A família Eclipse foi a escolhida como base para a fabricação dos componentes de vôo da Aeroflex, mais especificamente o modelo militar QL6325.

3.1.1 Eclipse QL6325

O componente QL6325 é fabricado em tecnologia CMOS de 0.25μm, com cinco camadas de metal. A família Eclipse possui uma grande variedade de pinos de E/S, com o intuito de maximizar o desempenho, a funcionalidade e a flexibilidade. Todos os pinos de E/S são tolerantes a 2,5V e 3,3V, exceto alguns pinos dedicados que suportam apenas 2,5V, como exemplo as entradas de *clock*. A Tabela 3 apresenta os padrões suportados pelos pinos de E/S e suas aplicações.

Tabela 3. Padrões de E/S Quicklogic

Padrão E/S	Tensão de referência	Tensão de saída	Aplicação
LVTTL	Não aplicável	3,3V	Aplicações gerais
LVCMOS	Não aplicável	2,5V	Aplicações gerais
PCI	Não aplicável	3,3V	Barramento PCI
GTL+	1.0V	Não aplicável	*Backplane*
SSTL3	1,5V	3,3V	SDRAM
SSTL2	1,25V	2,5V	SDRAM

Fonte: QUICKLOGIC (2007a).

A Figura 3-1 mostra o diagrama de blocos interno da família Eclipse. Como se pode verificar, ela possui quatro blocos de *Phase Locked Loop* (PLL) e possui também dois blocos de memória RAM embarcados.

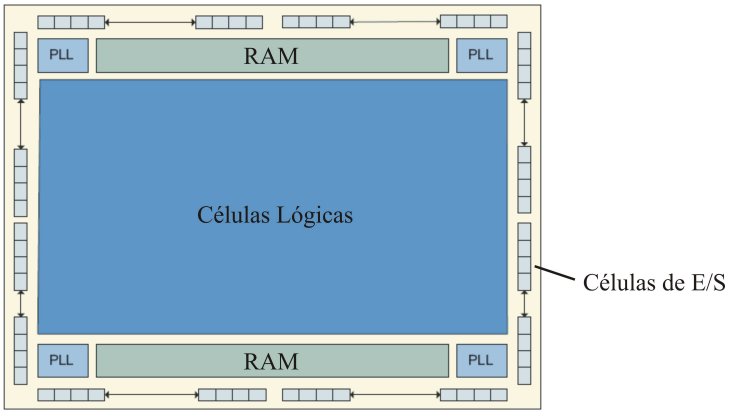


Figura 3-1. Diagrama de blocos interno - família Eclipse.
Fonte: QUICKLOGIC (2007a).

As células lógicas estão distribuídas no centro do componente. A estrutura da célula lógica é apresentada na Figura 3-2 e a estrutura da célula de E/S é apresentada na Figura 3-3.

A programação da família Eclipse é realizada com a tecnologia *ViaLink*®, uma tecnologia patenteada da Quicklogic (2007b). A Figura 3-4 apresenta uma visão geral da tecnologia *ViaLink*®.

Os *ViaLinks* são eletricamente programados, permitindo a programação do componente rapidamente e de forma permanente.

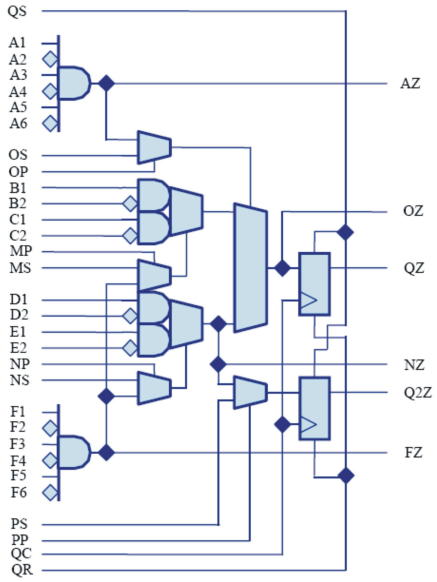


Figura 3-2. Célula lógica Quicklogic.
Fonte: QUICKLOGIC (2007a).

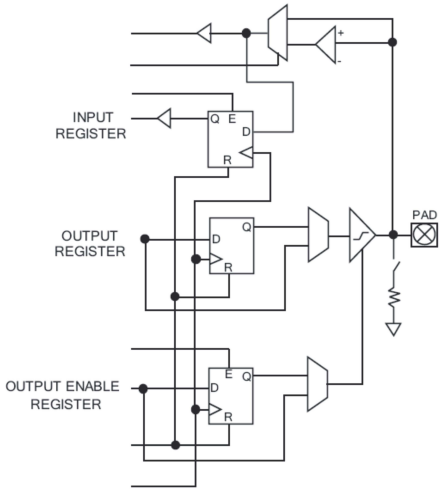


Figura 3-3. Célula de E/S Quicklogic.
Fonte: QUICKLOGIC (2007a).

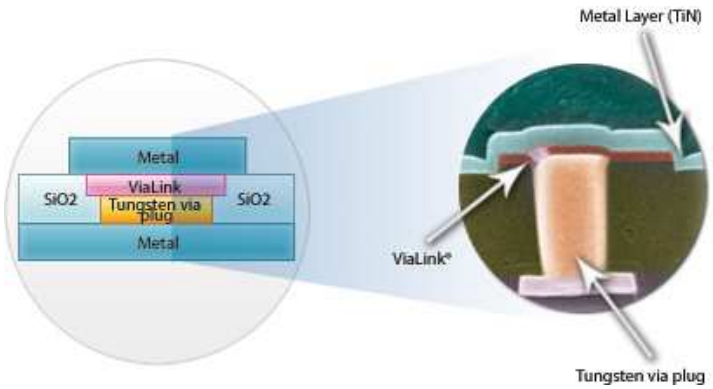


Figura 3-4. Tecnologia *ViaLink*® Quicklogic.
Fonte: QUICKLOGIC (2007b).

A tecnologia *ViaLink*® é baseada no padrão de interconexão utilizada em componentes ASIC. As vantagens apresentadas pelo fabricante com a utilização de *ViaLinks* são:

- Baixo consumo;
- Tamanho físico de interconexão pequeno;
- Programação permanente (não volátil).

A tecnologia de programação permanente é amplamente utilizada em aplicações aeroespaciais, devido ao longo período de retenção de dados apresentada por esta tecnologia.

3.2 Estrutura básica Aeroflex

A empresa Aeroflex é fabricante de componentes eletrônicos militares e espaciais. Seus componentes são utilizados como soluções em aplicações críticas. Seus produtos estão embarcados em vários projetos espacias, como exemplo na sonda Cassini, na Estação Espacial Internacional (*International Space Station* - ISS) entre outros (AEROFLEX, 2007a).

3.2.1 Eclipse UT6325

A família Eclipse *RadHard* UT6325 possui 320.000 portas lógicas, incluindo módulos *Dual-Port RadHard SRAM*. Esta família é fabricada em tecnologia CMOS de 0.25µm com 5 camadas de metal *ViaLink*®, contendo um máximo de 1.536 células lógicas e 24 módulos de *Dual-Port RadHard SRAM*, sobre uma arquitetura *RadHard.* Cada módulo de *RAM* possui 2304 bits para um conjunto máximo de 55.300 bits. (AEROFLEX, 2007b).

A família Eclipse *RadHard* UT6325 é fornecida em diferentes encapsulamentos, sendo o *Ceramic Quad Flat Pack* (CQFP) de 208 pinos

escolhida pela Opto Eletrônica S/A, permitindo o acesso à 99 pinos de E/S, 1 entrada exclusiva para *clock*, 8 entradas programáveis de *clock* e 16 entradas *high drive*.

Todas as células lógicas e as células de E/S são iguais às células utilizadas na família Eclipse da Quicklogic (2007a), descritas no item 3.1.1, inclusive suas temporizações. (AEROFLEX, 2007b).

3.3 Linguagem VHDL

VHDL é uma linguagem de programação padrão, aceita na indústria, utilizada para modelar e descrever o funcionamento de *hardware* do nível abstrato até o nível concreto. VHDL é a sigla para *VHSIC Hardware Description Language*, sendo que VHSIC é uma outra sigla, que significa *Very High Speed Integrated Circuits* (PERRY, 1994).

As linguagens para descrição de *hardware* ou *Hardware Description Language* (HDL) podem ser empregadas de diversas formas. A primeira forma é uma alternativa para a representação de diagramas de circuitos digitais e a segunda forma é a de um algoritmo de alto nível para resolver um determinado problema. Esses dois tipos de representação, estrutural (diagrama) e comportamental (algoritmo), representam duas maneiras de se descrever um modelo de um sistema digital.

A linguagem VHDL pode ser utilizada também para documentação, verificação e síntese de grandes sistemas digitais, assim como as linguagens Verilog e SystemC. Em VHDL, o mesmo código pode atingir todos os propósitos mencionados anteriormente, reduzindo, em grande parte, o esforço do projetista e

minimizando a introdução de erros durante tradução das especificações do projeto para sua implementação.

3.3.1 Histórico

A linguagem VHDL foi desenvolvida como parte do programa *Very High Speed Integrated Circuits* (VHSIC), iniciado pelo departamento de defesa dos Estados Unidos no final dos anos 70 e começo dos anos 80. O objetivo do programa VHSIC era produzir a nova geração de circuitos integrados, utilizando-se os limites da tecnologia em cada etapa do processo de projeto e fabricação. Os objetivos do programa foram atingidos com sucesso, porém, no processo de desenvolvimento de circuitos integrados extremamente complexos, os projetistas notaram que as ferramentas utilizadas para desenvolver grandes sistemas eram inadequadas. As ferramentas disponíveis para os projetistas na época, operavam, em sua maioria, em nível de portas lógicas e por este motivo, o projeto de sistemas com centenas de milhares de portas era uma tarefa complexa, exigindo, portanto, um novo método de descrição.

Para sanar este problema, em 1983, uma equipe de engenheiros da IBM, Texas Instruments e Intermetrics foi contratada pelo departamento de defesa dos Estados Unidos para completar a especificação e a implementação de um novo método de descrição de projeto baseado em linguagem. Em 1985 foi lançada a primeira versão pública da linguagem de descrição, chamada de VHDL. Essa linguagem tinha inicialmente dois objetivos principais: primeiro, os projetistas queriam uma linguagem que pudesse descrever circuitos complexos. Segundo, desejava-se uma linguagem que fosse padrão, para que todos os integrantes do programa VHSIC pudessem trocar projetos entre si de um modo padronizado (DOULOS, 2007).

Em 1986, os direitos da linguagem VHDL, foram transferidos ao *Institute of Electrical and Eletronic Engineers* (IEEE), sendo que em 1987 foi publicado o primeiro padrão IEEE VHDL-1076. Após sua primeira publicação, a linguagem passou por quatro revisões, realizadas nos anos de 1993, 2000, 2002 e 2006.

3.3.2 Vantagens e desvantagens

A utilização da linguagem VHDL traz algumas vantagens e uma desvantagem. As principais vantagens são:

- projeto independente de tecnologia;
- ferramentas de diferentes fabricantes compatíveis;
- modularização e reutilização de *software*;
- atualização de projetos facilitada;
- simulação compatível com comportamento real;
- redução do tempo para colocaçãode novos produtos no mercado.

A implementação em VHDL produz um código fonte independente de plataforma, até o processo de síntese. O processo de síntese é responsável pela conversão do software para portas lógicas, produzindo um circuito eletrônico que depende das características construtivas do *hardware* reconfigurável. A partir deste momento, a linguagem se torna dependente de tecnologia. Todas as ferramentas de desenvolvimento são compatíveis até o processo de síntese. As ferramentas de simulação são compatíveis entre si, garantindo assim uma grande flexibilidade durante o desenvolvimento.

A principal desvantagem da linguagem VHDL é a geração de *hardware* não otimizado, ou seja, toda a geração é realizada de forma automática pela ferramenta de síntese. Esta característica faz com que o processo de verificação e validação

do *hardware* gerado se torne muito trabalhoso, uma vez que o *hardware* gerado pode se tornar muito extenso. Esta desvantagem é apresentada em maiores detalhes no capítulo 4, onde analisamos a ferramenta de desenvolvimento.

3.4 Efeitos da radiação em circuitos digitais

Circuitos eletrônicos digitais presentes em ambiente espacial são afetados por partículas carregadas geradas principalmente pelos ventos solares.

Circuitos eletrônicos digitais de baixa tensão e baixo consumo de potência, necessitam de menos energia para o armazenamento de dados, tornando-se mais susceptíveis à radiação, onde partículas com uma pequena quantidade de carga podem produzir danos irreversíveis.

Segundo LABEL (1999), existem duas classes de efeitos da radiação que devem ser considerados quando da utilização de circuitos eletrônicos digitais de larga escala de integração (*Very Large Scale Integration* - VLSI) em projetos espaciais, sendo eles:

- *Total Ionizing Dose* (TID);
- *Single Event Effects* (SEE).

Total Ionizing Dose considera a degradação promovida no circuito eletrônico digital devido ao longo tempo de exposição à radiação. Esta degradação se dá devido ao acúmulo de partículas depositadas sobre o material, principalmente prótons e elétrons. As maiores fontes destas partículas estão relacionadas aos eventos solares, principalmente a Anomalia do Atlântico Sul (*South Atlantic Anomaly* - SAA) (NASA, 2008).

Devido à exposição dos circuitos eletrônicos digitais a estas partículas, podem ocorrer alterações de características, entre elas:

- aumento de níveis de *threshold*;

- aumento de correntes de polarização;
- aumento de consumo;
- alteração de atrasos internos;
- alteração de funcionalidade.

A blindagem destes circuitos eletrônicos digitais pode ajudar, mas há muitos fatores que devem ser levados em conta, entre eles a geometria da blindagem, o material e a composição química dos circuitos eletrônicos digitais. A blindagem de alumínio favorece a atenuação de elétrons e prótons de baixa energia, mas é ineficiente para prótons de alta energia (maior que 30MeV) (LABEL, 1999). O chumbo não é utilizado em blindagem pois produz radiação secundária (HOLMES-SIEDLE, 2002).

Single Event Efects são efeitos causados pela passagem de uma partícula carregada pelo silício. Uma partícula carregada passa pelo material, provoca uma ionização, gerando um pulso de corrente que pode ser destrutivo ou não. Segundo (O'BRYAN, 1998), existem várias fontes destas partículas, entre elas prótons solares, nêutrons e íons pesados dos raios cósmicos. Os íons pesados, presos pelo magnetismo da terra, não causam contribuições significativas ao TID, mas possuem energia suficiente para causar ionização, conseqüentemente, gerando SEE.

Os SEE podem ser divididos em três categorias, de acordo com os níveis de corrente gerados:

- SEE leves: a radiação provoca um transiente em um circuito eletrônico linear ou pode causar a alteração de bits em circuitos eletrônicos digitais. Estes efeitos não são permanentes, isto é, podem ser revertidos pela reprogramação ou *reset* do circuito eletrônico

digital. Um exemplo é o *Single Event Upset* (SEU), que altera os bits de uma memória, mas não de uma forma permanente.

- SEE pesados: a radiação provoca uma alteração de bits permanente em circuitos eletrônicos digitais.
- SEE destrutivo: a radiação provoca a destruição dos circuitos eletrônicos digitais. Alguns exemplos são: *Single Event Latch-up* (SEL), *Single Event Gate Rupture* (SEGR) e *Single Event Burnout* (SEB).

O erro de *software* mais comum é a ocorrência do SEL, que geralmente curto circuita a linha de alimentação com o terra do circuito eletrônico. Muitas vezes este efeito causa um dano permanente ao circuito. Caso a corrente máxima do circuito eletrônico digital ultrapasse o valor nominal durante um SEL, a destruição por efeito térmico pode ocorrer. A Tabela 4 apresenta os diferentes tipos de SEE classificados por circuitos eletrônicos e áreas sensíveis.

Tabela 4. S.E.E. em circuitos eletrônicos e áreas sensíveis.

Tipo de componente	**Área sensível**	**SEE**
Memórias	Células de memória	Inversão de Bit
	Controle Lógico	Inversão de Bit se seqüencial, transiente se combinacional
Lógica Combinacional	Lógica Combinacional	Transiente
Lógica Seqüencial	Lógica Seqüencial	Inversão de Bit
FPGAs	Lógica Combinacional	Transiente se combinacional, Inversão de Bit se *Lookup Tables*
	Lógica Seqüencial	Inversão de Bit
Microprocessadores	Registradores seqüenciais e controle lógico	Inversão de Bit
	Lógica Combinacional	Transiente
ADCs. DACs	Área analógica	Transiente
	Área digital	Transiente
Circuitos Lineares	Área analógica	Transiente
Fotodiodo	Fotodiodo	Transiente

Fonte: DENTAN (2000).

Em circuitos lógicos programáveis com programação do tipo *Anti-Fuse*, os circuitos lógicos e as linhas de interconexão podem ser considerados imunes aos SEE, porém todos os *latches* e *flip-flops* são sensíveis, assim como circuitos lógicos programáveis com tecnologia SRAM (KATZ, 1998).

A principal solução utilizada para tornar um circuito eletrônico inume a SEE é a utilização da Redundância Modular Tripla (*Triplicate Modular Redundancy* - TMR), que consiste em realizar a triplicação de toda a memória do sistema juntamente com um circuito votante. Este circuito votante realiza a escolha do dado que será utilizado, baseado na comparação dos três valores de entrada. A Figura 3-5 apresenta um *flip-flop* do tipo D implementado em TMR.

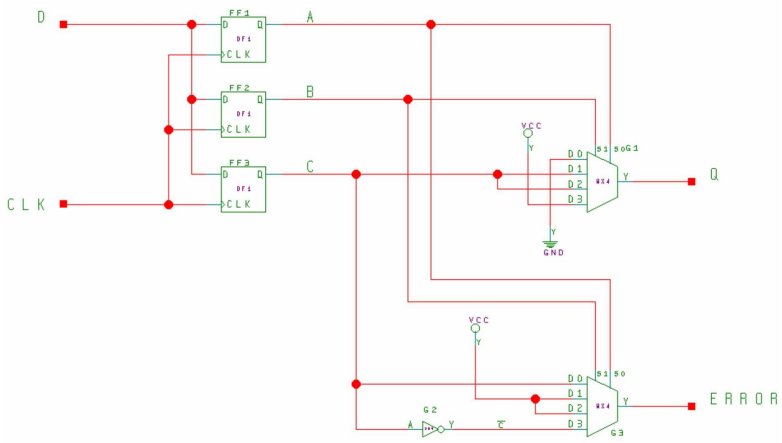


Figura 3-5. *Flip-flop* tipo D implementado em TMR.

Este método de mitigação é utilizado em circuitos ASIC e em hardware reconfigurável, como por exemplo a família de FPGA RTSX32SU, da Actel (2007). Este tipo de redundância exige um aumento de área, já que triplica todo o circuito eletrônico. Este tipo de redundância em hardware não é utilizado pelo fabricante Aeroflex.

Outro método utilizado para reduzir os efeitos da radiação em hardware reconfigurável, é a utilização de bibliotecas especiais que geram circuitos imunes

aos efeitos da radiação. Estas bibliotecas geram TMR via *software*, em linguagem VHDL, sem a necessidade de qualquer alteração em nível de circuito eletrônico.

A utilização de bibliotecas para a utilização de TMR em hardware reconfigurável que não possui tal tecnologia embarcada é prejudicada pelo processo de síntese e otimização lógica. Como veremos no capítulo 4, a ferramenta de desenvolvimento promove alterações significativas e de difícil visualização na geração em nível de portas lógicas (*gate level)*. Desta forma, deve-se verificar todo o circuito implementado para comprovar a geração de TMR em nível de portas lógicas (*gate level).*

3.5 Conclusões

A linguagem VHDL é claramente uma das mais importantes linguagens de descrição de hardware. Como uma linguagem de projeto, a linguagem VHDL apresenta um mérito significante e em virtude de seu *status* de padrão industrial, provê uma alternativa de risco muito menor se comparada às linguagens proprietárias. Um fator importante, neste contexto, é o rápido surgimento de uma grande variedade de ferramentas de produção que operam com a linguagem VHDL de vários distribuidores. Esta linguagem, porém apresenta problemas de otimização de hardware, promovidas pela ferramenta de síntese, como veremos nos capítulos seguintes.

Vê-se que existe a necessidade de se avaliar os efeitos causados pela radiação em circuitos eletrônicos digitais nos equipamentos que serão colocados em órbita. Existem diversas técnicas utilizadas para a diminuição da susceptibilidade dos circuitos eletrônicos digitais à radiação, dentre as quais podemos citar a técnica de TMR, utilizada para diminuir os efeitos dos SEU. A

utilização de bibliotecas já existentes para a geração de TMR é prejudicada pela ferramenta de síntese, que pode realizar alterações automáticas no circuito implementado, buscando otimização e eliminando circuitos idênticos.

A TID pode alterar parâmetros internos como por exemplo os atrasos e temporizações dos circuitos lógicos.

4 Ferramenta de desenvolvimento

Neste capítulo discutiremos as principais características da ferramenta de desenvolvimento QuickWorks™, da empresa Quicklogic (2007c), com o objetivo de levantar todas as características que possam influenciar no ciclo de desenvolvimento de hardware reconfigurável para aplicações aeroespaciais. Esta ferramenta foi escolhida pois não possui restrições comerciais relacionadas a projetos aeroespaciais. Neste capítulo também analisamos o comportamento da linguagem VHDL na ferramenta de síntese. O ciclo de desenvolvimento utilizado nas FPGAs comerciais e militares da Quicklogic é também utilizado nas FPGAs para aplicações aeroespaciais da empresa Aeroflex, desde o processo de síntese até o processo de gravação dos componentes. A gravação dos componentes comerciais, militares e espaciais é realizada em uma mesma plataforma de hardware, com a utilização de diferentes soquetes, uma vez que a tecnologia de gravação utilizada é a mesma para todos os componentes (AEROFLEX, 2007b).

A Figura 4-1 apresenta a janela principal da ferramenta QuickWorks™. A interface é simples e intuitiva, representando em forma de fluxograma todo o ciclo de desenvolvimento.

O processo de desenvolvimento se inicia pela simulação funcional (*Functional Simulation*) do *software* gerado em forma de esquemático ou através de linguagens de programação de alto nível como SystemC e VHDL. No caso da ferramenta QuickWorks™ escolhida para o desenvolvimento, o *software* deve ser criado em linguagem VHDL. A simulação funcional não utiliza informações de temporização, conseqüentemente apenas o funcionamento lógico do projeto pode ser verificado. Todas as simulações são realizadas no *software* ModelSim da

empresa Mentor Graphics (2008a), que faz parte do conjunto de ferramentas da plataforma QuickWorks™.

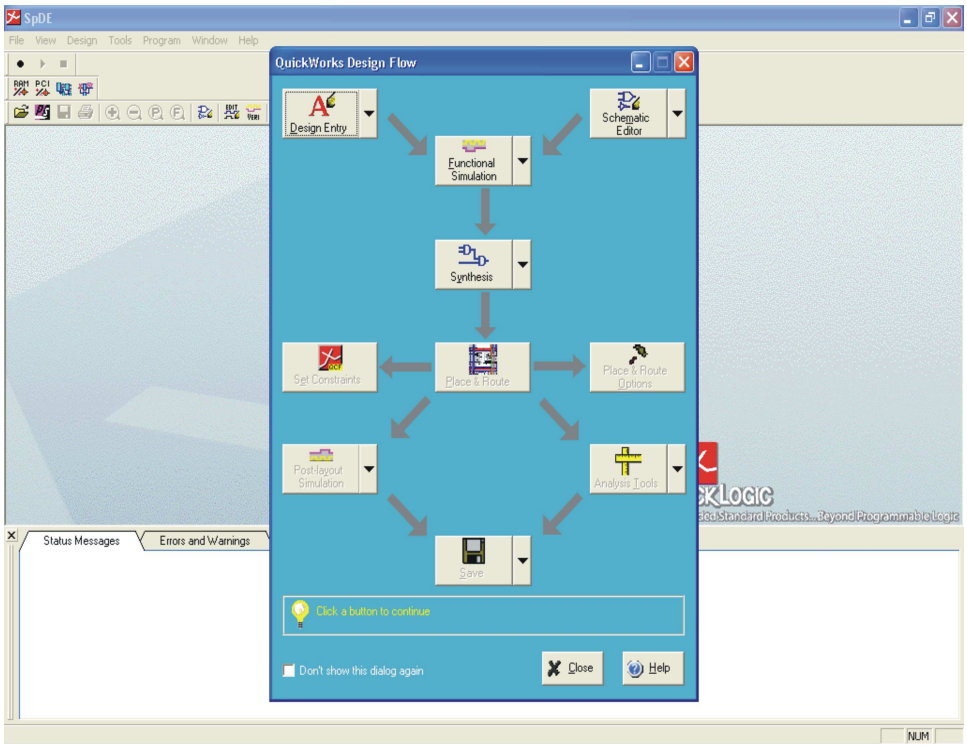


Figura 4-1. Janela principal da ferramenta QuickWorks™.
Fonte: QUICKLOGIC (2007c).

O próximo passo no ciclo de desenvolvimento é o processo de síntese (*Synthesis*), realizado pelo *software* Precision RTL Synthesis™ da empresa Mentor Graphics (2008b), também presente na ferramenta QuickWorks™. O processo de síntese é responsável pela transformação do *software* escrito em linguagem de alto nível para um nível intermediário, também chamado de *Register Transfer Level* (RTL) e finalmente, para uma representação em mais baixo nível, em nível de portas lógicas (*gate level)*. As linguagens de programação que são sintetizáveis geralmente são chamadas de linguagem RTL.

A representação em nível de portas lógicas (*gate level)* é composta por elementos digitais básicos presentes nas FPGAs e que são dependentes da tecnologia e arquitetura utilizadas. Devido a esta dependência, faz-se necessário a

inclusão das características do componente a ser utilizado antes do processo de síntese. No caso da Quicklogic, é permitido ao usuário a escolha entre diversos modelos de FPGAs. Verificamos, porém, que não existe a opção de componentes da empresa Aeroflex, evidenciando assim que o processo de síntese utilizado para componentes comerciais, militares e aeroespaciais são os mesmos, para todos os componentes fabricados pela empresas Quicklogic e Aeroflex. Como o *core* dos dois fabricantes é idêntico, conforme citado no item 3.2, o processo de síntese não é alterado. A Figura 4-2 mostra um detalhe da tela de escolha de componentes da ferramenta de síntese Precision RTL Synthesis™.

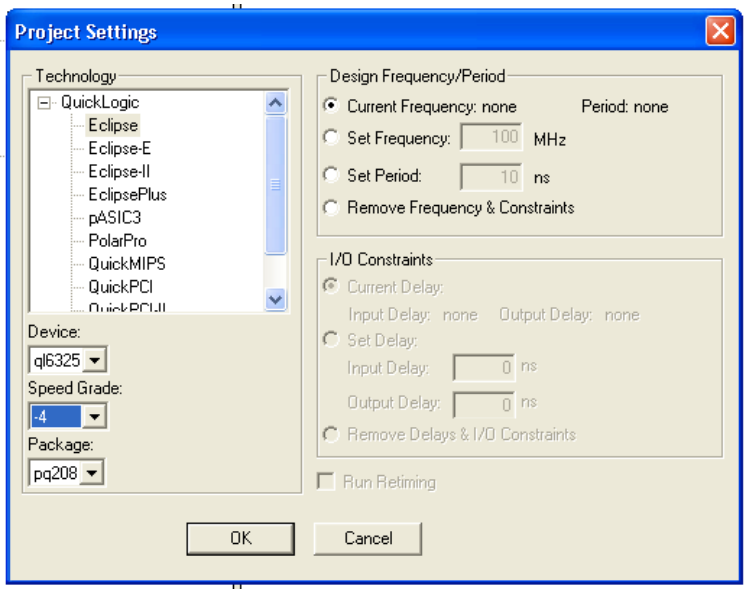


Figura 4-2. Detalhe da ferramenta de síntese Precision RTL Synthesis™.
Fonte: QUICKLOGIC (2007c).

Durante o processo de síntese são criados vários arquivos, dentre eles um arquivo *NetList*, que lista todas as interconexões entre todos os elementos básicos colocados em nível de portas lógicas (*gate level*). A geração desta lista é realizada de forma automatizada, não permitindo ao usuário qualquer interação. As nomenclaturas utilizadas internamente em nível de portas lógicas (*gate level)* não armazenam informações sobre o projeto escrito em alto nível, alterando *labels* e

nomes internos, dificultando assim sua visualização. Esta característica também está presente em outras ferramentas de síntese (BERG, 2004).

A Figura 4-3 apresenta parte de um projeto após o processo de síntese, composto de elementos lógicos básicos e linhas de interconexão. Nesta etapa, todo o esquemático do circuito a ser implementado é gerado. Verificamos que os nomes utilizados para as interconexões, por serem gerados automaticamente, possuem nomes extremamente diferentes. Este fato faz com que haja uma dificuldade muito grande durante o processo de verificação e validação do *software* implementado.

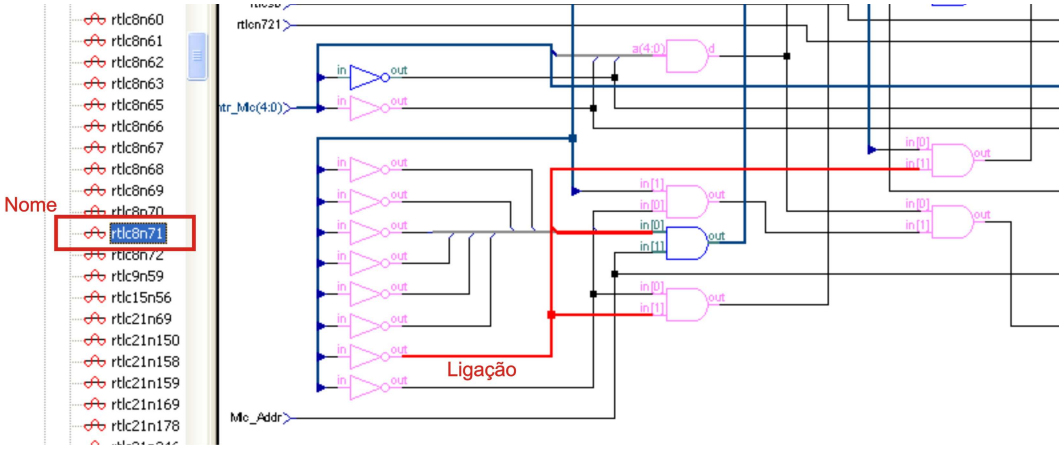


Figura 4-3. Parte de projeto após processo de síntese.
Fonte: QUICKLOGIC (2007c).

As aplicações aeroespaciais devem passar por processos de verificação e validação de *software* muito rígidos devido à alta confiabilidade exigida. São recomendados os processos de verificação e validação completa de todos os circuitos gerados em nível de portas lógicas (*gate level*), considerando principalmente as temporizações internas (KATZ, 2004). A utilização das ferramentas de análise, também fornecidas na plataforma QuickWorks™ fica prejudicada, devido à grande dificuldade de levantamento do esquemático em

nível de portas lógicas (*gate level*) gerado pelo processo de síntese, uma vez que estas ferramentas utilizam os nomes gerados automaticamente.

Após o processo de síntese, a ferramenta QuickWorks™ disponibiliza a ferramenta de *Place & Route*, responsável pela alocação dos circuitos lógicos em nível de portas lógicas (*gate level*) na matriz do hardware reconfigurável. A ferramenta disponibiliza um conjunto de opções ao usuário permitindo a alteração de parâmetros que podem influenciar no processo de alocação, conseqüentemente, influenciando no funcionamento do projeto final. A Figura 4-4 apresenta uma aba da janela de opções da ferramenta de *Place & Route* da ferramenta QuickWorks™. Este processo é realizado pela própria plataforma de desenvolvimento, não utilizando nenhum *software* externo, como o processo de simulação e síntese.

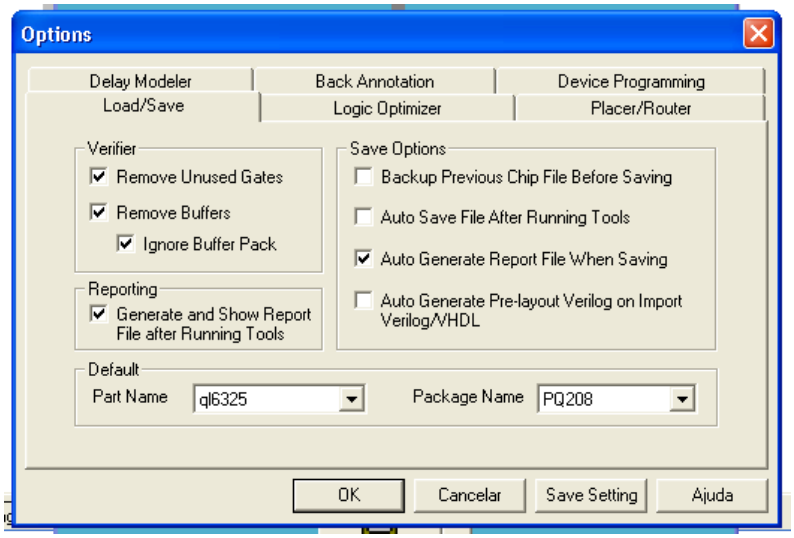


Figura 4-4. Janela de opções de *Place & Route.*
Fonte: QUICKLOGIC (2007c).

A Figura 4-5 apresenta a aba de opções do otimizador lógico utilizado no processo de *Place & Route.*

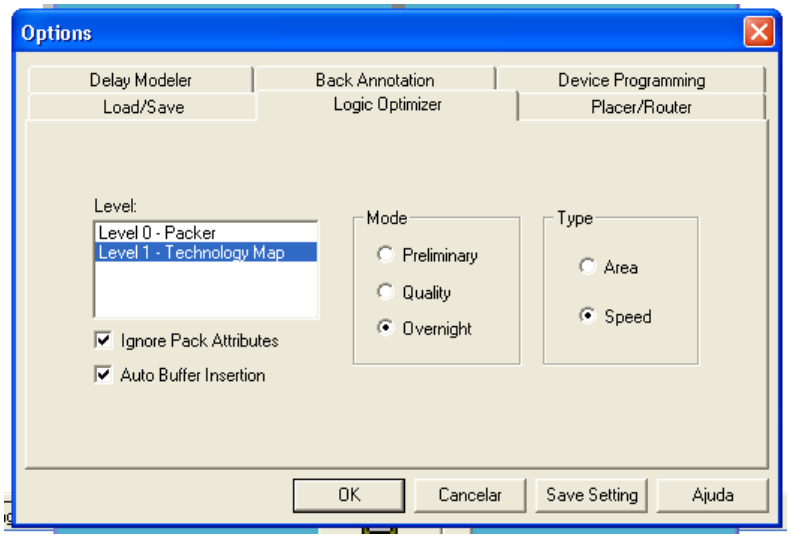


Figura 4-5. Opções do otimizador lógico.
Fonte: QUICKLOGIC (2007c).

A Figura 4-6 apresenta a aba de opções de modelagem de atraso utilizado no processo de *Place & Route* para estimar os atrasos reais para simulação posterior ao processo de síntese.

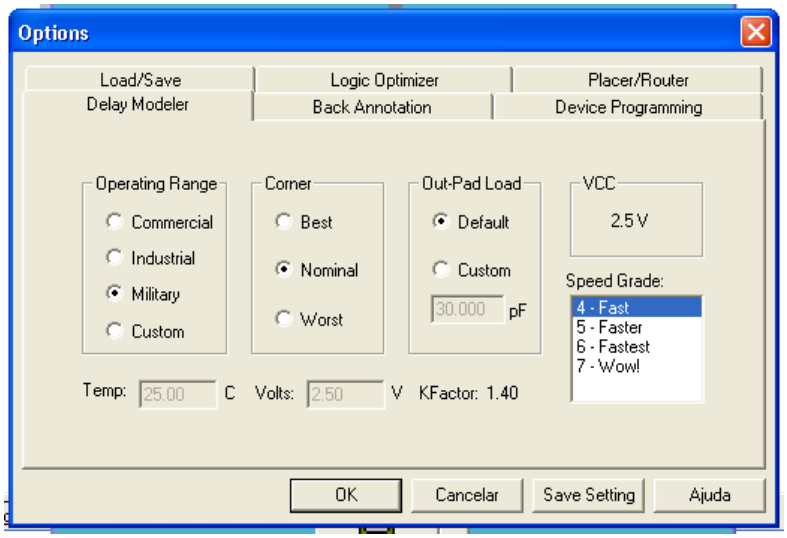


Figura 4-6. Opções de modelagem de atraso.
Fonte: QUICKLOGIC (2007c).

Além das opções relacionadas à ferramenta de *Place & Route*, existe a opção de criação de regras utilizadas para a criação do circuito lógico. As opções disponibilizadas ao usuário são:

- Posicionamento;
- Localização dos pinos de E/S;
- Definição de *pull-down* para cada pino de E/S;
- Definição de *slew rate* para cada pino de E/S;
- Definição de padrão elétrico para cada pino de E/S;
- Posicionamento por janelas;
- Regras de temporização.

A Figura 4-7 apresenta a janela de localização dos pinos de E/S. Nesta janela podemos verificar que o nome das *nets* são os mesmos nomes utilizados no projeto. Este fato permite a assimilação de um pino de E/S com uma *net* interna.

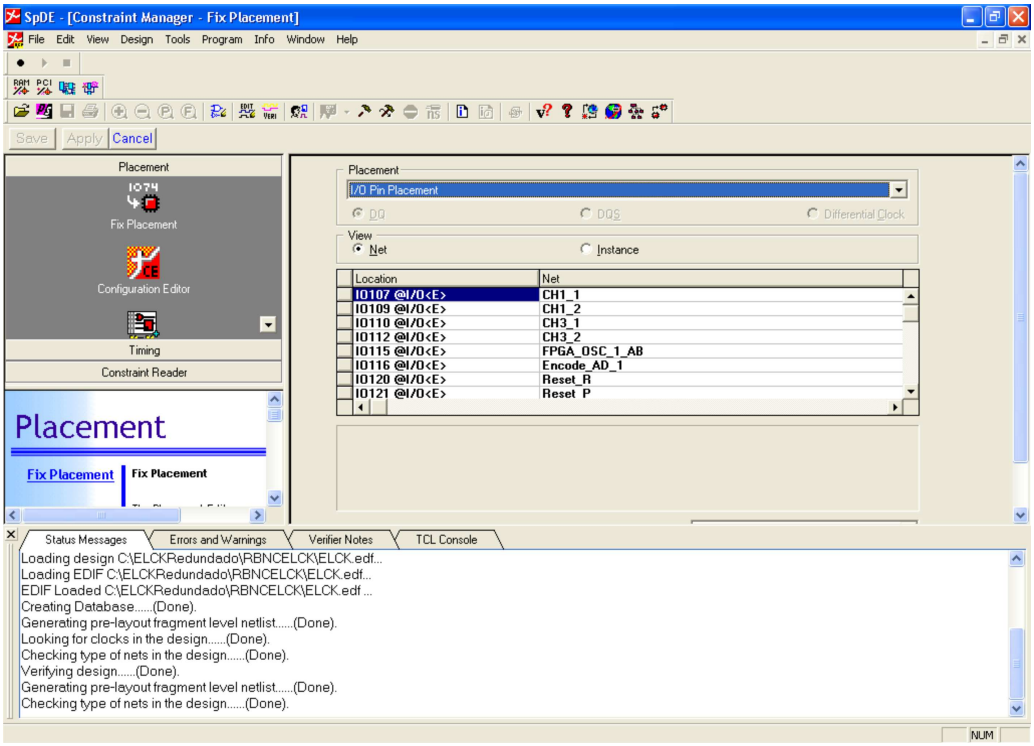


Figura 4-7. Janela de localização dos pinos de E/S.
Fonte: QUICKLOGIC (2007c).

A Figura 4-8 apresenta a janela de configuração na qual são escolhidos os valores de *pull-down*, *slew-rate* e padrão elétrico para cada pino de *E/S* independentemente.

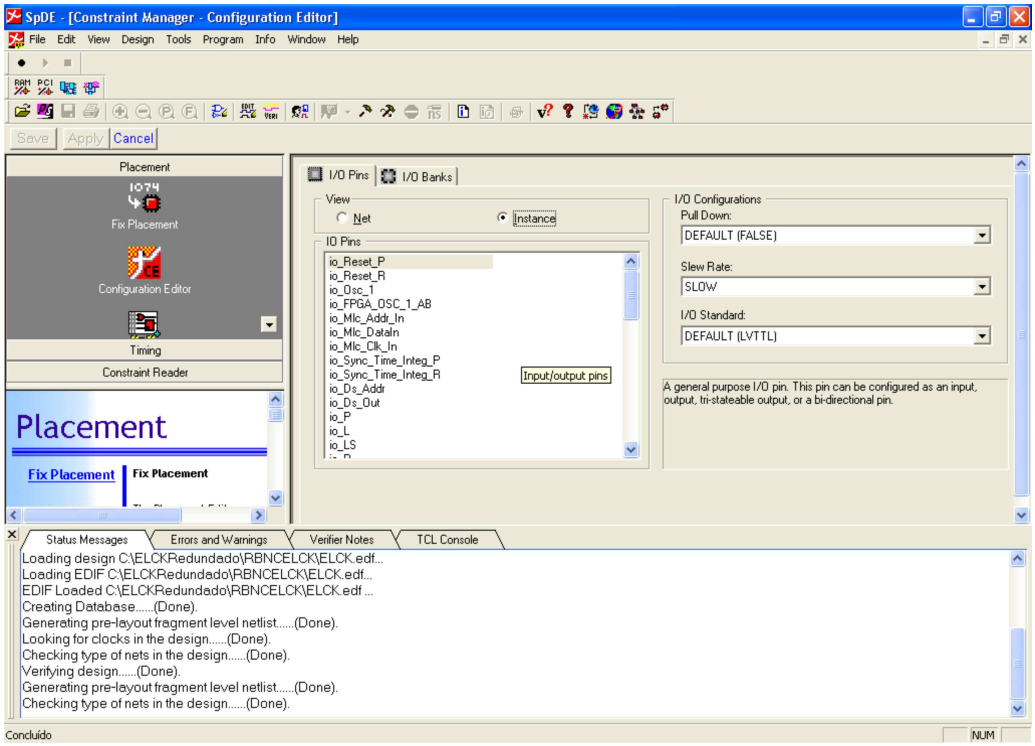


Figura 4-8. Janela de configuração de *pull-down*, *slew-rate* e padrão elétrico.
Fonte: QUICKLOGIC (2007c).

A Figura 4-9 apresenta a janela que permite o posicionamento na matriz da FPGA de células lógicas em janelas de tamanho e posição definidos pelo usuário. Esta opção permite a colocação de células lógicas em lugares específicos definidos pelo usuário, garantindo assim uma flexibilidade no posicionamento das células lógicas.

Podemos verificar nesta janela que as *nets* possuem uma nomenclatura diferente para cada conexão entre células lógicas, como ocorre na visualização em nível de portas lógicas (*gate level*). Esta característica prejudica em muito a utilização desta ferramenta de posicionamento de células lógicas, pois exige que o usuário conheça previamente todo o esquemático em nível de portas lógicas (*gate level*), para a colocação de células lógicas específicas em lugares específicos.

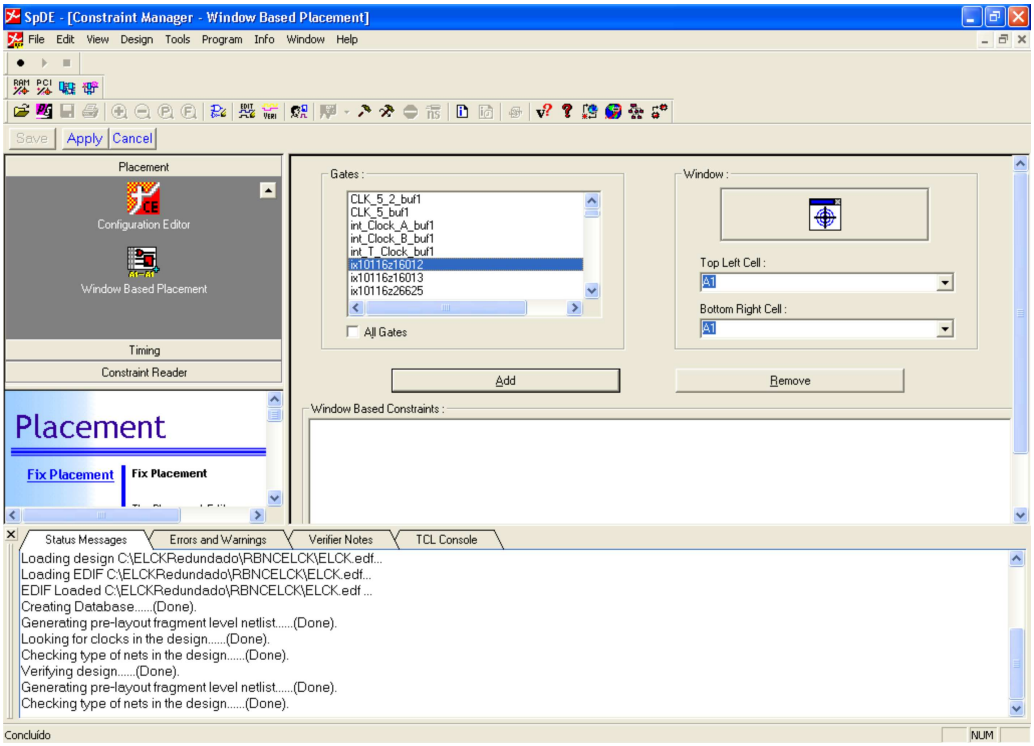


Figura 4-9. Janela de posicionamento de células lógicas.
Fonte: QUICKLOGIC (2007c).

A janela de regras de temporização é apresentada na Figura 4-10. Esta janela permite que sejam definidas regras de temporização entre sinais.

Após a realização do processo de *Place & Route*, é apresentado ao usuário uma visão global do *hardware* implementado. A Figura 4-11 apresenta uma visão geral de um projeto implementado. A ferramenta possui vários recursos visuais como por exemplo *zoom*, que permitem a investigação de todos os sinais, desde os pinos de E/S até as ligações internas das células lógicas.

A Figura 4-12 apresenta uma visão ampliada do projeto citado anteriormente. Como podemos visualizar, temos as informações de todas as ligações internas realizadas, assim como o posicionamento da célula lógica. Vemos que os nomes utilizados nas *nets* possuem uma nomenclatura diferente, seguindo os nomes gerados automaticamente durante a síntese.

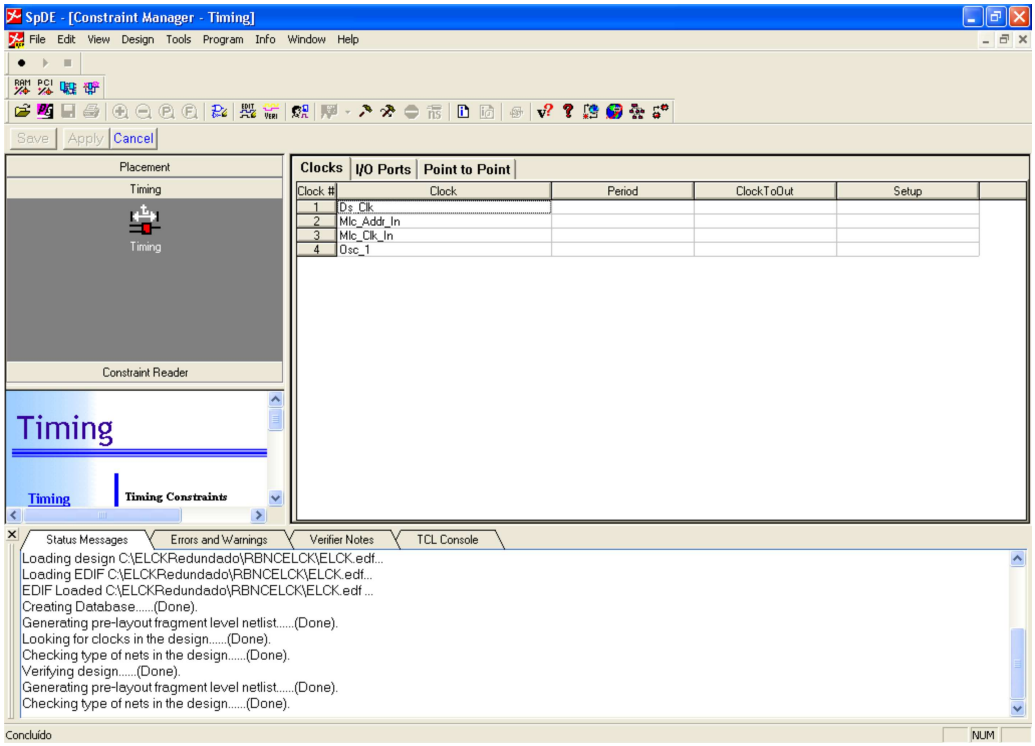


Figura 4-10. Janela de regras de temporização.
Fonte: QUICKLOGIC (2007c).

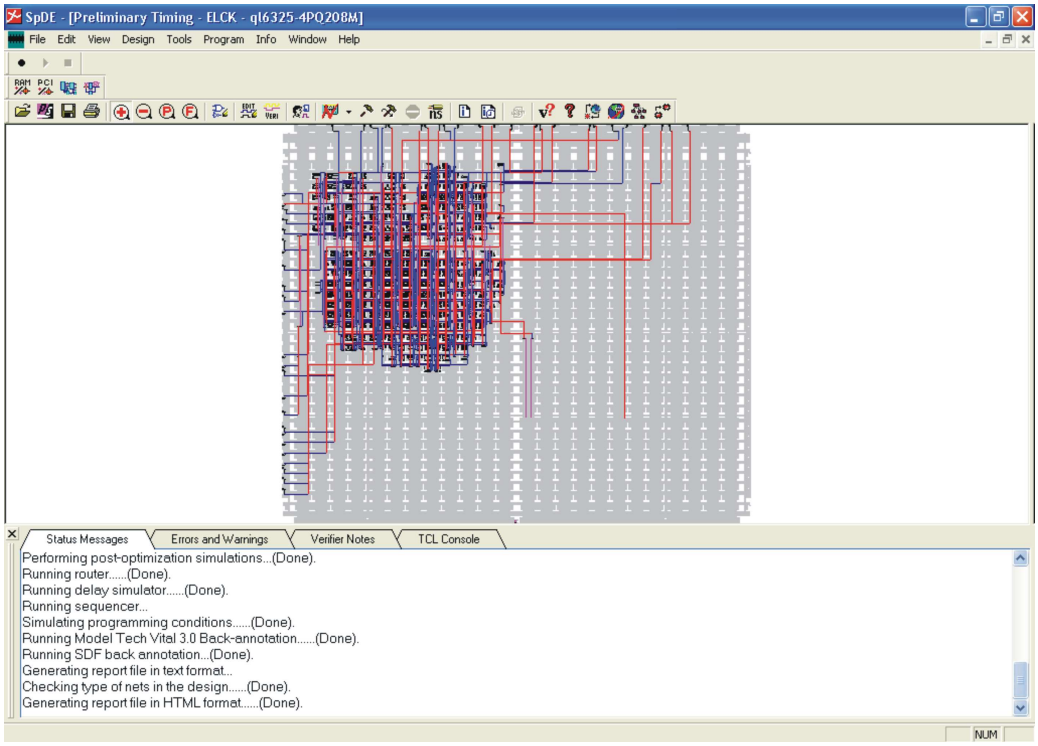


Figura 4-11. Visão geral do projeto implementado.
Fonte: QUICKLOGIC (2007c).

Durante o processo de *Place & Route* ainda pode ocorrer a inserção de novos componentes e *buffers*, em função da utilização dos otimizadores lógicos e das opções escolhidas para estes. Vemos que os otimizadores lógicos podem ser desabilitados na fase *Place & Route*, mas constatamos que os nomes das *nets* ainda permanecem diferentes, seguindo as definições geradas em nível de portas lógicas (*gate level*).

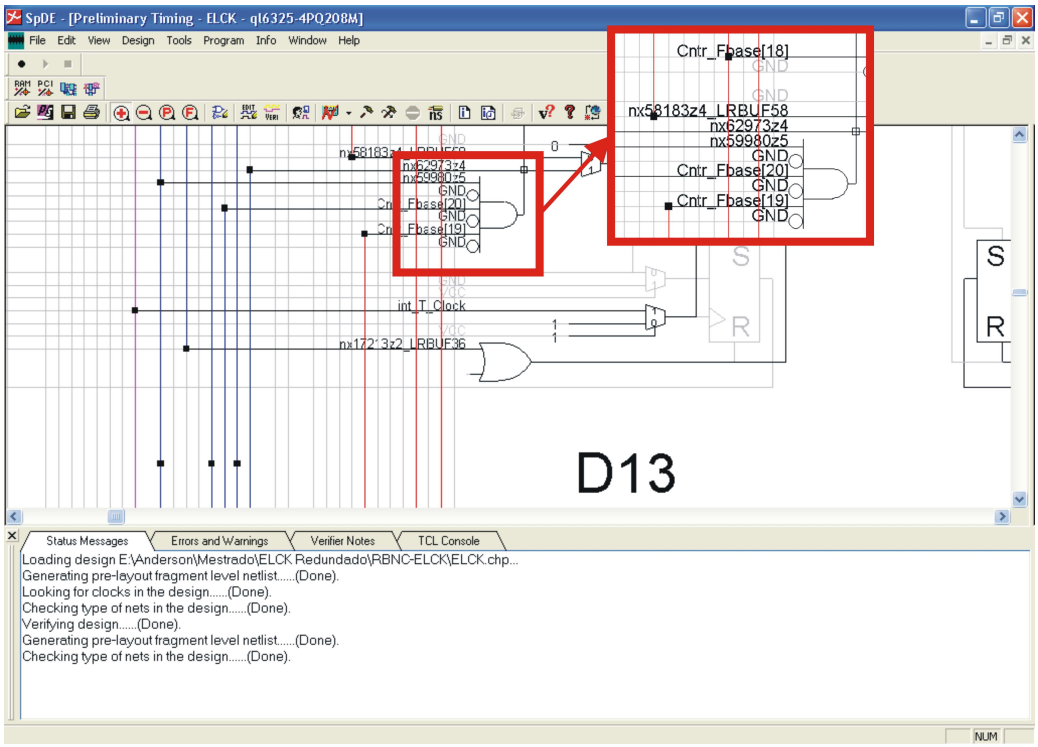


Figura 4-12. Visão ampliada do projeto implementado.
Fonte: QUICKLOGIC (2007c).

A ferramenta QuickWorks™ disponibiliza, após o processo de síntese, ferramentas de análise, entre elas aquelas para estimação de potência consumida, analisador de caminhos (*Path Analyzer*) e um relatório de temporização.

A ferramenta que realiza a análise de caminhos ou ligações, chamada *Path Analyzer*, é bastante útil e muito utilizada durante o ciclo de desenvolvimento, pois permite ao usuário a verificação das temporizações e dos caminhos de todos

os sinais internos. Com esta ferramenta é possível verificar e garantir os tempos combinacionais, *setup time*, *hold time*, *clock out to delay*, entre outros. Estas características são muito importantes para um projeto em hardware reconfigurável, pois elas definem as margens de segurança envolvidas no projeto. Variações paramétricas devem ser previstas em uma aplicação de alta confiabilidade como as aplicações aeroespaciais, pois elas podem comprometer toda uma missão.

4.1 Conclusões

Conclui-se que o processo de síntese realiza a alteração de nomes de ligações internas, uma vez que são geradas aleatoriamente e automaticamente. A cada ciclo de síntese realizada, ocorre a alteração de todos os nomes internos, dificultando assim o processo de levantamento do circuito gerado. Este levantamento é imprescindível durante o ciclo de desenvolvimento e durante a realização do processo de verificação e validação de *software* para aplicações que exigem alta confiabilidade.

5 Metodologias de desenvolvimento de *software*

5.1 Histórico

Os primeiros métodos de desenvolvimento de *software* começam a surgir em 1950, criados para utilização em aplicações militares e científicas. Com o surgimento dos equipamentos eletrônicos, houve a necessidade de criação de *software* mais elaborados, exigindo um grande esforço de desenvolvimento. Devido à grande velocidade de mudanças, este movimento foi considerado como uma nova revolução industrial (PRESSMAN, 2005).

Em uma segunda fase, por volta de 1970, surgem as empresas especializadas na criação de *software*, sendo conhecidas como *software houses*. O desenvolvimento de *software* passou a ser realizado para distribuição em larga escala (NONEMACHER,2003). Nesta segunda fase, algumas metodologias já existiam, e se difundiam em ampla escala.

Em uma terceira fase, por volta de 1980, o *software* passou a ter uma relevância maior, pois inúmeros produtos eletrônicos, inclusive eletrodomésticos, passaram a utilizá-lo. As redes de dados ganharam dimensões internacionais, juntamente com um aumento do número de usuários domésticos.

Devido à crescente expansão de utilização de *software*, surgiram paralelamente vários métodos alternativos de desenvolvimento, com diversas características em comum, derivando em sua maioria do conhecimento prático de programadores, desenvolvedores e pesquisadores.

5.2 Modelo Cascata

Na primeira fase da Engenharia de *Software*, o planejamento foi muito enfatizado, com o objetivo de tornar a programação mais previsível. Desta forma, todas as definições realizadas por um analista poderiam ser facilmente implementadas por um programador, baseando-se nos diagramas gerados.

Estas idéias são claramente visualizadas no modelo primordial de desenvolvimento, conhecido por Cascata (PRESSMAN, 2005). O fluxograma de desenvolvimento do modelo Cascata é apresentado na Figura 5-1.

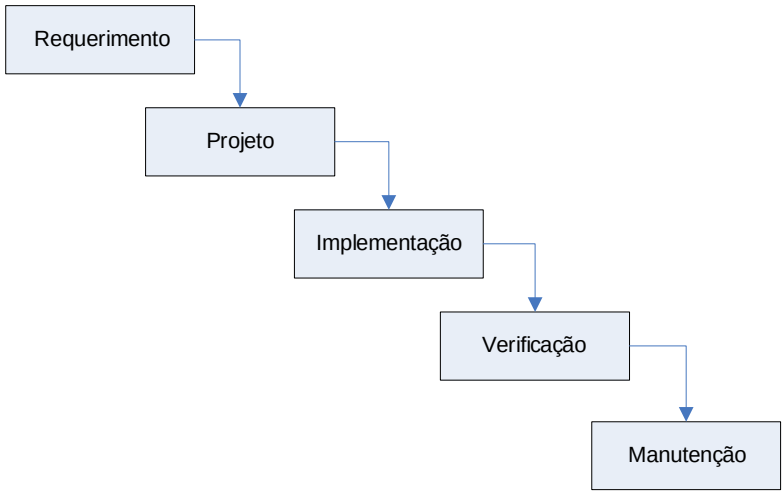


Figura 5-1. Modelo de desenvolvimento Cascata.

O modelo Cascata surgiu inicialmente nos trabalhos realizados pelo Departamento de Defesa dos Estados Unidos no desenvolvimento de circuitos integrados.

O modelo Cascata, mesmo sendo muito utilizado nos dias atuais, não é o modelo ideal para desenvolvimento comercial, por possuir diversas falhas. Este modelo de desenvolvimento não deve ser aplicado a processos cujos requisitos sofram muitas alterações ao longo do projeto. Uma alteração das necessidades da

aplicação pode gerar muita documentação, que será rapidamente levada à obsolescência.

Um outro problema do modelo Cascata, é a falta de visão do *software* final durante um grande período de projeto, causado pela distância do deste com sua especificação. (PRESSMAN, 2005).

Apesar dos diversos problemas encontrados no modelo Cascata, ele ainda é o mais adequado para aplicações aeroespaciais. Nestas aplicações, ao contrário das aplicações comerciais, não existe alterações de requisitos durante o ciclo de desenvolvimento, favorecendo assim sua utilização (COCKBURN, 1999). Devido a este fato, os outros modelos existentes não serão citados.

5.3 Metodologia clássica

O fluxograma da metodologia clássica é apresentado na Figura 5-2.

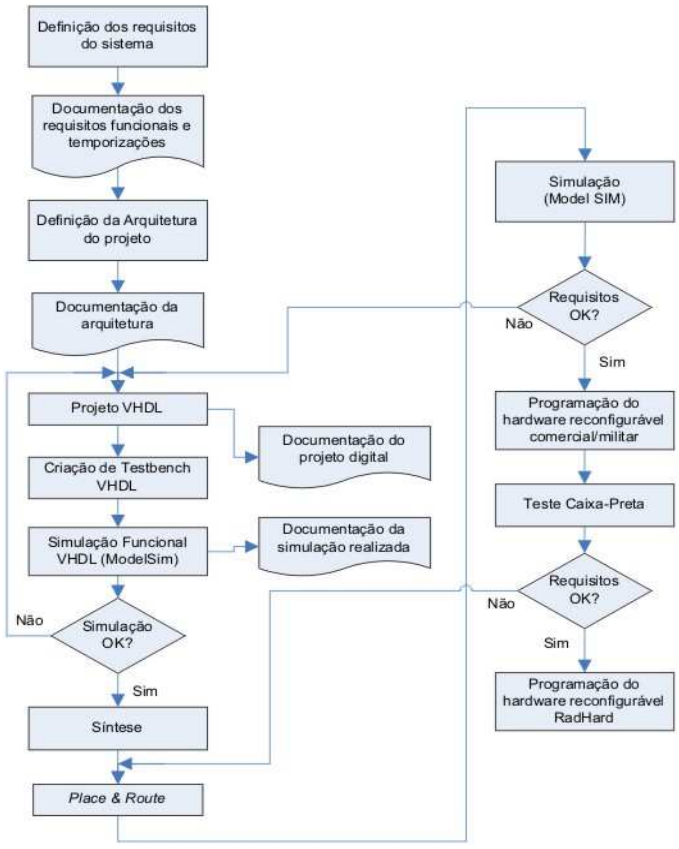


Figura 5-2. Fluxograma da metodologia clássica

O primeiro passo da metodologia clássica é a definição dos requisitos do sistema. Todos os requisitos a serem cumpridos na finalização do projeto devem ser definidos e documentados. A documentação gerada deve ser clara e objetiva, listando em forma de matriz todos os requisitos a serem cumpridos. O principal objetivo desta etapa é formalizar os requisitos iniciais para delinear os caminhos a serem adotados durante o ciclo de desenvolvimento.

Após a etapa de definição dos requisitos e sua documentação, deve-se determinar a arquitetura a ser utilizada para o projeto. Esta arquitetura deve possuir uma documentação específica e possuir controle de versões. Toda documentação criada ao longo do projeto deve possuir um controle de versões baseado no Sistema de Versões Concorrentes (*Concurrent Version System* - CVS). A principal finalidade do controle de versões é permitir a rastreabilidade durante o ciclo de desenvolvimento do projeto. O sistema de controle de versões CVS é amplamente utilizado pela comunidade de desenvolvedores, justificando assim sua utilização.

Com a finalização da definição da arquitetura a ser utilizada juntamente com sua documentação, passamos para a sua implementação em linguagem VHDL. Juntamente com a implementação da arquitetura escolhida, deve-se realizar a documentação do projeto, visando facilitar possíveis alterações futuras.

Com a finalização da implementação, é necessário a criação de *testbenches* para a realização de testes funcionais. Os *testbenches* devem ser escritos também em linguagem VHDL e devem possuir uma documentação específica com controle de versões. A utilização da linguagem VHDL para a criação de *testbenches* se justifica pelas características já apresentadas. Todos os *testbenches*

devem ser simulados e verificados antes de sua utilização para a verificação de requisitos funcionais.

Nesta etapa, o desenvolvedor deve se preocupar apenas com o funcionamento lógico do circuito, gerando um *testbench* que simule todos os requisitos funcionais do sistema.

Após a implementação do *testbench*, deve ser realizada uma simulação funcional. Esta simulação deve ser documentada e tem por finalidade a verificação funcional do sistema, sem levar em consideração as temporizações. Os resultados obtidos na simulação funcional devem ser documentados com controle de versões.

Caso a simulação apresente divergências entre os resultados e os requisitos funcionais do sistema, o re-projeto do sistema deve ser realizado, com alterações no *software* em VHDL. Caso a simulação mostre que todos os requisitos funcionais foram cumpridos, devemos dar prosseguimento ao desenvolvimento.

Com a verificação do cumprimento dos requisitos funcionais, deve ser realizado o processo de síntese. O processo de síntese é responsável pela implementação em células lógicas do *software* descrito em linguagem VHDL.

Com a finalização do processo de síntese, deve-se realizar o processo de *Place & Route*, que realiza o roteamento e posicionamento das células lógicas obtidas a partir do processo de síntese.

Com o término do processo de *Place & Route*, deve ser realizada uma simulação, buscando verificar o cumprimento de todos os requisitos do projeto. A verificação de requisitos nesta etapa deve ser realizada por meio de simulação, utilizando *testbenches* escritos em VHDL, criados especificamente para a simulação do projeto. Estes testbenches devem possuir documentação com controle de versões.

A utilização da linguagem VHDL para a criação de *testbenches* se justifica pelas características já apresentadas. Todos os *testbenches* devem ser simulados e verificados antes de sua utilização para a verificação de requisitos. Todo o processo de verificação dos *testbenches* deve ser documentado e possuir controle de versões.

A simulação deve utilizar os arquivos gerados durante o processo de *Place & Route* e que contém todas as informações de temporizações internas do projeto criado. A simulação deve ser realizada na ferramenta ModelSim da empresa Mentor Graphics (2008a), por ser compatível com a maioria das ferramentas de desenvolvimento.

Com a finalização das simulações, todos os resultados obtidos, assim como todas as configurações utilizadas durante a simulação, devem ser documentados em um relatório de testes. Deve-se realizar este relatório para permitir uma profunda análise do comportamento do sistema caso alterações sejam necessárias.

Caso os requisitos de projeto não sejam atingidos, deve-se realizar a alteração do *software* em VHDL buscando o cumprimento dos requisitos. Após as alterações, deve-se seguir o fluxograma, passando por uma nova simulação funcional, processo de síntese, processo de *Place & Route* e simulação final.

Com o cumprimento dos requisitos de projeto, o *software* final gerado pela ferramenta de desenvolvimento pode ser liberado para a programação do hardware reconfigurável comercial/militar. O procedimento de programação deve ser documentado, apresentando a versão do *software* final a ser gravado com o número de série do componente. O *software* final para gravação deve possuir um controle de versões.

Após a programação do hardware reconfigurável, deve ser realizado um teste funcional, utilizando o método de teste caixa-preta. Este método também é chamado de Teste Funcional, em que o *software* a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o seu comportamento interno. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado, previamente conhecido. Segundo Inthurn (2001), o teste de caixa-preta é utilizado para verificar os seguintes erros:

- erros de interface;
- erros de desempenho;
- erros de inicialização e término.

O teste caixa-preta deve ser realizado com o hardware mais próximo ao modelo de vôo, para que se possam verificar possíveis problemas futuros. Este teste deve possuir um procedimento, descrevendo claramente os objetivos, requisitos e resultados esperados. Após a realização do teste, deve ser gerado um relatório de teste contendo todas as informações geradas, devendo possuir também um controle de versões. Caso todos os requisitos sejam atingidos, a programação do componente de vôo deve ser liberada.

6 Metodologia de desenvolvimento proposta

6.1 Introdução

Todo desenvolvimento aeroespacial deve possuir margens de segurança grandes e algum tipo de redundância, com o principal objetivo de corrigir possíveis falhas, tanto de projeto, quanto causadas pelo ambiente inóspito. Devido, principalmente, à radiação presente no espaço, recomenda-se, para correção de seus efeitos em hardware reconfigurável, a criação de sistemas e *software* com algum tipo de redundância, como vimos no capítulo 3.4. Devido principalmente às características intrínsecas das ferramentas de síntese e de otimização lógica, como vimos no capítulo 4, a inserção de margens de segurança e redundâncias só são possíveis com o desenvolvimento de todo o *software* em forma de esquemático, no nível de desenvolvimento mais baixo, ou seja, em nível de portas lógicas (*gate level*). Com base nestas informações, a proposta deste trabalho é uma metodologia de desenvolvimento baseada no modelo Cascata, utilizando programação em nível de portas lógicas (*gate level*). Desta maneira, os problemas apresentados pela ferramenbta de síntese serão contornados, garantindo a presença da arquitetura definida inicialmente durante todo o ciclo de desenvolvimento.

6.2 Metodologia

O fluxograma da metodologia proposta é apresentado na Figura 6-1. O primeiro passo da metodologia é a definição dos requisitos do projeto. Todos os requisitos a serem cumpridos na finalização do projeto devem ser definidos e documentados.

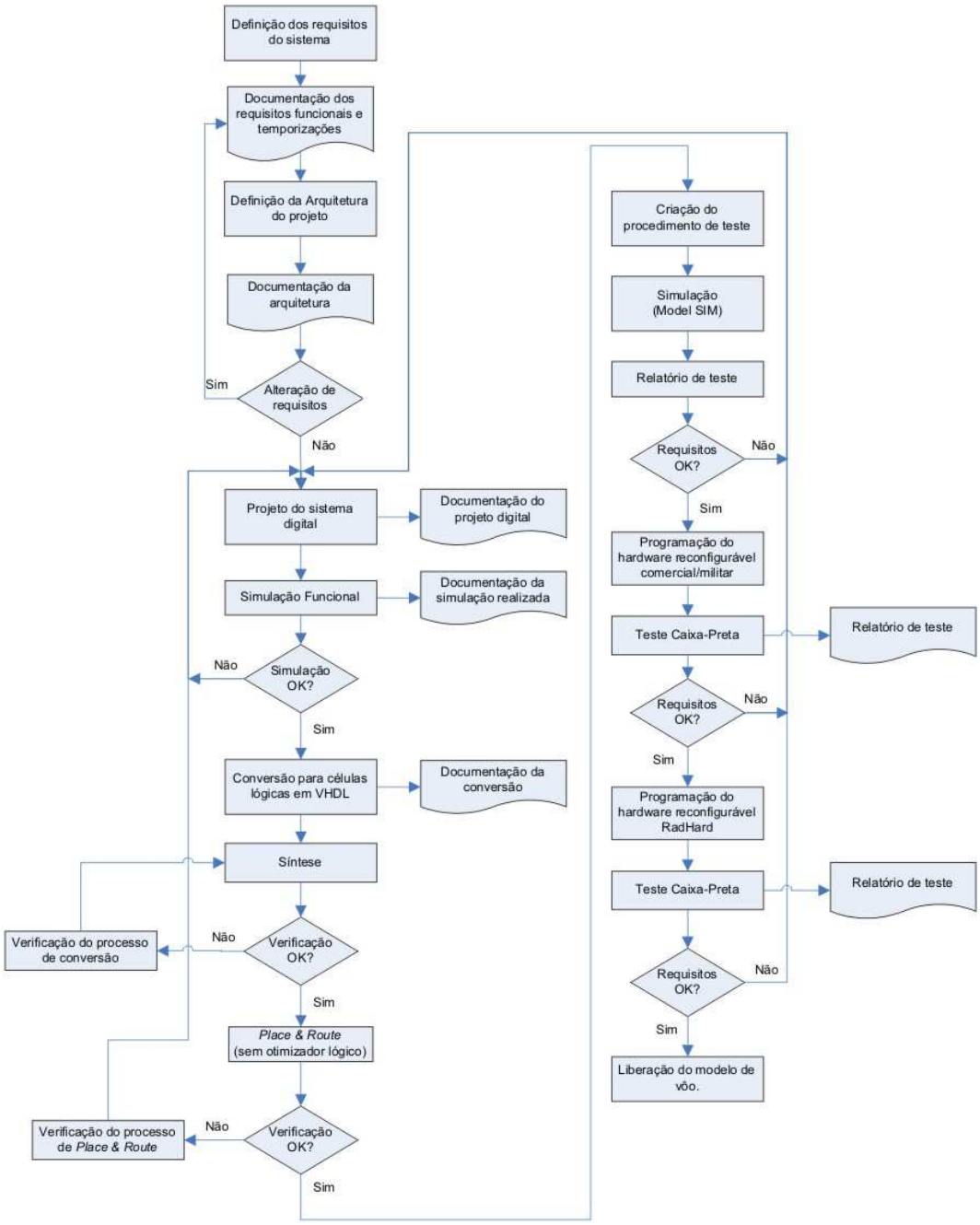


Figura 6-1. Fluxograma da metodologia proposta.

A documentação gerada deve ser clara e objetiva, listando em forma de matriz todos os requisitos a serem cumpridos. O principal objetivo desta etapa é formalizar os requisitos para delinear os caminhos a serem adotados durante o ciclo de desenvolvimento.

Após a etapa de definição dos requisitos e sua documentação, deve-se determinar a arquitetura a ser utilizada para o projeto. Esta arquitetura deve

possuir uma documentação específica e possuir controle de versões. Toda documentação criada ao longo do projeto deve possuir um controle de versões baseado no Sistema de Versões Concorrentes (*Concurrent Version System* - CVS). A principal finalidade do controle de versões é permitir a rastreabilidade durante o ciclo de desenvolvimento do projeto. O sistema de controle de versões CVS é amplamente utilizado pela comunidade de desenvolvedores, justificando assim sua utilização.

Com a finalização da definição da arquitetura a ser utilizada, deve-se realizar uma revisão dos requisitos do sistema, com o intuito de verificar a necessidade de adição ou modificação de algum requisito, devido a restrições da arquitetura escolhida.

Após a revisão dos requisitos, deve-se iniciar o projeto do sistema digital. O projeto do sistema digital deve ser realizado utilizando-se de circuitos lógicos de mais baixo nível, utilizando portas lógicas e *flip-flops* semelhantes à tecnologia empregada no hardware reconfigurável escolhido. O projeto do sistema digital pode ser realizado em qualquer ferramenta de simulação digital, desde que forneça todos os meios para documentação. A linguagem de programação VHDL será utilizada somente para a transcrição do projeto digital, gerado nesta etapa, para o hardware reconfigurável em nível de portas lógicas. O projeto digital deverá ser realizado totalmente em nível de portas lógicas (*gate level*) devido às características apresentadas pela ferramenta de desenvolvimento, detalhadas no capítulo 4. Desta forma, todo o desenvolvimento do *software* para o hardware reconfigurável presente na MUXCAM deverá ser realizado em nível de portas lógicas. Nesta etapa, o desenvolvedor deve se preocupar com o funcionamento lógico do circuito, considerando para o projeto, os atrasos dos componentes

presentes no hardware reconfigurável. Como esta metodologia será aplicada na MUXCAM, as temporizações que devem ser levadas em consideração serão da FPGA QL6325 da Quicklogic, uma vez que as temporizações dos componentes internos do modelo comercial, militar e aeroespacial são iguais (Quicklogic, 2007a) (Aeroflex, 2007b). Nesta etapa todas as margens de segurança para temporizações internas do projeto devem ser definidas e documentadas. A utilização de redundâncias é recomendada para a diminuição de problemas gerados pelos efeitos da radiação, como apresentado no capítulo 3.4.

Após a finalização do projeto digital e de toda documentação envolvida, deve ser realizada uma simulação funcional. Esta simulação deve ser documentada e tem por finalidade a verificação funcional do sistema, sem levar em consideração as temporizações. Os resultados obtidos na simulação funcional devem ser documentados com controle de versões. A simulação funcional deve ser realizada na mesma plataforma de *software* utilizada para o projeto digital.

Caso a simulação apresente divergências entre os resultados e os requisitos funcionais, o re-projeto do sistema digital deve ser realizado. Caso a simulação mostre que todos os requisitos funcionais foram cumpridos, devemos dar prosseguimento ao desenvolvimento.

A próxima etapa é a realização da conversão do circuito atual desenvolvido para células lógicas do hardware reconfigurável. Esta etapa é dependente de tecnologia, pois leva em consideração a arquitetura do hardware reconfigurável utilizada.

No caso da MUXCAM, a ferramenta de desenvolvimento QuickWorks™ permite a utilização do hardware reconfigurável em nível de portas lógicas por meio de programação em linguagem VHDL. Somente a utilização da linguagem

VHDL é necessária, pois permite que a ferramenta de síntese identifique a utilização de células lógicas, mantendo a descrição e ligações internas do circuito desenvolvido. Na MUXCAM, todo o circuito deve ser convertido utilizando células lógicas apresentadas nos itens 3.1.1e 3.2.1. Todo o processo de conversão deve ser documentado e possuir controle de versões.

Após o processo de conversão, deve ser realizado o processo de síntese. Como o circuito foi descrito em nível de portas lógicas (*gate level*) em VHDL, a ferramenta de síntese apenas transfere o circuito por meio de um *Netlist*, mantendo os nomes das *nets*. Esta característica é muito importante, pois faz com que o circuito a ser implementado seja exatamente idêntico ao projetado.

Com a finalização do processo de síntese, há a necessidade de se realizar uma verificação em nível de portas lógicas (*gate level*) do circuito gerado, a fim de identificar possíveis diferenças entre o circuito projetado e o gerado pela ferramenta de síntese. Todo o processo de verificação deve ser documentado e possuir controle de versões.

Caso o processo de verificação identifique características diferentes das esperadas, uma verificação do código escrito em VHDL deve ser realizada. Caso a verificação mostre que a geração do circuito em nível de portas lógicas (*gate level*) está igual à projetada, deve-se prosseguir o ciclo de desenvolvimento. Todo este processo deve ser documentado e possuir controle de versões.

O próximo passo no ciclo de desenvolvimento é a realização do processo de *Place & Route*. Este processo deve ser realizado sem a utilização de otimizadores lógicos, para que não ocorram alterações no circuito a ser implementado, como descrito no capítulo 4. Nesta etapa, nenhuma regra de temporização deve ser utilizada. Os circuitos lógicos devem ser agrupados o mais próximo possível, para

que os atrasos de interligações sejam mínimos. A ferramenta de desenvolvimento permite o posicionamento independente de todas as células lógicas, como apresentado no capítulo 4. Nesta etapa devem ser configurados todos os pinos de E/S utilizados no projeto.

Com o término do processo de *Place & Route*, deve-se realizar uma verificação de todo o circuito gerado a fim de identificar divergências entre o projeto implementado e o esperado, inclusive com relação às margens de segurança e redundâncias definidas no início do desenvolvimento. O desenvolvedor deve garantir que o hardware implementado seja igual ao descrito na etapa de projeto digital. Caso ocorram divergências, todo o processo, a partir do projeto digital, deve passar por uma verificação. Caso seja realizada alguma alteração, o ciclo de desenvolvimento deve ser seguido a partir do ponto de alteração, incluindo a revisão de toda documentação.

Com a finalização da verificação do processo de *Place & Route*, tem início o processo de verificação e validação do *software*. Estes processos se iniciam com uma simulação completa do sistema, para a verificação do cumprimento de requisitos de projeto, inclusive de temporizações e atrasos. Nesta etapa é necessária a criação de um procedimento de teste, onde serão definidos todos os procedimentos adotados para a verificação dos requisitos por meio de simulação. Este procedimento deve ser documentado e deve possuir controle de versões.

A verificação de requisitos nesta etapa deve ser realizada por meio de simulação, utilizando *testbenches* criados especificamente para a simulação do projeto. Um *testbench* utilizado neste trabalho é apresentando no Apêndice B. Estes *testbenches* devem ser escritos em linguagem VHDL e devem ser devidamente documentados no procedimento de teste, possuindo controle de

versões. A utilização da linguagem VHDL para a criação de *testbenches* se justifica pelas características apresentadas no capítulo 3.3. Todos os *testbenches* devem ser simulados e verificados antes de sua utilização para a verificação de requisitos. Todo o processo de verificação dos *testbenches* deve ser documentado e possuir controle de versões.

Esta simulação deve utilizar os arquivos gerados durante o processo de *Place & Route* e que contém todas as informações de temporizações do projeto criado. A simulação deve ser realizada na ferramenta ModelSim da empresa Mentor Graphics (2008a), por ser compatível com a maioria das ferramentas de desenvolvimento.

Com a finalização das simulações, todos os resultados obtidos, assim como todas as configurações utilizadas durante a simulação, devem ser documentados em um relatório de testes. Deve-se realizar este relatório para permitir uma profunda análise do comportamento do sistema caso alterações sejam necessárias.

Caso os requisitos de projeto não sejam atingidos, primeiramente deve ser realizado o reposicionamento das células lógicas, verificando sua influência nos resultados obtidos. Se os requisitos do projeto não forem cumpridos apenas pela alteração de posicionamento das células lógicas, todo o processo de desenvolvimento deve ser verificado.

Com o cumprimento dos requisitos de projeto, o *software* final gerado pela ferramenta de desenvolvimento pode ser liberado para a programação do hardware reconfigurável comercial/militar. O procedimento de programação deve ser documentado, apresentando a versão do *software* final a ser gravado com o número de série do componente. O *software* final para gravação deve possuir um controle de versões.

Após a programação do hardware reconfigurável, deve ser realizado um teste funcional, utilizando o método de teste caixa-preta. Este método também é chamado de Teste Funcional, em que o *software* a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o seu comportamento interno. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado, previamente conhecido.

O teste caixa-preta deve ser realizado com o hardware mais próximo ao modelo de vôo, para que se possam verificar possíveis problemas futuros. Este teste deve possuir um procedimento, descrevendo claramente os objetivos, requisitos e resultados esperados. Após a realização do teste, deve ser gerado um relatório de teste contendo todas as informações geradas, devendo possuir também um controle de versões.

Caso todos os requisitos sejam atingidos, a programação do componente de vôo deve ser liberada. Após a programação do componente de vôo, um teste caixa-preta deverá ser realizado. A exemplo do caso recém mencionado, este teste também deve possuir um procedimento, descrevendo claramente os objetivos, requisitos e resultados esperados. Após a realização do teste, deve ser gerado um relatório de teste contendo todas as informações geradas, devendo possuir também um controle de versões.

Caso todos os requisitos sejam cumpridos, o hardware reconfigurável está apto para ser utilizado em sistemas aeroespaciais.

7 Resultados e Discussões

7.1 Introdução

Como parte do processo de validação da metodologia proposta, utilizar-se-ão a metodologia clássica e a metodologia proposta, em um mesmo projeto.

Esta ação tem como objetivo principal a comparação direta entre os procedimentos adotados, visando identificar suas principais características. Todas as documentações relativas aos procedimentos de testes não foram realizadas por não serem o objetivo principal do trabalho.

O projeto proposto para comparação dos procedimentos adotados é um contador de 16 bits que deverá ser implementado em *hardware* reconfigurável.

7.2 Aplicação da metodologia clássica

Será descrito a seguir todo o procedimento adotado na criação de um contador de 16 bits utilizando a metodologia clássica. O fluxograma da metodologia adotada é reapresentado na Figura 7-1.

O primeiro procedimento é a definição dos requisitos de projeto. Os requisitos definidos nesta etapa serão utilizados durante todo o processo de validação da metodologia.

No projeto do contador, os requisitos principais do sistema são:

- contagem correta em 16 bits na subida de borda do sinal de *clock*;
- trabalhar em freqüência de até 20MHz;
- possuir circuito de *Enable* e *Reset*;

Após a definição dos requisitos do sistema, passa-se a realizar sua documentação.

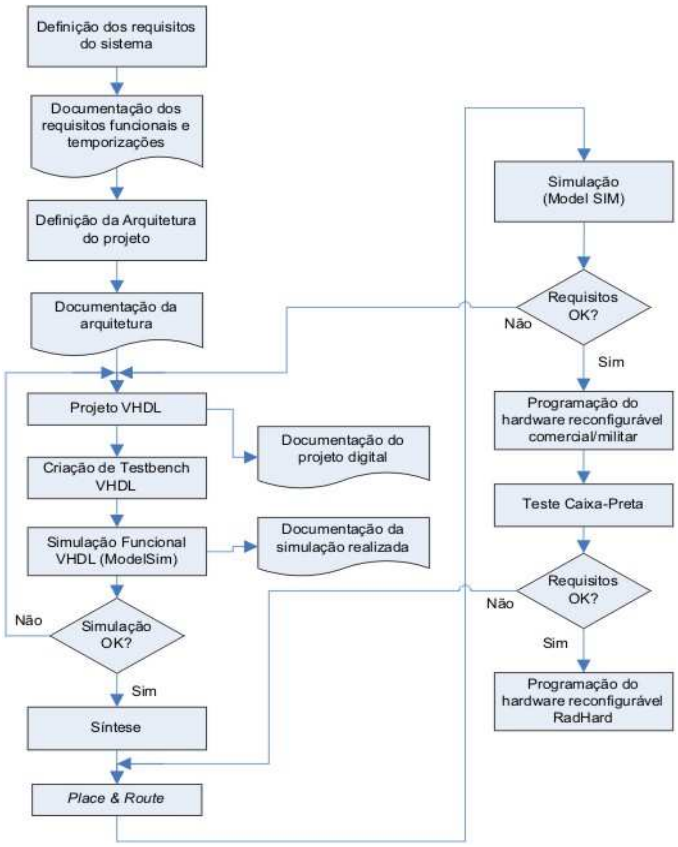


Figura 7-1. Fluxograma da metodologia clássica.

Em seguida, passa-se a realizar a definição da arquitetura proposta. Com a finalização da definição de arquitetura, realizou-se a implementação do contador de 16 bits em linguagem VHDL, como previsto pela metodologia clássica. O contador implementado é apresentado no Apêndice A.

Após a criação do arquivo em VHDL, cria-se um *testbench* em linguagem VHDL para verificar o funcionamento do contador de 16 bits implementado. Esta verificação é apenas funcional, uma vez que os atrasos reais do *hardware* reconfigurável não são considerados. O *testbench* implementado é apresentado no Apêndice B.

O próximo passo é a realização da simulação na ferramenta ModelSim, da empresa Mentor Graphics (2008a). Os resultados obtidos na simulação são

apresentados na Figura 7-2 e Figura 7-3. O funcionamento do sistema foi comprovado.

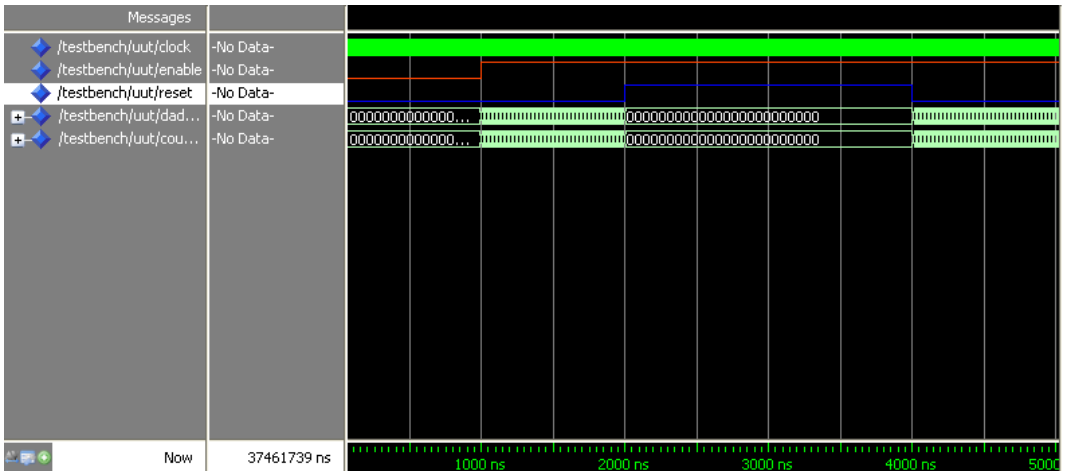


Figura 7-2. Visão geral da simulação pré-sintese do contador de 16 bits

Vemos na Figura 7-2 que o contador permanece em estado inicial 0 enquanto o sinal de *Enable* permanece no estado 0. O contador passa a realizar a contagem quando este sinal passa para o nível 1. Assim que o sinal de *Reset* altera seu valor de 0 para 1, o contador pára a contagem imediatamente, voltando para um estado inicial 0, até que o sinal de *Reset* retorne para o nível 0. Com este comportamento podemos concluir que os principais requisitos funcionais foram cumpridos.

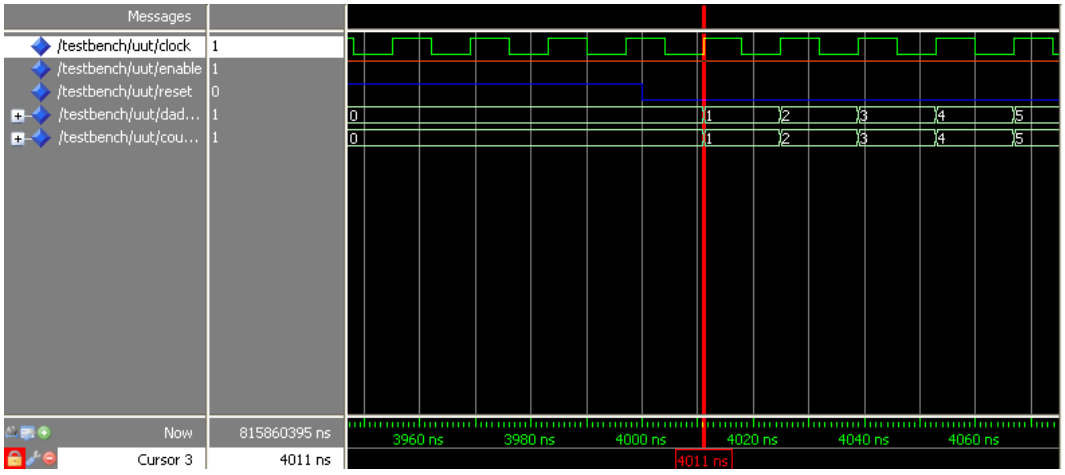


Figura 7-3. Visão ampliada da simulação pré-sintese do contador de 16 bits

Com a comprovação do funcionamento, seguindo o fluxograma, passamos para a realização do processo de síntese. A ferramenta de síntese utilizada é a Precision RTL da empresa Mentor Graphics (2008b), disponibilizada pela ferramenta QuickWorks™. As opções utilizadas no processo de síntese foram:

- Device: Eclipse QL6325
- Speed Grade: -4
- Package: pq208
- Design Frequency/Period: None

O esquemático RTL gerado pela ferramenta de síntese é apresentado na Figura 7-4.

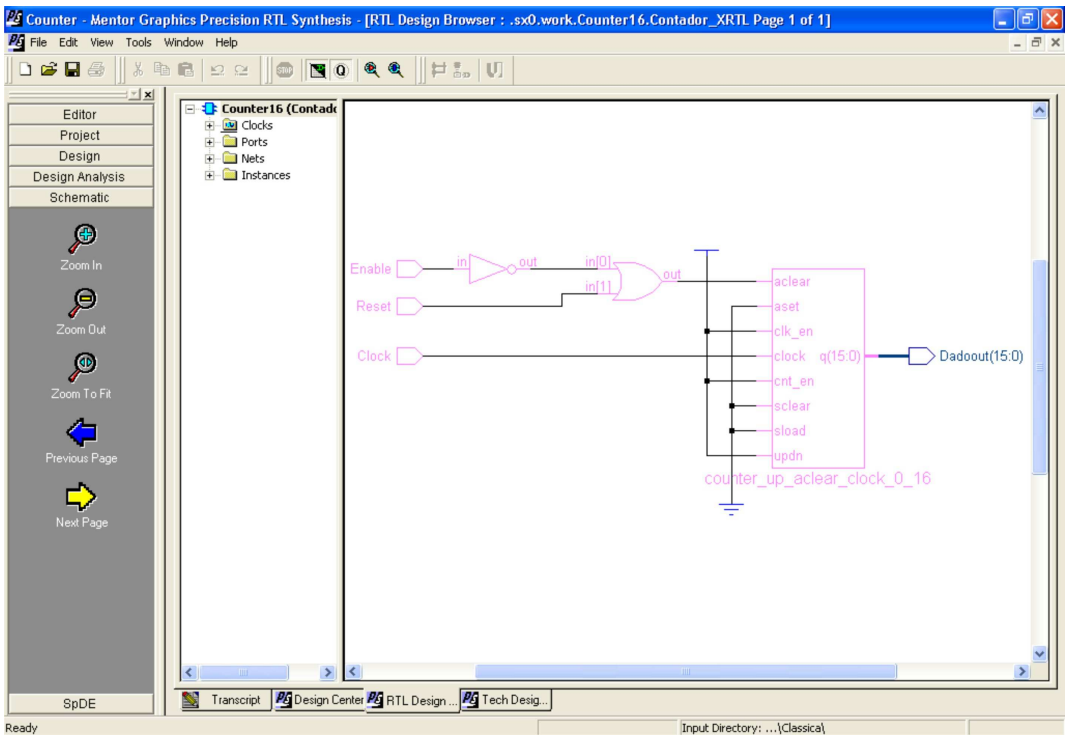


Figura 7-4. Esquemático *Register Transfer Level* RTL.

Devido à grande extensão do esquemático tecnológico disponibilizado pela ferramenta de síntese, apenas uma parte do circuito é apresentado na Figura 7-5.

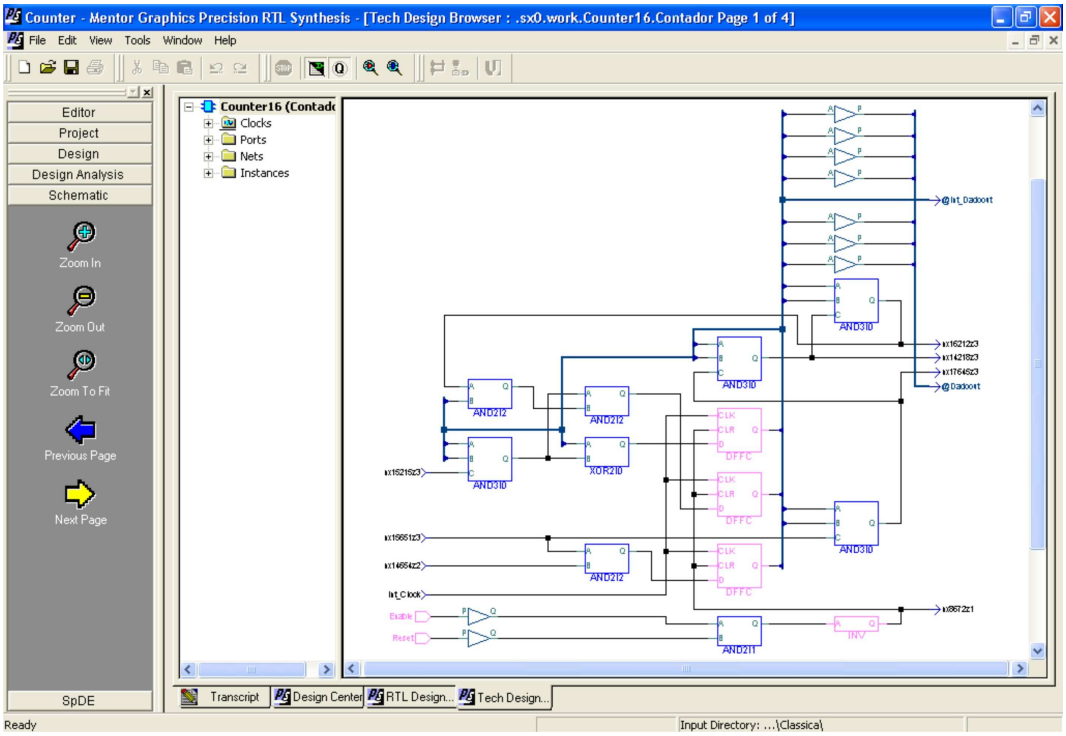


Figura 7-5. Esquemático tecnológico.

O relatório de área disponibilizado pela ferramenta de síntese é apresentado no Anexo A.

Dando seqüência ao fluxograma de desenvolvimento, realiza-se o processo de *Place & Route* do contador. Foram utilizadas no processo as opções *default*.

As regras criadas para a realização do *Place & Route* são apresentadas no Anexo B. Os pinos de E/S foram escolhidos de acordo com o hardware disponível para a realização do teste caixa-preta. A Figura 7-6 apresenta a janela de edição de regras, especificamente a de posicionamento por janelas, já apresentada no capítulo 4.

Nesta janela podemos verificar os nomes das *nets* gerados internamente. Conclui-se que a localização das *nets* do circuito implementado é prejudicada, uma vez que fica praticamente impossível a localização de um circuito específico. Esta característica impossibilita o posicionamento adequado de cada componente ou célula lógica.

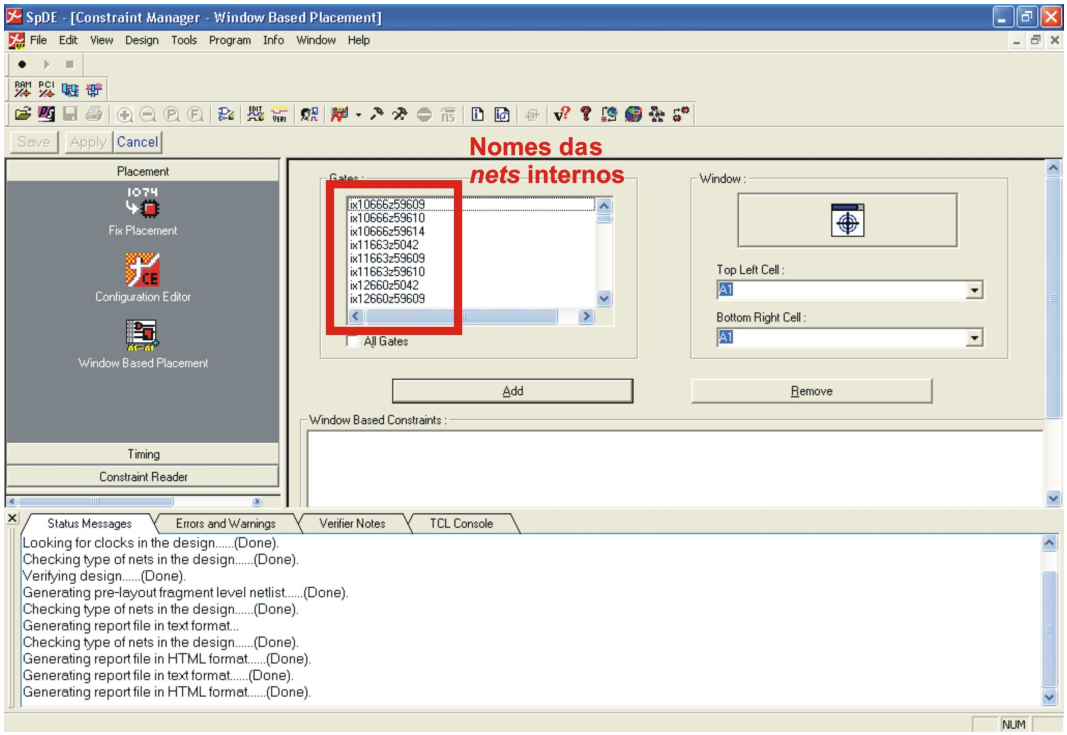


Figura 7-6. Janela de edição de regras da ferramenta QuickWorks™.

A localização das *nets* só é possível após o levantamento manual do circuito implementado. As características dos pinos de E/S foram mantidas em seus valores *default.* Os processos realizados pelo processo de *Place & Route* são apresentados na Figura 7-7.

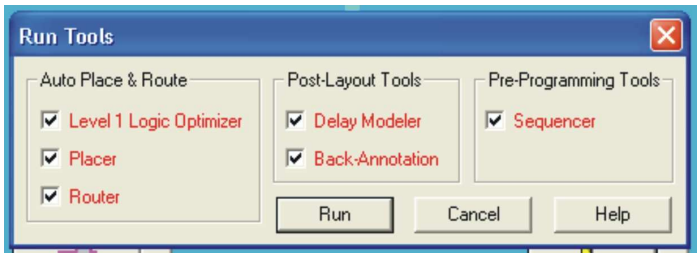


Figura 7-7. Processos realizados dentro do *Place & Route*

Após a realização do processo *Place & Route*, um relatório contendo um resumo deste é apresentado ao usuário. O resumo relativo a este projeto é apresentado no Anexo C. Em seguida verificou-se o circuito implementado no hardware reconfigurável pela ferramenta de *Place & Route*. A Figura 7-8 apresenta uma visão geral do circuito implementado. Devido ao elevado número de células lógicas presentes no hardware reconfigurável, a visão geral não permite

o detalhamento do circuito gerado. Após o levantamento manual, devido a inexistência de uma ferramenta específica, do circuito implementado criou-se um circuito em forma de esquemático que representa o circuito a ser gravado na FPGA. Este circuito é apresentado no Apêndice C.

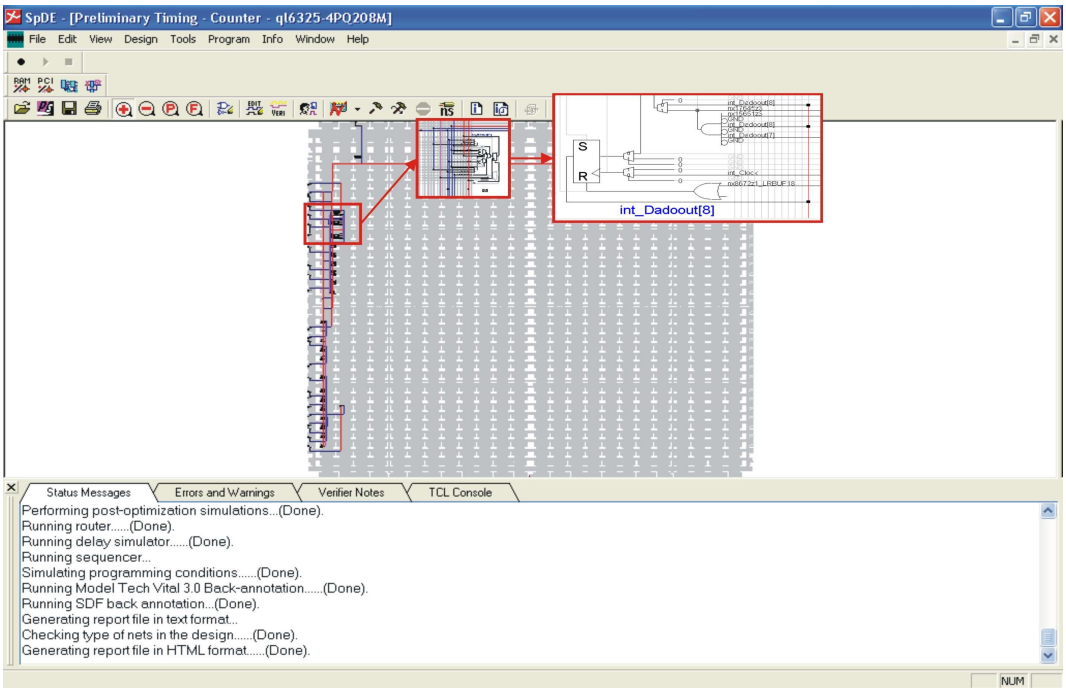


Figura 7-8. Visão geral do circuito implementado.

Seguindo o fluxograma de desenvolvimento, realizou-se a simulação na ferramenta ModelSim, da empresa Mentor Graphics (2008a). Nesta simulação, foi utilizado o arquivo gerado pela ferramenta de *Place & Route* que contém todas as informações sobre as temporizações internas. Este arquivo é gerado no formato Model Tech Vital 3.0 e possui a extensão SDF, possibilitando sua utilização no ModelSim. Nesta simulação, o arquivo com extensão .VHD, utilizado na simulação funcional, é substituído pelo arquivo com extensão .VHQ. Este arquivo é gerado pela ferramenta de *Place & Route*, que em conjunto com o arquivo .SDF, permite a simulação com todas as temporizações internas estimadas.

O funcionamento correto do contador foi verificado com uma resolução de 1ns. A resolução é uma opção fornecida pela ferramenta de simulação, indicando

a escala de tempo utilizada para a realização de cada iteração do processo de simulação. Nesta etapa foram alteradas apenas as resoluções utilizadas pela ferramenta de simulação.

Os resultados obtidos na simulação são apresentados na Figura 7-9 e Figura 7-10. Na Figura 7-10 vemos que a contagem está defasada de 8ns aproximadamente com relação à subida de borda do sinal de *clock*.

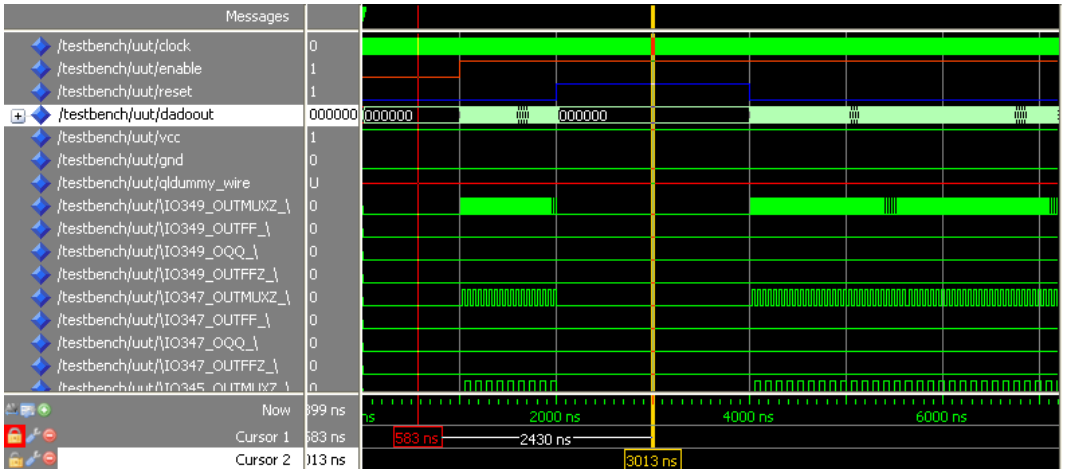


Figura 7-9. Visão geral da simulação com resolução de 1ns.

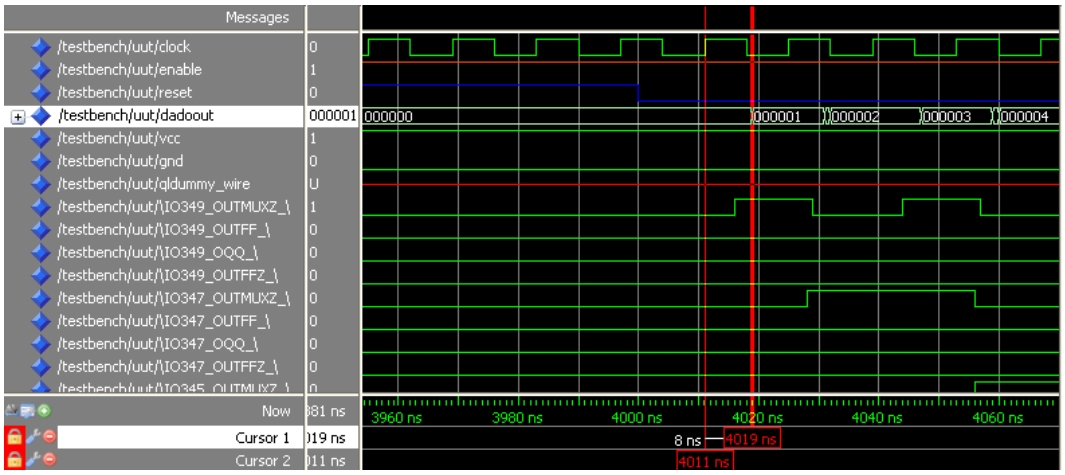


Figura 7-10. Visão ampliada da simulação com resolução de 1ns.

Através da simulação pode-se comprovar o funcionamento do sistema, uma vez que todos os requisitos de projeto foram cumpridos.

A ferramenta de simulação disponibiliza também sinais internos do circuito em simulação, mas os nomes das *nets* não são familiares, apresentando nomenclaturas ligadas aos nomes das células lógicas do hardware reconfigurável.

Estas nomenclaturas não são as mesmas utilizadas pela ferramenta QuickWorks™ durante a exibição do circuito criado pela ferramenta de *Place & Route*.

Seguindo o fluxograma proposto, com o cumprimento dos requisitos de projeto, passa-se para a programação do hardware reconfigurável. O hardware utilizado foi a FPGA QL6325 da Quicklogic (2007a). A FPGA programada será utilizada em uma placa já existente da MUXCAM para a realização do teste caixa-preta. A placa com a FPGA que foi utilizada é apresentada na Figura 7-11. Para este teste caixa-preta foi criado um fluxograma simplificado, parte integrante do procedimento de teste. Em sistemas complexos, com muitos requisitos, é necessária a criação de um procedimento, resultando posteriormente, um relatório, como previsto no fluxograma. O fluxograma simplificado do teste caixa-preta é apresentado na Figura 7-12.

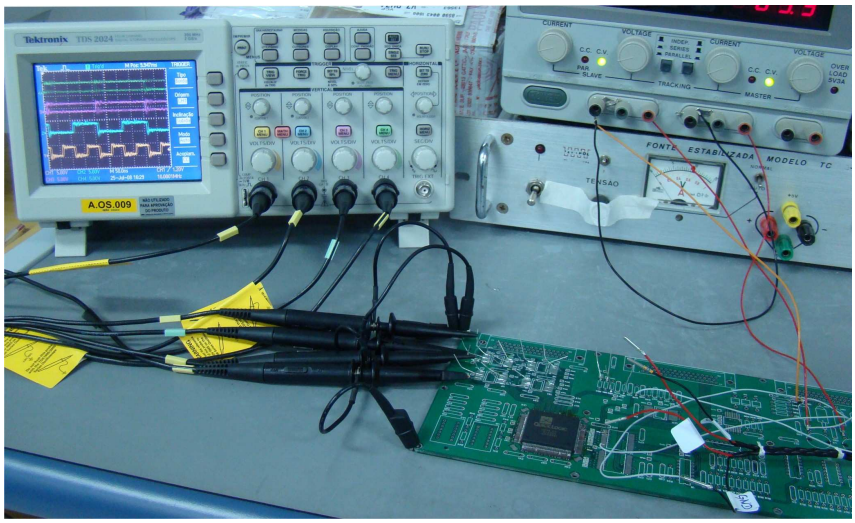


Figura 7-11. Placa para teste caixa-preta.

Os resultados obtidos são apresentados a seguir. A Figura 7-13 apresenta o atraso do sinal de *clock* em relação a alteração do primeiro bit do contador. Este atraso é de 8 ns, exatamente o mesmo atraso apresentado pela ferramenta de simulação. A Figura 7-14 apresenta o funcionamento dos bits0-5. A Figura 7-15 apresenta o funcionamento dos bits6-9 e a Figura 7-16 apresenta o funcionamento

dos bits10-15. Verifica-se que o contador está funcionando corretamente. Com a comprovação dos requisitos, este *software* estaria apto para ser embarcado em um hardware reconfigurável *RadHard* para aplicações aeroespaciais.

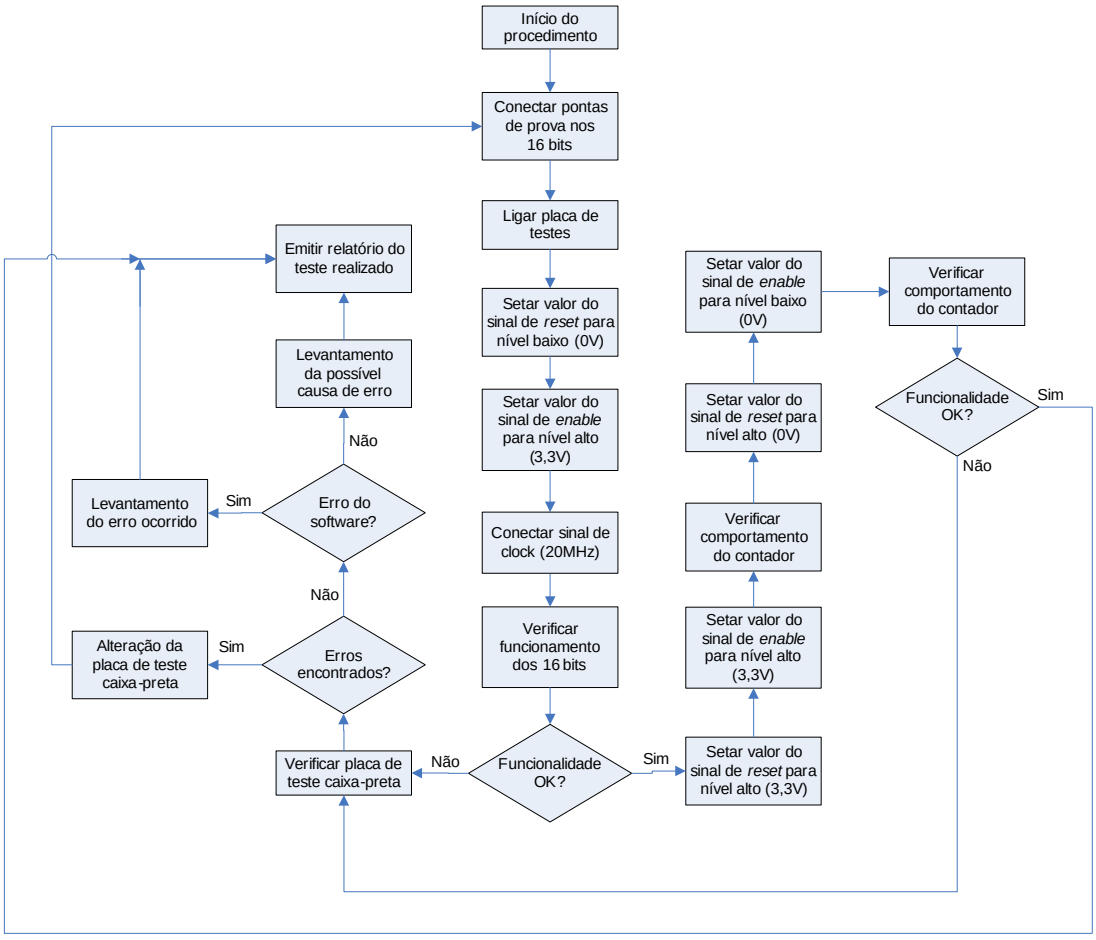


Figura 7-12. Fluxograma simplificado do teste caixa-preta.

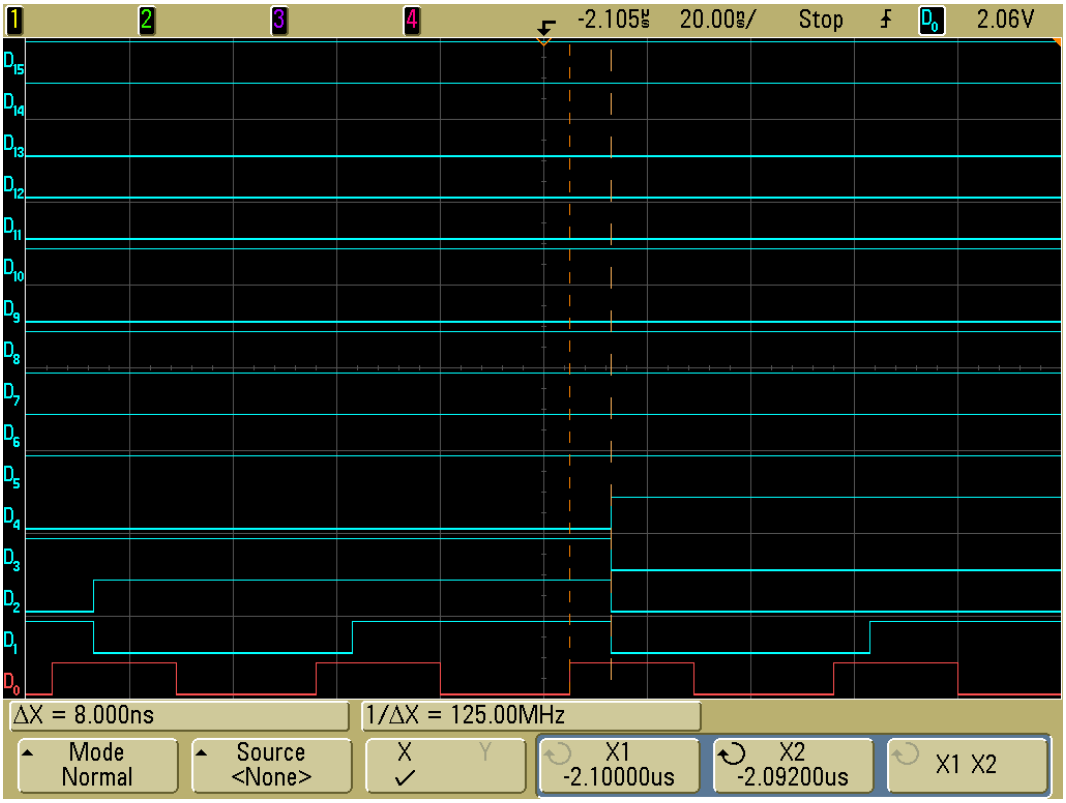


Figura 7-13. Resultados teste caixa-preta – Atraso do sinal de *clock* para o primeiro bit do contador.

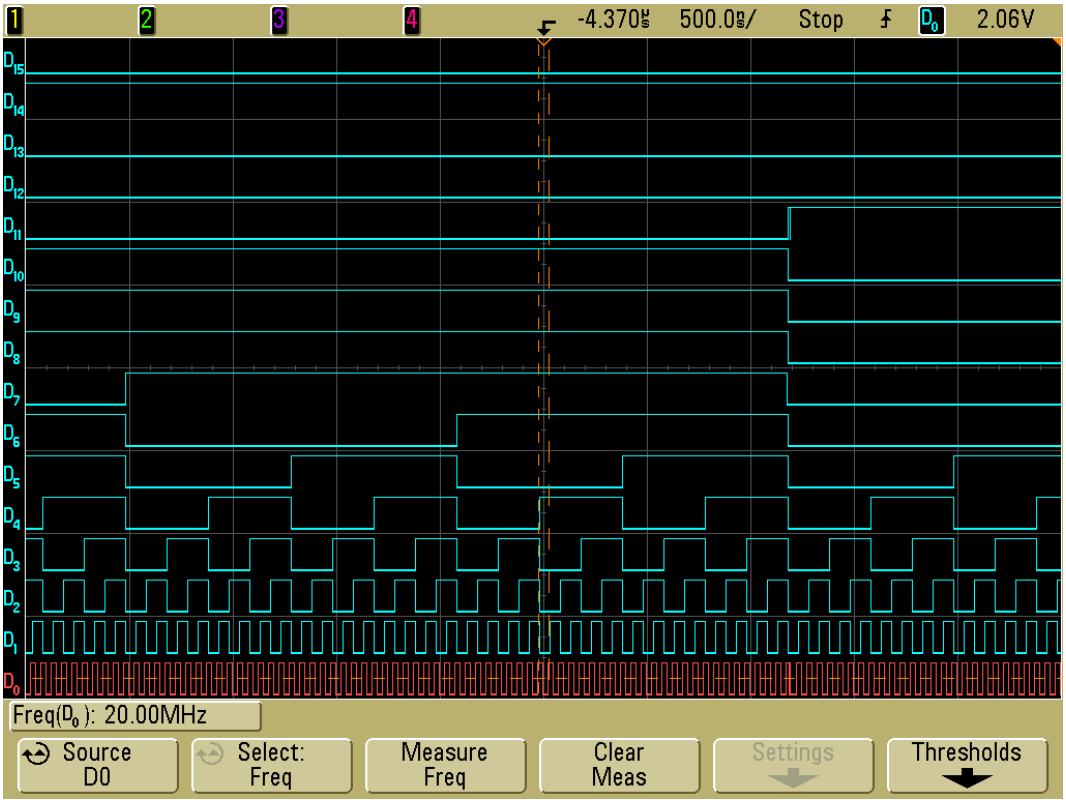


Figura 7-14. Resultados teste caixa-preta –Bits 0-5.

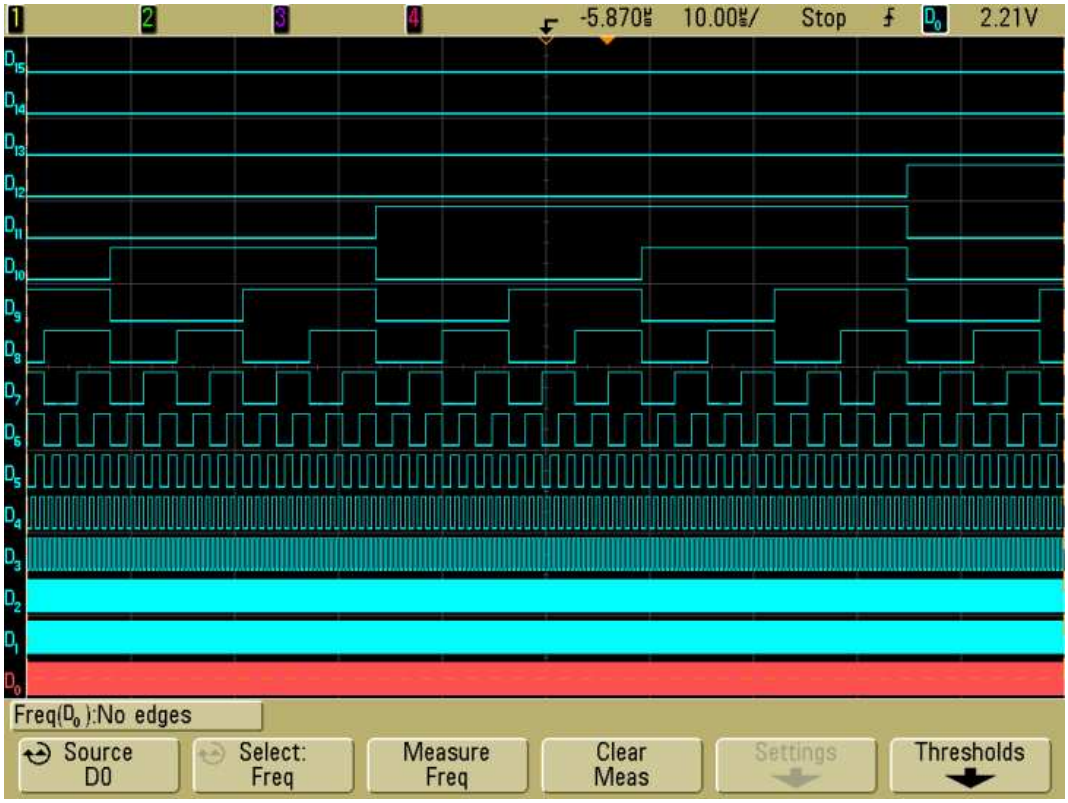


Figura 7-15. Resultados teste caixa-preta –Bits 6-9.

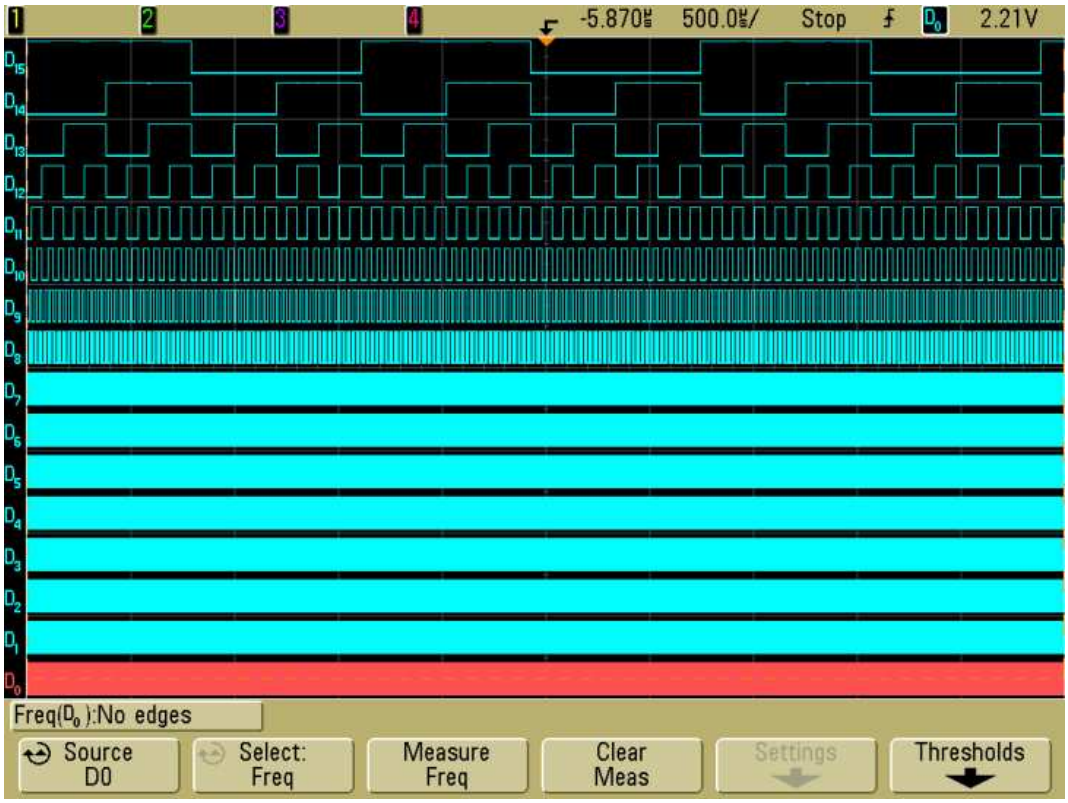


Figura 7-16. Resultados teste caixa-preta –Bits 10-15.

7.3 Aplicação da metodologia proposta

A seguir, aplica-se a metodologia proposta no mesmo projeto, verificando suas principais características. O fluxograma da metodologia proposta foi apresentado na seção 5.3, e será aplicado no projeto de um contador de 16 bits.

Seguindo o fluxograma proposto, primeiramente definem-se todos os requisitos do projeto. Estes requisitos já foram definidos durante o desenvolvimento do projeto anterior, e não serão alterados.

Como a metodologia prevê o uso da linguagem VHDL apenas como forma de descrição de circuitos em nível de portas lógicas (*gate level*), devemos proceder a escolha de uma arquitetura para o contador de 16 bits. Dentre as diversas arquiteturas de contadores existentes, escolheu-se a arquitetura apresentada na Figura 7-17. Finalizada a escolha da arquitetura, realizou-se toda documentação.

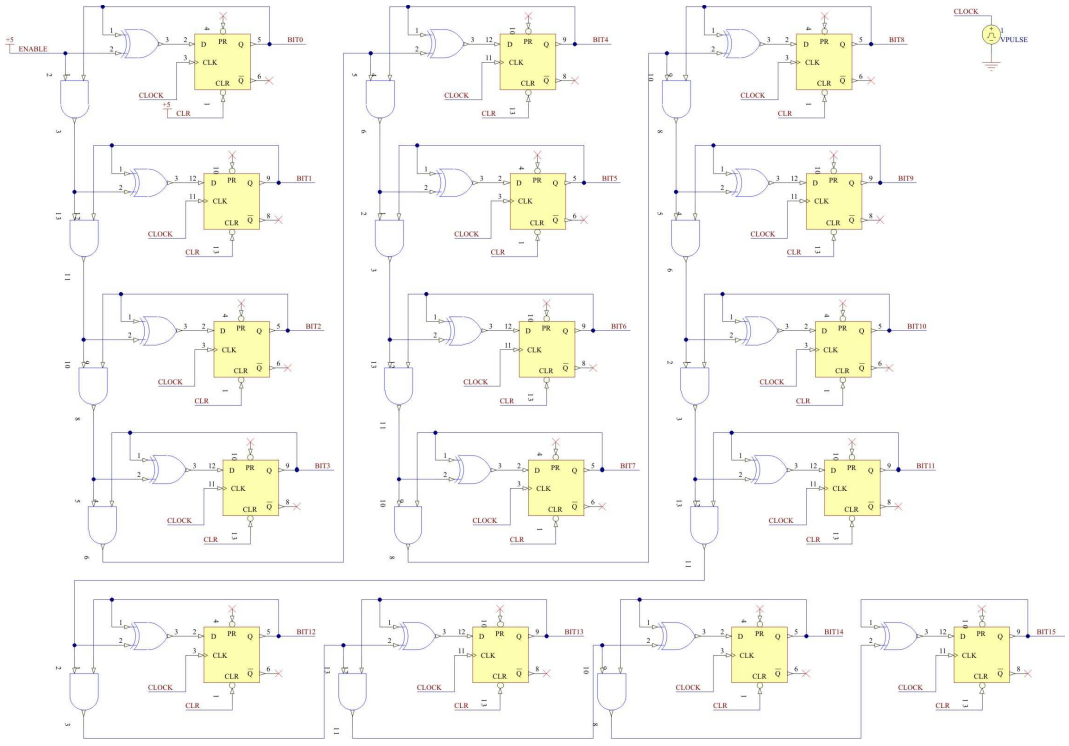


Figura 7-17. Arquitetura de contador proposta.

Finalizada a documentação da arquitetura escolhida, verifica-se a necessidade de alteração de alguns requisitos. Os requisitos podem ser alterados ou adicionados após a definição da arquitetura a ser utilizada.

No caso do contador de 16 bits proposto, foi necessário a adição de requisitos referentes à temporização interna da lógica utilizada. A Figura 7-18 apresenta uma visão ampliada da arquitetura proposta.

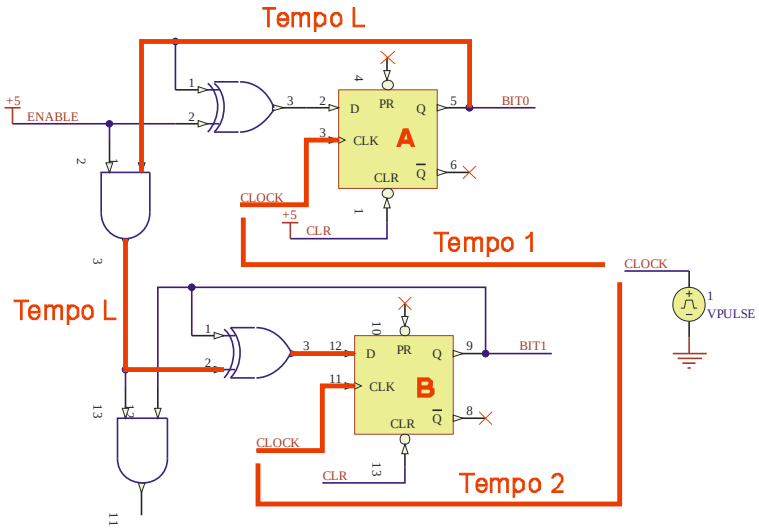


Figura 7-18. Figura ampliada da arquitetura proposta.

O funcionamento do contador proposto será assegurado somente se o atraso Tempo 2 for menor que Tempo 1 + Tco + Tempo L. O tempo Tco é o tempo *clock to data out*, que é o tempo entre a transição do *clock* e a saída do dado de entrada. O tempo Tco é de no máximo 0,427ns (QUICKLOGIC, 2007a). A Figura 7-19 ilustra a temporização esperada. Caso esta temporização não seja respeitada, como mostra a Figura 7-20, o contador apresentará um comportamento errôneo. Após a documentação deste novo requisito, passou-se a desenvolver o projeto do circuito digital. Realizou-se o projeto com o *software* Altium Designer, da empresa Altium Limited (2008).

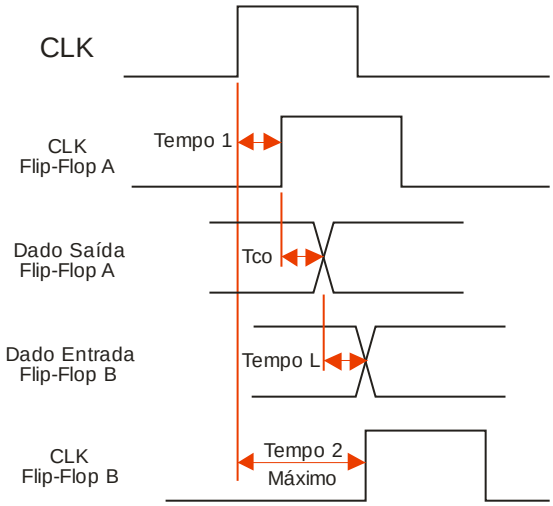


Figura 7-19. Temporização esperada.

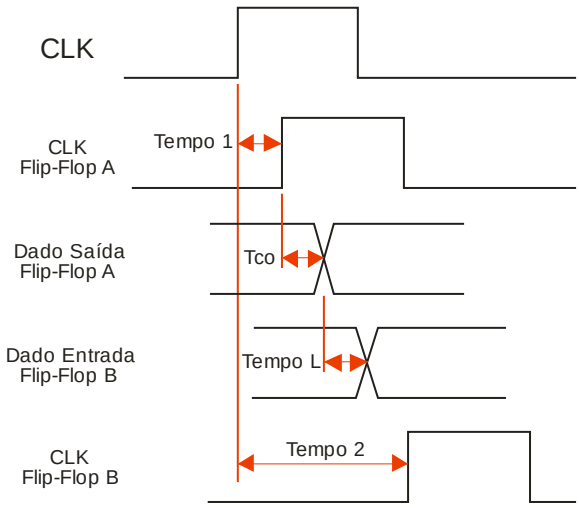


Figura 7-20. Temporização que acarreta erro.

Nesta plataforma, é possível realizar o projeto do sistema digital e sua simulação. A arquitetura escolhida foi transferida para um circuito compatível com as células lógicas da FPGA a ser utilizada. Cada porta E apresentada na arquitetura do contador foi substituída por uma célula lógica configurada como mostra a Figura 7-21 e cada porta XOR apresentada na arquitetura do contador foi substituída por uma célula lógica configurada como mostra a Figura 7-22.

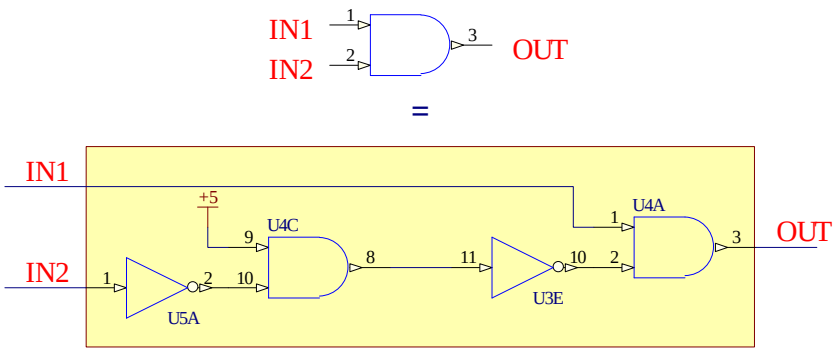


Figura 7-21. Porta E em nível de portas lógicas (*gate level*).

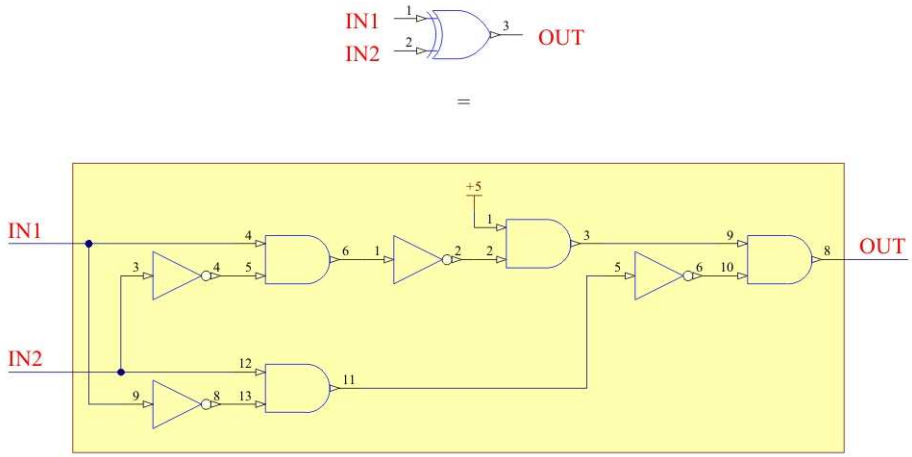


Figura 7-22. Porta XOR em nível de portas lógicas (*gate level*).

Esta implementação é apresentada na Figura 7-27 e visa utilizar os mesmos circuitos lógicos disponíveis em nível de portas lógicas (*gate level*). Os nomes em vermelho indicam a nomenclatura utilizada pelo fabricante Quicklogic (2007a) nas células lógicas. O índice de cada nome indica a qual célula lógica pertence a ligação, ou seja, a ligação A1(0) pertence a uma célula lógica diferente da ligação A1(1). Devido ao grande número de esquemáticos gerados, a Figura 7-27 apresenta apenas uma parte do esquemático do contador implementado. Todo o esquemático do contador é apresentado no Apêndice D.

Com a implementação do contador, seguindo o fluxograma de desenvolvimento, passa-se a realizar a simulação funcional. Os resultados desta simulação são apresentados a seguir. A Figura 7-23 apresenta o resultado da

simulação do bit 0 até o bit 4. A Figura 7-24 apresenta o resultado da simulação do bit 5 até o bit 9. A Figura 7-25 apresenta o resultado da simulação do bit 10 até o bit 14. A Figura 7-26 apresenta o resultado da simulação do bit 15.

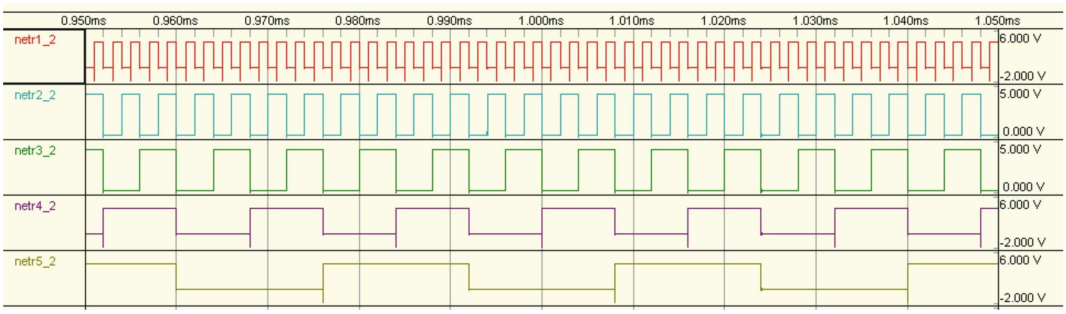


Figura 7-23. Simulação funcional - bit 0-4.

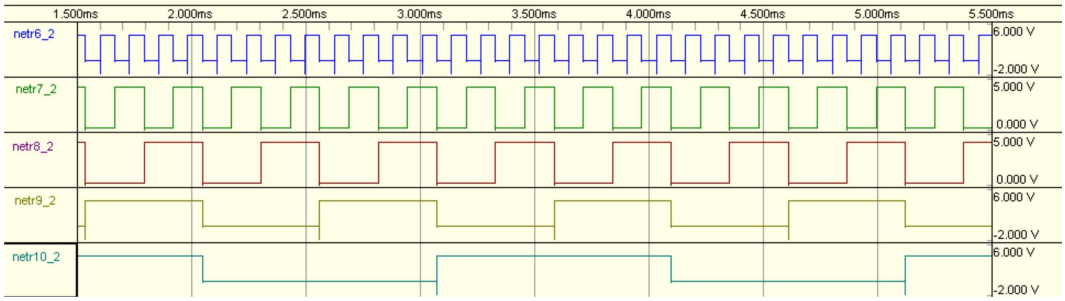


Figura 7-24. Simulação funcional - bit 5-9.

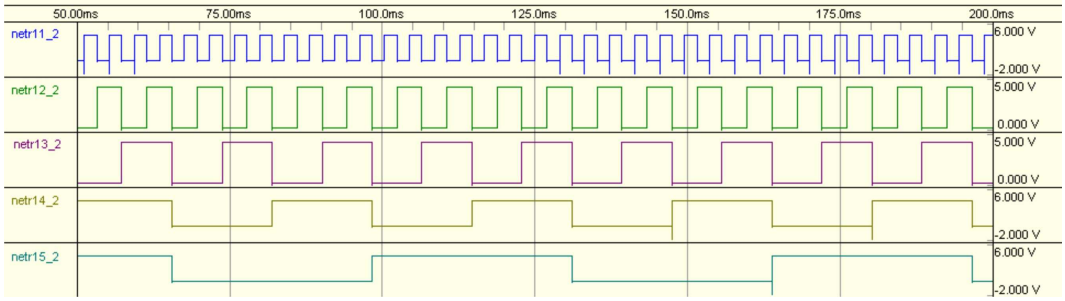


Figura 7-25. Simulação funcional - bit 10-14.

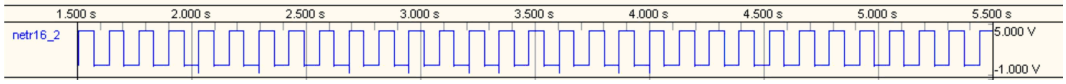


Figura 7-26. Simulação funcional - bit 15.

Verificou-se que o contador implementado funciona corretamente. Verificou-se porém que o tempo gasto para a realização da simulação do contador é um pouco elevado na ferramenta Altium Designer.

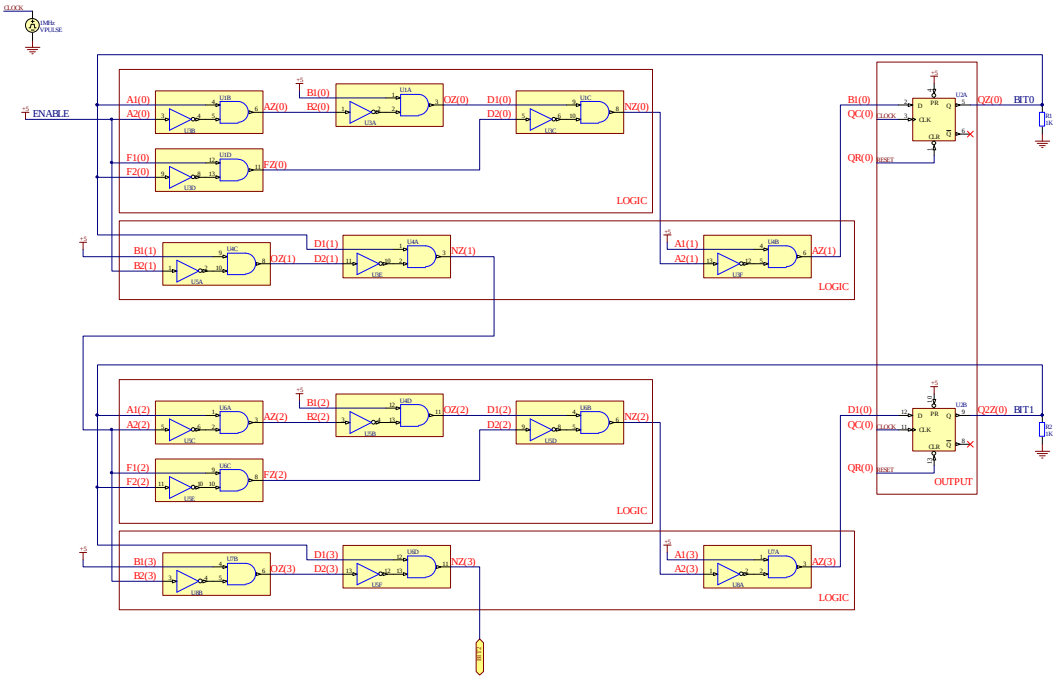


Figura 7-27. Circuito em células lógicas.

Seguindo o fluxograma proposto, realizou-se a conversão do circuito implementado para a linguagem VHDL. O arquivo VHDL do contador implementado em nível de portas lógicas (*gate level*) é apresentado no Apêndice E. A estrutura utilizada em VHDL na transcrição do contador é específica para implementação em nível de portas lógicas (*gate level*).

Finalizada a conversão, realiza-se o processo de síntese. Devido à grande extensão do esquemático RTL fornecido pela ferramenta de síntese, apenas uma parte do circuito é apresentado na Figura 7-28. Também devido à grande extensão do esquemático tecnológico disponibilizado pela ferramenta de síntese, apenas uma parte do circuito é apresentado na Figura 7-29. Verificou-se que os dois circuitos gerados são idênticos. Isto mostra que todo circuito definido no arquivo VHDL será implementado, incluindo redundâncias caso sejam implementadas. O relatório de área disponibilizado pela ferramenta de síntese é apresentado no Anexo D.

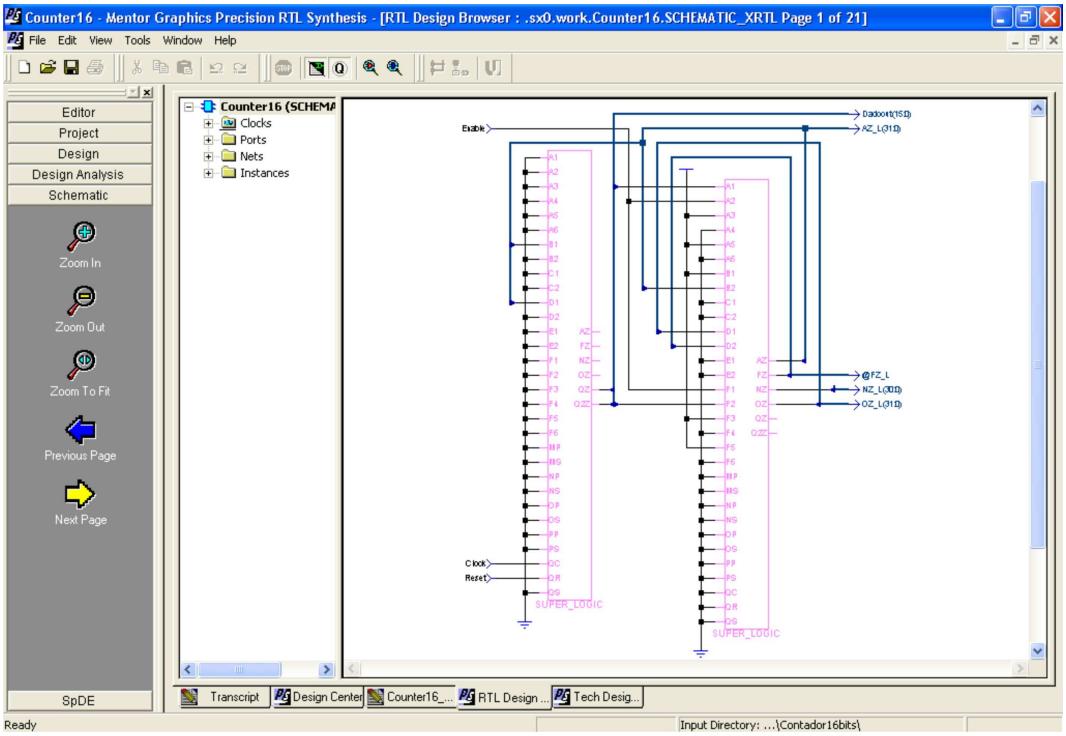


Figura 7-28. Esquemático *Register Transfer Level* RTL.

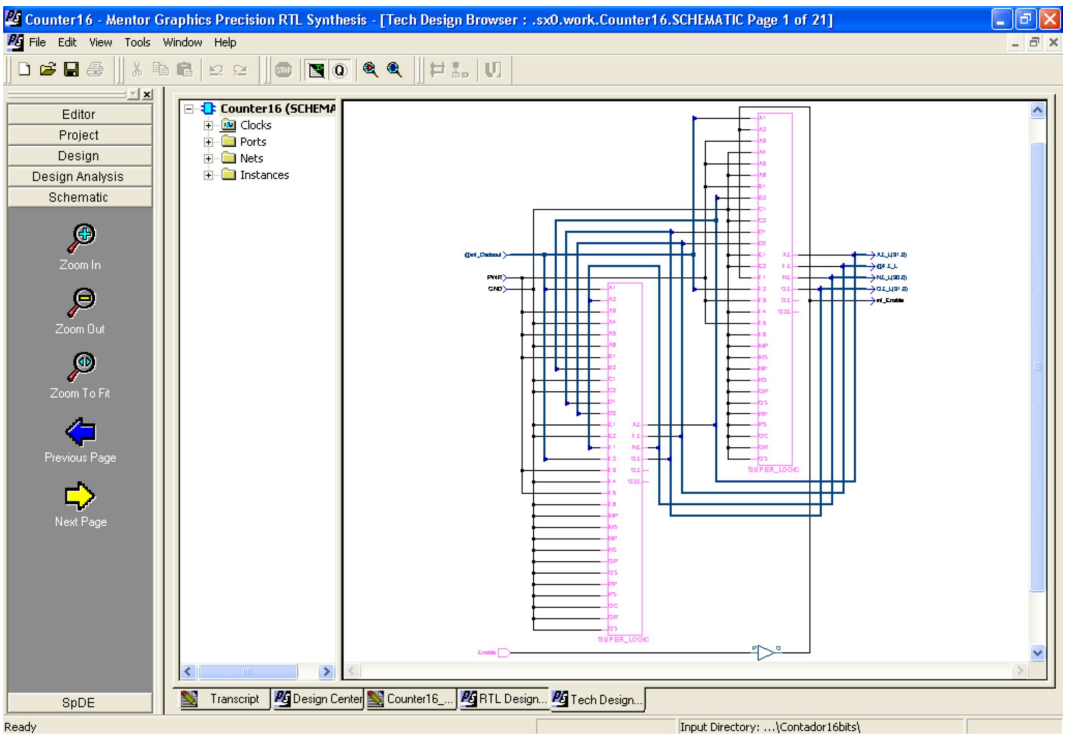


Figura 7-29. Esquemático tecnológico.

Este relatório não pode ser utilizado para comparação entre as duas metodologias, por que a ferramenta de síntese considera o circuito transcrito em VHDL como blocos caixa-preta, chamados de SUPER_LOGIC. Verificou-se

também que a documentação impressa ficará comprometida devido ao tamanho de todos os circuitos gerados.

Dando seqüência ao fluxograma de desenvolvimento, passa-se para a ferramenta de *Place & Route*. O processo de *Place & Route* foi realizado sem a utilização de otimizadores lógicos. As regras geradas para a realização do *Place & Route* são apresentadas no Anexo E. Os pinos de E/S foram escolhidos de acordo com o hardware disponível para a realização de teste caixa-preta e são os mesmos utilizados na verificação da metodologia clássica. A Figura 7-30 apresenta a janela de edição de regras, especificamente a de posicionamento por janelas, já apresentada no capítulo 4.

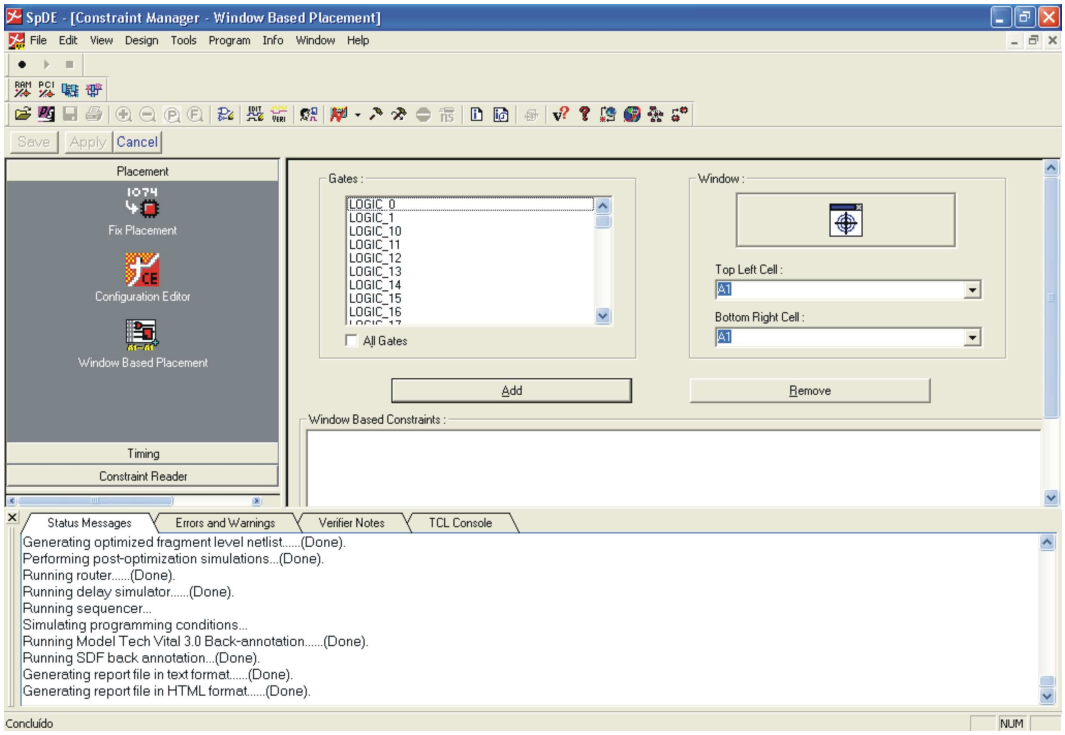


Figura 7-30. Janela de edição de regras.

Nesta janela verifica-se que os nomes das *nets* gerados são exatamente iguais aos definidos na transcrição VHDL, permitindo assim a escolha e o posicionamento independente de qualquer parte do circuito. As características dos pinos de E/S foram mantidas em seus valores *default.* Os processos realizados pela ferramenta de *Place & Route* são os mesmos apresentados na Figura 7-7.

Após a realização do processo de *Place & Route*, um relatório contendo um resumo é apresentado ao usuário. O resumo apresentado para este projeto é apresentado no Anexo F. Em seguida verificou-se o circuito montado no hardware reconfigurável pela ferramenta de *Place & Route*. A Figura 7-31 apresenta uma visão geral do circuito implementado. Devido ao elevado número de células lógicas presentes no hardware reconfigurável, a visão geral não permite o detalhamento do circuito gerado. Após o levantamento manual do circuito implementado, verificou-se que ele é exatamente o mesmo circuito transcrito em VHDL. Os nomes das *nets* são relacionados ao descrito em VHDL, mantendo assim uma coerência na implementação. Esta característica facilita qualquer verificação de temporização interna e garante presença de redundância caso esta seja implementada.

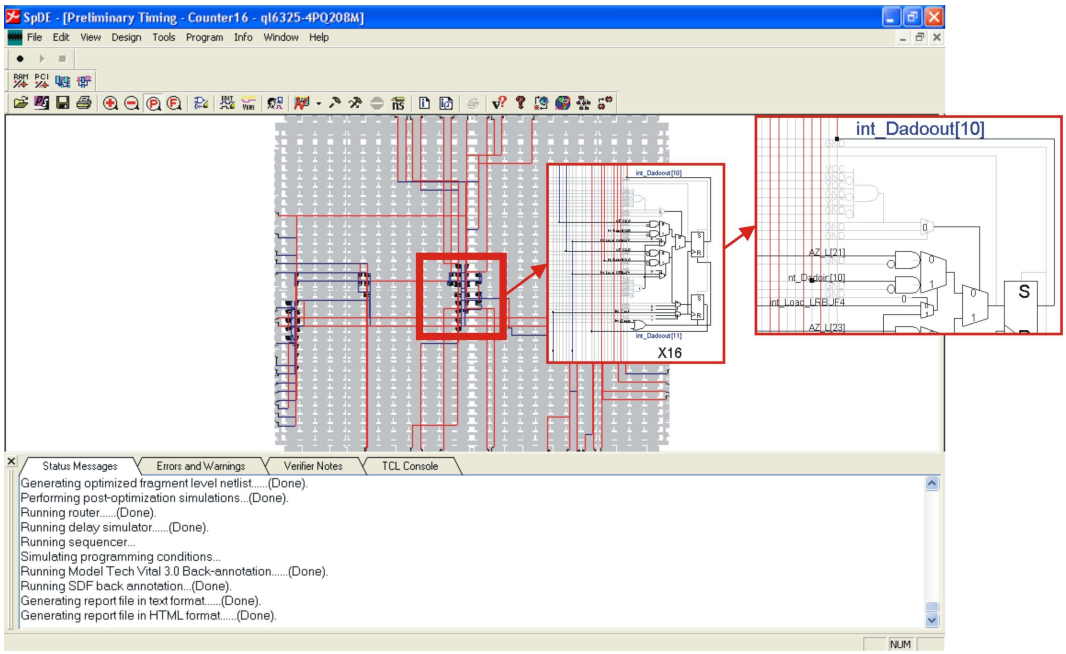


Figura 7-31. Visão geral do circuito implementado.

Nesta etapa do projeto, os requisitos de temporização interna são verificados. Verificou-se que a ferramenta *Path Analyzer* fornece as temporizações de todo o caminho da célula lógica. Os nomes utilizados estão ligados aos nomes utilizados no arquivo VHDL. Utilizando a Figura 7-27 como

referência, verifica-se o tempo gasto para o sinal sair do primeiro *flip-flop* (QZ(0)) e chegar até a entrada do próximo *flip-flop* (AZ(3)). Na ferramenta *Path Analyzer* escolhe-se como sinal de origem o sinal int_Dadoout(0) e o sinal AZ_L(3), como mostra a Figura 7-32. O sinal int_Dadoout(0) é o nome interno do sinal QZ(0), por se tratar de um pino de E/S.

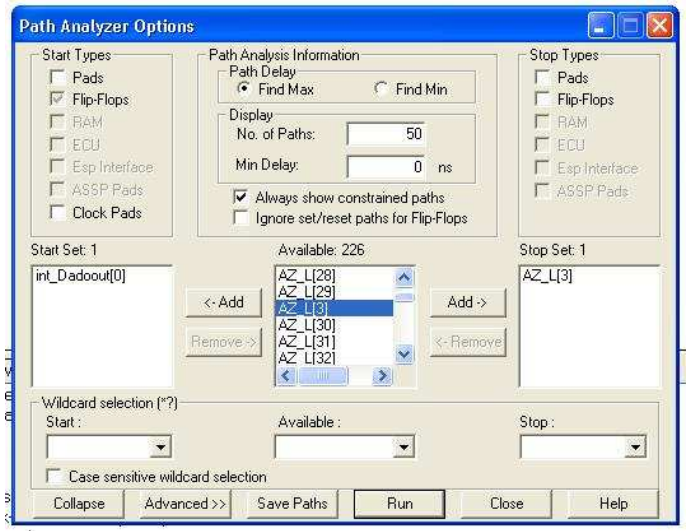


Figura 7-32. Janela da ferramenta *Path Analyzer*.

Como resultado, tem-se duas temporizações, como mostra a Figura 7-33. Estas temporizações se referem aos caminhos existentes. Analisando a Figura 7-27, verifica-se que o sinal de saída do *flip-flop* do bit 0, após ser enviado para outra célula lógica, entra em duas portas distintas, sendo elas a porta A2(2) e F1(2). As duas temporizações se referem cada uma a uma destas portas.

O primeiro valor de temporização, de cor azul, de 0,51ns, se refere ao tempo necessário para a saída de um dado no *flip-flop* do bit 0 após o recebimento do sinal de *clock*. Este tempo é chamado de Tco, como citado anteriormente, e seu valor está acima do valor máximo definido no *datasheet* do componente. Isto mostra que a ferramenta apresenta um valor próximo, mas não exato das temporizações internas.

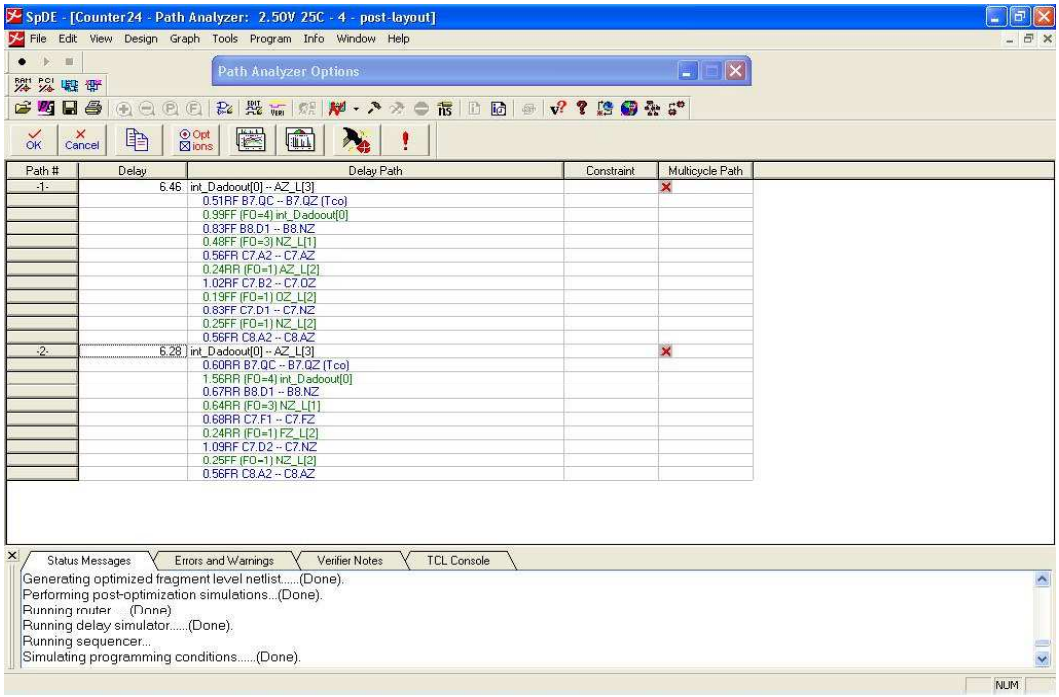


Figura 7-33. Janela com resultados da ferramenta *Path Analyzer*.

É necessário portanto, realizar uma verificação das temporizações internas, buscando identificar se as temporizações apresentadas pela ferramenta são equivalentes às temporizações reais. O valor em verde, de 0,99ns, é o tempo gasto pelo sinal para sair do *flip-flop* até chegar a próxima entrada, no caso a entrada D1(1). O próximo valor de temporização, de cor azul, de 0,83ns, se refere ao tempo gasto pelo sinal para entrar na porta D1(1) e sair na porta NZ(1). O valor em verde, de 0,48ns, é o tempo gasto pelo sinal para sair da porta NZ(1) até chegar à próxima entrada, no caso a entrada A2(1). O próximo valor de temporização, de cor azul, de 0,56ns, se refere ao tempo gasto pelo sinal para entrar na porta A2(2) e sair na porta AZ(2). O valor em verde, de 0,24ns, é o tempo gasto pelo sinal para sair da porta AZ(2) até chegar a próxima entrada, no caso a entrada B2(2). O próximo valor de temporização, de cor azul, de 1,02ns, se refere ao tempo gasto pelo sinal para entrar na porta B2(2) e sair na porta OZ(2). O valor em verde, de 0,19ns, é o tempo gasto pelo sinal para sair da porta OZ(2) até chegar a próxima entrada, no caso a entrada D1(2).

O próximo valor de temporização, de cor azul, de 0,83ns, se refere ao tempo gasto pelo sinal para entrar na porta D1(2) e sair na porta NZ(2). O valor em verde, de 0,25ns, é o tempo gasto pelo sinal para sair da porta NZ(2) até chegar a próxima entrada, no caso a entrada A2(3). O próximo valor de temporização, de cor azul, de 0,56ns, se refere ao tempo gasto pelo sinal para entrar na porta A2(3) e sair na porta AZ(3), que é a entrada do segundo *flip-flop*. A temporização total envolvida é de 6,46ns. Seguindo a mesma análise, a segunda temporização de 6,28ns é o tempo gasto pelo sinal seguindo o caminho da porta F2(2).

A próxima verificação deve ser a temporização do sinal de *clock*. Para esta verificação, utiliza-se a mesma ferramenta utilizada anteriormente. O tempo gasto pelo sinal de *clock* para chegar até o primeiro *flip-flop* foi de 3,72ns, como mostra a Figura 7-34. O tempo gasto pelo sinal de *clock* até o segundo *flip-flop* foi de 3,72ns, como mostra também a Figura 7-34.

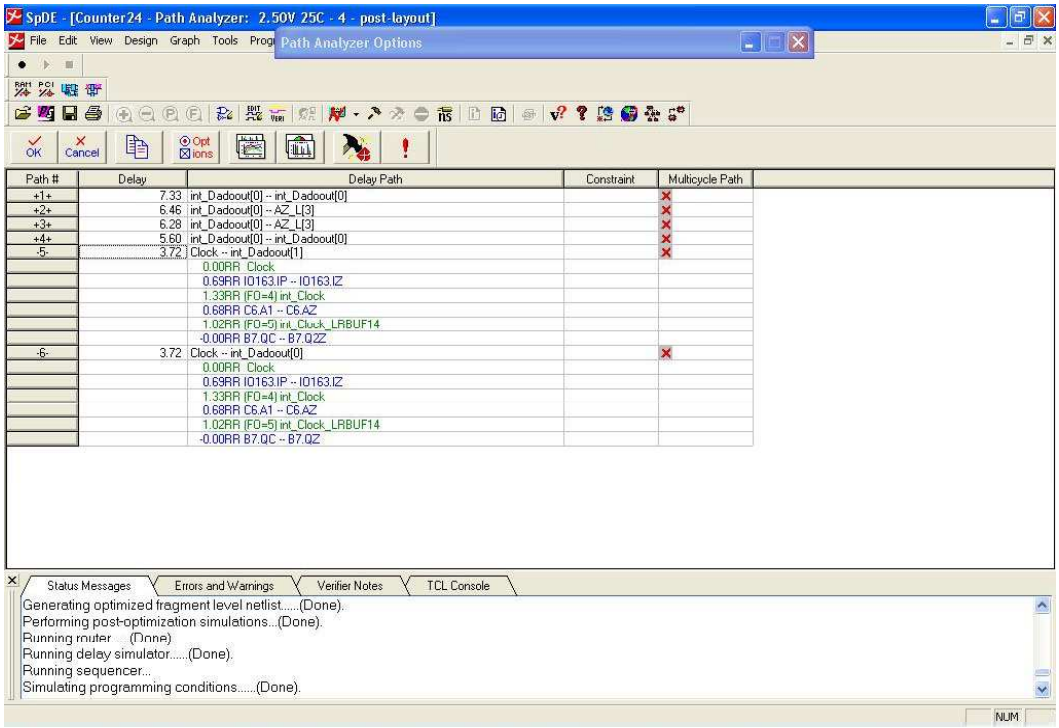


Figura 7-34. Janela com resultados da ferramenta *Path Analyzer*.

Com a soma das temporizações, vê-se que a soma Tempo 1 + Tco + Tempo L é maior que o atraso Tempo 2. Com isso, conclui-se que a contagem dos dois

primeiros *bits* do contador será correta, pois cumpre a especificação de temporização interna. Além disso, verificamos a presença de uma grande margem do cumprimento deste requisito. Para garantir o funcionamento do contador completo, esta mesma análise deve ser aplicada a todos os *bits*. Desta forma, garantem¿-se todas as temporizações internas de todo o sistema. Esta abordagem é bastante trabalhosa, mas produz um sistema bastante robusto e confiável. Qualquer alteração necessária para garantir temporizações pode ser feita em nível de portas lógicas (*gate level*) sem a necessidade de um novo levantamento do circuito gerado, pois a ferramenta de síntese irá implementar exatamente o que foi descrito em VHDL em nível de portas lógicas (*gate level*).

Seguindo o fluxograma proposto, apresentado na Figura 6-1, e reapresentado na Figura 7-35, cria-se o procedimento de teste para o contador de 16 bits. No caso do contador, o procedimento de teste adotado é o mesmo utilizado na simulação pós-síntese, apresentada no capítulo 4.

Desta forma, utilizou-se o mesmo *testbench* apresentado no Apêndice B para a realização da simulação. A simulação utiliza o arquivo gerado pela ferramenta *Place & Route* com extensão .SDF, que contém todas as temporizações internas estimadas. Os resultados da simulação com resolução de 1ns são apresentados na Figura 7-36, Figura 7-37 e Figura 7-38. Na Figura 7-36 apresenta-se o funcionamento dos bits 0 até 5. Na Figura 7-37 apresenta-se o funcionamento dos bits 6 até 11. Na Figura 7-38 apresenta-se o funcionamento dos bits 12 até 15.

Como verificação da ferramenta QuickWorks™, levantou-se o atraso entre o sinal de *clock* e a saída do bit 0 do contador. A Figura 7-39 apresenta uma visão ampliada dos resultados da simulação, mostrando a temporização obtida no bit 0,

que é de 3ns. Vê-se que o tempo apresentado pela ferramenta *Path Analyzer* é idêntico ao apresentado na simulação. Devido ao problema de resolução, discutida anteriormente, não podemos comprovar o valor exato, mas o valor obtido se aproxima em muito ao valor apresentado.

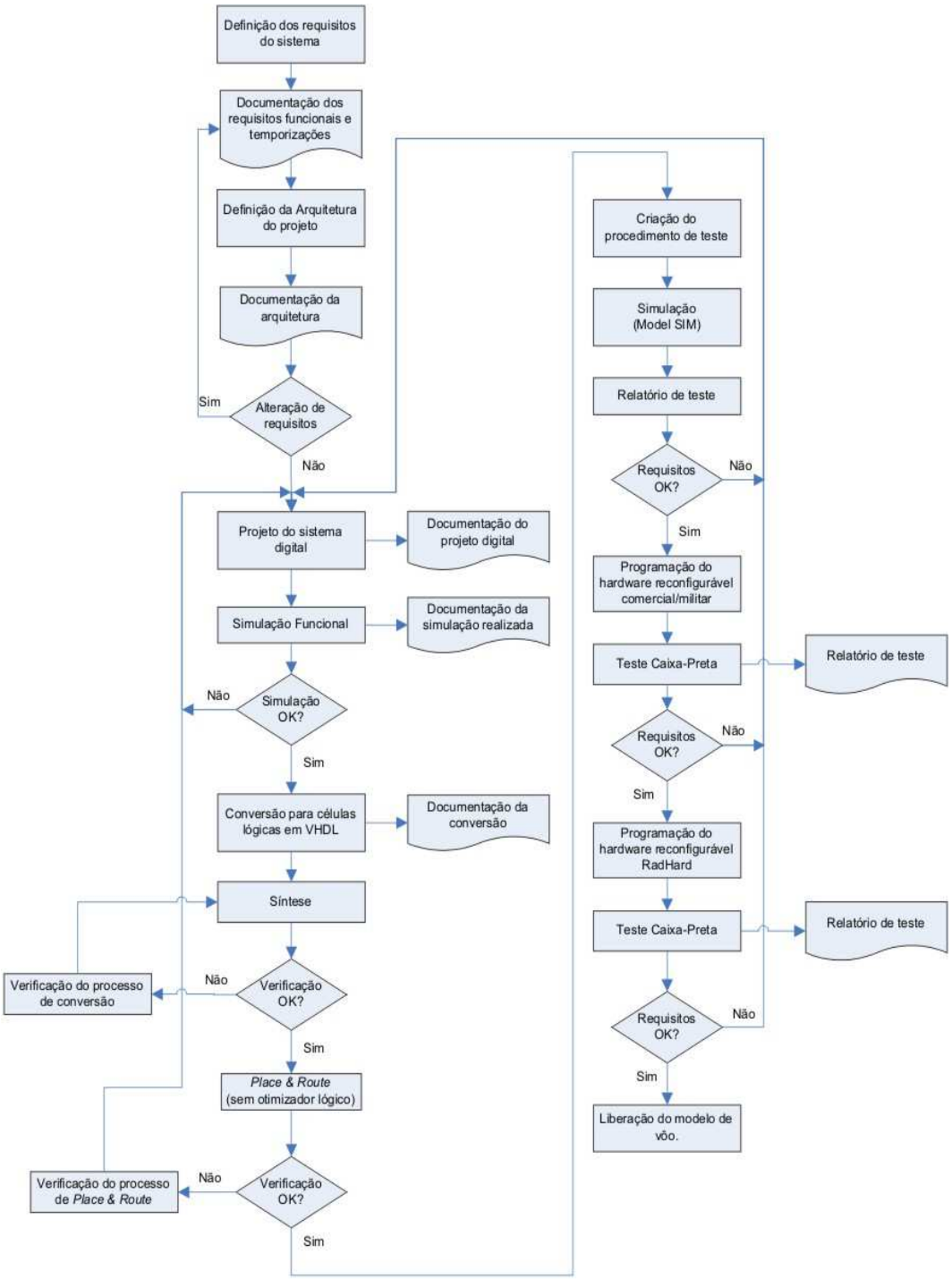


Figura 7-35. Fluxograma da metodologia proposta

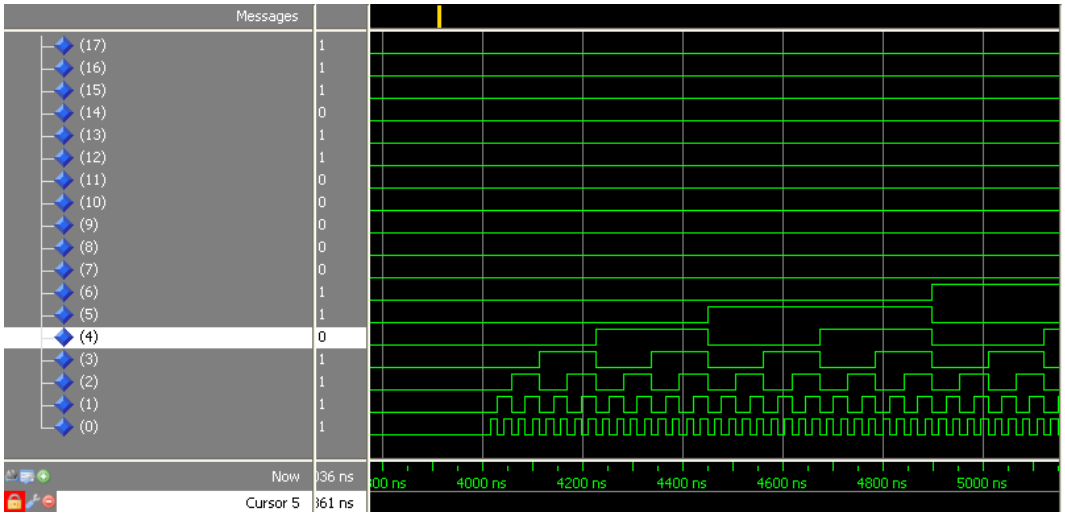


Figura 7-36. Simulação do contador – bits 0-5.

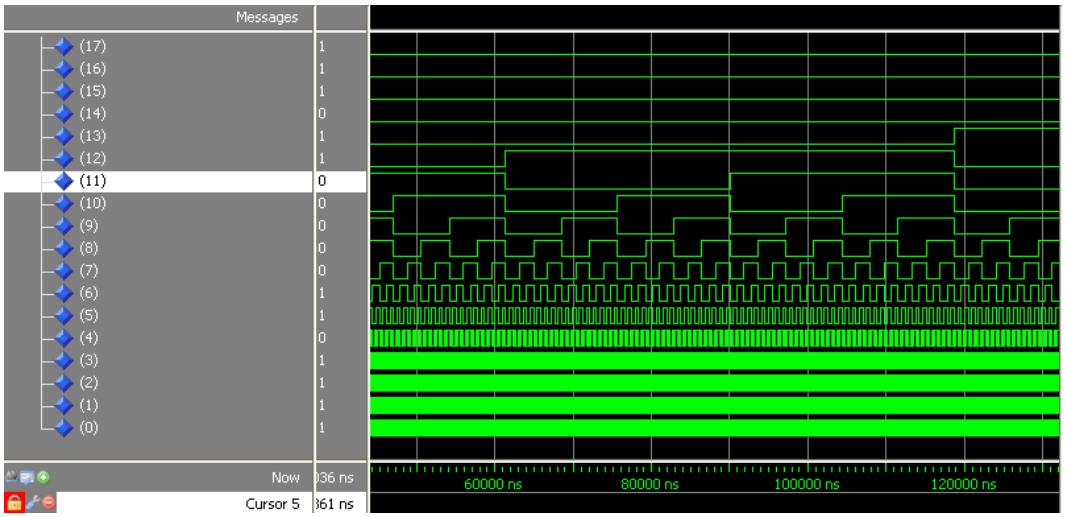


Figura 7-37. Simulação do contador – bits 6-11.

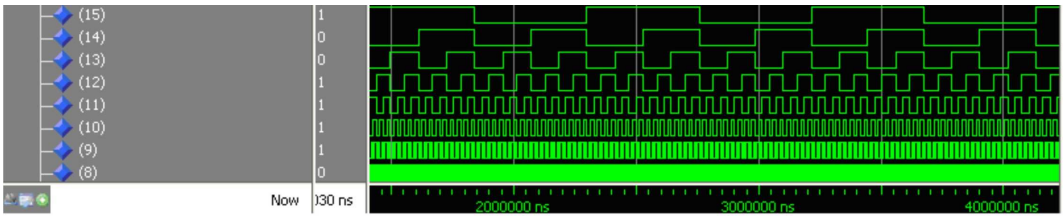


Figura 7-38. Simulação do contador – bits 12-15.

Com o término da simulação, passa-se a realizar o relatório de testes. Dando seqüência ao fluxograma, com o cumprimento dos requisitos de projeto, passa-se para a programação do hardware reconfigurável. O hardware reconfigurável utilizado foi o QL6325 da Quicklogic (2007a). Este hardware reconfigurável será utilizado em uma placa já existente da MUXCAM para a realização do teste caixa-preta.

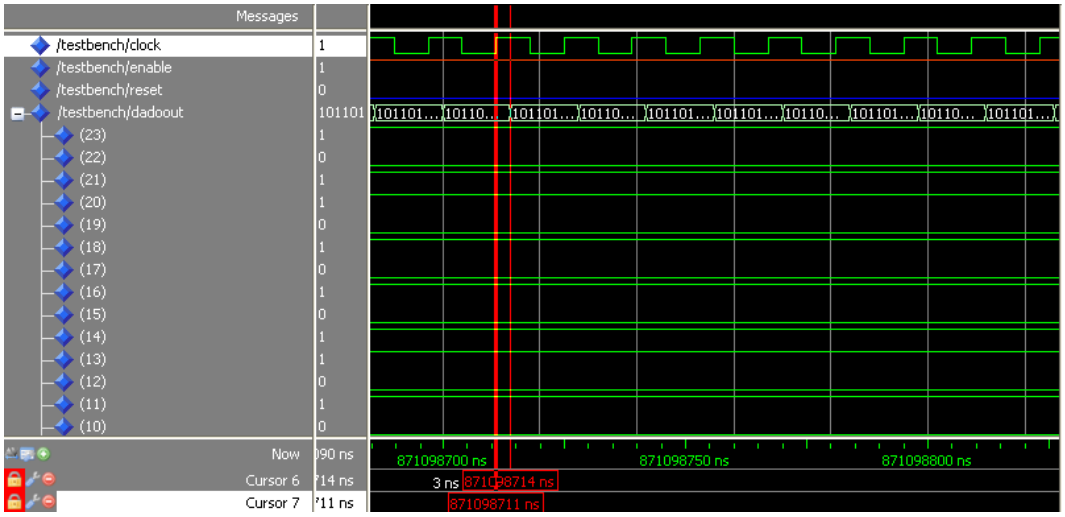


Figura 7-39. Visão ampliada dos resultados da simulação.

A placa com a FPGA que foi utilizada para o teste caixa-preta é apresentada na Figura 7-11. O fluxograma do teste caixa-preta realizado é igual ao apresentado na Figura 7-12.

Os resultados obtidos são apresentados a seguir. A Figura 7-40 apresenta o atraso do sinal de *clock* em relação à alteração do primeiro bit do contador. Este atraso é de 7 ns, valor muito diferente do valor apresentado pela ferramenta de simulação. A Figura 7-41 apresenta o funcionamento dos bits0-5. A Figura 7-42 apresenta o funcionamento dos bits6-9 e a Figura 7-43 apresenta o funcionamento dos bits10-15. Verifica-se que o funcionamento do contador está correto, assim como na metodologia clássica. Com os resultados obtidos, verificou-se o funcionamento e conseqüentemente o cumprimento de todos os requisitos do sistema, com exceção das temporizações internas, que são garantidas por projeto.

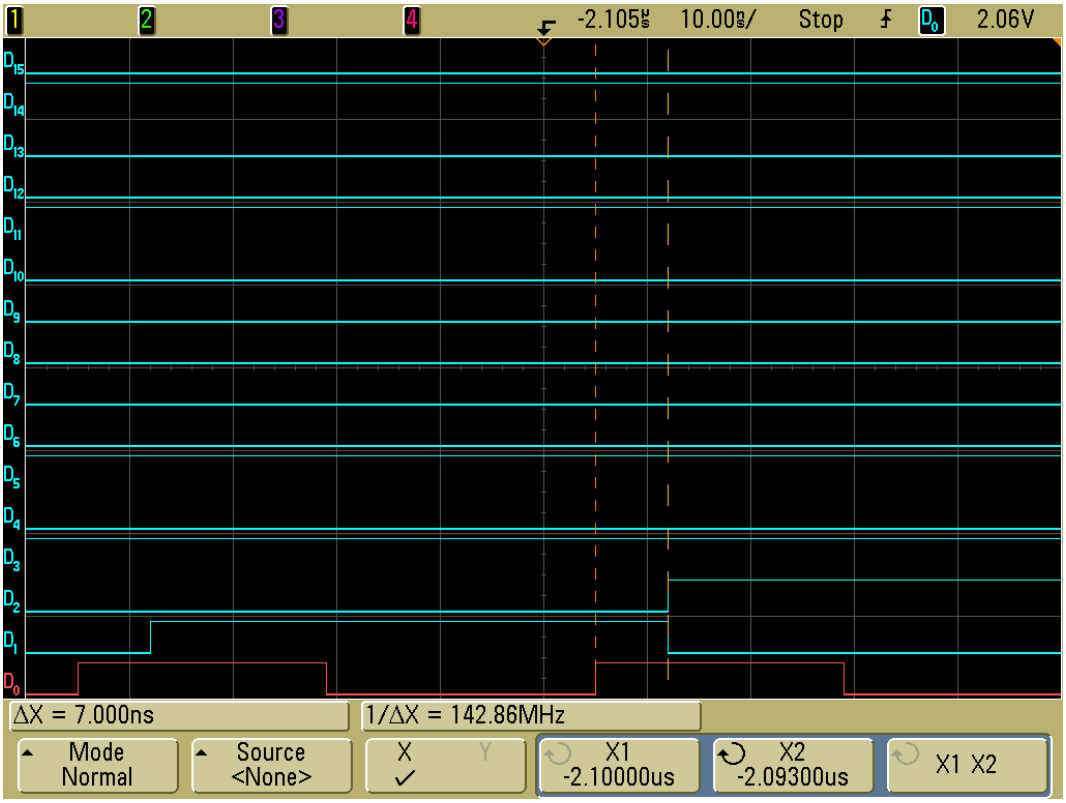


Figura 7-40. Resultados teste caixa-preta – Atraso do sinal de *clock* para o primeiro bit do contador.

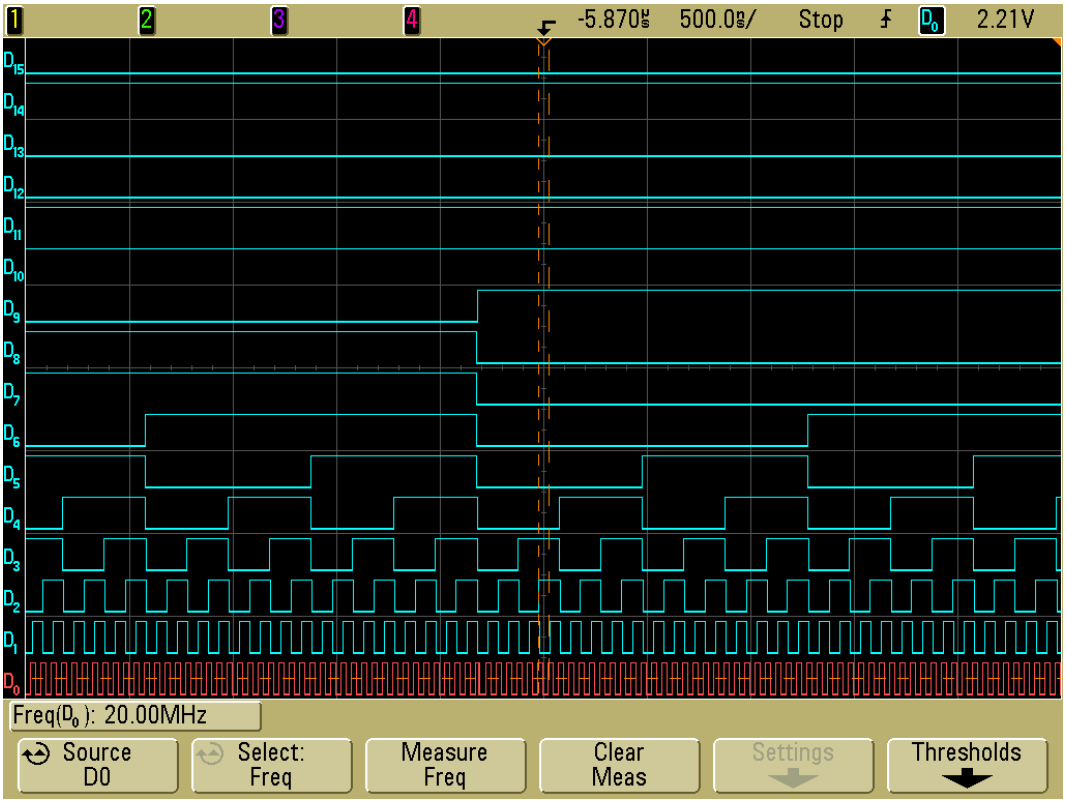


Figura 7-41. Resultados teste caixa-preta –Bits 0-5.

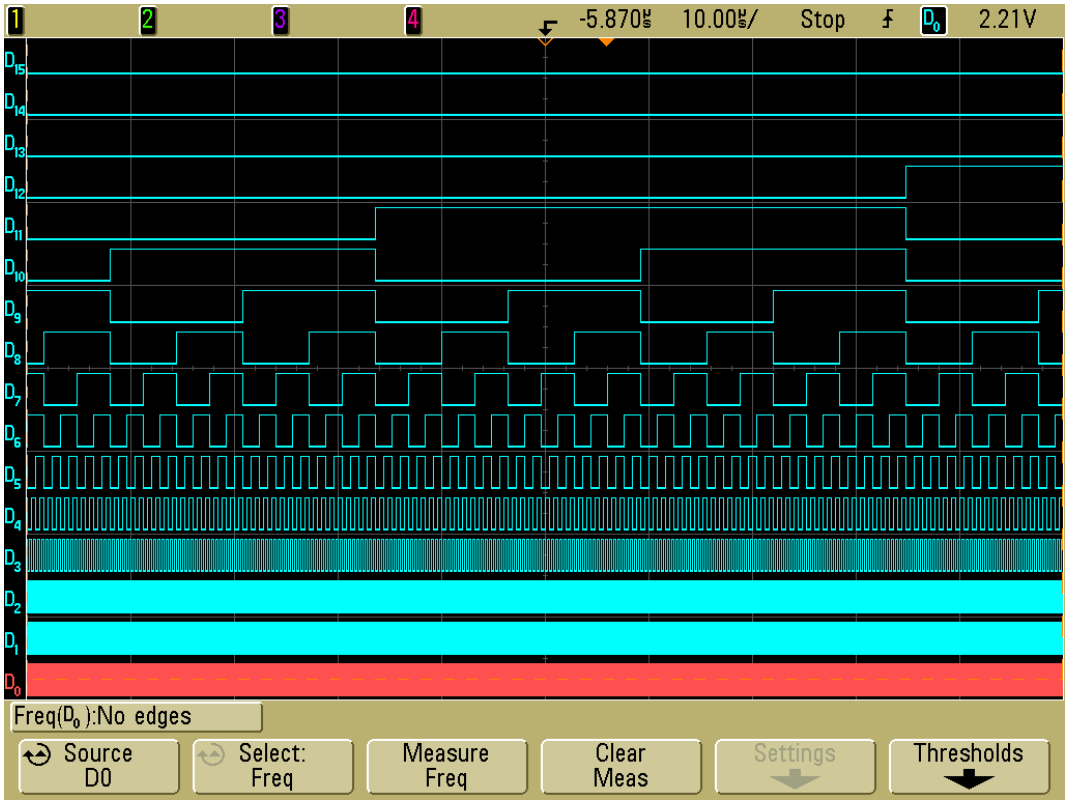


Figura 7-42. Resultados teste caixa-preta –Bits 6-9.

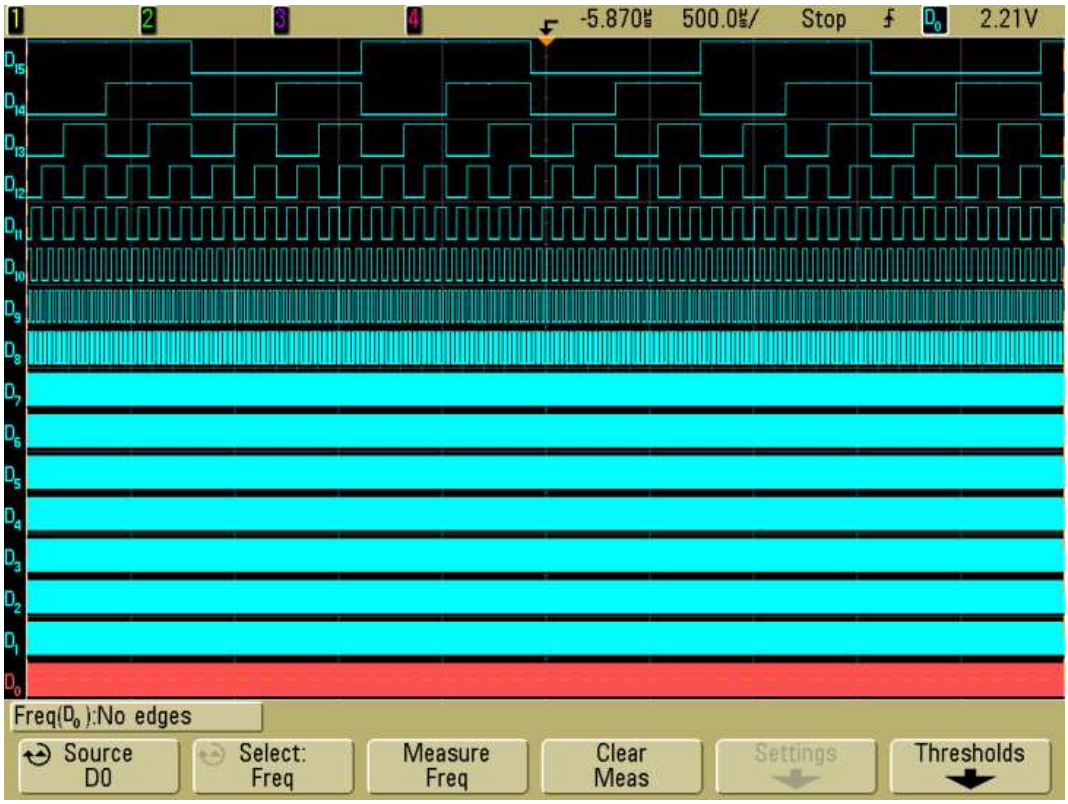


Figura 7-43. Resultados teste caixa-preta –Bits 10-15.

Seguindo o fluxograma apresentado na Figura 7-35, seria realizada a programação do hardware reconfigurável RadHard da empresa Aeroflex (2007b),

que não foi realizada neste trabalho devido à impossibilidade de compra em tempo hábil do dispositivo. A compra de componentes RadHard está sujeita as regulamentações ITAR, sendo estas impostas pelo governo norte americano para exportação de tecnologias estratégicas. Até o momento, o INPE, que é responsável pela compra das FPGAs, possui uma permissão de importação de componentes RadHard do fabricante Aeroflex.

7.4 Conclusões

Verifica-se que a utilização da metodologia clássica permite o desenvolvimento rápido de qualquer sistema através da descrição direta em linguagem VHDL, reduzindo o tempo de colocação de produtos no mercado, ao contrário da metodologia proposta. Em equipamentos comerciais, o tempo de desenvolvimento e laçamento de um novo produto é crucial, portanto, a verificação do software gerado pela ferramenta de síntese não é muito profunda, porém, em sistemas de alto custo e alta confiabilidade, a verificação do circuito criado pela ferramenta de síntese é crítica, e se torna inviável em sistemas de grande porte desenvolvidos somente em linguagem VHDL. O esforço necessário para o levantamento do circuito criado é bastante grande, sem levar em conta as possíveis iterações que não foram necessárias no projeto do contador de 16 bits, mas que certamente serão em projetos de grande porte.

Na metodologia clássica não está prevista nenhuma especificação por projeto das temporizações internas, uma vez que a arquitetura interna implementada é desconhecida até o levantamento manual do circuito implementado. Na metodologia proposta, a arquitetura interna, com atrasos e temporizações está presente desde o início até o final do ciclo de desenvolvimento. Na metodologia clássica, somente com o levantamento manual

do circuito gerado e a verificação da arquitetura utilizada é que se torna possível gerar especificações de temporizações internas e a utilização de todas as ferramentas fornecidas pela plataforma QuickWorks™.

Qualquer alteração de *software* na metodologia clássica deve ser realizada em linguagem VHDL, o que acarreta em uma alteração significativa do circuito implementado pela ferramenta de síntese. Na metodologia proposta, a alteração do circuito digital não implica em alterações significativas, pois o circuito se mantém idêntico até a gravação do software no *hardware* reconfigurável.

8 Conclusões

Conclui-se que a utilização da metodologia clássica permite o desenvolvimento rápido do sistema através da descrição direta em linguagem VHDL, porém, a verificação do circuito criado pela ferramenta de síntese é praticamente inviável em sistemas de grande porte. O esforço necessário para o levantamento do circuito é bastante grande..

Na metodologia clássica não está prevista nenhuma especificação por projeto das temporizações internas, uma vez que a arquitetura interna implementada é desconhecida até o levantamento manual do circuito implementado. Conclui-se que somente com o levantamento manual do circuito gerado e a verificação da arquitetura utilizada é que se torna possível gerar especificações de temporizações internas e a utilização de todas as ferramentas fornecidas pela plataforma QuickWorks™.

Qualquer alteração de *software* na metodologia clássica deve ser realizada em linguagem VHDL, o que acarreta em uma alteração significativa do circuito implementado pela ferramenta de síntese. Este comportamento foi detectado em experiências de desenvolvimento anteriores e que não apareceram durante o desenvolvimento do contador de 16 bits, justamente devido à não necessidade de alterações ao longo do desenvolvimento. Concui-se que o resultado direto de uma alteração é a necessidade de um novo levantamento de todo o circuito implementado.

Na metodologia proposta, verifica-se que a documentação impressa de todos os circuitos gerados, principalmente pela ferramenta de síntese, é prejudicada, devido à elevada quantidade de esquemáticos gerados. Conclui-se que o processo de documentação é problemático na metodologia proposta.

Na metodologia proposta, todo o circuito descrito em VHDL em nível de portas lógicas (*gate level*) é implementado pela ferramenta de síntese. Durante o processo de *Place & Route*, o circuito implementado também é mantido. Assim, conclui-se que todo o circuito descrito será implementado no hardware reconfigurável, inclusive redundâncias.

O processo de simulação é facilitado, pois os nomes das *nets* utilizados durante todo processo de desenvolvimento são os mesmos. Desta maneira, conclui-se que a utilização das ferramentas de temporização fornecidas na plataforma QuickWorks™ é facilitada durante todo ciclo de desenvolvimento. Com a utilização destas ferramentas, as margens de projeto para a arquitetura definida são garantidas.

Conclui-se que na metodologia proposta, a arquitetura utilizada é definida pelo usuário e não pela ferramenta de síntese, como ocorre na metodologia clássica, e esta arquitetura se mantêm durante todo o ciclo de desenvolvimento.

9 Trabalhos futuros

Sugerem-se as seguintes atividades para trabalhos futuros:

- aplicação da metodologia em um sistema de grande porte como o da MUXCAM, identificando outros pontos fortes e fracos da metodologia.
- o estudo da ferramenta de síntese, buscando identificar formas de interação com o usuário que permita a especificação de temporizações e arquiteturas utilizadas.
- o estudo da ferramenta de simulação ModelSim para identificar seu comportamento em resoluções de simulação diferentes, assim como a verificação das diferenças entre as temporizações de simulação e encontradas no hardware reconfigurável.
- o estudo dos arquivos *back annotation* fornecidos pela ferramenta QuickWorks™ com objetivo de levantar possíveis causas relacionadas ao problema de resolução apresentado pela ferramenta de simulação.
- desenvolvimento de ferramentas de síntese voltadas para aplicações que exigem alta confiabilidade, agregando as vantagens do desenvolvimento em linguagem VHDL e a garantia de implementação da arquitetura definida pelo usuário.

10 Referências

ACTEL. ***RTSX-SU RadTolerant FPGAs (UMC).*** Disponível em: <http://www.actel.com/documents/RTSXSU_DS.pdf>. Acesso em: 01 dez. 2007.

AEROFLEX. ***Corporate Profile.*** Disponível em: <http://www.aeroflex.com/aboutus/profile.cfm>. Acesso em: 01 dez 2007a.

_____. ***UT6325 RadHard Eclipse FPGA Data Sheet***. Disponível em: <http://ams.aeroflex.com/ProductFiles/DataSheets/FPGA/RadHardEclipseFPGA.pdf>. Acesso em: 01 dez 2007b.

_____. ***RadHard FPGAs***. Disponível em: <http://ams.aeroflex.com/ProductPages/RH_fpga.cfm>. Acesso em: 21 jan 2008.

ALTERA. ***AHDL Examples.*** Disponível em: <http://www.altera.com/support/examples/ahdl/ahdl.html?GSA_pos=1&WT.oss_r=1&WT.oss=AHDL>. Acesso em: 20 abr 2008.

ALTIUM LIMITED. ***Altium Designer.*** Disponível em: <http://www.altium.com/Products/AltiumDesigner/>. Acesso em: 20 abr 2008.

BERG, M., ***VHDL Synthesis for High-Reliability Systems.*** MAPLD International Conference, 2004.

COCKBURN, A. **Characterizing people as non-linear, first-order components in *software* development,** 1999. Disponível em: <http://alistair.cockburn.us/index.php/Characterizing_people_as_non-linear,_first-order_components_in_*software*_development>, Acesso em: 20 out. 2007.

DENTRAN, M. ***Radiation effects on electronic components and circuits.*** Disponível em: <http://atlas.web.cern.ch/Atlas/GROUPS/FRONTEND/radhard.htm>. Acesso em: 20 out. 2007.

DOULOS HOME PAGE. ***The Designer's Guide to VHDL.*** Disponível em: <http://www.doulos.com/knowhow/vhdl_designers_guide/>. Acesso em: 01 dez. 2007.

EPIPHANIO, J. C. N. CBERS – **Satélite Sino-Brasileiro de Recursos Terrestres.** In:SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 12., 2005, Goiânia. **Anais...** São José dos Campos: INPE, 2005. p. 915-922. CD-ROM.

HOLMES-SIEDLE, A., ADAMS, L. **Handbook of radiation effects.** Ed. Oxford, 2002.

INSTITUTO NACIONAL DE PESQUISA ESPACIAIS. **Plano Diretor do INPE 2007-2011: planejamento estratégico do INPE,** São José dos Campos, 2007a. 38p.

_____.**CBERS – Galeria de fotos.** Disponível em: <http://www.cbers.inpe.br/pt/imprensa/fotografia2.htm>. Acesso em: 01 dez. 2007b.

_____.**CBERS – Galeria de fotos.** Disponível em: <http://www.cbers.inpe.br/?content=cameras3e4>. Acesso em: 01 dez. 2007c.

_____.**RB-DCS-0001 v03 Design and Construction Specification**, São José dos Campos, 2007d. 83p.

_____.**RBN-HDS-0014-02 CBERS 3&4 Multispectral Camera Subsystem (MUX) Specification**, São José dos Campos, 2004. 18p.

INTHURN, C. **Qualidade & teste de *software,*** Florianópolis: Visual Books, 2001.

KATZ, R. et al. **Current Radiation Issues for Programmable Elements and Devices.** NSREC Conference, 1998.

KATZ, R. B., ***VHDL Design Review and Presentation.*** MAPLD International Conference, 2004.

KATZ, R. B., ROSE, J. et al., ***Architecture of Field-Programmable Gate Arrays***. Proceedings of the IEEE, Vol. 81, No. 7, p. 1013-1028, Julho 1993.

LABEL, K. et al. **Commercial microelectronics technologies for applications in the satellite radiation environment,** 1999. Disponível em: http://flick.gsfc.nasa.gov/ radhome.htm. Acesso em: (Nov. 1999).

MENTOR GRAPHICS. ***ModelSim.*** Disponível em: <http://www.mentor.com/products/fpga_pld/simulation/modelsim_se/>. Acesso em: 20 abr 2008a.

_____. ***Precision RTL.*** Disponível em: <http://www.mentor.com/products/fpga_pld/simulation/modelsim_se/>. Acesso em: 20 abr 2008b.

NASA. ***Rosat Guest Observer Facility.*** Disponível em: <http://heasarc.gsfc.nasa.gov/docs/rosat/gallery/display/saa.html>. Acesso em: 20 abr 2008.

_____. ***Qualification Strategies of Field Programmable Gate Arrays (FPGAs) for Space Application.*** Outubro 2005.

NONEMACHER, M. L. **Comparação e avaliação entre o processo RUP de desenvolvimento de *software* e a metodologia *Extreme Programming***. 2003. 164 p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Santa Catarina, Florianópolis, 2003.

O'BRYAN, Martha; et al. ***Single Event Effect and Radiation Damage Results For Candidate Spacecraft***. NSREC Conference, 1998.

OPTO ELETRÔNICA S/A. **RBN-TRP-8015-02 - Descrição Geral do Projeto MUX,** São Carlos, 2006a. 244p.

_____.**RBN-TRP-8020/04 – Descrição do Projeto Eletrônico**, São Carlos, 2006b. 244p.

PRESSMAN, R. S. **Software engineering: A practitioner's approach.** Ed. McGraw-Hill, 2005.

PERRY, D. L. **VHDL**, McGraw-Hill, 1994.

QUICKLOGIC. ***Eclipse Family Data Sheet.*** Disponível em: <http://www.quicklogic.com/images/eclipse_family_DS.pdf>. Acesso em: 01 dez 2007a.

_____. ***ViaLink Technology.*** Disponível em: <http://www.quicklogic.com/home.asp?PageID=745&sMenuID=90&p1=90&p2=213>. Acesso em: 01 dez 2007b.

_____. ***Development Software*** Disponível em: <http://www.quicklogic.com/home.asp?PageID=742&sMenuID=94&p1=94&p2=229&p3=498>. Acesso em: 01 dez 2007c.

ROSE, J. et al., ***Architecture of Field-Programmable Gate Arrays***. Proceedings of the IEEE, Vol. 81, No. 7, p. 1013-1028, Julho 1993.

VIRTUAL COMPUTER CORPORATION. ***H.O.T. Users Guide,*** Setembro 1997.

APÊNDICE A – Contador 16 bits em VHDL

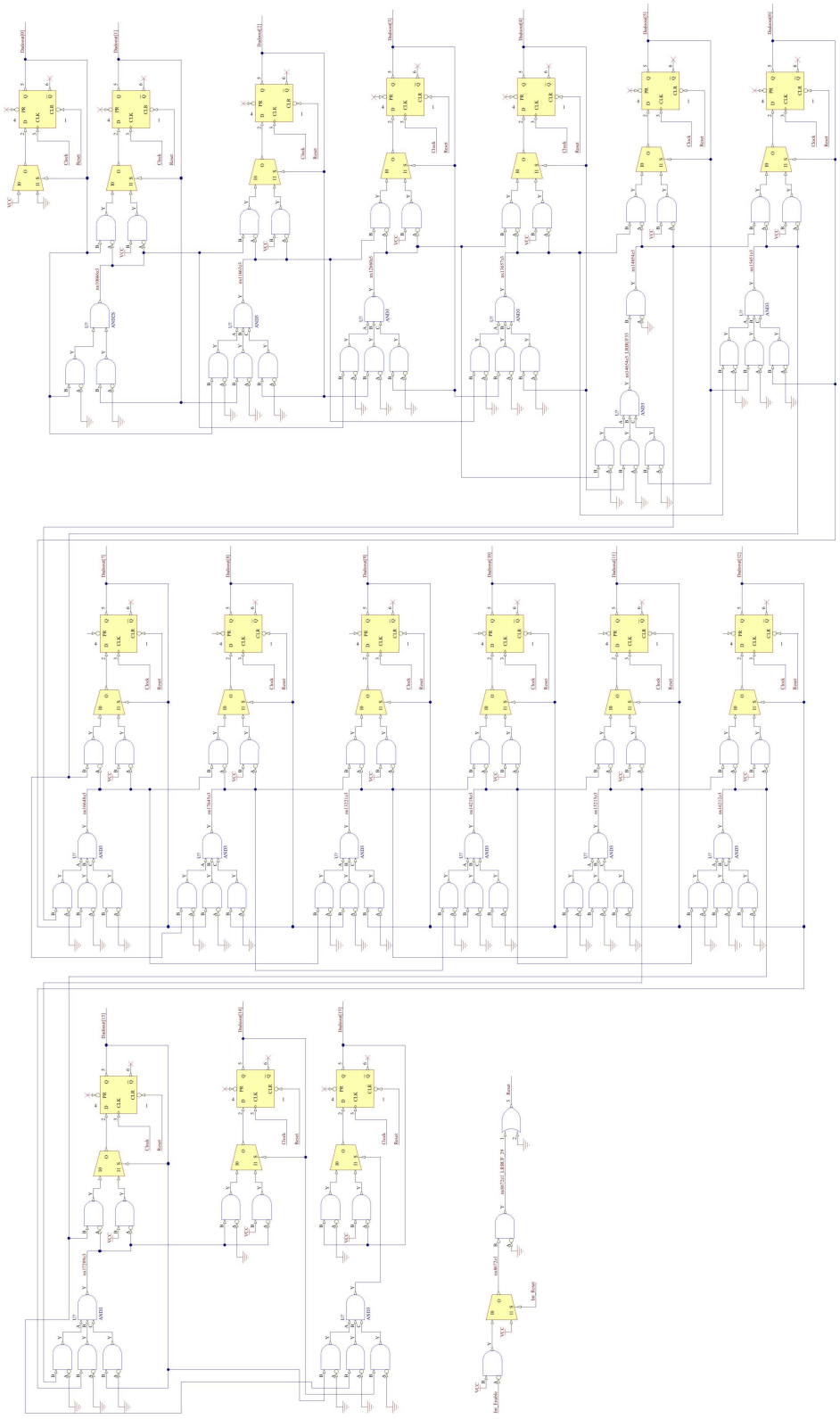
```
--=================================================================
-- File name   : Counter.vhd

-- Description : 16 bits Counter
-----------------------------------------------------------------
-----------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;


-----------------------------------------------------------------
--Entity declaration
-----------------------------------------------------------------
entity Counter16 is
      Port (
                  Clock   : in std_logic;
                  Enable  : in std_logic;
                  Reset   : in std_logic;
                  Dadoout : out std_logic_vector(15 downto 0)
            );

end Counter16;

-----------------------------------------------------------------
--Architecture declaration
-----------------------------------------------------------------
architecture Contador of Counter16 is

-----------------------------------------------------------------
--Signal declaration
-----------------------------------------------------------------
signal Counter : std_logic_vector(15 downto 0);

begin

-----------------------------------------------------------------
-- Process Header
-----------------------------------------------------------------
Principal: process (Reset,Enable,Clock)
begin

   if (Reset='1') then
       Counter <= X"0000";
   elsif (Enable='0') then
       Counter <= X"0000";
   elsif (Clock'event and Clock='1') then
      Counter <= Counter + X"0001";
   end if;

 end process Principal;
-----------------------------------------------------------------
-- End Process
-----------------------------------------------------------------

Dadoout <= Counter;

end Contador;
```

APÊNDICE B – *Testbench* do contador de 16 bits em VHDL

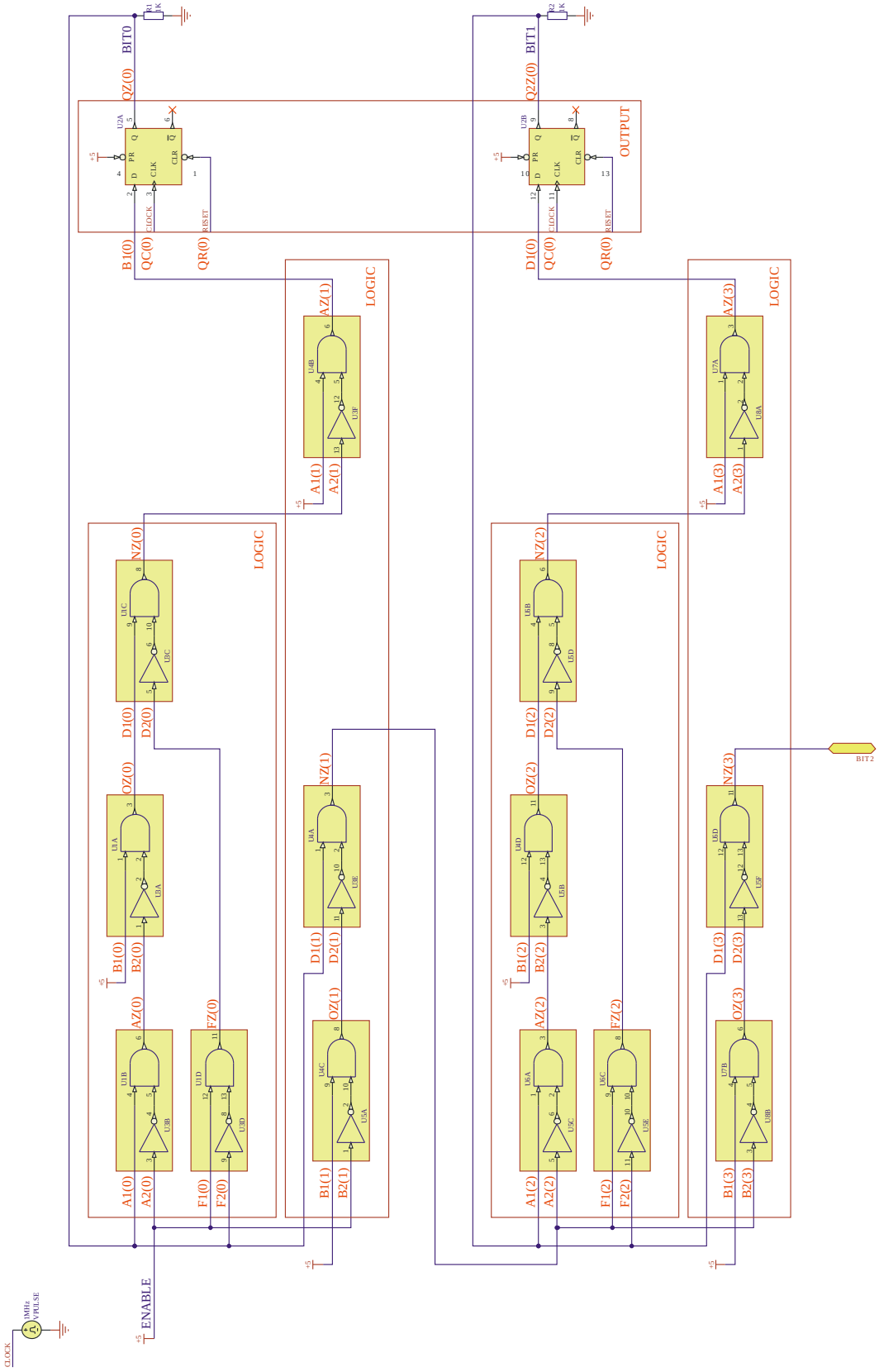
```
--================================================================
-- File name   : Counter_Testbench.vhd

-- Description : 16  bits Counter Testbench
-------------------------------------------------------------------
-------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_textio.all;

-------------------------------------------------------------------
--Entity declaration
-------------------------------------------------------------------
entity TESTBENCH is
end TESTBENCH;

-------------------------------------------------------------------
--Architecture declaration
-------------------------------------------------------------------
architecture TEST of TESTBENCH is

component Counter16

   Port (      Clock   : in std_logic;
               Enable  : in std_logic;
               Reset   : in std_logic;
               Dadoout : out std_logic_vector(15 downto 0)
         );

end component;

-------------------------------------------------------------------
--signal declaration
-------------------------------------------------------------------
signal   Clock   : std_logic:='0';
signal   Enable  : std_logic:='0';
signal   Reset   : std_logic:='0';
signal   Dadoout : std_logic_vector(15 downto 0):=X"0000";

begin

-------------------------------------------------------------------
-- Process Header
-- Clock - Used to generate the Clock signal
-------------------------------------------------------------------
P_Clock: process
begin
   Clock   <=   '0';
    wait for 7 ns;
   Clock   <=   '1';
    wait for 7 ns;
end process;
-------------------------------------------------------------------

-------------------------------------------------------------------
-- Process Header
-- P_Enable - Used to generate the Enable signal
-------------------------------------------------------------------
P_Enable: process
begin
   Enable   <=   '0';
```

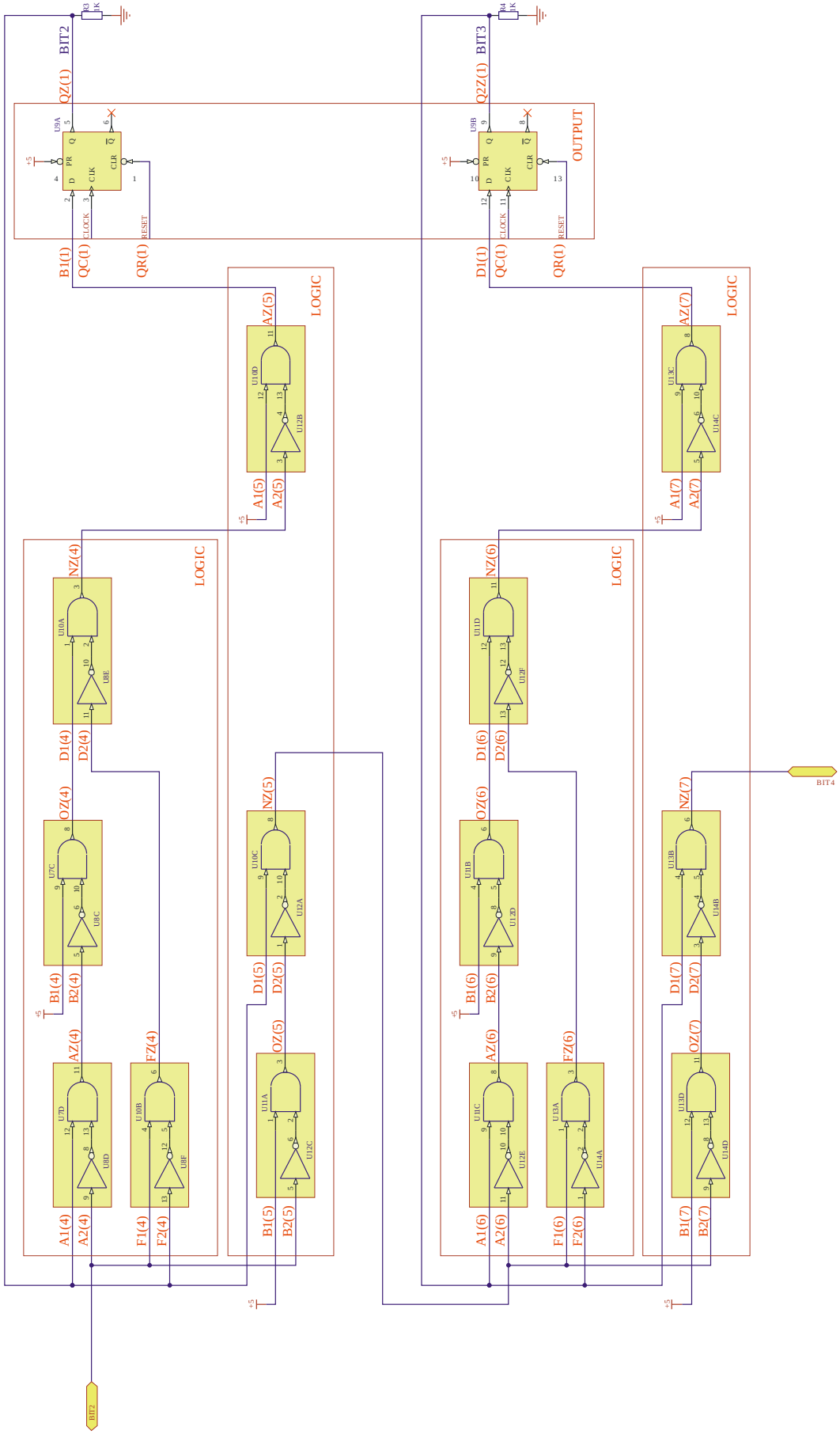
```
    wait for 1 us;
   Enable   <=   '1';
    wait for 500 ms;
end process;
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
-- Process Header
-- P_Reset - Used to generate the Reset signal
--------------------------------------------------------------------------------
P_Reset: process
begin
   Reset   <=   '0';
    wait for 2 us;
   Reset   <=   '1';
    wait for 2 us;
    Reset   <=   '0';
    wait for 5000 ms;
end process;
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
--Unit under test
--------------------------------------------------------------------------------
uut: Counter16
port map (

Clock,
Enable,
Reset,
Dadoout
);

end TEST;
```

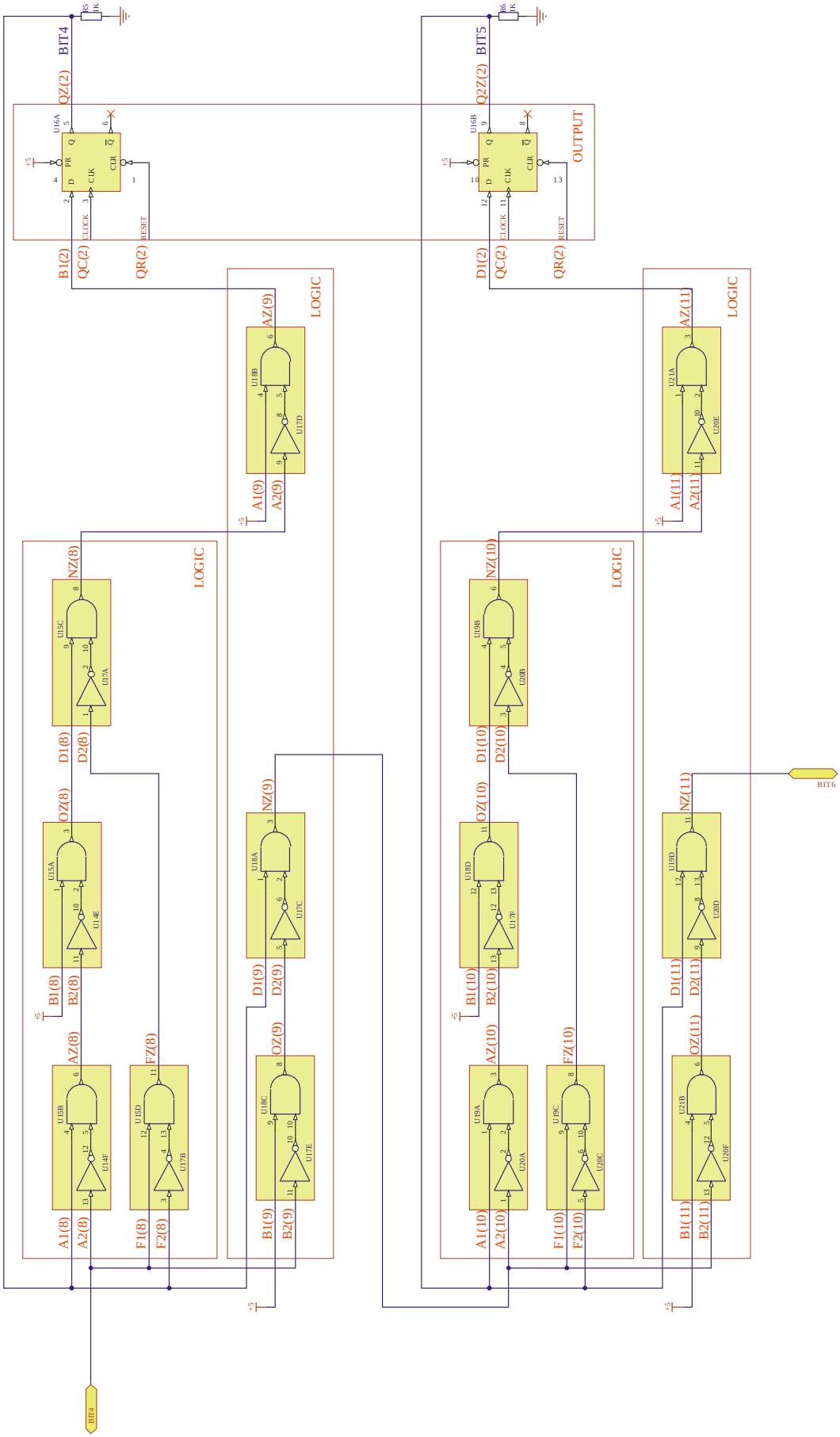
APÊNDICE C – Circuito implementado pelo processo *Place & Route* – Metodologia clássica

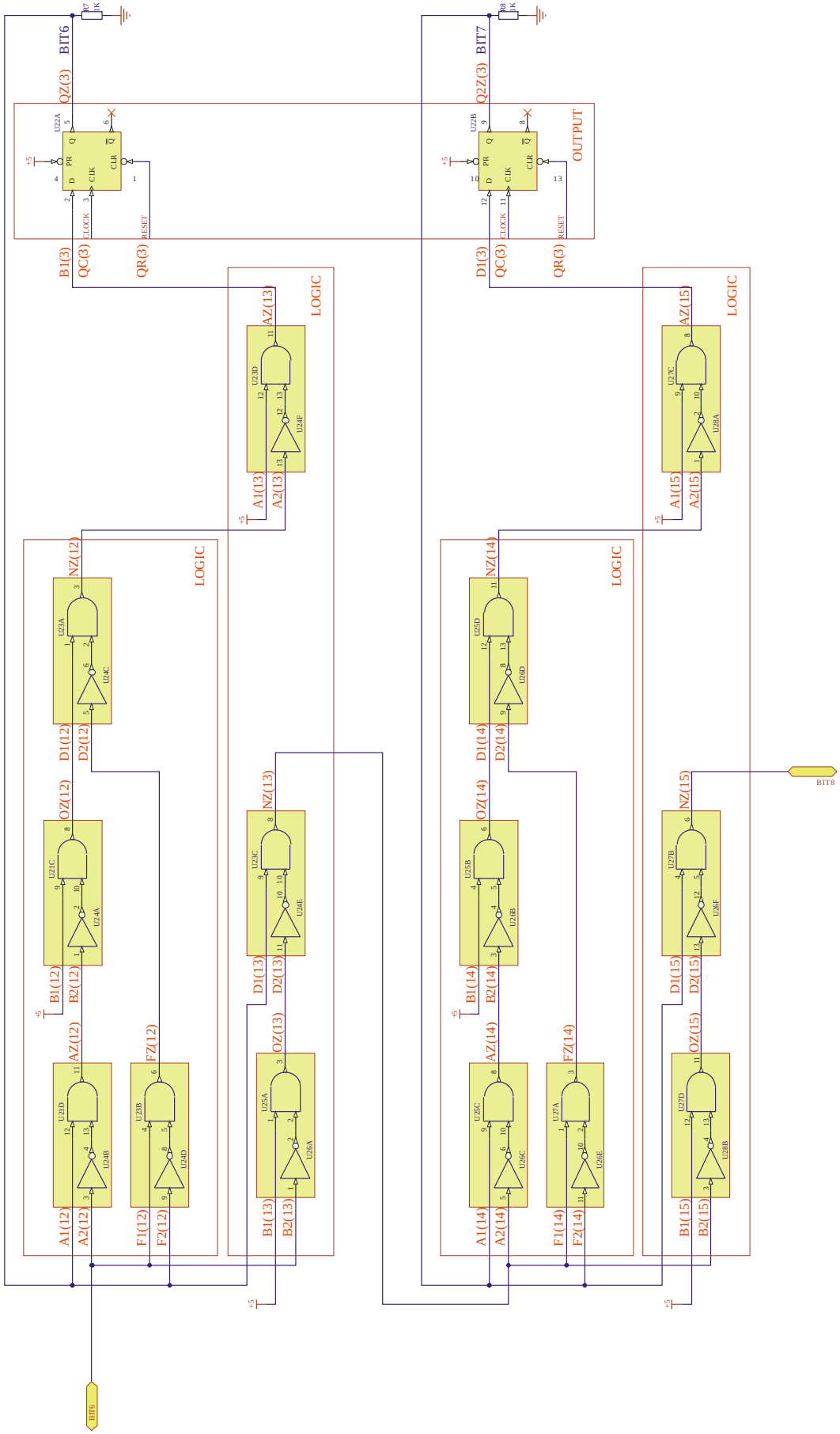


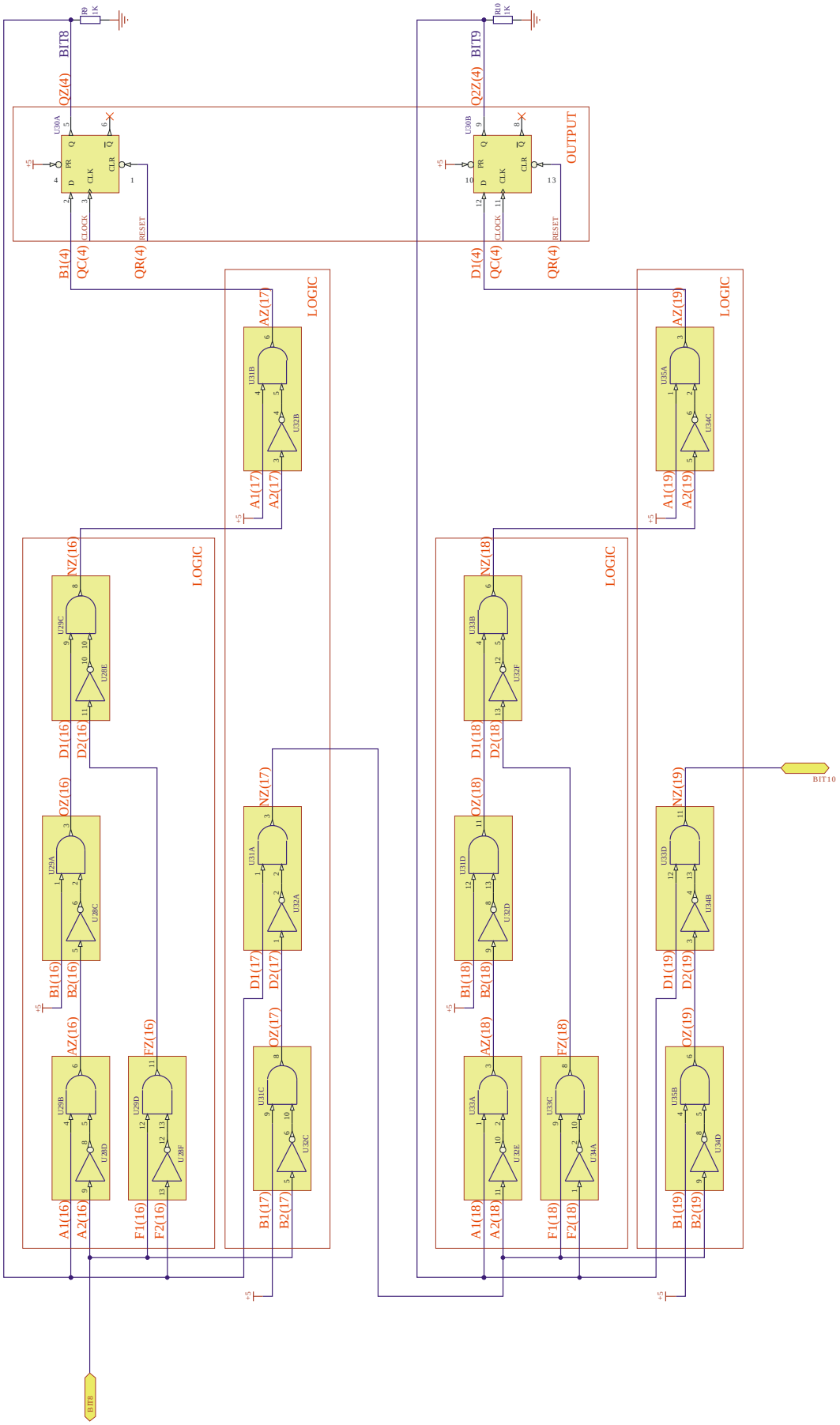
APÊNDICE D - Esquemático do contador de 16 bits

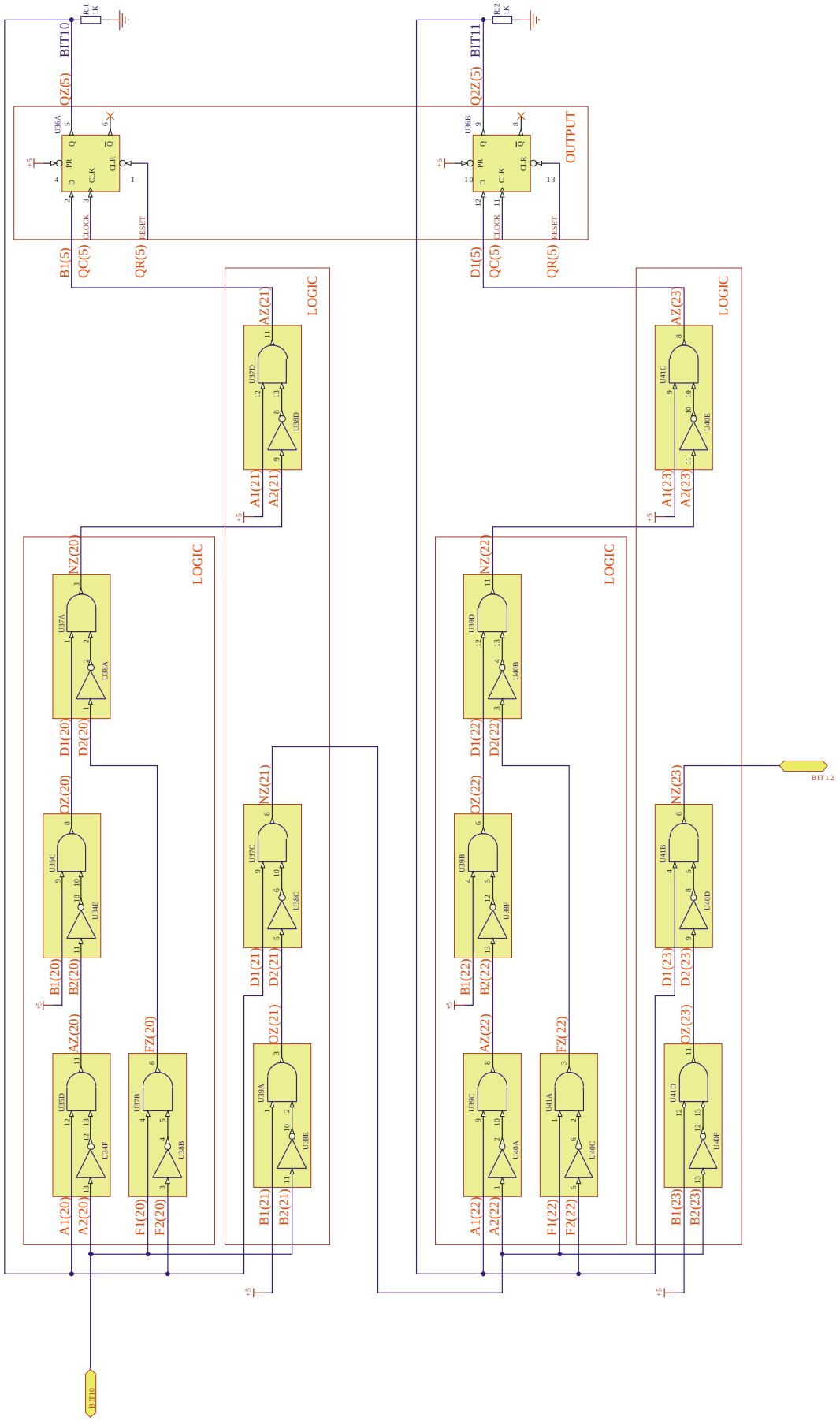


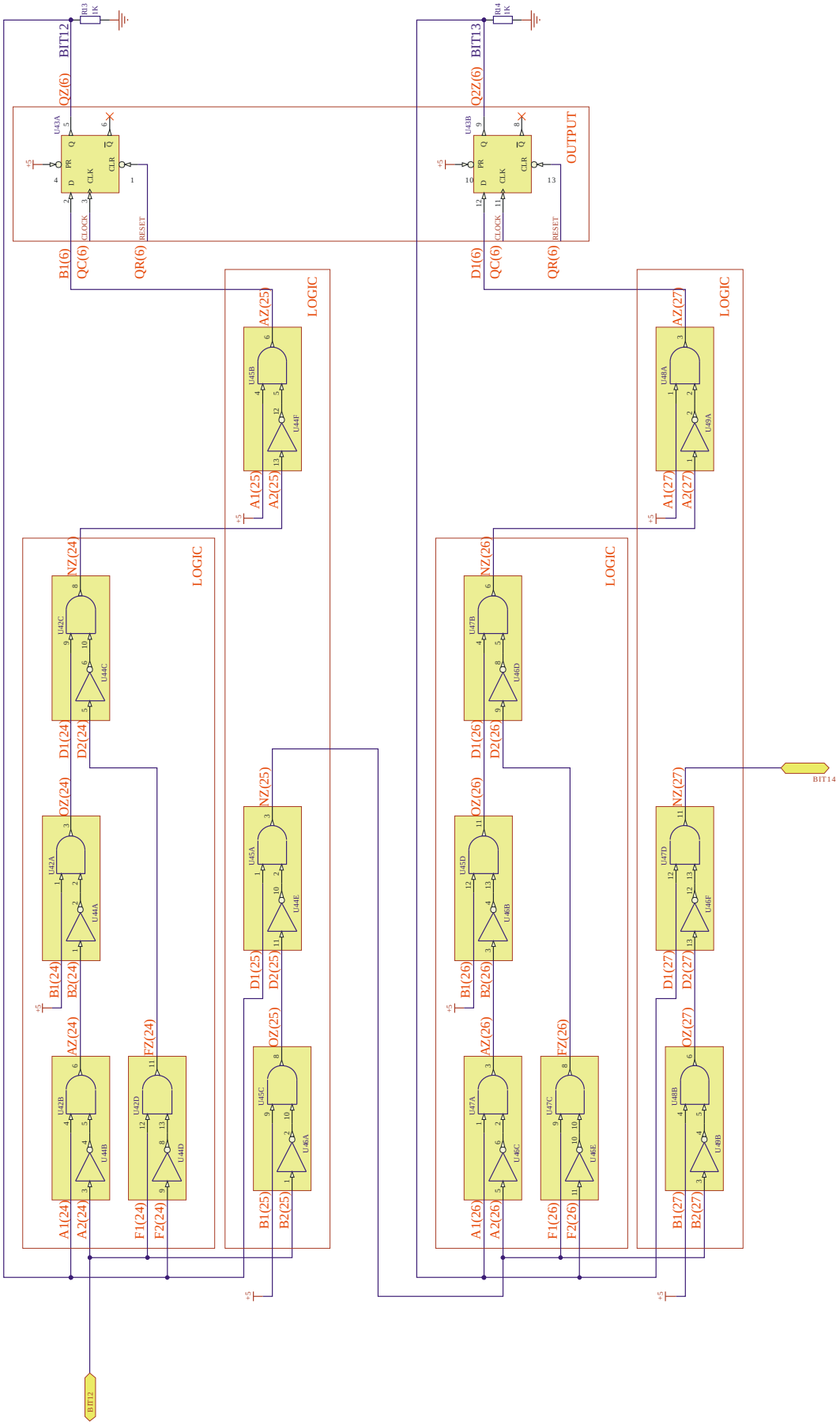

BIT2
QZ(1)
OUTPUT
B1(1)
QC(1)
QR(1)
LOGIC
AZ(5)
A1(5)
A2(5)
NZ(4)
D1(4)
D2(4)
OZ(4)
B1(4)
B2(4)
AZ(4)
FZ(4)
A1(4)
A2(4)
F1(4)
F2(4)
NZ(5)
D1(5)
D2(5)
OZ(5)
B1(5)
B2(5)
BIT3
QZ(1)
D1(1)
QC(1)
QR(1)
AZ(7)
A1(7)
A2(7)
NZ(6)
D1(6)
D2(6)
OZ(6)
B1(6)
B2(6)
AZ(6)
FZ(6)
A1(6)
A2(6)
F1(6)
F2(6)
NZ(7)
D1(7)
D2(7)
OZ(7)
B1(7)
B2(7)
BIT4

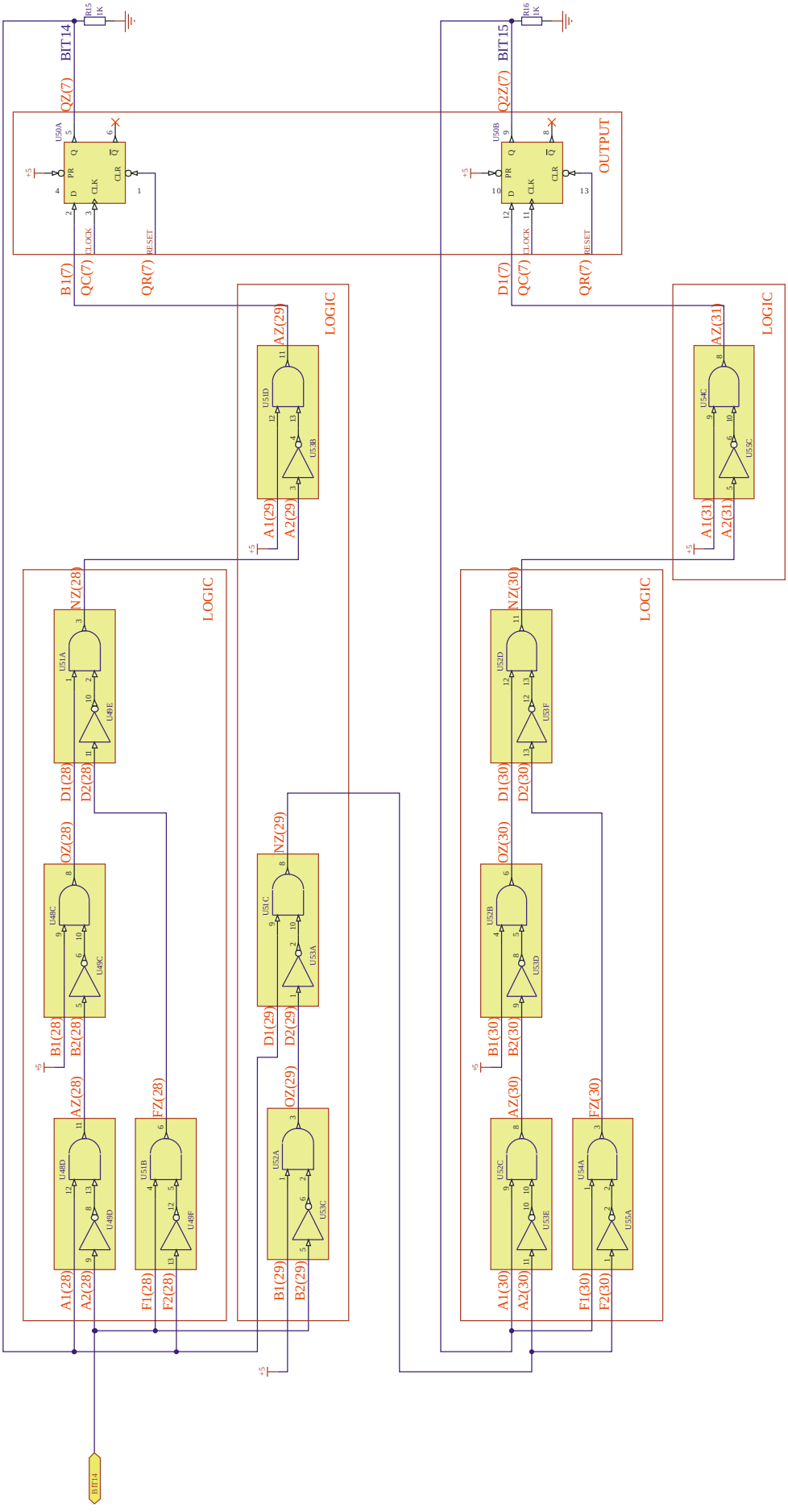

BIT4
BIT5
BIT6
OUTPUT
LOGIC
LOGIC
LOGIC
LOGIC
QZ(2)
Q2Z(2)
B1(2)
QC(2)
QR(2)
D1(2)
QC(2)
QR(2)
AZ(9)
A1(9)
A2(9)
AZ(11)
A1(11)
A2(11)
NZ(8)
D1(8)
D2(8)
NZ(10)
D1(10)
D2(10)
OZ(8)
B1(8)
B2(8)
OZ(10)
B1(10)
B2(10)
NZ(9)
D1(9)
D2(9)
NZ(11)
D1(11)
D2(11)
AZ(8)
A1(8)
A2(8)
FZ(8)
F1(8)
F2(8)
AZ(10)
A1(10)
A2(10)
FZ(10)
F1(10)
F2(10)
OZ(9)
B1(9)
B2(9)
OZ(11)
B1(11)
B2(11)


BIT6
BIT7
OUTPUT
LOGIC
BIT8


BIT8
BIT9
OUTPUT
LOGIC
BIT10
BIT8


BIT10
BIT11
QZ(5)
Q2Z(5)
OUTPUT
B1(5)
QC(5)
QR(5)
D1(5)
LOGIC
AZ(21)
AZ(23)
A1(21)
A2(21)
A1(23)
A2(23)
NZ(20)
NZ(22)
D1(20)
D2(20)
D1(22)
D2(22)
OZ(20)
OZ(22)
NZ(21)
NZ(23)
B1(20)
B2(20)
B1(22)
B2(22)
D1(21)
D2(21)
D1(23)
D2(23)
AZ(20)
AZ(22)
FZ(20)
FZ(22)
OZ(21)
OZ(23)
A1(20)
A2(20)
F1(20)
F2(20)
A1(22)
A2(22)
F1(22)
F2(22)
B1(21)
B2(21)
B1(23)
B2(23)
BIT12
BIT10


BIT12
BIT13
BIT14
OUTPUT
LOGIC
LOGIC
LOGIC
LOGIC
QZ(6)
Q2Z(6)
B1(6)
QC(6)
QR(6)
D1(6)
QC(6)
QR(6)
AZ(25)
AZ(27)
A1(25)
A2(25)
A1(27)
A2(27)
NZ(24)
NZ(26)
D1(24)
D2(24)
D1(26)
D2(26)
OZ(24)
NZ(25)
OZ(26)
NZ(27)
B1(24)
B2(24)
D1(25)
D2(25)
B1(26)
B2(26)
D1(27)
D2(27)
AZ(24)
FZ(24)
OZ(25)
AZ(26)
FZ(26)
OZ(27)
A1(24)
A2(24)
F1(24)
F2(24)
B1(25)
B2(25)
A1(26)
A2(26)
F1(26)
F2(26)
B1(27)
B2(27)


BIT14
BIT15
QZ(7)
QZZ(7)
OUTPUT
B1(7)
QC(7)
QR(7)
D1(7)
LOGIC
AZ(29)
AZ(31)
NZ(28)
NZ(30)
D1(28)
D2(28)
D1(30)
D2(30)
OZ(28)
OZ(30)
NZ(29)
D1(29)
D2(29)
B1(28)
B2(28)
B1(30)
B2(30)
AZ(28)
AZ(30)
FZ(28)
FZ(30)
OZ(29)
A1(28)
A2(28)
F1(28)
F2(28)
B1(29)
B2(29)
A1(30)
A2(30)
F1(30)
F2(30)
A1(29)
A2(29)
A1(31)
A2(31)
+5
BIT14

APÊNDICE E – Contador 16 bits em VHDL (*gate level*)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity Counter16 is
      Port (
                  Clock   : in std_logic;
                  Enable  : in std_logic;
                  Reset   : in std_logic;
                  Dadoout : out std_logic_vector(15 downto 0)
            );

end Counter16;


architecture SCHEMATIC of Counter16 is

constant        GND : std_logic := '0';
constant        VCC : std_logic := '1';

signal Load : std_logic := '0';
signal A1_O : std_logic_vector(11 downto 0);
signal A2_O : std_logic_vector(11 downto 0);
signal A3_O : std_logic_vector(11 downto 0);
signal A4_O : std_logic_vector(11 downto 0);
signal A5_O : std_logic_vector(11 downto 0);
signal A6_O : std_logic_vector(11 downto 0);
signal B1_O : std_logic_vector(11 downto 0);
signal B2_O : std_logic_vector(11 downto 0);
signal C1_O : std_logic_vector(11 downto 0);
signal C2_O : std_logic_vector(11 downto 0);
signal D1_O : std_logic_vector(11 downto 0);
signal D2_O : std_logic_vector(11 downto 0);
signal E1_O : std_logic_vector(11 downto 0);
signal E2_O : std_logic_vector(11 downto 0);
signal F1_O : std_logic_vector(11 downto 0);
signal F2_O : std_logic_vector(11 downto 0);
signal F3_O : std_logic_vector(11 downto 0);
signal F4_O : std_logic_vector(11 downto 0);
signal F5_O : std_logic_vector(11 downto 0);
signal F6_O : std_logic_vector(11 downto 0);
signal MP_O : std_logic_vector(11 downto 0);
signal MS_O : std_logic_vector(11 downto 0);
signal NP_O : std_logic_vector(11 downto 0);
signal NS_O : std_logic_vector(11 downto 0);
signal OP_O : std_logic_vector(11 downto 0);
signal OS_O : std_logic_vector(11 downto 0);
signal PP_O : std_logic_vector(11 downto 0);
signal PS_O : std_logic_vector(11 downto 0);
signal QC_O : std_logic_vector(11 downto 0);
signal QR_O : std_logic_vector(11 downto 0);
signal QS_O : std_logic_vector(11 downto 0);
signal AZ_O : std_logic_vector(11 downto 0);
signal FZ_O : std_logic_vector(11 downto 0);
signal NZ_O : std_logic_vector(11 downto 0);
signal OZ_O : std_logic_vector(11 downto 0);
signal Q2Z_O : std_logic_vector(11 downto 0);
signal QZ_O : std_logic_vector(11 downto 0);

signal A1_L : std_logic_vector(47 downto 0);
signal A2_L : std_logic_vector(47 downto 0);
signal A3_L : std_logic_vector(47 downto 0);
signal A4_L : std_logic_vector(47 downto 0);
signal A5_L : std_logic_vector(47 downto 0);
signal A6_L : std_logic_vector(47 downto 0);
signal B1_L : std_logic_vector(47 downto 0);
signal B2_L : std_logic_vector(47 downto 0);
signal C1_L : std_logic_vector(47 downto 0);
signal C2_L : std_logic_vector(47 downto 0);
signal D1_L : std_logic_vector(47 downto 0);
signal D2_L : std_logic_vector(47 downto 0);
signal E1_L : std_logic_vector(47 downto 0);
```

```
signal E2_L : std_logic_vector(47 downto 0);
signal F1_L : std_logic_vector(47 downto 0);
signal F2_L : std_logic_vector(47 downto 0);
signal F3_L : std_logic_vector(47 downto 0);
signal F4_L : std_logic_vector(47 downto 0);
signal F5_L : std_logic_vector(47 downto 0);
signal F6_L : std_logic_vector(47 downto 0);
signal MP_L : std_logic_vector(47 downto 0);
signal MS_L : std_logic_vector(47 downto 0);
signal NP_L : std_logic_vector(47 downto 0);
signal NS_L : std_logic_vector(47 downto 0);
signal OP_L : std_logic_vector(47 downto 0);
signal OS_L : std_logic_vector(47 downto 0);
signal PP_L : std_logic_vector(47 downto 0);
signal PS_L : std_logic_vector(47 downto 0);
signal QC_L : std_logic_vector(47 downto 0);
signal QR_L : std_logic_vector(47 downto 0);
signal QS_L : std_logic_vector(47 downto 0);
signal AZ_L : std_logic_vector(47 downto 0);
signal FZ_L : std_logic_vector(47 downto 0);
signal NZ_L : std_logic_vector(47 downto 0);
signal OZ_L : std_logic_vector(47 downto 0);
signal Q2Z_L : std_logic_vector(47 downto 0);
signal QZ_L : std_logic_vector(47 downto 0);

signal Dadooutbuffer : std_logic_vector(16 downto 0):=X"0000";

component SUPER_LOGIC
      Port (      A1 : In    STD_LOGIC;
                  A2 : In    STD_LOGIC;
                  A3 : In    STD_LOGIC;
                  A4 : In    STD_LOGIC;
                  A5 : In    STD_LOGIC;
                  A6 : In    STD_LOGIC;
                  B1 : In    STD_LOGIC;
                  B2 : In    STD_LOGIC;
                  C1 : In    STD_LOGIC;
                  C2 : In    STD_LOGIC;
                  D1 : In    STD_LOGIC;
                  D2 : In    STD_LOGIC;
                  E1 : In    STD_LOGIC;
                  E2 : In    STD_LOGIC;
                  F1 : In    STD_LOGIC;
                  F2 : In    STD_LOGIC;
                  F3 : In    STD_LOGIC;
                  F4 : In    STD_LOGIC;
                  F5 : In    STD_LOGIC;
                  F6 : In    STD_LOGIC;
                  MP : In    STD_LOGIC;
                  MS : In    STD_LOGIC;
                  NP : In    STD_LOGIC;
                  NS : In    STD_LOGIC;
                  OP : In    STD_LOGIC;
                  OS : In    STD_LOGIC;
                  PP : In    STD_LOGIC;
                  PS : In    STD_LOGIC;
                  QC : In    STD_LOGIC;
                  QR : In    STD_LOGIC;
                  QS : In    STD_LOGIC;
                  AZ : Out   STD_LOGIC;
                  FZ : Out   STD_LOGIC;
                  NZ : Out   STD_LOGIC;
                  OZ : Out   STD_LOGIC;
                 Q2Z : Out   STD_LOGIC;
                  QZ : Out   STD_LOGIC );
   end component;

begin

      OUTPUT_0 : SUPER_LOGIC
      Port Map ( A1=>A1_O(0),A2=>A2_O(0),A3=>A3_O(0),A4=>A4_O(0),
                 A5=>A5_O(0),A6=>A6_O(0),B1=>B1_O(0),B2=>B2_O(0),
                 C1=>C1_O(0),C2=>C2_O(0),D1=>D1_O(0),D2=>D2_O(0),
                 E1=>E1_O(0),E2=>E2_O(0),F1=>F1_O(0),F2=>F2_O(0),
                 F3=>F3_O(0),F4=>F4_O(0),F5=>F5_O(0),F6=>F6_O(0),
                 MP=>MP_O(0),MS=>MS_O(0),NP=>NP_O(0),NS=>NS_O(0),
                 OP=>OP_O(0),OS=>OS_O(0),PP=>PP_O(0),PS=>PS_O(0),
```

```
            QC=>QC_O(0),QR=>QR_O(0),QS=>QS_O(0),AZ=>AZ_O(0),
            FZ=>FZ_O(0),NZ=>NZ_O(0),OZ=>OZ_O(0),Q2Z=>Q2Z_O(0),
            QZ=>QZ_O(0)
            );

OUTPUT_1 : SUPER_LOGIC
Port Map ( A1=>A1_O(1),A2=>A2_O(1),A3=>A3_O(1),A4=>A4_O(1),
            A5=>A5_O(1),A6=>A6_O(1),B1=>B1_O(1),B2=>B2_O(1),
            C1=>C1_O(1),C2=>C2_O(1),D1=>D1_O(1),D2=>D2_O(1),
            E1=>E1_O(1),E2=>E2_O(1),F1=>F1_O(1),F2=>F2_O(1),
            F3=>F3_O(1),F4=>F4_O(1),F5=>F5_O(1),F6=>F6_O(1),
            MP=>MP_O(1),MS=>MS_O(1),NP=>NP_O(1),NS=>NS_O(1),
            OP=>OP_O(1),OS=>OS_O(1),PP=>PP_O(1),PS=>PS_O(1),
            QC=>QC_O(1),QR=>QR_O(1),QS=>QS_O(1),AZ=>AZ_O(1),
            FZ=>FZ_O(1),NZ=>NZ_O(1),OZ=>OZ_O(1),Q2Z=>Q2Z_O(1),
            QZ=>QZ_O(1)
            );

OUTPUT_2 : SUPER_LOGIC
Port Map ( A1=>A1_O(2),A2=>A2_O(2),A3=>A3_O(2),A4=>A4_O(2),
            A5=>A5_O(2),A6=>A6_O(2),B1=>B1_O(2),B2=>B2_O(2),
            C1=>C1_O(2),C2=>C2_O(2),D1=>D1_O(2),D2=>D2_O(2),
            E1=>E1_O(2),E2=>E2_O(2),F1=>F1_O(2),F2=>F2_O(2),
            F3=>F3_O(2),F4=>F4_O(2),F5=>F5_O(2),F6=>F6_O(2),
            MP=>MP_O(2),MS=>MS_O(2),NP=>NP_O(2),NS=>NS_O(2),
            OP=>OP_O(2),OS=>OS_O(2),PP=>PP_O(2),PS=>PS_O(2),
            QC=>QC_O(2),QR=>QR_O(2),QS=>QS_O(2),AZ=>AZ_O(2),
            FZ=>FZ_O(2),NZ=>NZ_O(2),OZ=>OZ_O(2),Q2Z=>Q2Z_O(2),
            QZ=>QZ_O(2)
            );

OUTPUT_3 : SUPER_LOGIC
Port Map ( A1=>A1_O(3),A2=>A2_O(3),A3=>A3_O(3),A4=>A4_O(3),
            A5=>A5_O(3),A6=>A6_O(3),B1=>B1_O(3),B2=>B2_O(3),
            C1=>C1_O(3),C2=>C2_O(3),D1=>D1_O(3),D2=>D2_O(3),
            E1=>E1_O(3),E2=>E2_O(3),F1=>F1_O(3),F2=>F2_O(3),
            F3=>F3_O(3),F4=>F4_O(3),F5=>F5_O(3),F6=>F6_O(3),
            MP=>MP_O(3),MS=>MS_O(3),NP=>NP_O(3),NS=>NS_O(3),
            OP=>OP_O(3),OS=>OS_O(3),PP=>PP_O(3),PS=>PS_O(3),
            QC=>QC_O(3),QR=>QR_O(3),QS=>QS_O(3),AZ=>AZ_O(3),
            FZ=>FZ_O(3),NZ=>NZ_O(3),OZ=>OZ_O(3),Q2Z=>Q2Z_O(3),
            QZ=>QZ_O(3)
            );

OUTPUT_4 : SUPER_LOGIC
Port Map ( A1=>A1_O(4),A2=>A2_O(4),A3=>A3_O(4),A4=>A4_O(4),
            A5=>A5_O(4),A6=>A6_O(4),B1=>B1_O(4),B2=>B2_O(4),
            C1=>C1_O(4),C2=>C2_O(4),D1=>D1_O(4),D2=>D2_O(4),
            E1=>E1_O(4),E2=>E2_O(4),F1=>F1_O(4),F2=>F2_O(4),
            F3=>F3_O(4),F4=>F4_O(4),F5=>F5_O(4),F6=>F6_O(4),
            MP=>MP_O(4),MS=>MS_O(4),NP=>NP_O(4),NS=>NS_O(4),
            OP=>OP_O(4),OS=>OS_O(4),PP=>PP_O(4),PS=>PS_O(4),
            QC=>QC_O(4),QR=>QR_O(4),QS=>QS_O(4),AZ=>AZ_O(4),
            FZ=>FZ_O(4),NZ=>NZ_O(4),OZ=>OZ_O(4),Q2Z=>Q2Z_O(4),
            QZ=>QZ_O(4)
            );

OUTPUT_5 : SUPER_LOGIC
Port Map ( A1=>A1_O(5),A2=>A2_O(5),A3=>A3_O(5),A4=>A4_O(5),
            A5=>A5_O(5),A6=>A6_O(5),B1=>B1_O(5),B2=>B2_O(5),
            C1=>C1_O(5),C2=>C2_O(5),D1=>D1_O(5),D2=>D2_O(5),
            E1=>E1_O(5),E2=>E2_O(5),F1=>F1_O(5),F2=>F2_O(5),
            F3=>F3_O(5),F4=>F4_O(5),F5=>F5_O(5),F6=>F6_O(5),
            MP=>MP_O(5),MS=>MS_O(5),NP=>NP_O(5),NS=>NS_O(5),
            OP=>OP_O(5),OS=>OS_O(5),PP=>PP_O(5),PS=>PS_O(5),
            QC=>QC_O(5),QR=>QR_O(5),QS=>QS_O(5),AZ=>AZ_O(5),
            FZ=>FZ_O(5),NZ=>NZ_O(5),OZ=>OZ_O(5),Q2Z=>Q2Z_O(5),
            QZ=>QZ_O(5)
            );

OUTPUT_6 : SUPER_LOGIC
Port Map ( A1=>A1_O(6),A2=>A2_O(6),A3=>A3_O(6),A4=>A4_O(6),
            A5=>A5_O(6),A6=>A6_O(6),B1=>B1_O(6),B2=>B2_O(6),
            C1=>C1_O(6),C2=>C2_O(6),D1=>D1_O(6),D2=>D2_O(6),
            E1=>E1_O(6),E2=>E2_O(6),F1=>F1_O(6),F2=>F2_O(6),
            F3=>F3_O(6),F4=>F4_O(6),F5=>F5_O(6),F6=>F6_O(6),
            MP=>MP_O(6),MS=>MS_O(6),NP=>NP_O(6),NS=>NS_O(6),
```

```
               OP=>OP_O(6),OS=>OS_O(6),PP=>PP_O(6),PS=>PS_O(6),
               QC=>QC_O(6),QR=>QR_O(6),QS=>QS_O(6),AZ=>AZ_O(6),
               FZ=>FZ_O(6),NZ=>NZ_O(6),OZ=>OZ_O(6),Q2Z=>Q2Z_O(6),
               QZ=>QZ_O(6)
               );

   OUTPUT_7 : SUPER_LOGIC
   Port Map ( A1=>A1_O(7),A2=>A2_O(7),A3=>A3_O(7),A4=>A4_O(7),
               A5=>A5_O(7),A6=>A6_O(7),B1=>B1_O(7),B2=>B2_O(7),
               C1=>C1_O(7),C2=>C2_O(7),D1=>D1_O(7),D2=>D2_O(7),
               E1=>E1_O(7),E2=>E2_O(7),F1=>F1_O(7),F2=>F2_O(7),
               F3=>F3_O(7),F4=>F4_O(7),F5=>F5_O(7),F6=>F6_O(7),
               MP=>MP_O(7),MS=>MS_O(7),NP=>NP_O(7),NS=>NS_O(7),
               OP=>OP_O(7),OS=>OS_O(7),PP=>PP_O(7),PS=>PS_O(7),
               QC=>QC_O(7),QR=>QR_O(7),QS=>QS_O(7),AZ=>AZ_O(7),
               FZ=>FZ_O(7),NZ=>NZ_O(7),OZ=>OZ_O(7),Q2Z=>Q2Z_O(7),
               QZ=>QZ_O(7)
               );

   OUTPUT_8 : SUPER_LOGIC
   Port Map ( A1=>A1_O(8),A2=>A2_O(8),A3=>A3_O(8),A4=>A4_O(8),
               A5=>A5_O(8),A6=>A6_O(8),B1=>B1_O(8),B2=>B2_O(8),
               C1=>C1_O(8),C2=>C2_O(8),D1=>D1_O(8),D2=>D2_O(8),
               E1=>E1_O(8),E2=>E2_O(8),F1=>F1_O(8),F2=>F2_O(8),
               F3=>F3_O(8),F4=>F4_O(8),F5=>F5_O(8),F6=>F6_O(8),
               MP=>MP_O(8),MS=>MS_O(8),NP=>NP_O(8),NS=>NS_O(8),
               OP=>OP_O(8),OS=>OS_O(8),PP=>PP_O(8),PS=>PS_O(8),
               QC=>QC_O(8),QR=>QR_O(8),QS=>QS_O(8),AZ=>AZ_O(8),
               FZ=>FZ_O(8),NZ=>NZ_O(8),OZ=>OZ_O(8),Q2Z=>Q2Z_O(8),
               QZ=>QZ_O(8)
               );

 LOGIC_0 : SUPER_LOGIC
   Port Map ( A1=>A1_L(0),A2=>A2_L(0),A3=>A3_L(0),A4=>A4_L(0),
               A5=>A5_L(0),A6=>A6_L(0),B1=>B1_L(0),B2=>B2_L(0),
               C1=>C1_L(0),C2=>C2_L(0),D1=>D1_L(0),D2=>D2_L(0),
               E1=>E1_L(0),E2=>E2_L(0),F1=>F1_L(0),F2=>F2_L(0),
               F3=>F3_L(0),F4=>F4_L(0),F5=>F5_L(0),F6=>F6_L(0),
               MP=>MP_L(0),MS=>MS_L(0),NP=>NP_L(0),NS=>NS_L(0),
               OP=>OP_L(0),OS=>OS_L(0),PP=>PP_L(0),PS=>PS_L(0),
               QC=>QC_L(0),QR=>QR_L(0),QS=>QS_L(0),AZ=>AZ_L(0),
               FZ=>FZ_L(0),NZ=>NZ_L(0),OZ=>OZ_L(0),Q2Z=>Q2Z_L(0),
               QZ=>QZ_L(0)
               );

LOGIC_1 : SUPER_LOGIC
   Port Map ( A1=>A1_L(1),A2=>A2_L(1),A3=>A3_L(1),A4=>A4_L(1),
               A5=>A5_L(1),A6=>A6_L(1),B1=>B1_L(1),B2=>B2_L(1),
               C1=>C1_L(1),C2=>C2_L(1),D1=>D1_L(1),D2=>D2_L(1),
               E1=>E1_L(1),E2=>E2_L(1),F1=>F1_L(1),F2=>F2_L(1),
               F3=>F3_L(1),F4=>F4_L(1),F5=>F5_L(1),F6=>F6_L(1),
               MP=>MP_L(1),MS=>MS_L(1),NP=>NP_L(1),NS=>NS_L(1),
               OP=>OP_L(1),OS=>OS_L(1),PP=>PP_L(1),PS=>PS_L(1),
               QC=>QC_L(1),QR=>QR_L(1),QS=>QS_L(1),AZ=>AZ_L(1),
               FZ=>FZ_L(1),NZ=>NZ_L(1),OZ=>OZ_L(1),Q2Z=>Q2Z_L(1),
               QZ=>QZ_L(1)
               );

LOGIC_2 : SUPER_LOGIC
   Port Map ( A1=>A1_L(2),A2=>A2_L(2),A3=>A3_L(2),A4=>A4_L(2),
               A5=>A5_L(2),A6=>A6_L(2),B1=>B1_L(2),B2=>B2_L(2),
               C1=>C1_L(2),C2=>C2_L(2),D1=>D1_L(2),D2=>D2_L(2),
               E1=>E1_L(2),E2=>E2_L(2),F1=>F1_L(2),F2=>F2_L(2),
               F3=>F3_L(2),F4=>F4_L(2),F5=>F5_L(2),F6=>F6_L(2),
               MP=>MP_L(2),MS=>MS_L(2),NP=>NP_L(2),NS=>NS_L(2),
               OP=>OP_L(2),OS=>OS_L(2),PP=>PP_L(2),PS=>PS_L(2),
               QC=>QC_L(2),QR=>QR_L(2),QS=>QS_L(2),AZ=>AZ_L(2),
               FZ=>FZ_L(2),NZ=>NZ_L(2),OZ=>OZ_L(2),Q2Z=>Q2Z_L(2),
               QZ=>QZ_L(2)
               );

LOGIC_3 : SUPER_LOGIC
   Port Map ( A1=>A1_L(3),A2=>A2_L(3),A3=>A3_L(3),A4=>A4_L(3),
               A5=>A5_L(3),A6=>A6_L(3),B1=>B1_L(3),B2=>B2_L(3),
               C1=>C1_L(3),C2=>C2_L(3),D1=>D1_L(3),D2=>D2_L(3),
               E1=>E1_L(3),E2=>E2_L(3),F1=>F1_L(3),F2=>F2_L(3),
               F3=>F3_L(3),F4=>F4_L(3),F5=>F5_L(3),F6=>F6_L(3),
```

```
               MP=>MP_L(3),MS=>MS_L(3),NP=>NP_L(3),NS=>NS_L(3),
               OP=>OP_L(3),OS=>OS_L(3),PP=>PP_L(3),PS=>PS_L(3),
               QC=>QC_L(3),QR=>QR_L(3),QS=>QS_L(3),AZ=>AZ_L(3),
               FZ=>FZ_L(3),NZ=>NZ_L(3),OZ=>OZ_L(3),Q2Z=>Q2Z_L(3),
               QZ=>QZ_L(3)
               );

LOGIC_4 : SUPER_LOGIC
   Port Map ( A1=>A1_L(4),A2=>A2_L(4),A3=>A3_L(4),A4=>A4_L(4),
               A5=>A5_L(4),A6=>A6_L(4),B1=>B1_L(4),B2=>B2_L(4),
               C1=>C1_L(4),C2=>C2_L(4),D1=>D1_L(4),D2=>D2_L(4),
               E1=>E1_L(4),E2=>E2_L(4),F1=>F1_L(4),F2=>F2_L(4),
               F3=>F3_L(4),F4=>F4_L(4),F5=>F5_L(4),F6=>F6_L(4),
               MP=>MP_L(4),MS=>MS_L(4),NP=>NP_L(4),NS=>NS_L(4),
               OP=>OP_L(4),OS=>OS_L(4),PP=>PP_L(4),PS=>PS_L(4),
               QC=>QC_L(4),QR=>QR_L(4),QS=>QS_L(4),AZ=>AZ_L(4),
               FZ=>FZ_L(4),NZ=>NZ_L(4),OZ=>OZ_L(4),Q2Z=>Q2Z_L(4),
               QZ=>QZ_L(4)
               );

LOGIC_5 : SUPER_LOGIC
   Port Map ( A1=>A1_L(5),A2=>A2_L(5),A3=>A3_L(5),A4=>A4_L(5),
               A5=>A5_L(5),A6=>A6_L(5),B1=>B1_L(5),B2=>B2_L(5),
               C1=>C1_L(5),C2=>C2_L(5),D1=>D1_L(5),D2=>D2_L(5),
               E1=>E1_L(5),E2=>E2_L(5),F1=>F1_L(5),F2=>F2_L(5),
               F3=>F3_L(5),F4=>F4_L(5),F5=>F5_L(5),F6=>F6_L(5),
               MP=>MP_L(5),MS=>MS_L(5),NP=>NP_L(5),NS=>NS_L(5),
               OP=>OP_L(5),OS=>OS_L(5),PP=>PP_L(5),PS=>PS_L(5),
               QC=>QC_L(5),QR=>QR_L(5),QS=>QS_L(5),AZ=>AZ_L(5),
               FZ=>FZ_L(5),NZ=>NZ_L(5),OZ=>OZ_L(5),Q2Z=>Q2Z_L(5),
               QZ=>QZ_L(5)
               );

LOGIC_6 : SUPER_LOGIC
   Port Map ( A1=>A1_L(6),A2=>A2_L(6),A3=>A3_L(6),A4=>A4_L(6),
               A5=>A5_L(6),A6=>A6_L(6),B1=>B1_L(6),B2=>B2_L(6),
               C1=>C1_L(6),C2=>C2_L(6),D1=>D1_L(6),D2=>D2_L(6),
               E1=>E1_L(6),E2=>E2_L(6),F1=>F1_L(6),F2=>F2_L(6),
               F3=>F3_L(6),F4=>F4_L(6),F5=>F5_L(6),F6=>F6_L(6),
               MP=>MP_L(6),MS=>MS_L(6),NP=>NP_L(6),NS=>NS_L(6),
               OP=>OP_L(6),OS=>OS_L(6),PP=>PP_L(6),PS=>PS_L(6),
               QC=>QC_L(6),QR=>QR_L(6),QS=>QS_L(6),AZ=>AZ_L(6),
               FZ=>FZ_L(6),NZ=>NZ_L(6),OZ=>OZ_L(6),Q2Z=>Q2Z_L(6),
               QZ=>QZ_L(6)
               );

LOGIC_7 : SUPER_LOGIC
   Port Map ( A1=>A1_L(7),A2=>A2_L(7),A3=>A3_L(7),A4=>A4_L(7),
               A5=>A5_L(7),A6=>A6_L(7),B1=>B1_L(7),B2=>B2_L(7),
               C1=>C1_L(7),C2=>C2_L(7),D1=>D1_L(7),D2=>D2_L(7),
               E1=>E1_L(7),E2=>E2_L(7),F1=>F1_L(7),F2=>F2_L(7),
               F3=>F3_L(7),F4=>F4_L(7),F5=>F5_L(7),F6=>F6_L(7),
               MP=>MP_L(7),MS=>MS_L(7),NP=>NP_L(7),NS=>NS_L(7),
               OP=>OP_L(7),OS=>OS_L(7),PP=>PP_L(7),PS=>PS_L(7),
               QC=>QC_L(7),QR=>QR_L(7),QS=>QS_L(7),AZ=>AZ_L(7),
               FZ=>FZ_L(7),NZ=>NZ_L(7),OZ=>OZ_L(7),Q2Z=>Q2Z_L(7),
               QZ=>QZ_L(7)
               );

LOGIC_8 : SUPER_LOGIC
   Port Map ( A1=>A1_L(8),A2=>A2_L(8),A3=>A3_L(8),A4=>A4_L(8),
               A5=>A5_L(8),A6=>A6_L(8),B1=>B1_L(8),B2=>B2_L(8),
               C1=>C1_L(8),C2=>C2_L(8),D1=>D1_L(8),D2=>D2_L(8),
               E1=>E1_L(8),E2=>E2_L(8),F1=>F1_L(8),F2=>F2_L(8),
               F3=>F3_L(8),F4=>F4_L(8),F5=>F5_L(8),F6=>F6_L(8),
               MP=>MP_L(8),MS=>MS_L(8),NP=>NP_L(8),NS=>NS_L(8),
               OP=>OP_L(8),OS=>OS_L(8),PP=>PP_L(8),PS=>PS_L(8),
               QC=>QC_L(8),QR=>QR_L(8),QS=>QS_L(8),AZ=>AZ_L(8),
               FZ=>FZ_L(8),NZ=>NZ_L(8),OZ=>OZ_L(8),Q2Z=>Q2Z_L(8),
               QZ=>QZ_L(8)
               );

LOGIC_9 : SUPER_LOGIC
   Port Map ( A1=>A1_L(9),A2=>A2_L(9),A3=>A3_L(9),A4=>A4_L(9),
               A5=>A5_L(9),A6=>A6_L(9),B1=>B1_L(9),B2=>B2_L(9),
               C1=>C1_L(9),C2=>C2_L(9),D1=>D1_L(9),D2=>D2_L(9),
               E1=>E1_L(9),E2=>E2_L(9),F1=>F1_L(9),F2=>F2_L(9),
```

```
               F3=>F3_L(9),F4=>F4_L(9),F5=>F5_L(9),F6=>F6_L(9),
               MP=>MP_L(9),MS=>MS_L(9),NP=>NP_L(9),NS=>NS_L(9),
               OP=>OP_L(9),OS=>OS_L(9),PP=>PP_L(9),PS=>PS_L(9),
               QC=>QC_L(9),QR=>QR_L(9),QS=>QS_L(9),AZ=>AZ_L(9),
               FZ=>FZ_L(9),NZ=>NZ_L(9),OZ=>OZ_L(9),Q2Z=>Q2Z_L(9),
               QZ=>QZ_L(9)
               );

LOGIC_10 : SUPER_LOGIC
   Port Map ( A1=>A1_L(10),A2=>A2_L(10),A3=>A3_L(10),A4=>A4_L(10),
               A5=>A5_L(10),A6=>A6_L(10),B1=>B1_L(10),B2=>B2_L(10),
               C1=>C1_L(10),C2=>C2_L(10),D1=>D1_L(10),D2=>D2_L(10),
               E1=>E1_L(10),E2=>E2_L(10),F1=>F1_L(10),F2=>F2_L(10),
               F3=>F3_L(10),F4=>F4_L(10),F5=>F5_L(10),F6=>F6_L(10),
               MP=>MP_L(10),MS=>MS_L(10),NP=>NP_L(10),NS=>NS_L(10),
               OP=>OP_L(10),OS=>OS_L(10),PP=>PP_L(10),PS=>PS_L(10),
               QC=>QC_L(10),QR=>QR_L(10),QS=>QS_L(10),AZ=>AZ_L(10),
               FZ=>FZ_L(10),NZ=>NZ_L(10),OZ=>OZ_L(10),Q2Z=>Q2Z_L(10),
               QZ=>QZ_L(10)
               );

LOGIC_11 : SUPER_LOGIC
   Port Map ( A1=>A1_L(11),A2=>A2_L(11),A3=>A3_L(11),A4=>A4_L(11),
               A5=>A5_L(11),A6=>A6_L(11),B1=>B1_L(11),B2=>B2_L(11),
               C1=>C1_L(11),C2=>C2_L(11),D1=>D1_L(11),D2=>D2_L(11),
               E1=>E1_L(11),E2=>E2_L(11),F1=>F1_L(11),F2=>F2_L(11),
               F3=>F3_L(11),F4=>F4_L(11),F5=>F5_L(11),F6=>F6_L(11),
               MP=>MP_L(11),MS=>MS_L(11),NP=>NP_L(11),NS=>NS_L(11),
               OP=>OP_L(11),OS=>OS_L(11),PP=>PP_L(11),PS=>PS_L(11),
               QC=>QC_L(11),QR=>QR_L(11),QS=>QS_L(11),AZ=>AZ_L(11),
               FZ=>FZ_L(11),NZ=>NZ_L(11),OZ=>OZ_L(11),Q2Z=>Q2Z_L(11),
               QZ=>QZ_L(11)
               );

LOGIC_12 : SUPER_LOGIC
   Port Map ( A1=>A1_L(12),A2=>A2_L(12),A3=>A3_L(12),A4=>A4_L(12),
               A5=>A5_L(12),A6=>A6_L(12),B1=>B1_L(12),B2=>B2_L(12),
               C1=>C1_L(12),C2=>C2_L(12),D1=>D1_L(12),D2=>D2_L(12),
               E1=>E1_L(12),E2=>E2_L(12),F1=>F1_L(12),F2=>F2_L(12),
               F3=>F3_L(12),F4=>F4_L(12),F5=>F5_L(12),F6=>F6_L(12),
               MP=>MP_L(12),MS=>MS_L(12),NP=>NP_L(12),NS=>NS_L(12),
               OP=>OP_L(12),OS=>OS_L(12),PP=>PP_L(12),PS=>PS_L(12),
               QC=>QC_L(12),QR=>QR_L(12),QS=>QS_L(12),AZ=>AZ_L(12),
               FZ=>FZ_L(12),NZ=>NZ_L(12),OZ=>OZ_L(12),Q2Z=>Q2Z_L(12),
               QZ=>QZ_L(12)
               );

LOGIC_13 : SUPER_LOGIC
   Port Map ( A1=>A1_L(13),A2=>A2_L(13),A3=>A3_L(13),A4=>A4_L(13),
               A5=>A5_L(13),A6=>A6_L(13),B1=>B1_L(13),B2=>B2_L(13),
               C1=>C1_L(13),C2=>C2_L(13),D1=>D1_L(13),D2=>D2_L(13),
               E1=>E1_L(13),E2=>E2_L(13),F1=>F1_L(13),F2=>F2_L(13),
               F3=>F3_L(13),F4=>F4_L(13),F5=>F5_L(13),F6=>F6_L(13),
               MP=>MP_L(13),MS=>MS_L(13),NP=>NP_L(13),NS=>NS_L(13),
               OP=>OP_L(13),OS=>OS_L(13),PP=>PP_L(13),PS=>PS_L(13),
               QC=>QC_L(13),QR=>QR_L(13),QS=>QS_L(13),AZ=>AZ_L(13),
               FZ=>FZ_L(13),NZ=>NZ_L(13),OZ=>OZ_L(13),Q2Z=>Q2Z_L(13),
               QZ=>QZ_L(13)
               );

LOGIC_14 : SUPER_LOGIC
   Port Map ( A1=>A1_L(14),A2=>A2_L(14),A3=>A3_L(14),A4=>A4_L(14),
               A5=>A5_L(14),A6=>A6_L(14),B1=>B1_L(14),B2=>B2_L(14),
               C1=>C1_L(14),C2=>C2_L(14),D1=>D1_L(14),D2=>D2_L(14),
               E1=>E1_L(14),E2=>E2_L(14),F1=>F1_L(14),F2=>F2_L(14),
               F3=>F3_L(14),F4=>F4_L(14),F5=>F5_L(14),F6=>F6_L(14),
               MP=>MP_L(14),MS=>MS_L(14),NP=>NP_L(14),NS=>NS_L(14),
               OP=>OP_L(14),OS=>OS_L(14),PP=>PP_L(14),PS=>PS_L(14),
               QC=>QC_L(14),QR=>QR_L(14),QS=>QS_L(14),AZ=>AZ_L(14),
               FZ=>FZ_L(14),NZ=>NZ_L(14),OZ=>OZ_L(14),Q2Z=>Q2Z_L(14),
               QZ=>QZ_L(14)
               );

LOGIC_15 : SUPER_LOGIC
   Port Map ( A1=>A1_L(15),A2=>A2_L(15),A3=>A3_L(15),A4=>A4_L(15),
               A5=>A5_L(15),A6=>A6_L(15),B1=>B1_L(15),B2=>B2_L(15),
               C1=>C1_L(15),C2=>C2_L(15),D1=>D1_L(15),D2=>D2_L(15),
```

```
                E1=>E1_L(15),E2=>E2_L(15),F1=>F1_L(15),F2=>F2_L(15),
                F3=>F3_L(15),F4=>F4_L(15),F5=>F5_L(15),F6=>F6_L(15),
                MP=>MP_L(15),MS=>MS_L(15),NP=>NP_L(15),NS=>NS_L(15),
                OP=>OP_L(15),OS=>OS_L(15),PP=>PP_L(15),PS=>PS_L(15),
                QC=>QC_L(15),QR=>QR_L(15),QS=>QS_L(15),AZ=>AZ_L(15),
                FZ=>FZ_L(15),NZ=>NZ_L(15),OZ=>OZ_L(15),Q2Z=>Q2Z_L(15),
                QZ=>QZ_L(15)
                );

LOGIC_16 : SUPER_LOGIC
   Port Map ( A1=>A1_L(16),A2=>A2_L(16),A3=>A3_L(16),A4=>A4_L(16),
                A5=>A5_L(16),A6=>A6_L(16),B1=>B1_L(16),B2=>B2_L(16),
                C1=>C1_L(16),C2=>C2_L(16),D1=>D1_L(16),D2=>D2_L(16),
                E1=>E1_L(16),E2=>E2_L(16),F1=>F1_L(16),F2=>F2_L(16),
                F3=>F3_L(16),F4=>F4_L(16),F5=>F5_L(16),F6=>F6_L(16),
                MP=>MP_L(16),MS=>MS_L(16),NP=>NP_L(16),NS=>NS_L(16),
                OP=>OP_L(16),OS=>OS_L(16),PP=>PP_L(16),PS=>PS_L(16),
                QC=>QC_L(16),QR=>QR_L(16),QS=>QS_L(16),AZ=>AZ_L(16),
                FZ=>FZ_L(16),NZ=>NZ_L(16),OZ=>OZ_L(16),Q2Z=>Q2Z_L(16),
                QZ=>QZ_L(16)
                );

LOGIC_17 : SUPER_LOGIC
   Port Map ( A1=>A1_L(17),A2=>A2_L(17),A3=>A3_L(17),A4=>A4_L(17),
                A5=>A5_L(17),A6=>A6_L(17),B1=>B1_L(17),B2=>B2_L(17),
                C1=>C1_L(17),C2=>C2_L(17),D1=>D1_L(17),D2=>D2_L(17),
                E1=>E1_L(17),E2=>E2_L(17),F1=>F1_L(17),F2=>F2_L(17),
                F3=>F3_L(17),F4=>F4_L(17),F5=>F5_L(17),F6=>F6_L(17),
                MP=>MP_L(17),MS=>MS_L(17),NP=>NP_L(17),NS=>NS_L(17),
                OP=>OP_L(17),OS=>OS_L(17),PP=>PP_L(17),PS=>PS_L(17),
                QC=>QC_L(17),QR=>QR_L(17),QS=>QS_L(17),AZ=>AZ_L(17),
                FZ=>FZ_L(17),NZ=>NZ_L(17),OZ=>OZ_L(17),Q2Z=>Q2Z_L(17),
                QZ=>QZ_L(17)
                );

LOGIC_18 : SUPER_LOGIC
   Port Map ( A1=>A1_L(18),A2=>A2_L(18),A3=>A3_L(18),A4=>A4_L(18),
                A5=>A5_L(18),A6=>A6_L(18),B1=>B1_L(18),B2=>B2_L(18),
                C1=>C1_L(18),C2=>C2_L(18),D1=>D1_L(18),D2=>D2_L(18),
                E1=>E1_L(18),E2=>E2_L(18),F1=>F1_L(18),F2=>F2_L(18),
                F3=>F3_L(18),F4=>F4_L(18),F5=>F5_L(18),F6=>F6_L(18),
                MP=>MP_L(18),MS=>MS_L(18),NP=>NP_L(18),NS=>NS_L(18),
                OP=>OP_L(18),OS=>OS_L(18),PP=>PP_L(18),PS=>PS_L(18),
                QC=>QC_L(18),QR=>QR_L(18),QS=>QS_L(18),AZ=>AZ_L(18),
                FZ=>FZ_L(18),NZ=>NZ_L(18),OZ=>OZ_L(18),Q2Z=>Q2Z_L(18),
                QZ=>QZ_L(18)
                );

LOGIC_19 : SUPER_LOGIC
   Port Map ( A1=>A1_L(19),A2=>A2_L(19),A3=>A3_L(19),A4=>A4_L(19),
                A5=>A5_L(19),A6=>A6_L(19),B1=>B1_L(19),B2=>B2_L(19),
                C1=>C1_L(19),C2=>C2_L(19),D1=>D1_L(19),D2=>D2_L(19),
                E1=>E1_L(19),E2=>E2_L(19),F1=>F1_L(19),F2=>F2_L(19),
                F3=>F3_L(19),F4=>F4_L(19),F5=>F5_L(19),F6=>F6_L(19),
                MP=>MP_L(19),MS=>MS_L(19),NP=>NP_L(19),NS=>NS_L(19),
                OP=>OP_L(19),OS=>OS_L(19),PP=>PP_L(19),PS=>PS_L(19),
                QC=>QC_L(19),QR=>QR_L(19),QS=>QS_L(19),AZ=>AZ_L(19),
                FZ=>FZ_L(19),NZ=>NZ_L(19),OZ=>OZ_L(19),Q2Z=>Q2Z_L(19),
                QZ=>QZ_L(19)
                );

LOGIC_20 : SUPER_LOGIC
   Port Map ( A1=>A1_L(20),A2=>A2_L(20),A3=>A3_L(20),A4=>A4_L(20),
                A5=>A5_L(20),A6=>A6_L(20),B1=>B1_L(20),B2=>B2_L(20),
                C1=>C1_L(20),C2=>C2_L(20),D1=>D1_L(20),D2=>D2_L(20),
                E1=>E1_L(20),E2=>E2_L(20),F1=>F1_L(20),F2=>F2_L(20),
                F3=>F3_L(20),F4=>F4_L(20),F5=>F5_L(20),F6=>F6_L(20),
                MP=>MP_L(20),MS=>MS_L(20),NP=>NP_L(20),NS=>NS_L(20),
                OP=>OP_L(20),OS=>OS_L(20),PP=>PP_L(20),PS=>PS_L(20),
                QC=>QC_L(20),QR=>QR_L(20),QS=>QS_L(20),AZ=>AZ_L(20),
                FZ=>FZ_L(20),NZ=>NZ_L(20),OZ=>OZ_L(20),Q2Z=>Q2Z_L(20),
                QZ=>QZ_L(20)
                );

 LOGIC_21 : SUPER_LOGIC
   Port Map ( A1=>A1_L(21),A2=>A2_L(21),A3=>A3_L(21),A4=>A4_L(21),
                A5=>A5_L(21),A6=>A6_L(21),B1=>B1_L(21),B2=>B2_L(21),
```

```
               C1=>C1_L(21),C2=>C2_L(21),D1=>D1_L(21),D2=>D2_L(21),
               E1=>E1_L(21),E2=>E2_L(21),F1=>F1_L(21),F2=>F2_L(21),
               F3=>F3_L(21),F4=>F4_L(21),F5=>F5_L(21),F6=>F6_L(21),
               MP=>MP_L(21),MS=>MS_L(21),NP=>NP_L(21),NS=>NS_L(21),
               OP=>OP_L(21),OS=>OS_L(21),PP=>PP_L(21),PS=>PS_L(21),
               QC=>QC_L(21),QR=>QR_L(21),QS=>QS_L(21),AZ=>AZ_L(21),
               FZ=>FZ_L(21),NZ=>NZ_L(21),OZ=>OZ_L(21),Q2Z=>Q2Z_L(21),
               QZ=>QZ_L(21)
               );

LOGIC_22 : SUPER_LOGIC
   Port Map ( A1=>A1_L(22),A2=>A2_L(22),A3=>A3_L(22),A4=>A4_L(22),
               A5=>A5_L(22),A6=>A6_L(22),B1=>B1_L(22),B2=>B2_L(22),
               C1=>C1_L(22),C2=>C2_L(22),D1=>D1_L(22),D2=>D2_L(22),
               E1=>E1_L(22),E2=>E2_L(22),F1=>F1_L(22),F2=>F2_L(22),
               F3=>F3_L(22),F4=>F4_L(22),F5=>F5_L(22),F6=>F6_L(22),
               MP=>MP_L(22),MS=>MS_L(22),NP=>NP_L(22),NS=>NS_L(22),
               OP=>OP_L(22),OS=>OS_L(22),PP=>PP_L(22),PS=>PS_L(22),
               QC=>QC_L(22),QR=>QR_L(22),QS=>QS_L(22),AZ=>AZ_L(22),
               FZ=>FZ_L(22),NZ=>NZ_L(22),OZ=>OZ_L(22),Q2Z=>Q2Z_L(22),
               QZ=>QZ_L(22)
               );

LOGIC_23 : SUPER_LOGIC
   Port Map ( A1=>A1_L(23),A2=>A2_L(23),A3=>A3_L(23),A4=>A4_L(23),
               A5=>A5_L(23),A6=>A6_L(23),B1=>B1_L(23),B2=>B2_L(23),
               C1=>C1_L(23),C2=>C2_L(23),D1=>D1_L(23),D2=>D2_L(23),
               E1=>E1_L(23),E2=>E2_L(23),F1=>F1_L(23),F2=>F2_L(23),
               F3=>F3_L(23),F4=>F4_L(23),F5=>F5_L(23),F6=>F6_L(23),
               MP=>MP_L(23),MS=>MS_L(23),NP=>NP_L(23),NS=>NS_L(23),
               OP=>OP_L(23),OS=>OS_L(23),PP=>PP_L(23),PS=>PS_L(23),
               QC=>QC_L(23),QR=>QR_L(23),QS=>QS_L(23),AZ=>AZ_L(23),
               FZ=>FZ_L(23),NZ=>NZ_L(23),OZ=>OZ_L(23),Q2Z=>Q2Z_L(23),
               QZ=>QZ_L(23)
               );

LOGIC_24 : SUPER_LOGIC
   Port Map ( A1=>A1_L(24),A2=>A2_L(24),A3=>A3_L(24),A4=>A4_L(24),
               A5=>A5_L(24),A6=>A6_L(24),B1=>B1_L(24),B2=>B2_L(24),
               C1=>C1_L(24),C2=>C2_L(24),D1=>D1_L(24),D2=>D2_L(24),
               E1=>E1_L(24),E2=>E2_L(24),F1=>F1_L(24),F2=>F2_L(24),
               F3=>F3_L(24),F4=>F4_L(24),F5=>F5_L(24),F6=>F6_L(24),
               MP=>MP_L(24),MS=>MS_L(24),NP=>NP_L(24),NS=>NS_L(24),
               OP=>OP_L(24),OS=>OS_L(24),PP=>PP_L(24),PS=>PS_L(24),
               QC=>QC_L(24),QR=>QR_L(24),QS=>QS_L(24),AZ=>AZ_L(24),
               FZ=>FZ_L(24),NZ=>NZ_L(24),OZ=>OZ_L(24),Q2Z=>Q2Z_L(24),
               QZ=>QZ_L(24)
               );

LOGIC_25 : SUPER_LOGIC
   Port Map ( A1=>A1_L(25),A2=>A2_L(25),A3=>A3_L(25),A4=>A4_L(25),
               A5=>A5_L(25),A6=>A6_L(25),B1=>B1_L(25),B2=>B2_L(25),
               C1=>C1_L(25),C2=>C2_L(25),D1=>D1_L(25),D2=>D2_L(25),
               E1=>E1_L(25),E2=>E2_L(25),F1=>F1_L(25),F2=>F2_L(25),
               F3=>F3_L(25),F4=>F4_L(25),F5=>F5_L(25),F6=>F6_L(25),
               MP=>MP_L(25),MS=>MS_L(25),NP=>NP_L(25),NS=>NS_L(25),
               OP=>OP_L(25),OS=>OS_L(25),PP=>PP_L(25),PS=>PS_L(25),
               QC=>QC_L(25),QR=>QR_L(25),QS=>QS_L(25),AZ=>AZ_L(25),
               FZ=>FZ_L(25),NZ=>NZ_L(25),OZ=>OZ_L(25),Q2Z=>Q2Z_L(25),
               QZ=>QZ_L(25)
               );

LOGIC_26 : SUPER_LOGIC
   Port Map ( A1=>A1_L(26),A2=>A2_L(26),A3=>A3_L(26),A4=>A4_L(26),
               A5=>A5_L(26),A6=>A6_L(26),B1=>B1_L(26),B2=>B2_L(26),
               C1=>C1_L(26),C2=>C2_L(26),D1=>D1_L(26),D2=>D2_L(26),
               E1=>E1_L(26),E2=>E2_L(26),F1=>F1_L(26),F2=>F2_L(26),
               F3=>F3_L(26),F4=>F4_L(26),F5=>F5_L(26),F6=>F6_L(26),
               MP=>MP_L(26),MS=>MS_L(26),NP=>NP_L(26),NS=>NS_L(26),
               OP=>OP_L(26),OS=>OS_L(26),PP=>PP_L(26),PS=>PS_L(26),
               QC=>QC_L(26),QR=>QR_L(26),QS=>QS_L(26),AZ=>AZ_L(26),
               FZ=>FZ_L(26),NZ=>NZ_L(26),OZ=>OZ_L(26),Q2Z=>Q2Z_L(26),
               QZ=>QZ_L(26)
               );

LOGIC_27 : SUPER_LOGIC
   Port Map ( A1=>A1_L(27),A2=>A2_L(27),A3=>A3_L(27),A4=>A4_L(27),
```

```
                A5=>A5_L(27),A6=>A6_L(27),B1=>B1_L(27),B2=>B2_L(27),
                C1=>C1_L(27),C2=>C2_L(27),D1=>D1_L(27),D2=>D2_L(27),
                E1=>E1_L(27),E2=>E2_L(27),F1=>F1_L(27),F2=>F2_L(27),
                F3=>F3_L(27),F4=>F4_L(27),F5=>F5_L(27),F6=>F6_L(27),
                MP=>MP_L(27),MS=>MS_L(27),NP=>NP_L(27),NS=>NS_L(27),
                OP=>OP_L(27),OS=>OS_L(27),PP=>PP_L(27),PS=>PS_L(27),
                QC=>QC_L(27),QR=>QR_L(27),QS=>QS_L(27),AZ=>AZ_L(27),
                FZ=>FZ_L(27),NZ=>NZ_L(27),OZ=>OZ_L(27),Q2Z=>Q2Z_L(27),
                QZ=>QZ_L(27)
                );

   LOGIC_28 : SUPER_LOGIC
      Port Map ( A1=>A1_L(28),A2=>A2_L(28),A3=>A3_L(28),A4=>A4_L(28),
                A5=>A5_L(28),A6=>A6_L(28),B1=>B1_L(28),B2=>B2_L(28),
                C1=>C1_L(28),C2=>C2_L(28),D1=>D1_L(28),D2=>D2_L(28),
                E1=>E1_L(28),E2=>E2_L(28),F1=>F1_L(28),F2=>F2_L(28),
                F3=>F3_L(28),F4=>F4_L(28),F5=>F5_L(28),F6=>F6_L(28),
                MP=>MP_L(28),MS=>MS_L(28),NP=>NP_L(28),NS=>NS_L(28),
                OP=>OP_L(28),OS=>OS_L(28),PP=>PP_L(28),PS=>PS_L(28),
                QC=>QC_L(28),QR=>QR_L(28),QS=>QS_L(28),AZ=>AZ_L(28),
                FZ=>FZ_L(28),NZ=>NZ_L(28),OZ=>OZ_L(28),Q2Z=>Q2Z_L(28),
                QZ=>QZ_L(28)
                );

   LOGIC_29 : SUPER_LOGIC
      Port Map ( A1=>A1_L(29),A2=>A2_L(29),A3=>A3_L(29),A4=>A4_L(29),
                A5=>A5_L(29),A6=>A6_L(29),B1=>B1_L(29),B2=>B2_L(29),
                C1=>C1_L(29),C2=>C2_L(29),D1=>D1_L(29),D2=>D2_L(29),
                E1=>E1_L(29),E2=>E2_L(29),F1=>F1_L(29),F2=>F2_L(29),
                F3=>F3_L(29),F4=>F4_L(29),F5=>F5_L(29),F6=>F6_L(29),
                MP=>MP_L(29),MS=>MS_L(29),NP=>NP_L(29),NS=>NS_L(29),
                OP=>OP_L(29),OS=>OS_L(29),PP=>PP_L(29),PS=>PS_L(29),
                QC=>QC_L(29),QR=>QR_L(29),QS=>QS_L(29),AZ=>AZ_L(29),
                FZ=>FZ_L(29),NZ=>NZ_L(29),OZ=>OZ_L(29),Q2Z=>Q2Z_L(29),
                QZ=>QZ_L(29)
                );

   LOGIC_30 : SUPER_LOGIC
      Port Map ( A1=>A1_L(30),A2=>A2_L(30),A3=>A3_L(30),A4=>A4_L(30),
                A5=>A5_L(30),A6=>A6_L(30),B1=>B1_L(30),B2=>B2_L(30),
                C1=>C1_L(30),C2=>C2_L(30),D1=>D1_L(30),D2=>D2_L(30),
                E1=>E1_L(30),E2=>E2_L(30),F1=>F1_L(30),F2=>F2_L(30),
                F3=>F3_L(30),F4=>F4_L(30),F5=>F5_L(30),F6=>F6_L(30),
                MP=>MP_L(30),MS=>MS_L(30),NP=>NP_L(30),NS=>NS_L(30),
                OP=>OP_L(30),OS=>OS_L(30),PP=>PP_L(30),PS=>PS_L(30),
                QC=>QC_L(30),QR=>QR_L(30),QS=>QS_L(30),AZ=>AZ_L(30),
                FZ=>FZ_L(30),NZ=>NZ_L(30),OZ=>OZ_L(30),Q2Z=>Q2Z_L(30),
                QZ=>QZ_L(30)
                );

   LOGIC_31 : SUPER_LOGIC
      Port Map ( A1=>A1_L(31),A2=>A2_L(31),A3=>A3_L(31),A4=>A4_L(31),
                A5=>A5_L(31),A6=>A6_L(31),B1=>B1_L(31),B2=>B2_L(31),
                C1=>C1_L(31),C2=>C2_L(31),D1=>D1_L(31),D2=>D2_L(31),
                E1=>E1_L(31),E2=>E2_L(31),F1=>F1_L(31),F2=>F2_L(31),
                F3=>F3_L(31),F4=>F4_L(31),F5=>F5_L(31),F6=>F6_L(31),
                MP=>MP_L(31),MS=>MS_L(31),NP=>NP_L(31),NS=>NS_L(31),
                OP=>OP_L(31),OS=>OS_L(31),PP=>PP_L(31),PS=>PS_L(31),
                QC=>QC_L(31),QR=>QR_L(31),QS=>QS_L(31),AZ=>AZ_L(31),
                FZ=>FZ_L(31),NZ=>NZ_L(31),OZ=>OZ_L(31),Q2Z=>Q2Z_L(31),
                QZ=>QZ_L(31)
                );

-------------------------------------------------------
-- Bloco de saída 0
-------------------------------------------------------
--Célula lógica de saída 0 - BIT0 e BIT1
---------------------------
Dadooutbuffer(0) <= QZ_O(0);
Dadooutbuffer(1) <= Q2Z_O(0);
QC_O(0) <= Clock;
QR_O(0) <= Reset;
A1_O(0) <= GND;
A2_O(0) <= GND;
A3_O(0) <= GND;
A4_O(0) <= GND;
A5_O(0) <= GND;
```

```
A6_O(0) <= GND;
F1_O(0) <= GND;
F2_O(0) <= GND;
F3_O(0) <= GND;
F4_O(0) <= GND;
F5_O(0) <= GND;
F6_O(0) <= GND;
MP_O(0) <= GND;
MS_O(0) <= Load;
NP_O(0) <= GND;
NS_O(0) <= Load;
OP_O(0) <= GND;
OS_O(0) <= GND;
PP_O(0) <= GND;
PS_O(0) <= GND;
QS_O(0) <= GND;
B1_O(0) <= AZ_L(1);--
B2_O(0) <= GND;
C1_O(0) <= GND;--
C2_O(0) <= GND;
D1_O(0) <= AZ_L(3);--
D2_O(0) <= GND;
E1_O(0) <= GND;--
E2_O(0) <= GND;
--------------------------

--Célula lógica 0 - BIT0
--------------------------
A1_L(0)<=Dadooutbuffer(0);--
A2_L(0)<=Enable;--
A3_L(0)<=VCC;
A4_L(0)<=GND;
A5_L(0)<=VCC;
A6_L(0)<=GND;
B1_L(0)<=VCC;
B2_L(0)<=AZ_L(0);--
C1_L(0)<=GND;
C2_L(0)<=GND;
D1_L(0)<=OZ_L(0);--
D2_L(0)<=FZ_L(0);--
E1_L(0)<=GND;
E2_L(0)<=GND;
F1_L(0)<=Enable;--
F2_L(0)<=Dadooutbuffer(0);--
F3_L(0)<=VCC;
F4_L(0)<=GND;
F5_L(0)<=VCC;
F6_L(0)<=GND;
MP_L(0)<=GND;
MS_L(0)<=GND;
NP_L(0)<=GND;
NS_L(0)<=GND;
OP_L(0)<=GND;
OS_L(0)<=GND;
PP_L(0)<=GND;
PS_L(0)<=GND;
QC_L(0)<=GND;
QR_L(0)<=GND;
QS_L(0)<=GND;
--------------------------

--Célula lógica 1 - BIT0
--------------------------
A1_L(1)<=VCC;--
A2_L(1)<=NZ_L(0);--
A3_L(1)<=VCC;
A4_L(1)<=GND;
A5_L(1)<=VCC;
A6_L(1)<=GND;
B1_L(1)<=VCC;
B2_L(1)<=Enable;--
C1_L(1)<=GND;
C2_L(1)<=GND;
D1_L(1)<=Dadooutbuffer(0);--
D2_L(1)<=OZ_L(1);--
E1_L(1)<=GND;
E2_L(1)<=GND;
```

```
F1_L(1)<=GND;
F2_L(1)<=GND;
F3_L(1)<=GND;
F4_L(1)<=GND;
F5_L(1)<=GND;
F6_L(1)<=GND;
MP_L(1)<=GND;
MS_L(1)<=GND;
NP_L(1)<=GND;
NS_L(1)<=GND;
OP_L(1)<=GND;
OS_L(1)<=GND;
PP_L(1)<=GND;
PS_L(1)<=GND;
QC_L(1)<=GND;
QR_L(1)<=GND;
QS_L(1)<=GND;
-------------------------

--Célula lógica 2 - BIT1
--------------------------
A1_L(2)<=Dadooutbuffer(1);--
A2_L(2)<=NZ_L(1);--
A3_L(2)<=VCC;
A4_L(2)<=GND;
A5_L(2)<=VCC;
A6_L(2)<=GND;
B1_L(2)<=VCC;
B2_L(2)<=AZ_L(2);--
C1_L(2)<=GND;
C2_L(2)<=GND;
D1_L(2)<=OZ_L(2);--
D2_L(2)<=FZ_L(2);--
E1_L(2)<=GND;
E2_L(2)<=GND;
F1_L(2)<=NZ_L(1);--
F2_L(2)<=Dadooutbuffer(1);--
F3_L(2)<=VCC;
F4_L(2)<=GND;
F5_L(2)<=VCC;
F6_L(2)<=GND;
MP_L(2)<=GND;
MS_L(2)<=GND;
NP_L(2)<=GND;
NS_L(2)<=GND;
OP_L(2)<=GND;
OS_L(2)<=GND;
PP_L(2)<=GND;
PS_L(2)<=GND;
QC_L(2)<=GND;
QR_L(2)<=GND;
QS_L(2)<=GND;
--------------------------

--Célula lógica 3 - BIT1
--------------------------
A1_L(3)<=VCC;--
A2_L(3)<=NZ_L(2);--
A3_L(3)<=VCC;
A4_L(3)<=GND;
A5_L(3)<=VCC;
A6_L(3)<=GND;
B1_L(3)<=VCC;
B2_L(3)<=NZ_L(1);--
C1_L(3)<=GND;
C2_L(3)<=GND;
D1_L(3)<=Dadooutbuffer(1);--
D2_L(3)<=OZ_L(3);--
E1_L(3)<=GND;
E2_L(3)<=GND;
F1_L(3)<=GND;
F2_L(3)<=GND;
F3_L(3)<=GND;
F4_L(3)<=GND;
F5_L(3)<=GND;
F6_L(3)<=GND;
MP_L(3)<=GND;
```

```
MS_L(3)<=GND;
NP_L(3)<=GND;
NS_L(3)<=GND;
OP_L(3)<=GND;
OS_L(3)<=GND;
PP_L(3)<=GND;
PS_L(3)<=GND;
QC_L(3)<=GND;
QR_L(3)<=GND;
QS_L(3)<=GND;
-------------------------

-----------------------------------------------------
-- Bloco de saída 1
-----------------------------------------------------
--Célula lógica de saída 1 - BIT2 e BIT3
-------------------------
Dadooutbuffer(2) <= QZ_O(1);
Dadooutbuffer(3) <= Q2Z_O(1);
QC_O(1) <= Clock;
QR_O(1) <= Reset;
A1_O(1) <= GND;
A2_O(1) <= GND;
A3_O(1) <= GND;
A4_O(1) <= GND;
A5_O(1) <= GND;
A6_O(1) <= GND;
F1_O(1) <= GND;
F2_O(1) <= GND;
F3_O(1) <= GND;
F4_O(1) <= GND;
F5_O(1) <= GND;
F6_O(1) <= GND;
MP_O(1) <= GND;
MS_O(1) <= Load;
NP_O(1) <= GND;
NS_O(1) <= Load;
OP_O(1) <= GND;
OS_O(1) <= GND;
PP_O(1) <= GND;
PS_O(1) <= GND;
QS_O(1) <= GND;
B1_O(1) <= AZ_L(5);--
B2_O(1) <= GND;
C1_O(1) <= GND;--
C2_O(1) <= GND;
D1_O(1) <= AZ_L(7);--
D2_O(1) <= GND;
E1_O(1) <= GND;--
E2_O(1) <= GND;
-------------------------

--Célula lógica 4 - BIT2
-------------------------
A1_L(4)<=Dadooutbuffer(2);--
A2_L(4)<=NZ_L(3);--
A3_L(4)<=VCC;
A4_L(4)<=GND;
A5_L(4)<=VCC;
A6_L(4)<=GND;
B1_L(4)<=VCC;
B2_L(4)<=AZ_L(4);--
C1_L(4)<=GND;
C2_L(4)<=GND;
D1_L(4)<=OZ_L(4);--
D2_L(4)<=FZ_L(4);--
E1_L(4)<=GND;
E2_L(4)<=GND;
F1_L(4)<=NZ_L(3);--
F2_L(4)<=Dadooutbuffer(2);--
F3_L(4)<=VCC;
F4_L(4)<=GND;
F5_L(4)<=VCC;
F6_L(4)<=GND;
MP_L(4)<=GND;
MS_L(4)<=GND;
NP_L(4)<=GND;
```

```
NS_L(4)<=GND;
OP_L(4)<=GND;
OS_L(4)<=GND;
PP_L(4)<=GND;
PS_L(4)<=GND;
QC_L(4)<=GND;
QR_L(4)<=GND;
QS_L(4)<=GND;
---------------------------

--Célula lógica 5 - BIT2
---------------------------
A1_L(5)<=VCC;--
A2_L(5)<=NZ_L(4);--
A3_L(5)<=VCC;
A4_L(5)<=GND;
A5_L(5)<=VCC;
A6_L(5)<=GND;
B1_L(5)<=VCC;
B2_L(5)<=NZ_L(3);--
C1_L(5)<=GND;
C2_L(5)<=GND;
D1_L(5)<=Dadooutbuffer(2);--
D2_L(5)<=OZ_L(5);--
E1_L(5)<=GND;
E2_L(5)<=GND;
F1_L(5)<=GND;
F2_L(5)<=GND;
F3_L(5)<=GND;
F4_L(5)<=GND;
F5_L(5)<=GND;
F6_L(5)<=GND;
MP_L(5)<=GND;
MS_L(5)<=GND;
NP_L(5)<=GND;
NS_L(5)<=GND;
OP_L(5)<=GND;
OS_L(5)<=GND;
PP_L(5)<=GND;
PS_L(5)<=GND;
QC_L(5)<=GND;
QR_L(5)<=GND;
QS_L(5)<=GND;
-------------------------

--Célula lógica 6 - BIT3
---------------------------
A1_L(6)<=Dadooutbuffer(3);--
A2_L(6)<=NZ_L(5);--
A3_L(6)<=VCC;
A4_L(6)<=GND;
A5_L(6)<=VCC;
A6_L(6)<=GND;
B1_L(6)<=VCC;
B2_L(6)<=AZ_L(6);--
C1_L(6)<=GND;
C2_L(6)<=GND;
D1_L(6)<=OZ_L(6);--
D2_L(6)<=FZ_L(6);--
E1_L(6)<=GND;
E2_L(6)<=GND;
F1_L(6)<=NZ_L(5);--
F2_L(6)<=Dadooutbuffer(3);--
F3_L(6)<=VCC;
F4_L(6)<=GND;
F5_L(6)<=VCC;
F6_L(6)<=GND;
MP_L(6)<=GND;
MS_L(6)<=GND;
NP_L(6)<=GND;
NS_L(6)<=GND;
OP_L(6)<=GND;
OS_L(6)<=GND;
PP_L(6)<=GND;
PS_L(6)<=GND;
QC_L(6)<=GND;
QR_L(6)<=GND;
```

```
QS_L(6)<=GND;
--------------------------

--Célula lógica 7 - BIT3
--------------------------
A1_L(7)<=VCC;--
A2_L(7)<=NZ_L(6);--
A3_L(7)<=VCC;
A4_L(7)<=GND;
A5_L(7)<=VCC;
A6_L(7)<=GND;
B1_L(7)<=VCC;
B2_L(7)<=NZ_L(5);--
C1_L(7)<=GND;
C2_L(7)<=GND;
D1_L(7)<=Dadooutbuffer(3);--
D2_L(7)<=OZ_L(7);--
E1_L(7)<=GND;
E2_L(7)<=GND;
F1_L(7)<=GND;
F2_L(7)<=GND;
F3_L(7)<=GND;
F4_L(7)<=GND;
F5_L(7)<=GND;
F6_L(7)<=GND;
MP_L(7)<=GND;
MS_L(7)<=GND;
NP_L(7)<=GND;
NS_L(7)<=GND;
OP_L(7)<=GND;
OS_L(7)<=GND;
PP_L(7)<=GND;
PS_L(7)<=GND;
QC_L(7)<=GND;
QR_L(7)<=GND;
QS_L(7)<=GND;
--------------------------

-----------------------------------------------------
-- Bloco de saída 2
-----------------------------------------------------
--Célula lógica de saída 2 - BIT4 e BIT5
--------------------------
Dadooutbuffer(4) <= QZ_O(2);
Dadooutbuffer(5) <= Q2Z_O(2);
QC_O(2) <= Clock;
QR_O(2) <= Reset;
A1_O(2) <= GND;
A2_O(2) <= GND;
A3_O(2) <= GND;
A4_O(2) <= GND;
A5_O(2) <= GND;
A6_O(2) <= GND;
F1_O(2) <= GND;
F2_O(2) <= GND;
F3_O(2) <= GND;
F4_O(2) <= GND;
F5_O(2) <= GND;
F6_O(2) <= GND;
MP_O(2) <= GND;
MS_O(2) <= Load;
NP_O(2) <= GND;
NS_O(2) <= Load;
OP_O(2) <= GND;
OS_O(2) <= GND;
PP_O(2) <= GND;
PS_O(2) <= GND;
QS_O(2) <= GND;
B1_O(2) <= AZ_L(9);--
B2_O(2) <= GND;
C1_O(2) <= GND;--
C2_O(2) <= GND;
D1_O(2) <= AZ_L(11);--
D2_O(2) <= GND;
E1_O(2) <= GND;--
E2_O(2) <= GND;
--------------------------
```

```
--Célula lógica 8 - BIT4
--------------------------
A1_L(8)<=Dadooutbuffer(4);--
A2_L(8)<=NZ_L(7);--
A3_L(8)<=VCC;
A4_L(8)<=GND;
A5_L(8)<=VCC;
A6_L(8)<=GND;
B1_L(8)<=VCC;
B2_L(8)<=AZ_L(8);--
C1_L(8)<=GND;
C2_L(8)<=GND;
D1_L(8)<=OZ_L(8);--
D2_L(8)<=FZ_L(8);--
E1_L(8)<=GND;
E2_L(8)<=GND;
F1_L(8)<=NZ_L(7);--
F2_L(8)<=Dadooutbuffer(4);--
F3_L(8)<=VCC;
F4_L(8)<=GND;
F5_L(8)<=VCC;
F6_L(8)<=GND;
MP_L(8)<=GND;
MS_L(8)<=GND;
NP_L(8)<=GND;
NS_L(8)<=GND;
OP_L(8)<=GND;
OS_L(8)<=GND;
PP_L(8)<=GND;
PS_L(8)<=GND;
QC_L(8)<=GND;
QR_L(8)<=GND;
QS_L(8)<=GND;
--------------------------

--Célula lógica 9 - BIT4
--------------------------
A1_L(9)<=VCC;--
A2_L(9)<=NZ_L(8);--
A3_L(9)<=VCC;
A4_L(9)<=GND;
A5_L(9)<=VCC;
A6_L(9)<=GND;
B1_L(9)<=VCC;
B2_L(9)<=NZ_L(7);--
C1_L(9)<=GND;
C2_L(9)<=GND;
D1_L(9)<=Dadooutbuffer(4);--
D2_L(9)<=OZ_L(9);--
E1_L(9)<=GND;
E2_L(9)<=GND;
F1_L(9)<=GND;
F2_L(9)<=GND;
F3_L(9)<=GND;
F4_L(9)<=GND;
F5_L(9)<=GND;
F6_L(9)<=GND;
MP_L(9)<=GND;
MS_L(9)<=GND;
NP_L(9)<=GND;
NS_L(9)<=GND;
OP_L(9)<=GND;
OS_L(9)<=GND;
PP_L(9)<=GND;
PS_L(9)<=GND;
QC_L(9)<=GND;
QR_L(9)<=GND;
QS_L(9)<=GND;
--------------------------

--Célula lógica 10 - BIT5
--------------------------
A1_L(10)<=Dadooutbuffer(5);--
A2_L(10)<=NZ_L(9);--
A3_L(10)<=VCC;
A4_L(10)<=GND;
```

```
A5_L(10)<=VCC;
A6_L(10)<=GND;
B1_L(10)<=VCC;
B2_L(10)<=AZ_L(10);--
C1_L(10)<=GND;
C2_L(10)<=GND;
D1_L(10)<=OZ_L(10);--
D2_L(10)<=FZ_L(10);--
E1_L(10)<=GND;
E2_L(10)<=GND;
F1_L(10)<=NZ_L(9);--
F2_L(10)<=Dadooutbuffer(5);--
F3_L(10)<=VCC;
F4_L(10)<=GND;
F5_L(10)<=VCC;
F6_L(10)<=GND;
MP_L(10)<=GND;
MS_L(10)<=GND;
NP_L(10)<=GND;
NS_L(10)<=GND;
OP_L(10)<=GND;
OS_L(10)<=GND;
PP_L(10)<=GND;
PS_L(10)<=GND;
QC_L(10)<=GND;
QR_L(10)<=GND;
QS_L(10)<=GND;
--------------------------

--Célula lógica 11 - BIT5
--------------------------
A1_L(11)<=VCC;--
A2_L(11)<=NZ_L(10);--
A3_L(11)<=VCC;
A4_L(11)<=GND;
A5_L(11)<=VCC;
A6_L(11)<=GND;
B1_L(11)<=VCC;
B2_L(11)<=NZ_L(9);--
C1_L(11)<=GND;
C2_L(11)<=GND;
D1_L(11)<=Dadooutbuffer(5);--
D2_L(11)<=OZ_L(11);--
E1_L(11)<=GND;
E2_L(11)<=GND;
F1_L(11)<=GND;
F2_L(11)<=GND;
F3_L(11)<=GND;
F4_L(11)<=GND;
F5_L(11)<=GND;
F6_L(11)<=GND;
MP_L(11)<=GND;
MS_L(11)<=GND;
NP_L(11)<=GND;
NS_L(11)<=GND;
OP_L(11)<=GND;
OS_L(11)<=GND;
PP_L(11)<=GND;
PS_L(11)<=GND;
QC_L(11)<=GND;
QR_L(11)<=GND;
QS_L(11)<=GND;
--------------------------

-----------------------------------------------------
-- Bloco de saída 3
-----------------------------------------------------
--Célula lógica de saída 3 - BIT6 e BIT7
--------------------------
Dadooutbuffer(6) <= QZ_O(3);
Dadooutbuffer(7) <= Q2Z_O(3);
QC_O(3) <= Clock;
QR_O(3) <= Reset;
A1_O(3) <= GND;
A2_O(3) <= GND;
A3_O(3) <= GND;
A4_O(3) <= GND;
```

```
A5_O(3) <= GND;
A6_O(3) <= GND;
F1_O(3) <= GND;
F2_O(3) <= GND;
F3_O(3) <= GND;
F4_O(3) <= GND;
F5_O(3) <= GND;
F6_O(3) <= GND;
MP_O(3) <= GND;
MS_O(3) <= Load;
NP_O(3) <= GND;
NS_O(3) <= Load;
OP_O(3) <= GND;
OS_O(3) <= GND;
PP_O(3) <= GND;
PS_O(3) <= GND;
QS_O(3) <= GND;
B1_O(3) <= AZ_L(13);--
B2_O(3) <= GND;
C1_O(3) <= GND;--
C2_O(3) <= GND;
D1_O(3) <= AZ_L(15);--
D2_O(3) <= GND;
E1_O(3) <= GND;--
E2_O(3) <= GND;
--------------------------

--Célula lógica 12 - BIT6
--------------------------
A1_L(12)<=Dadooutbuffer(6);--
A2_L(12)<=NZ_L(11);--
A3_L(12)<=VCC;
A4_L(12)<=GND;
A5_L(12)<=VCC;
A6_L(12)<=GND;
B1_L(12)<=VCC;
B2_L(12)<=AZ_L(12);--
C1_L(12)<=GND;
C2_L(12)<=GND;
D1_L(12)<=OZ_L(12);--
D2_L(12)<=FZ_L(12);--
E1_L(12)<=GND;
E2_L(12)<=GND;
F1_L(12)<=NZ_L(11);--
F2_L(12)<=Dadooutbuffer(6);--
F3_L(12)<=VCC;
F4_L(12)<=GND;
F5_L(12)<=VCC;
F6_L(12)<=GND;
MP_L(12)<=GND;
MS_L(12)<=GND;
NP_L(12)<=GND;
NS_L(12)<=GND;
OP_L(12)<=GND;
OS_L(12)<=GND;
PP_L(12)<=GND;
PS_L(12)<=GND;
QC_L(12)<=GND;
QR_L(12)<=GND;
QS_L(12)<=GND;
--------------------------

--Célula lógica 13 - BIT6
--------------------------
A1_L(13)<=VCC;--
A2_L(13)<=NZ_L(12);--
A3_L(13)<=VCC;
A4_L(13)<=GND;
A5_L(13)<=VCC;
A6_L(13)<=GND;
B1_L(13)<=VCC;
B2_L(13)<=NZ_L(11);--
C1_L(13)<=GND;
C2_L(13)<=GND;
D1_L(13)<=Dadooutbuffer(6);--
D2_L(13)<=OZ_L(13);--
E1_L(13)<=GND;
```

```
E2_L(13)<=GND;
F1_L(13)<=GND;
F2_L(13)<=GND;
F3_L(13)<=GND;
F4_L(13)<=GND;
F5_L(13)<=GND;
F6_L(13)<=GND;
MP_L(13)<=GND;
MS_L(13)<=GND;
NP_L(13)<=GND;
NS_L(13)<=GND;
OP_L(13)<=GND;
OS_L(13)<=GND;
PP_L(13)<=GND;
PS_L(13)<=GND;
QC_L(13)<=GND;
QR_L(13)<=GND;
QS_L(13)<=GND;
------------------------

--Célula lógica 14 - BIT7
--------------------------
A1_L(14)<=Dadooutbuffer(7);--
A2_L(14)<=NZ_L(13);--
A3_L(14)<=VCC;
A4_L(14)<=GND;
A5_L(14)<=VCC;
A6_L(14)<=GND;
B1_L(14)<=VCC;
B2_L(14)<=AZ_L(14);--
C1_L(14)<=GND;
C2_L(14)<=GND;
D1_L(14)<=OZ_L(14);--
D2_L(14)<=FZ_L(14);--
E1_L(14)<=GND;
E2_L(14)<=GND;
F1_L(14)<=NZ_L(13);--
F2_L(14)<=Dadooutbuffer(7);--
F3_L(14)<=VCC;
F4_L(14)<=GND;
F5_L(14)<=VCC;
F6_L(14)<=GND;
MP_L(14)<=GND;
MS_L(14)<=GND;
NP_L(14)<=GND;
NS_L(14)<=GND;
OP_L(14)<=GND;
OS_L(14)<=GND;
PP_L(14)<=GND;
PS_L(14)<=GND;
QC_L(14)<=GND;
QR_L(14)<=GND;
QS_L(14)<=GND;
--------------------------

--Célula lógica 15 - BIT7
--------------------------
A1_L(15)<=VCC;--
A2_L(15)<=NZ_L(14);--
A3_L(15)<=VCC;
A4_L(15)<=GND;
A5_L(15)<=VCC;
A6_L(15)<=GND;
B1_L(15)<=VCC;
B2_L(15)<=NZ_L(13);--
C1_L(15)<=GND;
C2_L(15)<=GND;
D1_L(15)<=Dadooutbuffer(7);--
D2_L(15)<=OZ_L(15);--
E1_L(15)<=GND;
E2_L(15)<=GND;
F1_L(15)<=GND;
F2_L(15)<=GND;
F3_L(15)<=GND;
F4_L(15)<=GND;
F5_L(15)<=GND;
F6_L(15)<=GND;
```

```
MP_L(15)<=GND;
MS_L(15)<=GND;
NP_L(15)<=GND;
NS_L(15)<=GND;
OP_L(15)<=GND;
OS_L(15)<=GND;
PP_L(15)<=GND;
PS_L(15)<=GND;
QC_L(15)<=GND;
QR_L(15)<=GND;
QS_L(15)<=GND;
-------------------------

-----------------------------------------------------
-- Bloco de saída 4
-----------------------------------------------------
--Célula lógica de saída 4 - BIT8 e BIT9
--------------------------
Dadooutbuffer(8) <= QZ_O(4);
Dadooutbuffer(9) <= Q2Z_O(4);
QC_O(4) <= Clock;
QR_O(4) <= Reset;
A1_O(4) <= GND;
A2_O(4) <= GND;
A3_O(4) <= GND;
A4_O(4) <= GND;
A5_O(4) <= GND;
A6_O(4) <= GND;
F1_O(4) <= GND;
F2_O(4) <= GND;
F3_O(4) <= GND;
F4_O(4) <= GND;
F5_O(4) <= GND;
F6_O(4) <= GND;
MP_O(4) <= GND;
MS_O(4) <= Load;
NP_O(4) <= GND;
NS_O(4) <= Load;
OP_O(4) <= GND;
OS_O(4) <= GND;
PP_O(4) <= GND;
PS_O(4) <= GND;
QS_O(4) <= GND;
B1_O(4) <= AZ_L(17);--
B2_O(4) <= GND;
C1_O(4) <= GND;--
C2_O(4) <= GND;
D1_O(4) <= AZ_L(19);--
D2_O(4) <= GND;
E1_O(4) <= GND;--
E2_O(4) <= GND;
--------------------------

--Célula lógica 16 - BIT8
--------------------------
A1_L(16)<=Dadooutbuffer(8);--
A2_L(16)<=NZ_L(15);--
A3_L(16)<=VCC;
A4_L(16)<=GND;
A5_L(16)<=VCC;
A6_L(16)<=GND;
B1_L(16)<=VCC;
B2_L(16)<=AZ_L(16);--
C1_L(16)<=GND;
C2_L(16)<=GND;
D1_L(16)<=OZ_L(16);--
D2_L(16)<=FZ_L(16);--
E1_L(16)<=GND;
E2_L(16)<=GND;
F1_L(16)<=NZ_L(15);--
F2_L(16)<=Dadooutbuffer(8);--
F3_L(16)<=VCC;
F4_L(16)<=GND;
F5_L(16)<=VCC;
F6_L(16)<=GND;
MP_L(16)<=GND;
MS_L(16)<=GND;
```

```
NP_L(16)<=GND;
NS_L(16)<=GND;
OP_L(16)<=GND;
OS_L(16)<=GND;
PP_L(16)<=GND;
PS_L(16)<=GND;
QC_L(16)<=GND;
QR_L(16)<=GND;
QS_L(16)<=GND;
--------------------------

--Célula lógica 17 - BIT8
--------------------------
A1_L(17)<=VCC;--
A2_L(17)<=NZ_L(16);--
A3_L(17)<=VCC;
A4_L(17)<=GND;
A5_L(17)<=VCC;
A6_L(17)<=GND;
B1_L(17)<=VCC;
B2_L(17)<=NZ_L(15);--
C1_L(17)<=GND;
C2_L(17)<=GND;
D1_L(17)<=Dadooutbuffer(8);--
D2_L(17)<=OZ_L(17);--
E1_L(17)<=GND;
E2_L(17)<=GND;
F1_L(17)<=GND;
F2_L(17)<=GND;
F3_L(17)<=GND;
F4_L(17)<=GND;
F5_L(17)<=GND;
F6_L(17)<=GND;
MP_L(17)<=GND;
MS_L(17)<=GND;
NP_L(17)<=GND;
NS_L(17)<=GND;
OP_L(17)<=GND;
OS_L(17)<=GND;
PP_L(17)<=GND;
PS_L(17)<=GND;
QC_L(17)<=GND;
QR_L(17)<=GND;
QS_L(17)<=GND;
------------------------

--Célula lógica 18 - BIT9
--------------------------
A1_L(18)<=Dadooutbuffer(9);--
A2_L(18)<=NZ_L(17);--
A3_L(18)<=VCC;
A4_L(18)<=GND;
A5_L(18)<=VCC;
A6_L(18)<=GND;
B1_L(18)<=VCC;
B2_L(18)<=AZ_L(18);--
C1_L(18)<=GND;
C2_L(18)<=GND;
D1_L(18)<=OZ_L(18);--
D2_L(18)<=FZ_L(18);--
E1_L(18)<=GND;
E2_L(18)<=GND;
F1_L(18)<=NZ_L(17);--
F2_L(18)<=Dadooutbuffer(9);--
F3_L(18)<=VCC;
F4_L(18)<=GND;
F5_L(18)<=VCC;
F6_L(18)<=GND;
MP_L(18)<=GND;
MS_L(18)<=GND;
NP_L(18)<=GND;
NS_L(18)<=GND;
OP_L(18)<=GND;
OS_L(18)<=GND;
PP_L(18)<=GND;
PS_L(18)<=GND;
QC_L(18)<=GND;
```

```
QR_L(18)<=GND;
QS_L(18)<=GND;
--------------------------

--Célula lógica 19 - BIT9
--------------------------
A1_L(19)<=VCC;--
A2_L(19)<=NZ_L(18);--
A3_L(19)<=VCC;
A4_L(19)<=GND;
A5_L(19)<=VCC;
A6_L(19)<=GND;
B1_L(19)<=VCC;
B2_L(19)<=NZ_L(17);--
C1_L(19)<=GND;
C2_L(19)<=GND;
D1_L(19)<=Dadooutbuffer(9);--
D2_L(19)<=OZ_L(19);--
E1_L(19)<=GND;
E2_L(19)<=GND;
F1_L(19)<=GND;
F2_L(19)<=GND;
F3_L(19)<=GND;
F4_L(19)<=GND;
F5_L(19)<=GND;
F6_L(19)<=GND;
MP_L(19)<=GND;
MS_L(19)<=GND;
NP_L(19)<=GND;
NS_L(19)<=GND;
OP_L(19)<=GND;
OS_L(19)<=GND;
PP_L(19)<=GND;
PS_L(19)<=GND;
QC_L(19)<=GND;
QR_L(19)<=GND;
QS_L(19)<=GND;
--------------------------

-----------------------------------------------------
-- Bloco de saída 5
-----------------------------------------------------
--Célula lógica de saída 5 - BIT10 e BIT11
--------------------------
Dadooutbuffer(10) <= QZ_O(5);
Dadooutbuffer(11) <= Q2Z_O(5);
QC_O(5) <= Clock;
QR_O(5) <= Reset;
A1_O(5) <= GND;
A2_O(5) <= GND;
A3_O(5) <= GND;
A4_O(5) <= GND;
A5_O(5) <= GND;
A6_O(5) <= GND;
F1_O(5) <= GND;
F2_O(5) <= GND;
F3_O(5) <= GND;
F4_O(5) <= GND;
F5_O(5) <= GND;
F6_O(5) <= GND;
MP_O(5) <= GND;
MS_O(5) <= Load;
NP_O(5) <= GND;
NS_O(5) <= Load;
OP_O(5) <= GND;
OS_O(5) <= GND;
PP_O(5) <= GND;
PS_O(5) <= GND;
QS_O(5) <= GND;
B1_O(5) <= AZ_L(21);--
B2_O(5) <= GND;
C1_O(5) <= GND;--
C2_O(5) <= GND;
D1_O(5) <= AZ_L(23);--
D2_O(5) <= GND;
E1_O(5) <= GND;--
E2_O(5) <= GND;
```

```
-------------------------

--Célula lógica 20 - BIT10
-------------------------
A1_L(20)<=Dadooutbuffer(10);--
A2_L(20)<=NZ_L(19);--
A3_L(20)<=VCC;
A4_L(20)<=GND;
A5_L(20)<=VCC;
A6_L(20)<=GND;
B1_L(20)<=VCC;
B2_L(20)<=AZ_L(20);--
C1_L(20)<=GND;
C2_L(20)<=GND;
D1_L(20)<=OZ_L(20);--
D2_L(20)<=FZ_L(20);--
E1_L(20)<=GND;
E2_L(20)<=GND;
F1_L(20)<=NZ_L(19);--
F2_L(20)<=Dadooutbuffer(10);--
F3_L(20)<=VCC;
F4_L(20)<=GND;
F5_L(20)<=VCC;
F6_L(20)<=GND;
MP_L(20)<=GND;
MS_L(20)<=GND;
NP_L(20)<=GND;
NS_L(20)<=GND;
OP_L(20)<=GND;
OS_L(20)<=GND;
PP_L(20)<=GND;
PS_L(20)<=GND;
QC_L(20)<=GND;
QR_L(20)<=GND;
QS_L(20)<=GND;
-------------------------

--Célula lógica 21 - BIT10
-------------------------
A1_L(21)<=VCC;--
A2_L(21)<=NZ_L(20);--
A3_L(21)<=VCC;
A4_L(21)<=GND;
A5_L(21)<=VCC;
A6_L(21)<=GND;
B1_L(21)<=VCC;
B2_L(21)<=NZ_L(19);--
C1_L(21)<=GND;
C2_L(21)<=GND;
D1_L(21)<=Dadooutbuffer(10);--
D2_L(21)<=OZ_L(21);--
E1_L(21)<=GND;
E2_L(21)<=GND;
F1_L(21)<=GND;
F2_L(21)<=GND;
F3_L(21)<=GND;
F4_L(21)<=GND;
F5_L(21)<=GND;
F6_L(21)<=GND;
MP_L(21)<=GND;
MS_L(21)<=GND;
NP_L(21)<=GND;
NS_L(21)<=GND;
OP_L(21)<=GND;
OS_L(21)<=GND;
PP_L(21)<=GND;
PS_L(21)<=GND;
QC_L(21)<=GND;
QR_L(21)<=GND;
QS_L(21)<=GND;
------------------------

--Célula lógica 22 - BIT11
-------------------------
A1_L(22)<=Dadooutbuffer(11);--
A2_L(22)<=NZ_L(21);--
A3_L(22)<=VCC;
```

```
A4_L(22)<=GND;
A5_L(22)<=VCC;
A6_L(22)<=GND;
B1_L(22)<=VCC;
B2_L(22)<=AZ_L(22);--
C1_L(22)<=GND;
C2_L(22)<=GND;
D1_L(22)<=OZ_L(22);--
D2_L(22)<=FZ_L(22);--
E1_L(22)<=GND;
E2_L(22)<=GND;
F1_L(22)<=NZ_L(21);--
F2_L(22)<=Dadooutbuffer(11);--
F3_L(22)<=VCC;
F4_L(22)<=GND;
F5_L(22)<=VCC;
F6_L(22)<=GND;
MP_L(22)<=GND;
MS_L(22)<=GND;
NP_L(22)<=GND;
NS_L(22)<=GND;
OP_L(22)<=GND;
OS_L(22)<=GND;
PP_L(22)<=GND;
PS_L(22)<=GND;
QC_L(22)<=GND;
QR_L(22)<=GND;
QS_L(22)<=GND;
--------------------------

--Célula lógica 23 - BIT11
--------------------------
A1_L(23)<=VCC;--
A2_L(23)<=NZ_L(22);--
A3_L(23)<=VCC;
A4_L(23)<=GND;
A5_L(23)<=VCC;
A6_L(23)<=GND;
B1_L(23)<=VCC;
B2_L(23)<=NZ_L(21);--
C1_L(23)<=GND;
C2_L(23)<=GND;
D1_L(23)<=Dadooutbuffer(11);--
D2_L(23)<=OZ_L(23);--
E1_L(23)<=GND;
E2_L(23)<=GND;
F1_L(23)<=GND;
F2_L(23)<=GND;
F3_L(23)<=GND;
F4_L(23)<=GND;
F5_L(23)<=GND;
F6_L(23)<=GND;
MP_L(23)<=GND;
MS_L(23)<=GND;
NP_L(23)<=GND;
NS_L(23)<=GND;
OP_L(23)<=GND;
OS_L(23)<=GND;
PP_L(23)<=GND;
PS_L(23)<=GND;
QC_L(23)<=GND;
QR_L(23)<=GND;
QS_L(23)<=GND;
--------------------------

-----------------------------------------------------
-- Bloco de saída 6
-----------------------------------------------------
--Célula lógica de saída 6 - BIT12 e BIT13
--------------------------
Dadooutbuffer(12) <= QZ_O(6);
Dadooutbuffer(13) <= Q2Z_O(6);
QC_O(6) <= Clock;
QR_O(6) <= Reset;
A1_O(6) <= GND;
A2_O(6) <= GND;
A3_O(6) <= GND;
```

```
A4_O(6) <= GND;
A5_O(6) <= GND;
A6_O(6) <= GND;
F1_O(6) <= GND;
F2_O(6) <= GND;
F3_O(6) <= GND;
F4_O(6) <= GND;
F5_O(6) <= GND;
F6_O(6) <= GND;
MP_O(6) <= GND;
MS_O(6) <= Load;
NP_O(6) <= GND;
NS_O(6) <= Load;
OP_O(6) <= GND;
OS_O(6) <= GND;
PP_O(6) <= GND;
PS_O(6) <= GND;
QS_O(6) <= GND;
B1_O(6) <= AZ_L(25);--
B2_O(6) <= GND;
C1_O(6) <= GND;--
C2_O(6) <= GND;
D1_O(6) <= AZ_L(27);--
D2_O(6) <= GND;
E1_O(6) <= GND;--
E2_O(6) <= GND;
--------------------------

--Célula lógica 24 - BIT12
--------------------------
A1_L(24)<=Dadooutbuffer(12);--
A2_L(24)<=NZ_L(23);--
A3_L(24)<=VCC;
A4_L(24)<=GND;
A5_L(24)<=VCC;
A6_L(24)<=GND;
B1_L(24)<=VCC;
B2_L(24)<=AZ_L(24);--
C1_L(24)<=GND;
C2_L(24)<=GND;
D1_L(24)<=OZ_L(24);--
D2_L(24)<=FZ_L(24);--
E1_L(24)<=GND;
E2_L(24)<=GND;
F1_L(24)<=NZ_L(23);--
F2_L(24)<=Dadooutbuffer(12);--
F3_L(24)<=VCC;
F4_L(24)<=GND;
F5_L(24)<=VCC;
F6_L(24)<=GND;
MP_L(24)<=GND;
MS_L(24)<=GND;
NP_L(24)<=GND;
NS_L(24)<=GND;
OP_L(24)<=GND;
OS_L(24)<=GND;
PP_L(24)<=GND;
PS_L(24)<=GND;
QC_L(24)<=GND;
QR_L(24)<=GND;
QS_L(24)<=GND;
--------------------------

--Célula lógica 25 - BIT12
--------------------------
A1_L(25)<=VCC;--
A2_L(25)<=NZ_L(24);--
A3_L(25)<=VCC;
A4_L(25)<=GND;
A5_L(25)<=VCC;
A6_L(25)<=GND;
B1_L(25)<=VCC;
B2_L(25)<=NZ_L(23);--
C1_L(25)<=GND;
C2_L(25)<=GND;
D1_L(25)<=Dadooutbuffer(12);--
D2_L(25)<=OZ_L(25);--
```

```
E1_L(25)<=GND;
E2_L(25)<=GND;
F1_L(25)<=GND;
F2_L(25)<=GND;
F3_L(25)<=GND;
F4_L(25)<=GND;
F5_L(25)<=GND;
F6_L(25)<=GND;
MP_L(25)<=GND;
MS_L(25)<=GND;
NP_L(25)<=GND;
NS_L(25)<=GND;
OP_L(25)<=GND;
OS_L(25)<=GND;
PP_L(25)<=GND;
PS_L(25)<=GND;
QC_L(25)<=GND;
QR_L(25)<=GND;
QS_L(25)<=GND;
-------------------------

--Célula lógica 26 - BIT13
--------------------------
A1_L(26)<=Dadooutbuffer(13);--
A2_L(26)<=NZ_L(25);--
A3_L(26)<=VCC;
A4_L(26)<=GND;
A5_L(26)<=VCC;
A6_L(26)<=GND;
B1_L(26)<=VCC;
B2_L(26)<=AZ_L(26);--
C1_L(26)<=GND;
C2_L(26)<=GND;
D1_L(26)<=OZ_L(26);--
D2_L(26)<=FZ_L(26);--
E1_L(26)<=GND;
E2_L(26)<=GND;
F1_L(26)<=NZ_L(25);--
F2_L(26)<=Dadooutbuffer(13);--
F3_L(26)<=VCC;
F4_L(26)<=GND;
F5_L(26)<=VCC;
F6_L(26)<=GND;
MP_L(26)<=GND;
MS_L(26)<=GND;
NP_L(26)<=GND;
NS_L(26)<=GND;
OP_L(26)<=GND;
OS_L(26)<=GND;
PP_L(26)<=GND;
PS_L(26)<=GND;
QC_L(26)<=GND;
QR_L(26)<=GND;
QS_L(26)<=GND;
--------------------------

--Célula lógica 27 - BIT13
--------------------------
A1_L(27)<=VCC;--
A2_L(27)<=NZ_L(26);--
A3_L(27)<=VCC;
A4_L(27)<=GND;
A5_L(27)<=VCC;
A6_L(27)<=GND;
B1_L(27)<=VCC;
B2_L(27)<=NZ_L(25);--
C1_L(27)<=GND;
C2_L(27)<=GND;
D1_L(27)<=Dadooutbuffer(13);--
D2_L(27)<=OZ_L(27);--
E1_L(27)<=GND;
E2_L(27)<=GND;
F1_L(27)<=GND;
F2_L(27)<=GND;
F3_L(27)<=GND;
F4_L(27)<=GND;
F5_L(27)<=GND;
```

```
F6_L(27)<=GND;
MP_L(27)<=GND;
MS_L(27)<=GND;
NP_L(27)<=GND;
NS_L(27)<=GND;
OP_L(27)<=GND;
OS_L(27)<=GND;
PP_L(27)<=GND;
PS_L(27)<=GND;
QC_L(27)<=GND;
QR_L(27)<=GND;
QS_L(27)<=GND;
--------------------------

-----------------------------------------------------
-- Bloco de saída 7
-----------------------------------------------------
--Célula lógica de saída 7 - BIT14 e BIT15
--------------------------
Dadooutbuffer(14) <= QZ_O(7);
Dadooutbuffer(15) <= Q2Z_O(7);
QC_O(7) <= Clock;
QR_O(7) <= Reset;
A1_O(7) <= GND;
A2_O(7) <= GND;
A3_O(7) <= GND;
A4_O(7) <= GND;
A5_O(7) <= GND;
A6_O(7) <= GND;
F1_O(7) <= GND;
F2_O(7) <= GND;
F3_O(7) <= GND;
F4_O(7) <= GND;
F5_O(7) <= GND;
F6_O(7) <= GND;
MP_O(7) <= GND;
MS_O(7) <= Load;
NP_O(7) <= GND;
NS_O(7) <= Load;
OP_O(7) <= GND;
OS_O(7) <= GND;
PP_O(7) <= GND;
PS_O(7) <= GND;
QS_O(7) <= GND;
B1_O(7) <= AZ_L(29);--
B2_O(7) <= GND;
C1_O(7) <= GND;--
C2_O(7) <= GND;
D1_O(7) <= AZ_L(31);--
D2_O(7) <= GND;
E1_O(7) <= GND;--
E2_O(7) <= GND;
--------------------------

--Célula lógica 28 - BIT14
--------------------------
A1_L(28)<=Dadooutbuffer(14);--
A2_L(28)<=NZ_L(27);--
A3_L(28)<=VCC;
A4_L(28)<=GND;
A5_L(28)<=VCC;
A6_L(28)<=GND;
B1_L(28)<=VCC;
B2_L(28)<=AZ_L(28);--
C1_L(28)<=GND;
C2_L(28)<=GND;
D1_L(28)<=OZ_L(28);--
D2_L(28)<=FZ_L(28);--
E1_L(28)<=GND;
E2_L(28)<=GND;
F1_L(28)<=NZ_L(27);--
F2_L(28)<=Dadooutbuffer(14);--
F3_L(28)<=VCC;
F4_L(28)<=GND;
F5_L(28)<=VCC;
F6_L(28)<=GND;
MP_L(28)<=GND;
```

```
MS_L(28)<=GND;
NP_L(28)<=GND;
NS_L(28)<=GND;
OP_L(28)<=GND;
OS_L(28)<=GND;
PP_L(28)<=GND;
PS_L(28)<=GND;
QC_L(28)<=GND;
QR_L(28)<=GND;
QS_L(28)<=GND;
--------------------------

--Célula lógica 29 - BIT14
--------------------------
A1_L(29)<=VCC;--
A2_L(29)<=NZ_L(28);--
A3_L(29)<=VCC;
A4_L(29)<=GND;
A5_L(29)<=VCC;
A6_L(29)<=GND;
B1_L(29)<=VCC;
B2_L(29)<=NZ_L(27);--
C1_L(29)<=GND;
C2_L(29)<=GND;
D1_L(29)<=Dadooutbuffer(14);--
D2_L(29)<=OZ_L(29);--
E1_L(29)<=GND;
E2_L(29)<=GND;
F1_L(29)<=GND;
F2_L(29)<=GND;
F3_L(29)<=GND;
F4_L(29)<=GND;
F5_L(29)<=GND;
F6_L(29)<=GND;
MP_L(29)<=GND;
MS_L(29)<=GND;
NP_L(29)<=GND;
NS_L(29)<=GND;
OP_L(29)<=GND;
OS_L(29)<=GND;
PP_L(29)<=GND;
PS_L(29)<=GND;
QC_L(29)<=GND;
QR_L(29)<=GND;
QS_L(29)<=GND;
-------------------------

--Célula lógica 30 - BIT15
--------------------------
A1_L(30)<=Dadooutbuffer(15);--
A2_L(30)<=NZ_L(29);--
A3_L(30)<=VCC;
A4_L(30)<=GND;
A5_L(30)<=VCC;
A6_L(30)<=GND;
B1_L(30)<=VCC;
B2_L(30)<=AZ_L(30);--
C1_L(30)<=GND;
C2_L(30)<=GND;
D1_L(30)<=OZ_L(30);--
D2_L(30)<=FZ_L(30);--
E1_L(30)<=GND;
E2_L(30)<=GND;
F1_L(30)<=NZ_L(29);--
F2_L(30)<=Dadooutbuffer(15);--
F3_L(30)<=VCC;
F4_L(30)<=GND;
F5_L(30)<=VCC;
F6_L(30)<=GND;
MP_L(30)<=GND;
MS_L(30)<=GND;
NP_L(30)<=GND;
NS_L(30)<=GND;
OP_L(30)<=GND;
OS_L(30)<=GND;
PP_L(30)<=GND;
PS_L(30)<=GND;
```

```
QC_L(30)<=GND;
QR_L(30)<=GND;
QS_L(30)<=GND;
--------------------------

--Célula lógica 31 - BIT15
--------------------------
A1_L(31)<=VCC;--
A2_L(31)<=NZ_L(30);--
A3_L(31)<=VCC;
A4_L(31)<=GND;
A5_L(31)<=VCC;
A6_L(31)<=GND;
B1_L(31)<=VCC;
B2_L(31)<=NZ_L(29);--
C1_L(31)<=GND;
C2_L(31)<=GND;
D1_L(31)<=Dadooutbuffer(15);--
D2_L(31)<=OZ_L(31);--
E1_L(31)<=GND;
E2_L(31)<=GND;
F1_L(31)<=GND;
F2_L(31)<=GND;
F3_L(31)<=GND;
F4_L(31)<=GND;
F5_L(31)<=GND;
F6_L(31)<=GND;
MP_L(31)<=GND;
MS_L(31)<=GND;
NP_L(31)<=GND;
NS_L(31)<=GND;
OP_L(31)<=GND;
OS_L(31)<=GND;
PP_L(31)<=GND;
PS_L(31)<=GND;
QC_L(31)<=GND;
QR_L(31)<=GND;
QS_L(31)<=GND;
--------------------------

Dadoout <= Dadooutbuffer;

end SCHEMATIC;
```

ANEXO A – Relatório de área – Metodologia clássica

```
************************************************
Device Utilization for ql6325pq208
************************************************
Resource              Used    Avail   Utilization
------------------------------------------------
IOs                   19      136     13.97%
Logic Cells           8       1536    0.51%
Flip Flops            11      3072    0.35%
ECU Cells             0       0       0.00%
RAM Cells             0       24      0.00%

------------------------------------------------

************************************************

Library: work    Cell: Counter16    View: Contador

************************************************

 Cell          Library  References     Total Area

 AND2I0        eclipse     1 x      0      0 Logic Cells
 AND2I1        eclipse     1 x      0      0 Logic Cells
 AND2I2        eclipse    26 x      0      4 Logic Cells
 AND3I0        eclipse    13 x      0      2 Logic Cells
 CKPAD_25um    eclipse     1 x      1      1 IOs
 DFFC          eclipse    16 x      1     11 Flip Flops
 INPAD_25um    eclipse     2 x      1      2 IOs
 INV           eclipse     2 x      0      0 Logic Cells
 OUTPAD_25um   eclipse    16 x      1     16 IOs
 XOR2I0        eclipse     2 x      0      1 Logic Cells

 Number of ports :                  19
 Number of nets :                   83
 Number of instances :              80
 Number of references to this view :    0

Total accumulated area :
 Number of Flip Flops :             11
 Number of IOs :                    19
 Number of Logic Cells :             8
 Number of accumulated instances :      80
```

ANEXO B – Regras do processo de *Place & Route* – Metodologia clássica

```
#[Part Name]
partname ql6325

#[Package Name]
packname PQ208

#[Path constraints from the Path Analyzer]

#[Fixed Pin Placement]
place Dadoout[0]                IO205
place Dadoout[1]                IO202
place Dadoout[2]                IO201
place Dadoout[3]                IO200
place Dadoout[4]                IO199
place Dadoout[5]                IO198
place Dadoout[6]                IO197
place Dadoout[7]                IO193
place Dadoout[8]                IO191
place Dadoout[9]                IO190
place Dadoout[10]               IO187
place Dadoout[11]               IO186
place Dadoout[12]               IO185
place Dadoout[13]               IO181
place Dadoout[14]               IO180
place Dadoout[15]               IO179
place Enable                    IO154
place Reset                     IO121
place Clock                      IO24

#[Fixed FF Placement]
#[Fixed Macro Placement]
#[Fixed Ram Placement]
#[Fixed ECU Placement]
#[Pull FF]
#[Duplication constraints]
#[False Path Constraints]
#[Multi-Cycle Path Constraints]
#[Clock Constraints]
#[Frequency Constraints]
#[Point To Point Constraints]
#[Unused pads tie-off]
UnusedIO GND

#[Pin Standards ]
#[IOBank Standards]
#[IO SlewRate]
#[module name and window size for Window based placer]
#[routing priority for net(s)]
```

ANEXO C – Relatório *Place & Route* – Metodologia clássica

Report of Counter

Design Information

Design	Counter
SpDE Version	SpDE 9.8 Release Build1
Report Generated	Tue Jul 29 13:57:01 2008
CHIP Last Updated	Tue Jul 29 13:56:41 2008
Part Type	ql6325
Speed Grade	4
Operating Range	Military
Package Type	208 PIN PQFP
PROM Loading	DISABLED
Design Check Sum	79207c

Operating Conditions

Voltage	2.50
Temperature	25.00

K-Factors

KpFactor	KvFactor	KtFactor	K-Factor
1.40	1.00	1.00	1.40

Go to Top

Utilization Information

Utilized cells (preplacement)	16 of 1536 (1.0)
Utilized cells (postplacement)	21 of 1536 (1.4)
Utilized Logic cell Frags (preplacement)	48 of 9216 (0.5)
Utilized Logic cell Frags (postplacement)	54 of 9216 (0.6)

Utilized Fragment A	14
Utilized Fragment F	7
Utilized Fragment O	8
Utilized Fragment N	9
IO control cells	0 of 16 (0.0)
Clock only cells	1 of 9 (11.1)
Bi directional cells	18 of 99 (18.2)
RAM cells	0 of 24 (0.0)
PLL cells	0 of 4 (0.0)
Flip-Flop of IO cells	0 of 316 (0.0)
1st Flip-Flop of Logic cells	7 of 1536 (0.5)
2nd Flip-Flop of Logic cells	9 of 1536 (0.6)
Routing resources	369 of 119431 (0.3)
ViaLink resources	305 of 3213992 (0.0)

Clock Network Utilization by clock pads

Clock Network	Net	Pin	Quad	Load
PLLMUX_BL2	int_Clock	IO30	Bottom Left	16

Clock Network Utilization by Internal Logic

Clock Network Utilization by PLL

Available HSCK Clock Networks

Quad TOP LEFT	5 of 5 QuadNets available (100.0)
Quad TOP RIGHT	5 of 5 QuadNets available (100.0)
Quad BOTTOM LEFT	5 of 5 QuadNets available (100.0)
Quad BOTTOM RIGHT	5 of 5 QuadNets available (100.0)

Clock Network Load Information

ClockNet	DriverCell	Quadnets	QuadNet_Details	Total_Load	Load_Details
int_Clock	IO30	N/A	N/A	16	Load_Types
					16 - LOGIC_CLOCK

Go to Top

Clock Timing Results
Summary

Clock	Frequency	Setup Time	Clock to Out
Clock	82 MHz / 12.3 ns	N/A	6.6 ns

Inter Clock Domain Delay Matrix

	Clock
Clock	12.3 ns

Go to Top

Tools run on design Counter

partdef	6.0	Run Time 0:00:00
design	9.8	Run Time 0:00:00
logic optimizer	9.8	Mode = Overnight Goal = Speed IgnorePack = TRUE UseNonBondedPads = TRUE Run Time 0:00:01
placer	9.8	Seed = 42 Mode = Overnight Run Time 0:00:05
router	9.81	Seed = 42 Run Time 0:00:00
delay modeler	9.8	Mode = Military Corner = Nominal SpeedGrade = 4 LowPower = FALSE Run Time 0:00:00
back annotation	9.8	Run Time 0:00:00
verifier	9.8	Strip = TRUE RemoveBuffersOnLoad = TRUE Run Time 0:00:00
sequencer	9.8	Run Time 0:01:27
auto buffer	9.8	Run Time 0:00:00

Go to Top

Pin Table

Pin #	Pad Name	Net Name	PinType	bankName	Fixed
1	NU (Connect to Vcc or Gnd)			PLLRST3	
2	VCCPLL			VCCPLL3	Y
3	GND				
4	GND				
5	io_Dadoout[6]	Dadoout[6]	OUTPUT	I/O(A)	N
6	io_Dadoout[7]	Dadoout[7]	OUTPUT	I/O(A)	N
7	io_Dadoout[9]	Dadoout[9]	OUTPUT	I/O(A)	N
8	VCCIO			VCCIO(A)	Y
9	io_Dadoout[8]	Dadoout[8]	OUTPUT	I/O(A)	N
10	io_Dadoout[10]	Dadoout[10]	OUTPUT	I/O(A)	N
11	NU		IOCONTROL	IOCTL(A)	
12	VCC				
13	INREF			INREF(A)	Y
14	NU		IOCONTROL	IOCTL(A)	
15	io_Dadoout[12]	Dadoout[12]	OUTPUT	I/O(A)	N

16	io_Dadoout[11]	Dadoout[11]	OUTPUT	I/O(A)	N
17	io_Dadoout[13]	Dadoout[13]	OUTPUT	I/O(A)	N
18	io_Dadoout[14]	Dadoout[14]	OUTPUT	I/O(A)	N
19	VCCIO			VCCIO(A)	Y
20	io_Dadoout[15]	Dadoout[15]	OUTPUT	I/O(A)	N
21	GND				
22	NU (GND)			I/O(A)	
23	JTAG: TDI (Connect to VCC)				
24	NU (Connect to Vcc or Gnd)				
25	NU (Connect to Vcc or Gnd)				
26	VCC				
27	NU (Connect to Vcc or Gnd)			PLLIN2	
28	NU (Connect to Vcc or Gnd)			PLLIN1	
29	VCC				
30	io_Clock	Clock	CLKPAD		N
31	NU (GND)			I/O(B)	
32	NU (GND)			I/O(B)	
33	GND				
34	VCCIO			VCCIO(B)	Y
35	NU (GND)			I/O(B)	
36	NU (GND)			I/O(B)	
37	NU (GND)			I/O(B)	
38	NU (GND)			I/O(B)	
39	NU		IOCONTROL	IOCTL(B)	
40	INREF			INREF(B)	Y
41	NU		IOCONTROL	IOCTL(B)	
42	NU (GND)			I/O(B)	
43	NU (GND)			I/O(B)	
44	VCCIO			VCCIO(B)	Y
45	NU (GND)			I/O(B)	
46	VCC				
47	NU (GND)			I/O(B)	
48	NU (GND)			I/O(B)	
49	GND				
50	JTAG: TDO (Floating)				
51	NU (Connect to Vcc or Gnd)			PLLOUT1	
52	GNDPLL			GNDPLL2	Y
53	GND				

54	VCCPLL			VCCPLL2	Y
55	NU (Connect to Vcc or Gnd)			PLLRST2	
56	VCC				
57	NU (GND)			I/O(C)	
58	GND				
59	NU (GND)			I/O(C)	
60	VCCIO			VCCIO(C)	Y
61	NU (GND)			I/O(C)	
62	NU (GND)			I/O(C)	
63	NU (GND)			I/O(C)	
64	NU (GND)			I/O(C)	
65	NU (GND)			I/O(C)	
66	NU (GND)			I/O(C)	
67	NU		IOCONTROL	IOCTL(C)	
68	INREF			INREF(C)	Y
69	NU		IOCONTROL	IOCTL(C)	
70	NU (GND)			I/O(C)	
71	NU (GND)			I/O(C)	
72	VCCIO			VCCIO(C)	Y
73	NU (GND)			I/O(C)	
74	NU (GND)			I/O(C)	
75	GND				
76	VCC				
77	NU (GND)			I/O(C)	
78	JTAG: TRSTB (Connect to GND)				
79	VCC				
80	NU (GND)			I/O(D)	
81	NU (GND)			I/O(D)	
82	NU (GND)			I/O(D)	
83	GND				
84	VCCIO			VCCIO(D)	Y
85	NU (GND)			I/O(D)	
86	VCC				
87	NU (GND)			I/O(D)	
88	NU (GND)			I/O(D)	
89	VCC				
90	NU (GND)			I/O(D)	
91	NU (GND)			I/O(D)	

92	NU		IOCONTROL	IOCTL(D)	
93	INREF			INREF(D)	Y
94	NU		IOCONTROL	IOCTL(D)	
95	NU (GND)			I/O(D)	
96	NU (GND)			I/O(D)	
97	NU (GND)			I/O(D)	
98	VCCIO			VCCIO(D)	Y
99	NU (GND)			I/O(D)	
100	NU (GND)			I/O(D)	
101	GND				
102	NU (Connect to Vcc or Gnd)			PLLOUT0	
103	GND				
104	GNDPLL			GNDPLL1	Y
105	NU (Connect to Vcc or Gnd)			PLLRST1	
106	VCCPLL			VCCPLL1	Y
107	NU (GND)			I/O(E)	
108	GND				
109	NU (GND)			I/O(E)	
110	NU (GND)			I/O(E)	
111	VCCIO			VCCIO(E)	Y
112	NU (GND)			I/O(E)	
113	VCC				
114	NU (GND)			I/O(E)	
115	NU (GND)			I/O(E)	
116	NU (GND)			I/O(E)	
117	NU		IOCONTROL	IOCTL(E)	
118	INREF			INREF(E)	Y
119	NU		IOCONTROL	IOCTL(E)	
120	NU (GND)			I/O(E)	
121	NU (GND)			I/O(E)	
122	VCCIO			VCCIO(E)	Y
123	GND				
124	NU (GND)			I/O(E)	
125	NU (GND)			I/O(E)	
126	NU (GND)			I/O(E)	
127	NU (Connect to Vcc or Gnd)			PLLIN3	
128	NU (Connect to Vcc or Gnd)				
129	VCC				

130	NU (Connect to Vcc or Gnd)				
131	VCC				
132	NU (Connect to Vcc or Gnd)				
133	JTAG: TMS (Connect to VCC)				
134	NU (GND)			I/O(F)	
135	NU (GND)			I/O(F)	
136	NU (GND)			I/O(F)	
137	GND				
138	VCCIO			VCCIO(F)	Y
139	NU (GND)			I/O(F)	
140	NU (GND)			I/O(F)	
141	NU (GND)			I/O(F)	
142	NU (GND)			I/O(F)	
143	NU (GND)			I/O(F)	
144	NU		IOCONTROL	IOCTL(F)	
145	INREF			INREF(F)	Y
146	VCC				
147	NU		IOCONTROL	IOCTL(F)	
148	NU (GND)			I/O(F)	
149	NU (GND)			I/O(F)	
150	VCCIO			VCCIO(F)	Y
151	NU (GND)			I/O(F)	
152	NU (GND)			I/O(F)	
153	GND				
154	NU (GND)			I/O(F)	
155	NU (Connect to Vcc or Gnd)			PLLOUT3	
156	GNDPLL			GNDPLL0	Y
157	GND				
158	VCCPLL			VCCPLL0	Y
159	NU (Connect to Vcc or Gnd)			PLLRST0	
160	GND				
161	NU (GND)			I/O(G)	
162	VCCIO			VCCIO(G)	Y
163	NU (GND)			I/O(G)	
164	NU (GND)			I/O(G)	
165	VCC				
166	NU (GND)			I/O(G)	
167	NU (GND)			I/O(G)	

168	NU (GND)			I/O(G)	
169	NU		IOCONTROL	IOCTL(G)	
170	INREF			INREF(G)	Y
171	NU		IOCONTROL	IOCTL(G)	
172	NU (GND)			I/O(G)	
173	NU (GND)			I/O(G)	
174	NU (GND)			I/O(G)	
175	VCC				
176	NU (GND)			I/O(G)	
177	VCCIO			VCCIO(G)	Y
178	GND				
179	NU (GND)			I/O(G)	
180	NU (GND)			I/O(G)	
181	NU (GND)			I/O(G)	
182	VCC				
183	JTAG: TCK (Connect to GND)				
184	VCC				
185	NU (GND)			I/O(H)	
186	NU (GND)			I/O(H)	
187	NU (GND)			I/O(H)	
188	GND				
189	VCCIO			VCCIO(H)	Y
190	NU (GND)			I/O(H)	
191	NU (GND)			I/O(H)	
192	NU		IOCONTROL	IOCTL(H)	
193	io_Dadoout[0]	Dadoout[0]	OUTPUT	I/O(H)	N
194	INREF			INREF(H)	Y
195	VCC				
196	NU		IOCONTROL	IOCTL(H)	
197	io_Dadoout[1]	Dadoout[1]	OUTPUT	I/O(H)	N
198	io_Dadoout[2]	Dadoout[2]	OUTPUT	I/O(H)	N
199	io_Dadoout[3]	Dadoout[3]	OUTPUT	I/O(H)	N
200	io_Dadoout[5]	Dadoout[5]	OUTPUT	I/O(H)	N
201	io_Dadoout[4]	Dadoout[4]	OUTPUT	I/O(H)	N
202	io_Enable	Enable	INPUT	I/O(H)	N
203	VCCIO			VCCIO(H)	Y
204	GND				
205	io_Reset	Reset	INPUT	I/O(H)	N

206	NU (Connect to Vcc or Gnd)			PLLOUT2	
207	GND				
208	GNDPLL			GNDPLL3	Y

Go to Top

IO Banks, Bank-I/Os info. and their VCCIO, VRef values

IO Bank	IO Standard	VCCIO	VREF	Pin #
BANK_A	DEFAULT			
				19
				8
				22
				20
				18
				17
				16
				15
				14
				11
				10
				9
				7
				6
				5
BANK_B	DEFAULT			
				44
				34
				48
				47
				45
				43
				42
				41
				39
				38
				37
				36
				35
				32

				31
BANK_C	DEFAULT			
				72
				60
				77
				74
				73
				71
				70
				69
				67
				66
				65
				64
				63
				62
				61
				59
				57
BANK_D	DEFAULT			
				98
				84
				100
				99
				97
				96
				95
				94
				92
				91
				90
				88
				87
				85
				82
				81
				80
BANK_E	DEFAULT			

				122
				111
				126
				125
				124
				121
				120
				119
				117
				116
				115
				114
				112
				110
				109
				107
BANK_F	DEFAULT			
				150
				138
				154
				152
				151
				149
				148
				147
				144
				143
				142
				141
				140
				139
				136
				135
				134
BANK_G	DEFAULT			
				177
				162
				181

				180
				179
				176
				174
				173
				172
				171
				169
				168
				167
				166
				164
				163
				161
BANK_H	DEFAULT			
				203
				189
				205
				202
				201
				200
				199
				198
				197
				196
				193
				192
				191
				190
				187
				186
				185

Each of the INREF pins should be connected to GND if no differential input is specified, otherwise connect the INREF pin to the appropriate reference voltage. Please make sure that different IO banks (VCCIO pins) are connected to the desirable voltage according to the IO configuration standard.

Go to Top

Fixed Flip Flops

None

Go to Top

Fixed RAM cells

None

Go to Top

Fixed ECU cells

None

Go to Top

Nets Removed by Technology Mapper

Net NOT_[428]
Net nx9669z1
Net nx10666z1
Net nx11663z1
Net nx12660z1
Net nx13657z1
Net nx14654z1
Net nx15651z1
Net nx16648z1
Net nx17645z1
Net nx13221z1
Net nx14218z1
Net nx15215z1
Net nx16212z1
Net nx17209z1
Net nx18206z2
Net nx18206z1
Net nx8672z2
Net nx9669z2
Net nx10666z2
Net nx11663z2
Net nx12660z2
Net nx13657z2
Net nx14654z2
Net nx15651z2
Net nx16648z2

Net nx17645z2
Net nx13221z2
Net nx14218z2
Net nx15215z2
Net nx16212z2

Go to Top

ANEXO D – Relatório de área – Metodologia proposta

```
************************************************
Device Utilization for ql6325pq208
************************************************
Resource             Used    Avail   Utilization
------------------------------------------------
IOs                  19      136     13.97%
Logic Cells          0       1536    0.00%
Flip Flops           0       3072    0.00%
ECU Cells            0       0       0.00%
RAM Cells            0       24      0.00%

------------------------------------------------

****************************************************

Library: work    Cell: Counter16    View: SCHEMATIC

****************************************************

 Cell         Library  References     Total Area

 INPAD_25um     eclipse     3 x      1      3 IOs
 OUTPAD_25um    eclipse    16 x      1     16 IOs
 SUPER_LOGIC    work       40 x      1     40 SUPER_LOGIC
 logic_0        eclipse     1 x
 logic_1        eclipse     1 x

 Number of ports :                  19
 Number of nets :                  151
 Number of instances :              61
 Number of references to this view :  0

Total accumulated area :
 Number of IOs :                    19
 Black Box SUPER_LOGIC :            40
 Number of accumulated instances :  61
```

ANEXO E – Regras do processo de *Place & Route* – Metodologia proposta

```
#[Part Name]
partname ql6325

#[Package Name]
packname PQ208

#[Path constraints from the Path Analyzer]

#[Fixed Pin Placement]
place Dadoout[0]                IO205
place Dadoout[1]                IO202
place Dadoout[2]                IO201
place Dadoout[3]                IO200
place Dadoout[4]                IO199
place Dadoout[5]                IO198
place Dadoout[6]                IO197
place Dadoout[7]                IO193
place Dadoout[8]                IO191
place Dadoout[9]                IO190
place Dadoout[10]               IO187
place Dadoout[11]               IO186
place Dadoout[12]               IO185
place Dadoout[13]               IO181
place Dadoout[14]               IO180
place Dadoout[15]               IO179
place Enable                    IO154
place Reset                     IO121
place Clock                      IO24

#[Fixed FF Placement]
#[Fixed Macro Placement]
#[Fixed Ram Placement]
#[Fixed ECU Placement]
#[Pull FF]
#[Duplication constraints]
#[False Path Constraints]
#[Multi-Cycle Path Constraints]
#[Clock Constraints]
#[Frequency Constraints]
#[Point To Point Constraints]
#[Unused pads tie-off]
UnusedIO GND

#[Pin Standards ]
#[IOBank Standards]
#[IO SlewRate]
#[module name and window size for Window based placer]
#[routing priority for net(s)]
```

ANEXO F – Relatório *Place & Route* – Metodologia proposta

Report of Counter16

- Design Information
- Operating Conditions
- Utilization Information
- Clock Network Utilization by clock pads
- Clock Network Utilization by Internal Logic
- Clock Network Utilization by PLL
- Available HSCK Clock Networks
- Clock Network Load Information
- Clock Timing Results
- Tools run on design Counter16
- Pin Table
- IO Banks, Bank-I/Os info. and their VCCIO, VRef values
- Fixed Flip Flops
- Fixed RAM cells
- Fixed ECU cells
- Nets Removed by Technology Mapper

Design Information

Design	Counter16
SpDE Version	SpDE 9.8 Release Build1
Report Generated	Tue Jul 29 13:43:40 2008
CHIP Last Updated	
Part Type	ql6325
Speed Grade	4
Operating Range	Military
Package Type	208 PIN PQFP
PROM Loading	DISABLED
Design Check Sum	17b47a

Operating Conditions

Voltage	2.50
Temperature	25.00

K-Factors

KpFactor	KvFactor	KtFactor	K-Factor
1.40	1.00	1.00	1.40

Go to Top

Utilization Information

Utilized cells (preplacement)	40 of 1536 (2.6)
Utilized cells (postplacement)	40 of 1536 (2.6)
Utilized Logic cell Frags (preplacement)	144 of 9216 (1.6)
Utilized Logic cell Frags (postplacement)	144 of 9216 (1.6)
Utilized Fragment A	32
Utilized Fragment F	16
Utilized Fragment O	40
Utilized Fragment N	40
IO control cells	0 of 16 (0.0)
Clock only cells	0 of 9 (0.0)
Bi directional cells	19 of 99 (19.2)
RAM cells	0 of 24 (0.0)
PLL cells	0 of 4 (0.0)
Flip-Flop of IO cells	0 of 316 (0.0)
1st Flip-Flop of Logic cells	8 of 1536 (0.5)
2nd Flip-Flop of Logic cells	8 of 1536 (0.5)
Routing resources	660 of 119431 (0.6)
ViaLink resources	530 of 3213992 (0.0)

Clock Network Utilization by clock pads

Clock Network Utilization by Internal Logic

Clock Network Utilization by PLL

Available HSCK Clock Networks

Quad TOP LEFT	5 of 5 QuadNets available (100.0)
Quad TOP RIGHT	5 of 5 QuadNets available (100.0)
Quad BOTTOM LEFT	5 of 5 QuadNets available (100.0)
Quad BOTTOM RIGHT	5 of 5 QuadNets available (100.0)

Clock Network Load Information

No such clock networks exist in the design

Go to Top

Clock Timing Results
Summary

Clock	Frequency	Setup Time	Clock to Out
Clock	23 MHz / 43.2 ns	40.7 ns	9.8 ns

Inter Clock Domain Delay Matrix

	Clock
Clock	43.2 ns

Go to Top

Tools run on design Counter16

partdef	6.0	Run Time 0:00:00
design	9.8	Run Time 0:00:00
logic optimizer	9.8	Mode = Overnight Goal = Speed IgnorePack = TRUE UseNonBondedPads = TRUE Run Time 0:00:00
placer	9.8	Seed = 42 Mode = Overnight Run Time 0:00:05
router	9.81	Seed = 42 Run Time 0:00:00
delay modeler	9.8	Mode = Military Corner = Nominal SpeedGrade = 4 LowPower = FALSE Run Time 0:00:01
back annotation	9.8	Run Time 0:00:01
verifier	9.8	Run Time 0:00:00
sequencer	9.8	Run Time 0:01:25
auto buffer	9.8	Run Time 0:00:00

Go to Top

Pin Table

Pin #	Pad Name	Net Name	PinType	bankName	Fixed
1	NU (Connect to Vcc or Gnd)			PLLRST3	

2	VCCPLL			VCCPLL3	Y
3	GND				
4	GND				
5	NU (GND)			I/O(A)	
6	NU (GND)			I/O(A)	
7	NU (GND)			I/O(A)	
8	VCCIO			VCCIO(A)	Y
9	NU (GND)			I/O(A)	
10	NU (GND)			I/O(A)	
11	NU		IOCONTROL	IOCTL(A)	
12	VCC				
13	INREF			INREF(A)	Y
14	NU		IOCONTROL	IOCTL(A)	
15	NU (GND)			I/O(A)	
16	NU (GND)			I/O(A)	
17	NU (GND)			I/O(A)	
18	NU (GND)			I/O(A)	
19	VCCIO			VCCIO(A)	Y
20	NU (GND)			I/O(A)	
21	GND				
22	NU (GND)			I/O(A)	
23	JTAG: TDI (Connect to VCC)				
24	NU (Connect to Vcc or Gnd)				
25	NU (Connect to Vcc or Gnd)				
26	VCC				
27	NU (Connect to Vcc or Gnd)			PLLIN2	
28	NU (Connect to Vcc or Gnd)			PLLIN1	
29	VCC				
30	NU (Connect to Vcc or Gnd)			PLLIN0	
31	NU (GND)			I/O(B)	

32	NU (GND)			I/O(B)	
33	GND				
34	VCCIO			VCCIO(B)	Y
35	io_Dadoout[15]	Dadoout[15]	OUTPUT	I/O(B)	N
36	io_Dadoout[14]	Dadoout[14]	OUTPUT	I/O(B)	N
37	io_Dadoout[13]	Dadoout[13]	OUTPUT	I/O(B)	N
38	io_Dadoout[12]	Dadoout[12]	OUTPUT	I/O(B)	N
39	NU		IOCONTROL	IOCTL(B)	
40	INREF			INREF(B)	Y
41	NU		IOCONTROL	IOCTL(B)	
42	io_Dadoout[10]	Dadoout[10]	OUTPUT	I/O(B)	N
43	io_Dadoout[11]	Dadoout[11]	OUTPUT	I/O(B)	N
44	VCCIO			VCCIO(B)	Y
45	io_Dadoout[9]	Dadoout[9]	OUTPUT	I/O(B)	N
46	VCC				
47	io_Dadoout[8]	Dadoout[8]	OUTPUT	I/O(B)	N
48	io_Dadoout[7]	Dadoout[7]	OUTPUT	I/O(B)	N
49	GND				
50	JTAG: TDO (Floating)				
51	NU (Connect to Vcc or Gnd)			PLLOUT1	
52	GNDPLL			GNDPLL2	Y
53	GND				
54	VCCPLL			VCCPLL2	Y
55	NU (Connect to Vcc or Gnd)			PLLRST2	
56	VCC				
57	io_Dadoout[6]	Dadoout[6]	OUTPUT	I/O(C)	N
58	GND				
59	NU (GND)			I/O(C)	
60	VCCIO			VCCIO(C)	Y
61	io_Reset	Reset	INPUT	I/O(C)	N
62	io_Dadoout[5]	Dadoout[5]	OUTPUT	I/O(C)	N
63	io_Dadoout[4]	Dadoout[4]	OUTPUT	I/O(C)	N

64	io_Clock	Clock	INPUT	I/O(C)	N
65	io_Dadoout[2]	Dadoout[2]	OUTPUT	I/O(C)	N
66	io_Dadoout[3]	Dadoout[3]	OUTPUT	I/O(C)	N
67	NU		IOCONTROL	IOCTL(C)	
68	INREF			INREF(C)	Y
69	NU		IOCONTROL	IOCTL(C)	
70	io_Dadoout[1]	Dadoout[1]	OUTPUT	I/O(C)	N
71	io_Dadoout[0]	Dadoout[0]	OUTPUT	I/O(C)	N
72	VCCIO			VCCIO(C)	Y
73	io_Enable	Enable	INPUT	I/O(C)	N
74	NU (GND)			I/O(C)	
75	GND				
76	VCC				
77	NU (GND)			I/O(C)	
78	JTAG: TRSTB (Connect to GND)				
79	VCC				
80	NU (GND)			I/O(D)	
81	NU (GND)			I/O(D)	
82	NU (GND)			I/O(D)	
83	GND				
84	VCCIO			VCCIO(D)	Y
85	NU (GND)			I/O(D)	
86	VCC				
87	NU (GND)			I/O(D)	
88	NU (GND)			I/O(D)	
89	VCC				
90	NU (GND)			I/O(D)	
91	NU (GND)			I/O(D)	
92	NU		IOCONTROL	IOCTL(D)	
93	INREF			INREF(D)	Y
94	NU		IOCONTROL	IOCTL(D)	
95	NU (GND)			I/O(D)	
96	NU (GND)			I/O(D)	

97	NU (GND)			I/O(D)	
98	VCCIO			VCCIO(D)	Y
99	NU (GND)			I/O(D)	
100	NU (GND)			I/O(D)	
101	GND				
102	NU (Connect to Vcc or Gnd)			PLLOUT0	
103	GND				
104	GNDPLL			GNDPLL1	Y
105	NU (Connect to Vcc or Gnd)			PLLRST1	
106	VCCPLL			VCCPLL1	Y
107	NU (GND)			I/O(E)	
108	GND				
109	NU (GND)			I/O(E)	
110	NU (GND)			I/O(E)	
111	VCCIO			VCCIO(E)	Y
112	NU (GND)			I/O(E)	
113	VCC				
114	NU (GND)			I/O(E)	
115	NU (GND)			I/O(E)	
116	NU (GND)			I/O(E)	
117	NU		IOCONTROL	IOCTL(E)	
118	INREF			INREF(E)	Y
119	NU		IOCONTROL	IOCTL(E)	
120	NU (GND)			I/O(E)	
121	NU (GND)			I/O(E)	
122	VCCIO			VCCIO(E)	Y
123	GND				
124	NU (GND)			I/O(E)	
125	NU (GND)			I/O(E)	
126	NU (GND)			I/O(E)	
127	NU (Connect to Vcc or Gnd)			PLLIN3	
128	NU (Connect to Vcc or				

	Gnd)				
129	VCC				
130	NU (Connect to Vcc or Gnd)				
131	VCC				
132	NU (Connect to Vcc or Gnd)				
133	JTAG: TMS (Connect to VCC)				
134	NU (GND)			I/O(F)	
135	NU (GND)			I/O(F)	
136	NU (GND)			I/O(F)	
137	GND				
138	VCCIO			VCCIO(F)	Y
139	NU (GND)			I/O(F)	
140	NU (GND)			I/O(F)	
141	NU (GND)			I/O(F)	
142	NU (GND)			I/O(F)	
143	NU (GND)			I/O(F)	
144	NU		IOCONTROL	IOCTL(F)	
145	INREF			INREF(F)	Y
146	VCC				
147	NU		IOCONTROL	IOCTL(F)	
148	NU (GND)			I/O(F)	
149	NU (GND)			I/O(F)	
150	VCCIO			VCCIO(F)	Y
151	NU (GND)			I/O(F)	
152	NU (GND)			I/O(F)	
153	GND				
154	NU (GND)			I/O(F)	
155	NU (Connect to Vcc or Gnd)			PLLOUT3	
156	GNDPLL			GNDPLL0	Y
157	GND				
158	VCCPLL			VCCPLL0	Y

159	NU (Connect to Vcc or Gnd)			PLLRST0	
160	GND				
161	NU (GND)			I/O(G)	
162	VCCIO			VCCIO(G)	Y
163	NU (GND)			I/O(G)	
164	NU (GND)			I/O(G)	
165	VCC				
166	NU (GND)			I/O(G)	
167	NU (GND)			I/O(G)	
168	NU (GND)			I/O(G)	
169	NU		IOCONTROL	IOCTL(G)	
170	INREF			INREF(G)	Y
171	NU		IOCONTROL	IOCTL(G)	
172	NU (GND)			I/O(G)	
173	NU (GND)			I/O(G)	
174	NU (GND)			I/O(G)	
175	VCC				
176	NU (GND)			I/O(G)	
177	VCCIO			VCCIO(G)	Y
178	GND				
179	NU (GND)			I/O(G)	
180	NU (GND)			I/O(G)	
181	NU (GND)			I/O(G)	
182	VCC				
183	JTAG: TCK (Connect to GND)				
184	VCC				
185	NU (GND)			I/O(H)	
186	NU (GND)			I/O(H)	
187	NU (GND)			I/O(H)	
188	GND				
189	VCCIO			VCCIO(H)	Y
190	NU (GND)			I/O(H)	

191	NU (GND)			I/O(H)	
192	NU		IOCONTROL	IOCTL(H)	
193	NU (GND)			I/O(H)	
194	INREF			INREF(H)	Y
195	VCC				
196	NU		IOCONTROL	IOCTL(H)	
197	NU (GND)			I/O(H)	
198	NU (GND)			I/O(H)	
199	NU (GND)			I/O(H)	
200	NU (GND)			I/O(H)	
201	NU (GND)			I/O(H)	
202	NU (GND)			I/O(H)	
203	VCCIO			VCCIO(H)	Y
204	GND				
205	NU (GND)			I/O(H)	
206	NU (Connect to Vcc or Gnd)			PLLOUT2	
207	GND				
208	GNDPLL			GNDPLL3	Y

Go to Top

IO Banks, Bank-I/Os info. and their VCCIO, VRef values

IO Bank	IO Standard	VCCIO	VREF	Pin #
BANK_A	DEFAULT			
				19
				8
				22
				20
				18
				17
				16
				15
				14

				11
				10
				9
				7
				6
				5
BANK_B	DEFAULT			
				44
				34
				48
				47
				45
				43
				42
				41
				39
				38
				37
				36
				35
				32
				31
BANK_C	DEFAULT			
				72
				60
				77
				74
				73
				71
				70
				69
				67
				66
				65

				64
				63
				62
				61
				59
				57
BANK_D	DEFAULT			
				98
				84
				100
				99
				97
				96
				95
				94
				92
				91
				90
				88
				87
				85
				82
				81
				80
BANK_E	DEFAULT			
				122
				111
				126
				125
				124
				121
				120
				119
				117

				116
				115
				114
				112
				110
				109
				107
BANK_F	DEFAULT			
				150
				138
				154
				152
				151
				149
				148
				147
				144
				143
				142
				141
				140
				139
				136
				135
				134
BANK_G	DEFAULT			
				177
				162
				181
				180
				179
				176
				174
				173

				172
				171
				169
				168
				167
				166
				164
				163
				161
BANK_H	DEFAULT			
				203
				189
				205
				202
				201
				200
				199
				198
				197
				196
				193
				192
				191
				190
				187
				186
				185

Each of the INREF pins should be connected to GND if no differential input is specified, otherwise connect the INREF pin to the appropriate reference voltage. Please make sure that different IO banks (VCCIO pins) are connected to the desirable voltage according to the IO configuration standard.

Go to Top

Fixed Flip Flops

None

Go to Top

Fixed RAM cells

None

Go to Top

Fixed ECU cells

None

Go to Top

Nets Removed by Technology Mapper

None

Go to Top

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo