PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROJETO E PROTOTIPAÇÃO DE INTERFACES E REDES INTRACHIP NÃO-SÍNCRONAS EM FPGAS

JULIAN JOSÉ HILGEMBERG PONTES

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Ney Laert Vilar Calazans

Porto Alegre

Fevereiro de 2008

Dados Internacionais de Catalogação na Publicação (CIP)

P813p Pontes, Julian José Hilgemberg
Projeto e prototipação de interfaces e redes intrachip não-síncronas em FPGAS / Julian José Hilgemberg Pontes. – Porto Alegre, 2008.
116 f.

Diss. (Mestrado) - Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Ney Laert Vilar Calazans.

1. Informática. 2. Redes de Computadores. 3. Arquitetura de Redes. I. Calazans, Ney Laert Vilar. II. Título.

CDD 004.6

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "**Projeto e Prototipação de Interfaces e Redes Intrachip Não-Síncronas em FPGAs**", apresentada por Julian José Hilgemberg Pontes, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas Embarcados e Sistemas Digitais, aprovada em 19/02/08 pela Comissão Examinadora:

Prof. Dr. Ney Laert Vilar Calazans– PPGCC/PUCRS
Orientador

Prof. Dr. Fernando Gehm Moraes– PPGCC/PUCRS

Prof. Dr. Sérgio Bampi– UFRGS

Prof. Dr. João Leonardo Fragoso– CEITEC

Homologada em ...../...../....., conforme Ata No. ..... pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

Campus Central
Av. Ipiranga, 6681 – P32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@inf.pucrs.br
www.pucrs.br/facin/pos

“Toda a sabedoria vem do Senhor Deus, ela sempre esteve com Ele. Ela existe antes de todos os séculos”.
(Livro do Eclesiástico Capítulo 1, versículo 1).

Agradecimentos

Gostaria de agradecer principalmente a Deus, a Nossa Senhora do Perpétuo Socorro, de quem sou devoto e grato por todas as graças recebidas desde que cheguei a Porto Alegre, e também a toda a minha família, que mesmo distante sempre me ofereceu suporte e acreditou na finalização desse trabalho. Agradeço aos meus pais: Luiz José Ferreira Pontes e Zaclis Terezinha Hilgemberg Pontes, por todo o apoio e empenho para educar-me. Aos meus irmãos: Andrea Hilgemberg Pontes, Emerson Luis Hilgemberg Pontes, Tais Hilgemberg Pontes e Marcos Javert Hilgemberg Pontes, que me apoiaram e me motivaram para que esse trabalho fosse concluído. Gostaria de agradecer a compreensão e carinho da minha namorada Patrícia Loren Inácio e de sua mãe Terezinha de Fátima Bueno. Gostaria de agradecer a todos os meus amigos de trabalho do GAPH, em especial ao Ewerson Carvalho, Edson Moreno, Rafael Soares, Leonel Tedesco e Matheus Trevisan por toda a ajuda e amizade durante esses dois anos. Gostaria de agradecer a CAPES pelo apoio financeiro. Agradeço ao meu orientador, Ney Laert Vilar Calazans pela orientação e pela motivação dada ao meu engajamento em trabalhos de pesquisa científica. Ao professor Fernando Gehm Moraes pelos conselhos, instruções e direcionamentos. Gostaria de agradecer aos amigos que me ajudaram desde a minha chegada em Porto Alegre: Anderson Mossi, Sebastião Romero, Marcio Farias e Laerte Strelow. Finalizando, gostaria de agradecer às pessoas da cidade de Ponta Grossa-PR que participaram direta ou indiretamente da minha formação, Mauro Silva e aos Professores da UEPG: Dr. Jorim das Virgens Filho e Dr. Márcio de Souza.

Muito Obrigado.

Resumo

Devido à evolução das tecnologias submicrônicas, hoje é possível o desenvolvimento de sistemas cada vez mais complexos dentro de um chip. Entretanto, esta evolução está inviabilizando algumas práticas de projeto tradicionais. O uso de comunicação intrachip multiponto, exemplificada por arquiteturas de barramento, e o desenvolvimento de sistemas completamente síncronos são exemplos destas práticas. Adicionalmente, a dissipação de potência está se tornando uma das principais restrições de projeto devido, por exemplo, ao aumento do uso e relevância de produtos baseados em baterias como PDAs, telefones celulares e computadores portáteis. Uma alternativa para superar estas práticas de projeto que estão perdendo viabilidade é a utilização de redes de comunicação intrachip que dêem suporte ao desenvolvimento de sistemas globalmente assíncronos e localmente síncronos (GALS). Este trabalho tem como principal alvo o desenvolvimento de suporte para o projeto utilizando o paradigma GALS em FPGAs. FPGAs foram selecionados como arquitetura alvo porque dispositivos comerciais atuais já possuem parte da infra-estrutura para dar suporte a sistemas GALS, incluindo múltiplos domínios de relógio em um único dispositivo. Também, FPGAs são dispositivos essenciais na etapa de verificação de projetos complexos que serão mais tarde sintetizados como circuitos integrados dedicados. Ao longo do trabalho, três eixos de viabilização de projeto GALS em FPGAs foram abordados, cada um gerando resultados práticos. Primeiro, foi proposta e desenvolvida uma biblioteca de macro blocos para dar suporte ao projeto de dispositivos assíncronos em FPGAs de forma compacta e eficiente. Segundo, após uma fase de comparação de interfaces assíncronas sugeridas na literatura para FPGAs e ASICs, foi proposta e validada SCAFFI, uma família de interfaces assíncronas para comunicação de módulos síncronos com relógios distintos. Terceiro, dois tipos de roteadores de redes intrachip com suporte para o projeto de sistemas GALS foram propostos e validados: Hermes GALS (Hermes-G) e Hermes GALS *Low Power* (Hermes-GLP). O roteador Hermes-GLP, além de dar suporte ao desenvolvimento de sistemas GALS, aproveita as características desse estilo de projeto para reduzir a dissipação de potência nos roteadores. Isto se dá através do emprego de mecanismos de chaveamento de freqüência internamente ao roteador. Alguns circuitos foram usados como estudos de caso para validar as duas primeiras estruturas propostas, exemplos sendo um núcleo de criptografia RSA e multiplicadores combinacionais e *pipeline*. A contribuição mais importante deste trabalho foi a geração de uma infra-estrutura básica para projeto de sistemas GALS em FPGAs.

Palavras Chave: Projeto GALS, interfaces assíncronas, redes intrachip, NoCs, FPGAs.

Abstract

The evolution of deep submicron technologies allows the development of increasingly complex Systems on a Chip. However, this evolution is rendering less viable some well-established design practices. Examples of these are the use of multipoint communication architectures (e. g. busses) and designing fully synchronous systems. In addition, power dissipation is becoming one of the main design concerns due e. g. to the increasing use of mobile products such as PDAs, mobile phones and laptop computers. An alternative to overcome the design practices becoming unviable is adopting Networks on Chip (NoCs) communication architectures supporting globally asynchronous locally synchronous (GALS) system design. This work has as main goal the development of features to support the design of GALS systems in FPGAs devices. The selection of FPGAs as target architecture occurred because several of these commercial devices already contain features supporting the design of GALS systems, such as the availability of multiple independent clock domains. Also, FPGAs are used in many scenarios as an important verification step in the design of complex integrated circuits. This works explores three development axes for enabling GALS design in FPGAs. Each one led to its own set of usable, practical results. First, there is the proposition and design of a macro block library of asynchronous devices for FPGAs. The cells of this library can be used to create compact and efficient non-synchronous modules in FPGAs. Second, after comparing a set of approaches for developing asynchronous interfaces in FPGAs and ASICS, the SCAFFI family of asynchronous interfaces was proposed. SCAFFI allows that modules operating in distinct clock domains interconnect to each other seamlessly. Third, two NoC routers supporting the GALS systems were proposed and validated: Hermes GALS (Hermes-G) and Hermes GALS Low Power (Hermes-GLP). The Hermes-GLP router, besides supporting the development of GALS systems, takes advantage of the GALS design style to reduce power dissipation in the routers. The way to achieve this is to add frequency switching mechanisms to the latter. Some circuits have been employed as case studies to validate the two first development axes, including an RSA cryptography core and combinational and pipeline multipliers. The most relevant strategic contribution of this work is the generation of a basic infrastructure for the design of GALS systems in FPGAs.

Keywords: GALS design, asynchronous interfaces, networks on chip, NoCs, FPGAs.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

BE	*Best Effort*
CHP	*Communicating Hardware Processes*
CLB	*Configurable Logic Block*
CSP	*Communicating Sequential Processes*
DFS	*Dynamic Frequency Scaling*
DI	*Delay Insensitive*
DIMS	*Delay Insensitive Minterm Synthesis*
DVS	*Dynamic Voltage Scaling*
EOP	*End of Packet*
FPGA	*Field-Programmable Gate Array*
GALS	*Globally Asynchronous Locally Synchronous*
GLP	*GALS Low Power*
GT	*Guaranteed Throughput*
HDL	*Hardware Description Language*
IP	*Intellectual Property*
ITRS	*International Technology Roadmap for Semiconductors*
LUT	*Look Up Table*
MANGO	*Message-passing Asynchronous Network-on-chip providing Guaranteed services through OCP interfaces*
Mbps	*Megabits por segundo*
MPSoC	*Multi Processor Systems on a Chip*
NoC	*Network on Chip*
OCP	*Open Core Protocol*
QDI	*Quasi Delay Insensitive*
OSI	*Open System Interconnection*
QoS	*Quality of Service*
RPM	*Relationally Placed Macro*
RSA	*Rivest, Shamir e Adleman (Algoritmo de Criptografia)*
RTL	*Register Transfer Level*
SCAFFI	*Stretchable Clock Asynchronous Flexible FPGA Interface*
SDF	*Standard Delay Format*
SoC	*System on a Chip*
TAST	*Tool for Asynchronous circuits SynThesis*

TDM	*Time-Division Multiplexing*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*

Sumário

1 INTRODUÇÃO

O desenvolvimento de SoCs até recentemente era quase que totalmente realizado partindo do pressuposto da discretização do tempo. Este pressuposto é implementado submetendo todas as entradas do sistema à temporização de um único sinal de controle, gerado externamente. Este sinal é normalmente denominado relógio, do inglês *clock*. Para que o sistema funcione corretamente, todas as entradas devem permanecer estáveis em torno do instante em que o sinal de relógio muda de estado, permitindo a amostragem de seus valores instantâneos. Desta forma, o funcionamento do sistema se dá como uma seqüência de estados definidos a cada transição do sinal de relógio. O tempo transcorrido entre transições do sinal de relógio permite que os valores de entrada amostrados sejam usados na computação de novos resultados. Este tempo garante a estabilização dos valores nas entradas e saídas de elementos de armazenamento, bem como a ocorrência e descarte de valores transitórios no interior do circuito. A estrutura de um sistema síncrono é exemplificada na Figura 1.

O pressuposto da discretização do tempo pelo uso de um sinal de relógio global no projeto de sistemas digitais é a característica principal do *estilo síncrono* de projeto. O uso maciço desse estilo de projeto não acontece por acaso. Esta simplificação torna o projeto de sistemas digitais muito mais simples. Projetos síncronos tornaram-se comuns a ponto da maioria dos projetistas de sistemas digitais não possuir conhecimento de estilos de projeto que não utilizem este pressuposto.

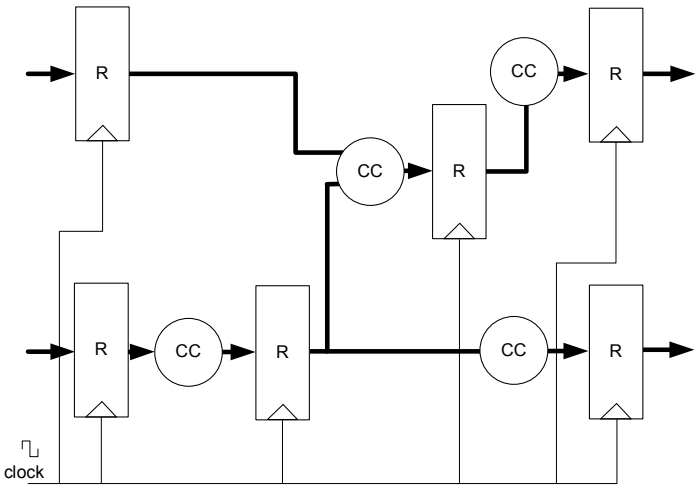

Figura 1 – Exemplo de estrutura de um circuito síncrono. Os módulos CC são blocos de lógica combinacional e os módulos R são elementos de armazenamento.

A adoção de um único relógio para controlar todo o sistema digital não traz apenas benefícios. Problemas de distribuição, escorregamento e consumo de potência do sinal relógio, que até meados da década de 80 eram desprezíveis ou facilmente tratáveis, começaram a se tornar difíceis de resolver, conforme a miniaturização dos dispositivos VLSI se acentuou. Segundo o

International Technology Roadmap for Semiconductors [ITR05], a complexidade de sistemas VLSI coloca em risco paradigmas consolidados. Entre estes se pode citar o uso do estilo síncrono, pois os circuitos resultantes tendem a se tornar menos confiáveis, devido aos limites de potência e ao custo para manter sua robustez, em relação ao esforço de projeto exigido.

O consumo de potência do sinal de relógio pode tornar sistemas síncronos complexos menos atraentes para futuros sistemas embarcados móveis. Processadores de alto desempenho apresentam um consumo dominado pelo sinal de relógio, atingindo hoje um valor médio de 45% do total da potência consumida [AMD05].

1.1 Alternativas ao Projeto Síncrono

Com o objetivo de superar as limitações do projeto síncrono, diversos grupos de pesquisa estão retomando o interesse no desenvolvimento de circuitos não-síncronos. Para facilitar a distinção entre os estilos de projeto, este trabalho utiliza o termo *não-síncrono* para englobar a classe dos circuitos totalmente assíncronos adicionada da classe dos sistemas globalmente assíncronos localmente síncronos (GALS).

Circuitos assíncronos são circuitos que assumem sinais binários mas não assumem o pressuposto de discretização do tempo, isto é, em circuitos assíncronos o tempo é tratado como uma variável contínua [SPA02]. Este tipo de circuito elimina os problemas de escorregamento e de dissipação de potência do sinal de relógio. Entretanto, o desenvolvimento de sistemas puramente assíncronos esbarra na falta de ferramentas adequadas para a automatização do processo de desenvolvimento.

Tendo em vistas as limitações do estilo síncrono de projeto, a falta de ferramentas para dar suporte a circuitos totalmente assíncronos, e a grande popularidade do estilo síncrono de projeto, alguns trabalhos de pesquisa propõe soluções intermediárias entre o projeto síncrono e o projeto assíncrono. O objetivo principal é manter as ferramentas do projeto síncrono e eliminar ou reduzir o uso de sincronização através do sinal de relógio. Entre essas propostas pode-se destacar o uso de sistemas globalmente assíncronos e localmente síncronos e o uso da *dessincronização*.

A dessincronização, do inglês *desynchronization*, é uma técnica de geração de circuitos assíncronos a partir do estilo síncrono [COR06]. A única alteração do estilo síncrono é a substituição da etapa de geração da árvore de relógio por uma etapa de inserção de circuitos de *handshake,* um para o controle de cada etapa do circuito síncrono. O período do sinal de relógio é substituído por um elemento de atraso que deve possuir atraso maior que o atraso de propagação de pior caso da lógica combinacional a qual está associado. A estrutura geral de um circuito que utiliza a estratégia de dessincronização é mostrada na Figura 2.

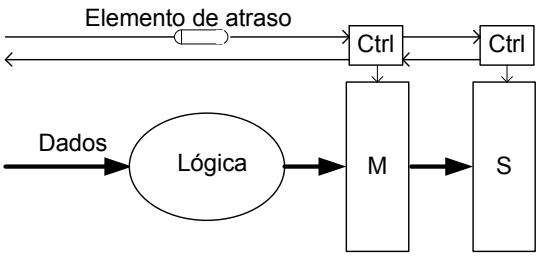

Figura 2 – Estrutura geral de um estágio de um circuito dessincronizado. Cada módulo de processamento (lógica combinacional) é sucedido por um ou mais estágios de sincronização, com elementos de armazenamento associados a módulos de sincronização cujo atraso de operação é ajustado durante o processo de projeto. Assim os resultados do módulo somente são armazenados e disponibilizados para outros estágios quando alcançam o final de sua computação.

Uma segunda alternativa que pode potencialmente preencher a lacuna entre sistemas síncronos e assíncronos é a decomposição de um sistema síncrono em diversos módulos que não trabalham globalmente sincronizados, ou seja, onde cada módulo possua um domínio de relógio distinto. Este estilo de projeto é conhecido como Globalmente Assíncrono e Localmente Síncrono (do inglês, *Globally Asynchronous, Locally Synchronous* ou GALS) [CHA84]. Em sistemas GALS, cada módulo trabalha sincronamente [TEE07] [KRS07]. Porém, a interação entre módulos utiliza uma interface de comunicação assíncrona, responsável por efetuar a transferência de informações entre os módulos [PON07].

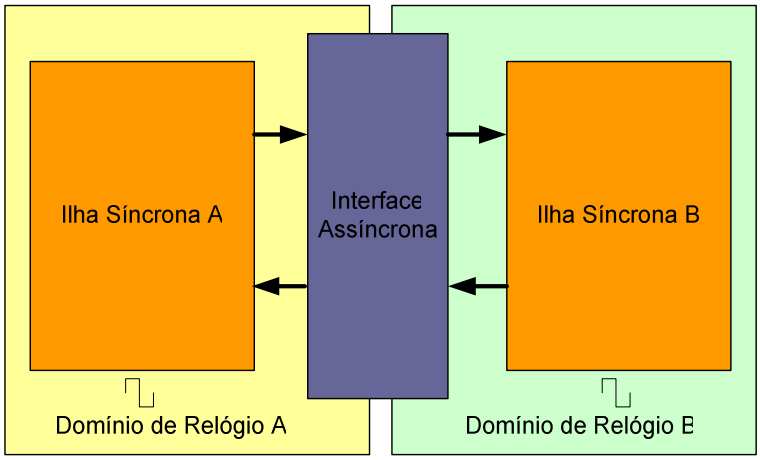

Figura 3 – Estrutura geral de um sistema GALS.

Observando as alternativas apresentadas, dessincronização e GALS, observa-se que mesmo utilizando um estilo síncrono é possível, empregando etapas adicionais de projeto, obter como resultado um circuito não-síncrono. A Tabela 1 resume a discussão acima com uma visão limitada da relação entre estilos de projeto, recursos de projeto adicionais e o tipo de circuito resultante segundo a estratégia de sincronização interna das operações deste.

A última linha da Tabela 1 é em si uma simplificação bastante grosseira da realidade de projeto de sistemas digitais. Ao assumir que um dado sistema digital não parte do pressuposto de tempo discretizado, uma grande restrição aos métodos de projeto é eliminada [CAL98]. O conseqüente aumento do número de graus de liberdade para implementar o sistema digital torna o espaço de soluções por demais amplo. Logo, costuma-se adicionar outras restrições para assim

definir um estilo de projeto. Como não é consenso quais restrições aplicar neste caso, o termo *estilo assíncrono* engloba uma família de estilos de projeto, nominalmente um para cada conjunto de restrições impostas, em substituição ao tempo discreto. Pode-se categoricamente afirmar que nenhum destes alcançou até hoje o grau de desenvolvimento do estilo síncrono de projeto, mas vários estilos assíncronos são hoje empregados em aplicações específicas. O leitor interessado pelo estilo assíncrono pode referir-se a obras como a de Sparsø e Furber [SPA02] e a de Myers [MYE01], que tratam o tema em maior profundidade.

Tabela 1 – Relação entre estilos de projeto e recursos de projeto adicionais e o tipo de circuito resultante de sua aplicação combinada.

Estilo de Projeto	Recursos Adicionais	Circuito Resultante
Síncrono	Nenhum	Síncrono
Síncrono	Dessincronização	Assíncrono
Localmente Síncrono	Interfaces de Comunicação Assíncrona	GALS
Assíncrono	Nenhum	Assíncrono

Este trabalho adota o estilo síncrono combinado com a utilização de interfaces assíncronas de comunicação para o desenvolvimento de sistemas GALS. Desta forma, os circuitos resultantes podem ser vistos como um sistema composto por um conjunto de módulos, cada qual com um domínio distinto de relógio, não existindo nenhuma relação de freqüência e/ou fase pré-estabelecida entre os módulos. Caso algum pressuposto seja assumido sobre a relação entre diferentes sinais de relógio, o problema de implementar sistemas GALS é simplificado.

1.2 Relevância da Comunicação em Sistemas Complexos

As restrições de tempo impostas para o lançamento de um produto no mercado estão tornando cada vez mais importante a adoção do reuso de núcleos de propriedade intelectual (do inglês, *Intellectual Property Cores*, *IP Cores* ou núcleos IP), dentro das equipes de projeto. Segundo [KUM02], a chave para o reuso e a integração de núcleos IP heterogêneos é a *comunicação*, desde o nível físico até o nível de sistema e conceitual. Keutzer et al. [KEU00] afirmam que uma das principais medidas que devem ser adotadas para promover o reuso é separar a computação da comunicação. Conseqüentemente, arquiteturas, plataformas e métodos centrados na comunicação têm sido amplamente investigados em pesquisas atuais.

A adoção do reuso remete a sistemas modulares, onde cada módulo possui suas próprias restrições, como a freqüência de operação, por exemplo. Desta forma, a composição de um projeto complexo por diversos módulos restringe os problemas de sincronização, complexidade de projeto e verificação da comunicação entre os módulos.

A forma mais difundida de comunicação em SoCs é através de barramentos. Porém, este meio de comunicação apresenta escalabilidade limitada, e o seu uso pode comprometer o desempenho de SoCs compostos por quantidades significativas de módulos com fluxo de dados intensivo. Em tais sistemas, o uso de redes intrachip, (do inglês, *Networks on Chip* ou NoCs) pode ser mais adequado [ZEF02].

Redes intrachip são estruturas de comunicação potencialmente capazes de atender melhor às necessidades de modularidade, escalabilidade, reuso e paralelismo na comunicação quando comparadas à barramentos. Uma rede intrachip é composta por elementos de chaveamento que realizam a transferência de mensagens entre módulos de processamento externos à rede por interconexões entre elementos de chaveamento. Diversas propostas de NoCs são encontradas na literatura [MOR04], mas apenas um pequeno conjunto destas dá suporte à comunicação de módulos operando com sinais de relógio distintos [BJE06].

Procurando atender às necessidades de futuros SoCs, algumas propostas de NoCs para sistemas GALS têm sido apresentadas [ROS05] [BEI05] [BJE07] [PON08]. Nestas propostas, SoCs são implementados como um conjunto de elementos síncronos interligados por uma NoC com estrutura de comunicação não-síncrona.

Este trabalho apresenta propostas de dois roteadores capazes de oferecer suporte ao desenvolvimento de sistemas GALS baseados em NoCs.

1.3 Importância de Interfaces Assíncronas em Sistemas GALS

A implementação de um roteador de NoC GALS pode utilizar duas abordagens diferentes. Na primeira, o roteador é um circuito totalmente assíncrono [BEI05] [BJE05] [BAI02] [ROS05] e as portas de comunicação com o módulo síncrono possuem interfaces responsáveis pela sincronização e pela conversão de comunicação assíncrona–síncrona e síncrona–assíncrona. Uma segunda abordagem consiste em implementar o roteador da NoC como um circuito síncrono [KIM05] [PON08] [BJE07] que pode ou não trabalhar sincronizado com o núcleo IP conectado a ele. Neste caso, as portas que realizam a comunicação entre roteadores e as portas responsáveis pela comunicação roteador-núcleo IP também podem possuir uma interface responsável pela sincronização e conversão dos protocolos.

As interfaces assíncronas estão presentes nas duas abordagens de implementação dos roteadores e a escolha da interface adotada pode ter um impacto significativo no desempenho do sistema como um todo.

O desenvolvimento de tais interfaces deve obedecer a um conjunto de restrições temporais intrínsecas de circuitos assíncronos [PON07]. A especificação e o desenvolvimento de circuitos que respeitem tais restrições não é uma tarefa trivial, e se torna ainda mais árdua quando o dispositivo

alvo em questão é um FPGA.

O uso de FPGAs na implementação de sistemas GALS é dificultado devido ao fato de sua estrutura ser desenvolvida para servir de base a sistemas projetados segundo o estilo síncrono. Alguns modelos de FPGAs foram propostos para dar suporte ao desenvolvimento de circuitos assíncronos [FAN05] [PAY96][TEI04], mas todos estes têm características que remetem a um estilo particular de circuito assíncrono. Recentemente, alguns trabalhos propuseram desenvolver circuitos assíncronos e GALS em FPGAs comerciais [HO02] [MOO98]. Entretanto, nenhum dos métodos propostos consegue garantir que as restrições temporais sejam realmente atendidas em todos os casos, sendo que todas as restrições devem ser verificadas após a etapa de posicionamento e roteamento do circuito.

O presente trabalho apresenta um estudo comparativo de interfaces assíncronas de comunicação intrachip para FPGAs em função de área e desempenho. Os resultados do estudo servem de suporte para a decisão de que interface utilizar para as portas das redes intrachip propostas. Além disso, este trabalho propõe um método de desenvolvimento de circuitos básicos necessários para o desenvolvimento de tais interfaces em FPGAs.

1.4 Objetivos do Trabalho

Dado o conjunto de motivações apresentado nas Seções anteriores deste Capítulo procede-se aqui ao estabelecimento de objetivos que determinam o escopo deste trabalho. Primeiro, são três os objetivos estratégicos:

Contribuir para a pesquisa em NoCs em geral. Dado que o assunto é vasto e mantido em constante evolução pela comunidade de pesquisa atuando em sistemas VLSI, considera-se relevante procurar contribuir em alguns aspectos ainda pouco explorados de redes intrachip. Um destes aspectos é a implementação de NoCs não-síncronas. Algumas propostas de NoCs não-síncronas e assíncronas são revisadas no Capítulo 3, que representa um apanhado razoável do que se pode encontrar na literatura atual.

Disponibilizar ao grupo de pesquisa do Autor, um conjunto coerente de conhecimentos sobre o desenvolvimento de sistemas GALS e circuitos assíncronos em geral, em particular sobre NoCs elaboradas usando princípios desses sistemas.

Disponibilizar métodos e ferramentas que dêem suporte ao desenvolvimento de sistemas GALS, bem como um conjunto de quantificações que possam justificar o emprego desta abordagem.

Por outro lado, divisa-se um conjunto de objetivos específicos para viabilizar e atingir os objetivos estratégicos:

O primeiro objetivo é desenvolver recursos habilitadores para o projeto de circuitos não-

síncronos em FPGAs.

O segundo objetivo é o estudo, prototipação, verificação e medição de desempenho de interfaces de comunicação assíncronas. Os resultados dessa etapa são a base para o objetivo apresentado a seguir.

O terceiro objetivo é disponibilizar um conjunto de implementações de NoCs não-síncronas, validadas, empregando o estilo GALS de projeto. A implementação das NoCs não-síncronas utiliza como ponto de partida a NoC Hermes, desenvolvida pelo grupo de pesquisa GAPH. Para permitir que estas dêem suporte a comunicação assíncrona, adiciona-se interfaces assíncronas às portas de comunicação.

1.5 Contribuições

As principais contribuições desse trabalho são:

- A disponibilização de uma biblioteca de *macros* para FPGAs que dá suporte ao desenvolvimento de circuitos assíncronos.
- A especificação, desenvolvimento, prototipação, validação e avaliação de uma interface de comunicação ponto a ponto intrachip para FPGAs [PON07].
- O desenvolvimento e validação de duas redes intrachip, Hermes GALS (Hermes-G) e Hermes GALS Low Power (Hermes-GLP), capazes de suportar o desenvolvimento de sistemas GALS [PON08]. Os roteadores da rede intrachip Hermes-GLP, além de permitir o desenvolvimento de sistemas GALS, ainda aproveitam as características de sistemas GALS para prover mecanismos de redução de potência.

1.6 Organização do Restante do Documento

O restante desse documento está organizado como descrito a seguir:

O Capítulo 2 apresenta alguns conceitos básicos relacionados a circuitos assíncronos, Seção 2.1, e redes intrachip, Seção 2.2, para facilitar a compreensão desse documento. O Capítulo 3 apresenta o levantamento do estado da arte de redes intrachip que dão suporte ao desenvolvimento de sistemas GALS. Esse Capítulo apresenta uma classificação das redes intrachip baseada no critério de implementação dos roteadores, síncronos ou assíncronos.

O desenvolvimento de uma biblioteca de circuitos assíncronos para FPGAs é apresentado no Capítulo 4. Nesse Capítulo é descrito o processo de desenvolvimento de *hard macros*, incluindo exemplos de desenvolvimento de alguns de seus componentes. Ao final do Capítulo os componentes que formam a biblioteca são listados, para dar uma idéia do estado atual da biblioteca ainda em expansão no momento da escrita desse texto.

O Capítulo 5 apresenta a implementação, prototipação e validação de interfaces assíncronas em FPGAs. Esse Capítulo compara três abordagens, cada uma com uma técnica distinta de sincronização, em função de desempenho e consumo de área. Em seguida, o Capítulo 6 apresenta a proposta de uma interface assíncrona de comunicação, desde a especificação até a validação para FPGAs. Ao final de Capítulo são mostradas a implementação e avaliação de um estudo de caso de aplicação utilizando a interface proposta.

O Capítulo 7 apresenta as principais contribuições desse trabalho, os roteadores das redes intrachip Hermes-G e Hermes-GLP. Além da descrição, são apresentados dados quantitativos de desempenho dos dois roteadores e de NoCs construídas com eles. Além disso, é apresentada uma revisão do estado da arte de redes intrachip que implementam mecanismos de redução de potência. Essa revisão foi separada do Capítulo 3 devido ao fato de o controle de potência não ser o principal objetivo desse trabalho, embora a NoC Hermes-GLP seja uma das contribuições originais desse trabalho. Ao final do Capítulo, são mostrados dados quantitativos obtidos de simulação que demonstram a capacidade de redução de potência dinâmica da rede Hermes-GLP.

O Capítulo 8 apresenta as conclusões gerais desse trabalho juntamente com rumos interessantes para a continuação do mesmo.

2 CONCEITOS BÁSICOS

Este Capítulo apresenta alguns conceitos básicos relacionados ao desenvolvimento desse trabalho. Conceitos sobre circuitos não-síncronos são abordados na Seção 2.1. A Seção 2.2 aborda conceitos relacionados a redes intra-chip.

2.1 Circuitos Não-Síncronos

Durante a evolução da eletrônica, alguns pressupostos foram criados para simplificar o desenvolvimento de circuitos. Um pressuposto fundamental da eletrônica consiste na discretização de valores de grandezas elétricas tais como a tensão. A eletrônica digital e toda a sua difusão nas tecnologias atuais decorrem da adoção desse pressuposto. Nele, dois níveis distintos de tensão são associados aos valores lógicos e números binários. As técnicas matemáticas derivadas das álgebras Booleanas podem assim ser usadas para sistematizar o tratamento de informações. Um segundo pressuposto criado foi a discretização do tempo, implementada utilizando o sinal de relógio para cadenciar as operações do circuito. O estilo de projeto síncrono assume esses dois pressupostos.

Estilos de projeto assíncronos mantêm o pressuposto de discretização dos níveis de tensão, mas não adotam nenhum pressuposto quanto ao tempo. Dessa forma, o tempo é tratado como uma variável contínua, o que torna tais circuitos mais sensíveis a fenômenos temporais que ocorrem em circuitos digitais.

A seguir serão descritos alguns conceitos sobre fenômenos temporais, protocolos de comunicação, codificação de dados extraídos na sua maioria do trabalho de Sparsø e Furber [SPA02]. Ao final dessa Seção é apresentada uma classificação dos circuitos assíncronos segundo esse Autor.

2.1.1 Fenômenos Temporais

Transitórios (em inglês *hazards*) são exemplos de fenômenos temporais que podem afetar o funcionamento de circuitos assíncronos. Em sistemas síncronos, valores transitórios podem ocorrer sem problemas, desde que no instante de amostragem todos os valores estejam estáveis. Em sistemas assíncronos, um valor transitório pode levar o circuito a um estado inválido ou indesejado. Transitórios podem ser classificados da seguinte forma [CAL98]:

- Estáticos: Quando alterações dos sinais de entrada deveriam manter algum sinal de saída estável, mas alteram este um número par de vezes antes de estabilizar.
- Dinâmicos: Quando um sinal de saída de um circuito deveria fazer uma transição,

mas ao invés disso, apresenta um número ímpar, maior que 1, de alterações.

A existência de transitórios acontece devido à ocorrência de atrasos diferenciados entre portas e fios ao longo do circuito. Entretanto, existem algumas técnicas que podem garantir o desenvolvimento de circuitos combinacionais com comportamento livre de transitórios [NOW95] [SPA02].

Outro fenômeno temporal que pode prejudicar o funcionamento de circuitos digitais é a *metaestabilidade*. Este fenômeno é mais difícil de tratar quando técnicas assíncronas de projeto são adotadas. A metaestabilidade é um fenômeno que pode ocorrer em dispositivos de armazenamento [WES94]. Quando um dado a ser registrado em um elemento de armazenamento altera o seu valor simultaneamente ou muito próximo à transição do sinal que habilita a amostragem, pode ocorrer de a saída apresentar problemas. Isto provavelmente ocorre se não forem respeitadas as restrições de tempos de *setup* e *hold* [WES94] do elementos de memória. Algumas falhas possíveis são a saída apresentar um valor indeterminado entre 0 e 1, ou haver uma demora arbitrariamente longa para que a transição de valor surja na saída. Quando este fenômeno ocorre, ele pode produzir uma falha de sincronização, isto é, o valor de saída do elemento de armazenamento pode ser interpretado como distinto do valor correto pelo estágio seguinte do circuito.

2.1.2 Interação com o Ambiente

Durante o desenvolvimento de circuitos assíncronos a imposição de restrições à interação entre o circuito e seu ambiente é uma etapa fundamental para determinar qual estilo de projeto assíncrono adotar. Quanto menor o conjunto de restrições impostas a essa interação, mais complexo é o processo de projeto. Tradicionalmente, a interação do circuito com o seu ambiente é classificada em três categorias:

- Modo Fundamental: neste modo, quando um circuito parte de um estado estável, ou seja, um estado onde todos os sinais de entrada, saída e sinais internos estão estáveis, é permitida a mudança de um único sinal de entrada. Após esta mudança, é necessário que todos os sinais alcancem novamente um estado estável antes que o ambiente possa realizar outra mudança unitária de alguma entrada do circuito.
- O modo rajada (do inglês, *burst mode*), é uma extensão do modo fundamental. Ele é uma forma restrita de permitir múltiplas alterações de entradas e múltiplas alterações de saídas. Quando um circuito operando nesse modo se encontra em um estado estável, ele aguarda por um conjunto definido de alterações na suas entradas. As alterações desse conjunto podem acontecer em ordem arbitrária. Uma vez que o conjunto de alterações na entrada é observado, esse computa um conjunto de saídas. O ambiente não pode gerar nenhuma nova alteração na sua entrada até que o circuito

se estabilize, isso é, gere o conjunto de saídas e estabilize seus sinais internos. Para funcionar de acordo com as restrições de ambiente, tanto circuitos operando no modo fundamental quanto circuitos operando no modo rajada utilizam elementos de atraso em seus sinais de realimentação. Isso ocorre por que esses sinais são vistos como novas entradas e devem obedecer ao tempo de estabilização do circuito.

- Modo Entrada e Saída: neste modo é permitida a alteração de mais de um sinal simultaneamente. O circuito opera sobre essas entradas e pode gerar ou não sinais de saída correspondentes. Assim que o ambiente observa que a saída correspondente foi definida, esse pode gerar novas entradas. Uma forma de o circuito interagir com o ambiente no modo entrada e saída é utilizando um protocolo de *handshake*, onde cada conjunto de entradas gerado pelo ambiente é assinalado com um sinal de requisição. Uma vez que estas entradas são processadas, o circuito responde com um sinal de reconhecimento, o qual é usado pelo ambiente como sinalização para geração de novos dados. O protocolo de *handshake* pode ser implementado de diversas maneiras, a Seção a seguir caracteriza este protocolo.

2.1.3 Protocolos de Comunicação

O protocolo de comunicação assíncrono mais simples e mais comum é conhecido como *handshake*. Este protocolo utiliza dois sinais, geralmente denominados de *request* e *acknowledge*, para controlar um processo de transmissão de dados ou de sincronização. A Figura 4 mostra um esquema básico de canal de comunicação utilizando *handshake*. O canal de dados é opcional, uma vez que o protocolo pode ser utilizado apenas para sincronização entre os módulos, sem troca de dados. No caso de troca de dados, a Figura 4 ilustra um canal do tipo *push channel*, isso é, um canal onde a fonte de dados é o mestre do protocolo de *handshake*. Uma forma alternativa desse protocolo é a utilização de *pull channels*, onde o destino dos dados atua como mestre.

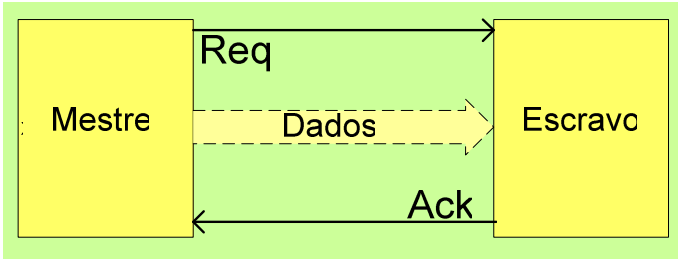

Figura 4 – Esquema básico de um canal de comunicação utilizando protocolo *handshake*. O canal de dados é opcional, Caso ele exista no sentido mostrado, o canal se denomina *push channel*.

De acordo com o número de sinalizações envolvidas no protocolo *handshake* tem-se duas versões deste: protocolo de *quatro fases* e protocolo de *duas fases*.

O protocolo de quatro fases, ilustrado na Figura 5, inicia uma transmissão com a requisição

de transmissão (*Req+*) (e eventual disponibilização do dado). Quando o receptor recebe esta requisição, ele a processa e sinaliza para o transmissor a finalização da recepção através de um reconhecimento (*Ack+*). O transmissor então retira a requisição (*Req-*) que, quando percebida pelo receptor, faz com que este retire o reconhecimento (*Ack-*).

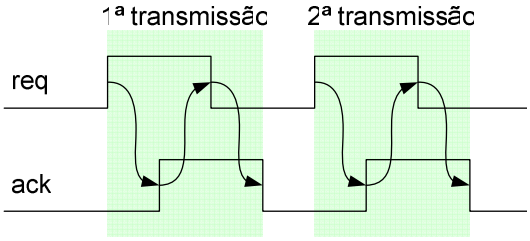

Figura 5 – Operação do protocolo de comunicação de quatro fases.

O protocolo de duas fases ilustrado na Figura 6, elimina metade das transições do protocolo de quatro fases. O transmissor sinaliza o início de uma comunicação invertendo o valor do sinal de requisição (*Req+/-*). O receptor, após processar o pedido, responde com a inversão do sinal de reconhecimento (*Ack+/-*).

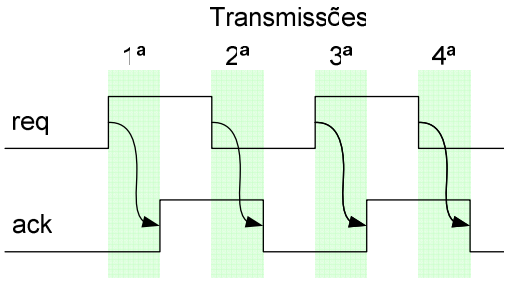

Figura 6 – Operação do protocolo de comunicação de duas fases.

Em circuitos assíncronos, cada sinalização pode ser abstraída como a movimentação de um *token* [SPA02]. Nesta abstração uma transmissão do protocolo de quatro fases é caracterizada como a movimentação de quatro *tokens (Req+, Ack+, Req-, Ack-)*, sendo que os *tokens* (*Req+)* e (*Req-)* fluem em um sentido enquanto os *tokens (Ack+)* e *(Ack-)* fluem em sentido oposto. No protocolo de duas fases cada transmissão caracteriza-se pela movimentação de dois *tokens (Req+, Ack+) ou (Req-, Ack-)*. Os *tokens* nos sinais (*Req*) e (*Ack*) fluem em sentidos opostos como no protocolo de quatro fases.

2.1.4 Codificação de Dados

A codificação de dados em um circuito digital pode ser feita de diversas formas. A codificação mais comumente empregada é conhecida como *trilha única* (em inglês *single-rail*). Em circuitos assíncronos, a utilização do protocolo de *handshake* associado a dados codificados em trilha única é chamada de *bundled data*. Nesse tipo de comunicação o sinal de requisição é responsável por sinalizar a validade dos dados transmitidos. Para o funcionamento do protocolo *bundled data* é necessário que o atraso do sinal de requisição seja maior que o atraso de todos os

sinais de dados. Dito de outra forma, a máquina de estados responsável por produzir o sinal de requisição deve ser projetada de forma que a geração do sinal de requisição só aconteça após os dados estabilizarem na entrada do receptor.

Para eliminar a restrição de temporização imposta ao sinal de requisição em um canal *bundled data*, projetistas de circuitos assíncronos exploram possibilidades de codificação capazes de carregar o dado juntamente com a sua validade. Este tipo de codificação permite o desenvolvimento de protocolos de comunicação onde não é necessária a imposição de restrições temporais na troca de dados. Esta codificação é conhecida como *insensível a atrasos* (do inglês *delay insensitive codes* ou *DI Codes*).

Em uma codificação dita DI o sinal de requisição, *Req* é parte dos dados, mas o sinal de reconhecimento (*Ack*) é mantido. Assim, é necessário um circuito capaz de verificar a *validade dos dados*, bem como sua presença ou ausência. Códigos especiais são reservados para indicar a ausência de dados, os ditos códigos *neutros*. Outro nome para códigos neutros é *espaçadores*.

Existem várias formas de codificação *DI*, mas as duas formas mais utilizadas são o código trilha dupla (em inglês *dual-rail*) e o código *m-de-N* [MAR06].

Na codificação trilha dupla, dois sinais são utilizados para cada *bit* de dados da representação binária. Uma forma de representar um *bit* de dados utilizando trilha dupla associada a um protocolo de quatro fases é mostrada na Tabela 2. Quando uma transmissão de dados DI utiliza o protocolo *handshake* de quatro fases, sempre entre dois dados válidos deve haver um espaçador. A Figura 7 mostra a troca de 1 bit de dados utilizando a codificação de trilha dupla e *handshake* de quatro fases.

Tabela 2 – Codificação trilha dupla de 1 bit de dados para transmissão utilizando protocolo *handshake* de quatro fases. O valor d.t=1 e d.f=1 é inválido e nunca deve ocorrer.

valor	d.t	d.f
neutro	0	0
0	0	1
1	1	0

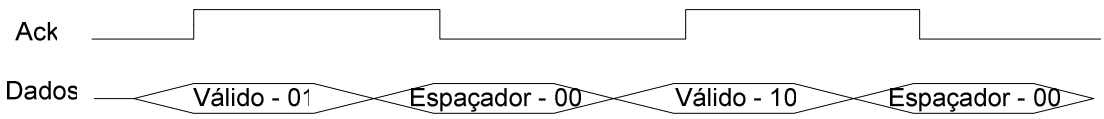

Figura 7 – Transmissão de dados utilizando codificação de trilha dupla e protocolo *handshake* de quatro fases.

A codificação de trilha dupla pode ser associada também a *handshake* de duas fases. Nesse caso não são inseridos espaçadores entre dados válidos. O funcionamento dessa abordagem requer que para mudar de um dado válido *A* para um dado válido *B* não se alcance nenhum valor intermediário que represente outro dado válido. Para alcançar essa condição, a transmissão de dados

utilizando trilha dupla de duas fases utiliza a noção de fases. A cada transmissão, a fase é alternada entre duas possíveis, par e ímpar, como mostra a Tabela 3. Cada fase indica qual a paridade do bit de trilha dupla. Nesta codificação, um fio codifica o valor dos dados, em azul na Tabela 3, enquanto o segundo codifica a paridade da trilha dupla, em vermelho na Tabela 3. A Figura 8 mostra uma transmissão de 1 *bit* de dados trilha dupla usando protocolo de duas fases.

Tabela 3 – Codificação trilha dupla de 1 bit de dado para transmissão utilizando protocolo de *handshake* de duas fases. O fio em azul codifica o valor e o fio em vermelho codifica a paridade.

		Valor	
		0	1
Fase	Par	00	11
	Ímpar	01	10

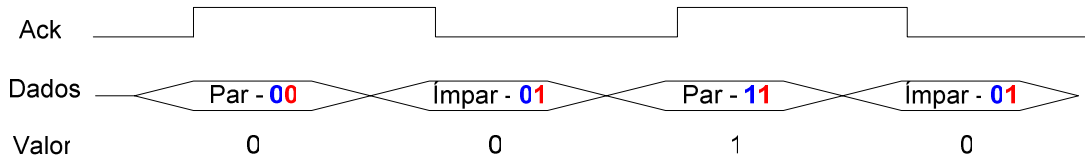

Figura 8 – Transmissão de dados utilizando codificação de trilha dupla e protocolo *handshake* de duas fases.

A codificação trilha dupla de quatro fases é uma codificação 1-de-2, um caso particular da codificação *m-de-N*. Na codificação *m-de-N* para um dado ser considerado válido *m* dos *N* fios devem estar em valor 1, enquanto que o espaçador possui todos os fios com valor 0. No caso da codificação *1-de-N,* também conhecida com *one-hot,* apenas um fio deve ter o valor igual a 1 para representar um dado válido. Dessa forma, cada sinal codifica um valor. A Tabela 4 mostra um exemplo de codificação *1-de-4*. A codificação de trilha dupla usando protocolo de quatro fases é outro caso especial de *m-de-N*, onde: $m = 1$ e $N = 4$.

Tabela 4 – Codificação 1-de-N para representar 4 valores.

Valor	d.3	d.2	d.1	d.0
Neutro	0	0	0	0
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

As codificações DI têm chamado a atenção de desenvolvedores de circuitos para aplicações de criptografia, como em *smart cards* [AND03]. O motivo é que estas codificações possuem características que aumentam a tolerância destes sistemas a ataques e apresentam consumo de

potência menos correlacionado com os dados que a codificação trilha única. Isto ocorre porque o número de transições para passar de um espaçador para um dado válido é sempre fixo.

2.1.5 Implementação de Componentes e Circuitos Assíncronos

A implementação de dispositivos e circuitos assíncronos pode ser feita tanto sobre dados codificados usando trilha única, através de circuitos combinacionais semelhantes aos utilizados em lógica síncrona, quanto sobre dados utilizando codificação DI. A seguir serão exploradas essas formas de implementação de circuitos assíncronos. Antes, contudo será apresentada a forma de implementação do C-element de Muller, um componente básico em circuitos assíncronos, usado em vários pontos neste trabalho.

2.1.5.1 O C-Element de Muller

Um componente de importância em circuitos assíncronos é o C-element de Muller [MAR06] [BER92]. O C-element funciona como um sincronizador de eventos, produzindo um evento na sua saída quando todas as suas entradas recebem eventos específicos. A Tabela 5 é uma especificação de comportamento de um C-element de 2 entradas, onde a e b são os sinais de entrada e Z_i é o sinal de saída. O C-element gera 0 na sua saída quando ambas as entradas são 0, 1 na saída quando ambas as entradas são 1 e mantém o valor anterior para qualquer outra configuração de entradas. Logo, trata-se de um circuito seqüencial que pode ser implementado de diversas formas.

Tabela 5 – Tabela verdade de um C-element de Muller com 2 entradas.

a	b	Z_i
0	0	0
0	1	Z_{i-1}
1	0	Z_{i-1}
1	1	1

Para FPGAs, a maneira mais simples de implementar um C-Element é apresentada na Figura 9. O circuito possui duas entradas e sua saída é realimentada. Assim, um C-Element de duas entradas pode ser implementado em uma LUT de 4 entradas (o que representa apenas metade das LUTs disponíveis em um *slice* nas famílias Virtex II e Spartan 3, por exemplo). C-elements são na realidade uma família de componentes onde outros tipos de C-element podem ser obtidos variando e. g. o número de entradas ou a polaridade de entradas.

O sinal de realimentação de um C-element constitui uma derivação que deve ser isócrona assimétrica, ou seja, o fio de realimentação deve possuir atraso menor que o fio de saída [BER92]. Essa restrição é necessária para garantir que o ambiente não gere um novo sinal antes da estabilização do circuito.

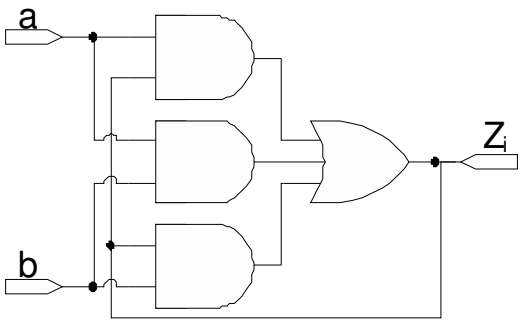

Figura 9 – Uma forma de implementação de um C-Element de Muller de 2 entradas usando portas lógicas.

A implementação de circuitos assíncronos pode optar ou não pelo uso de protocolos *handshake*. Entretanto, circuitos que não utilizam protocolo de *handshake* possuem um conjunto maior de restrições e são utilizados na maioria das vezes apenas como circuitos de controle de pequenas dimensões ou como circuitos de controle de fluxo. Tais circuitos são conhecidos como circuitos seqüenciais, e sua principal característica é que seus *tokens* fluem em apenas uma direção dentro do circuito. A definição de circuitos de controle de fluxo e alguns exemplos destes são apresentados a seguir.

2.1.5.2 Implementação de circuitos controlados por handshake

Circuitos assíncronos que realizam computação em vetores de dados, tais como circuitos aritméticos, são implementados em lógica assíncrona como *pipelines* e anéis. Nestes, os sinais do protocolo *handshake* são responsáveis por cadenciar o fluxo dos dados dentro do *pipeline* ou anel.

Tanto circuitos assíncronos que operam sobre dados usando protocolo *bundled data* quanto circuitos assíncronos que utilizam codificação DI são formados por três tipos de componentes: registradores, blocos funcionais e circuitos de controle de fluxo.

Registradores são responsáveis por armazenar valores temporários e dar suporte ao controle de *tokens* com os registradores vizinhos através de um protocolo *handshake*. Um registrador é classificado, entre outros, pela quantidade de *tokens* que pode registrar [OZD02]. Um registrador que é capaz de armazenar apenas um *token*, seja esse um dado válido ou espaçador, é chamado de *half buffer*, enquanto um registrador capaz de armazenar um dado válido e um espaçador é chamado de *full buffer* [YAH06].

Circuitos desenvolvidos utilizando codificação DI devem possuir capacidade de detectar a validade das suas entradas. Circuitos que utilizam codificação trilha dupla e protocolo de quatro fases utilizam componentes de *detecção de completude* ou *detecção de validade*, construídos com portas OR, para determinar a validade de cada bit de informação trilha dupla, e uma árvore de C-Elements. Um exemplo de componente de detecção de completude para um vetor de dados de 2 bits trilha dupla é mostrado na Figura 10, onde se mostra o circuito de um *half buffer* de dois bits. A Figura 11 mostra uma fila assíncrona composta pela conexão série de três registradores assíncronos.

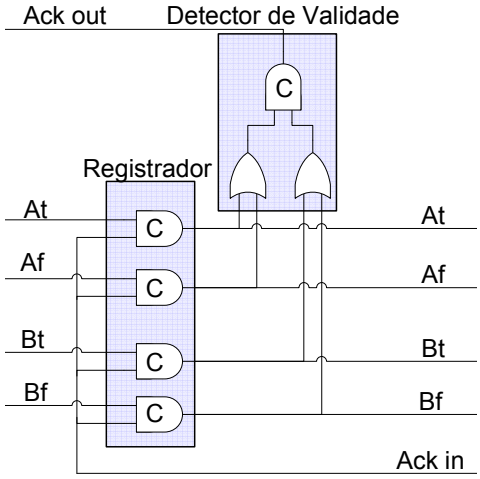

Figura 10 – Registrador de dois bits trilha dupla do tipo *half buffer* associado a detector de validade.

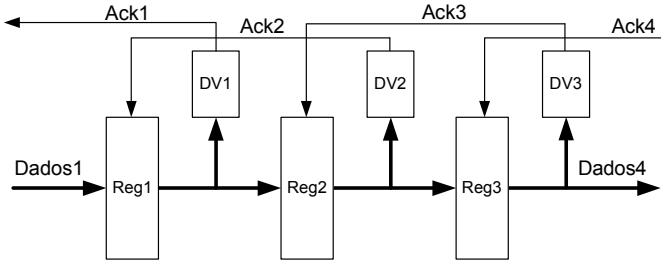

Figura 11 – Fila assíncrona para dados codificados em trilha dupla. Os blocos DVi são detectores de validade e Regi são registradores.

Entre as etapas de armazenamento podem ser inseridos blocos funcionais, conforme ilustrado na Figura 12. Esses blocos são equivalentes à lógica combinacional em circuitos síncronos. Blocos funcionais devem ser transparentes aos sinais de *handshake*. Isso quer dizer que uma seqüência de *tokens* observada na entrada de um bloco funcional é também observada na saída com um atraso. Esse atraso se deve ao tempo de propagação do bloco funcional.

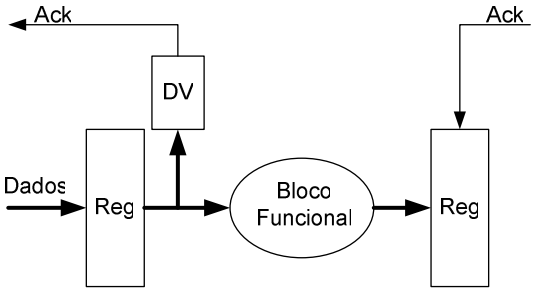

Figura 12 – Posição típica de um bloco funcional em um circuito que usa codificação de trilha dupla.

Um bloco funcional pode ser classificado de acordo com a forma como este reage às transições dos sinais de entrada. Blocos funcionais com *indicação forte* (em inglês *strongly indicating*) aguardam que todas as suas entradas estejam válidas para então iniciar a computar e gerar saídas. Da mesma forma, para gerar um espaçador na sua saída um bloco funcional com indicação forte espera até que todas as suas entradas sejam espaçadores. Blocos funcionais com *indicação fraca* (em inglês *weakly indicating*) reagem às alterações na suas entradas antes que essas estejam todas válidas ou todas sejam espaçadores. Entretanto, para que esses blocos funcionais

funcionem corretamente, somente é gerada uma saída com valor válido após todas as suas entradas apresentarem valores válidos. Tipicamente, indicação forte gera circuitos mais robustos e de mais baixo desempenho, enquanto indicação fraca tem características duais a estas.

Blocos funcionais *bundled data* são acompanhados por elementos de atraso chamados de *atrasos casados* (em inglês *matched delays*). A Figura 13 mostra um exemplo desse tipo de bloco funcional. O atraso de tais elementos deve cobrir o atraso do caminho crítico do bloco funcional. Isso garante que o bloco funcional subseqüente só irá amostrar os dados de saída depois que estes estejam estáveis. Assim, os circuitos assíncronos *bundled data*, da mesma forma que os circuitos síncronos, apresentam comportamento que acompanha o pior caso. Entretanto, os crcuitos *bundled data* eliminam a necessidade de uma árvore global de relógio. Procurando reduzir a limitação de caso médio em circuitos *bundled data*, Nowick [NOW96] apresenta uma abordagem de seleção dinâmica de elementos de atraso em tempo de execução. Essa seleção é feita ou com base nos valores das entradas do bloco funcional ou com base em um sinal interno. Essa abordagem é capaz de fazer com que um circuito usando *matched delays* apresente desempenho próximo ao de caso médio.

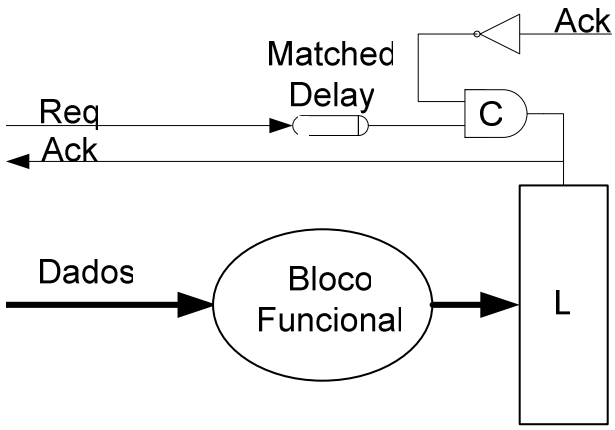

Figura 13 – Bloco funcional para codificação *bundled data* associado ao circuito de armazenamento. O bloco assinalado com *L* designa um *latch*.

Existem diversas técnicas de implementação de blocos funcionais para codificação trilha dupla [SPA02] [DAV92] [BRE05]. Um exemplo de técnica empregada para construir blocos funcionais é *Delay Insensitive Minterm Synthesis* (DIMS). Essa técnica utiliza um conjunto de C-elements para gerar todos os mintermos das variáveis de entrada. Uma porta *OR* é usada para somar os mintermos que levam a saída ao estado de *set* e outra para somar os mintermos que levam a saída ao estado de *reset*. A Figura 14 mostra um exemplo de porta XOR DIMS de dois *bits* que pode ser usada para construção de blocos funcionais.

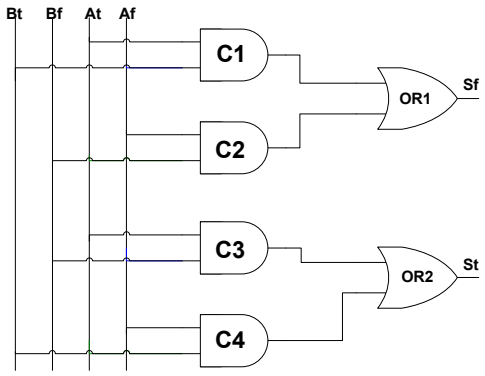

Figura 14 – Exemplo de uma porta XOR implementada com a técnica DIMS.

Além de registradores e blocos funcionais, circuitos assíncronos também são compostos por circuitos de controle responsáveis pelo controle de fluxo de *tokens* dentro do *pipeline* ou anel. Esses circuitos podem ser de controle de fluxo incondicional (*join*, *fork* e *merge*) ou condicional (*mux* e *demux*). A título de exemplo, a Figura 15 mostra o símbolo e a implementação *bundled data* do circuito de controle de fluxo *merge*. Além dos componentes citados, os árbitros e elementos de exclusão mútua também podem ser considerados como circuitos de controle de fluxo. A principal tarefa desses é *sequencializar* e controlar o acesso a recursos compartilhados.

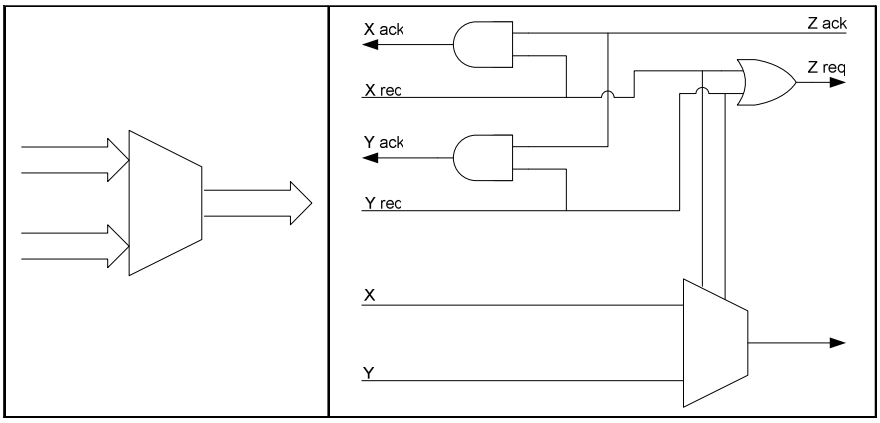

Figura 15 – Representação e implementação *bundled data* de um circuito de controle de fluxo incondicional *merge*.

2.1.6 Classificação de Circuitos Assíncronos

Estilos assíncronos de projeto possuem diversas variações e diversas classificações podem ser geradas para estes, sob diferentes critérios. A classificação mais tradicional desses estilos é em função dos pressupostos de atraso adotados. Quanto menor o número de pressupostos adotados mais robusta e restrita é a classe de circuitos.

A classe mais robusta e restrita de circuitos assíncronos é chamada de *insensível ao atrasos* (do inglês *delay insensitive*, DI). Os circuitos que se enquadram nessa classe possuem a característica de funcionar corretamente independente dos atrasos de fios e das portas lógicas. Entretanto, essa classe de circuitos é muito restrita. A adição de um pressuposto de temporização

referente ao atraso dos fios gera uma classe mais abrangente de circuitos assíncronos. Essa classe é chamada de quase insensível ao atraso, do inglês *quasi delay insensitive* (QDI). O pressuposto de temporização adicionado consiste na restrição dos atrasos de sinais que geram derivações, do inglês *forks*. Essas derivações devem ser isócronas, isso é, a diferença do tempo de propagação para que uma transição chegue a todas as extremidades da derivação deve ser limitada, de forma que essa diferença possa ser considerada nula.

Os circuitos *independentes de velocidade* (do inglês *speed independent*, SI) formam outra classe de circuitos assíncronos. Para que esses circuitos funcionem corretamente adiciona-se um pressuposto que impõe que atrasos de fio possam ser considerados nulos. Esses circuitos mantêm a necessidade das derivações isócronas, uma vez que se atrasos de fios são considerados nulos, a diferença de atraso entre as derivações também deve ser nula. O pressuposto de atraso nulo nos fios vai contra o a tendência apresentada na evolução de tecnologias VLSI, que apresentam um aumento relativo de atrasos de fios significativo quando comparados a atrasos de portas. Entretanto, esse tipo de circuito é muito aplicado para circuitos de controle. Eles podem ter tamanho reduzido e são associados a restrições de posicionamento que permitem a construção de áreas equipotenciais, onde os fios possuem comprimento reduzido e podem ser considerados nulos.

A quarta classe de circuitos é conhecida como *circuitos de Huffman*. Enquanto as classes de circuitos descritos anteriormente funcionam no modo entrada e saída, os circuitos pertencentes à classe dos circuitos de Huffman funcionam sob o modo fundamental. O modo fundamental restringe as possibilidades de geração de entradas do circuito. Para que os circuitos pertencentes a essa classe funcionem de forma correta, é permitida a alteração de um único sinal de entrada a cada instante. Alterada uma entrada o ambiente deve esperar que o circuito estabilize-se, somente então podendo ser gerada outra variação de entrada. O pressuposto de variação de um sinal a cada instante vale para os sinais de realimentação que armazenam o estado do sistema. Essa classe de circuitos é utilizada em circuitos de controle.

Outra classe de circuitos assíncronos é a classe dos circuitos *auto temporizados* (do inglês *self timed,* SI). Esses circuitos geralmente possuem elementos de atraso associados, chamados de *matched delays*, conforme ilustrado na Figura 13. Esses elementos devem possuir atrasos superiores ao atraso da lógica combinacional a qual estão associadas.

2.2 Redes Intra-Chip

Uma rede intrachip é uma arquitetura de comunicação normalmente composta de canais de comunicação ponto a ponto que interligam elementos de transporte de mensagens, os roteadores. Redes intrachip podem dar suporte ao requisito de escalabilidade necessário ao desenvolvimento de sistemas complexos em único chip. Além da escalabilidade, uma estrutura baseada em ligações

ponto a ponto entre roteadores permite que as redes intrachip se adaptem facilmente ao paradigma GALS, outra tendência no desenvolvimento de SoCs. Esta Seção introduz alguns dos princípios básicos subjacentes a redes intrachip. A Figura 16 mostra a estrutura genérica de um SoC composto por uma rede intrachip direta, onde cada roteador possui exatamente um núcleo IP associado.

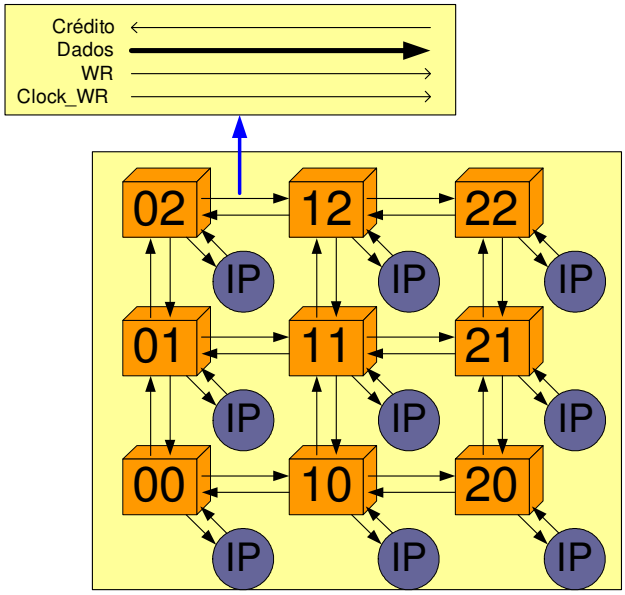

Figura 16 – Exemplo de um SoC composto por uma NoC e núcleos IP, onde cada roteador possui um núcleo IP associado. No detalhe apresenta-se a interface física de cada canal unidirecional entre dois roteadores.

A Figura 17 mostra a estrutura de um roteador composto por cinco portas (Norte, Sul, Leste, Oeste, Local) para comunicação entre roteadores e uma porta para a comunicação entre roteador e núcleo IP (Local). Neste modelo, cada porta de entrada do roteador possui uma fila associada.

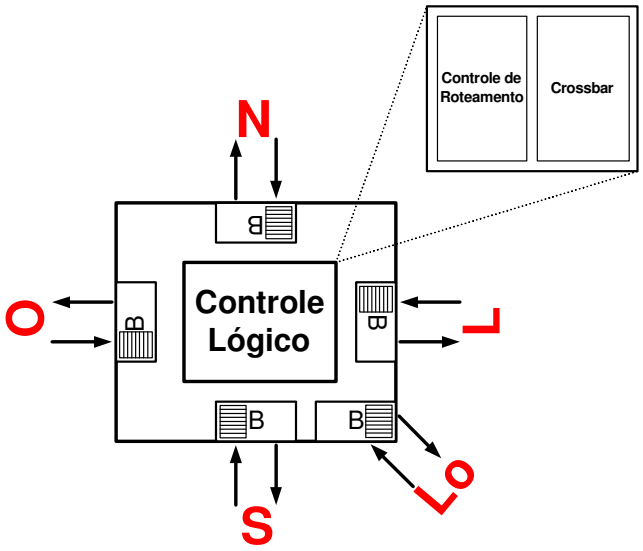

Figura 17 – Estrutura interna de um roteador de NoC de cinco portas utilizando a infra-estrutura Hermes.

2.2.1 Topologia

A topologia de uma rede intrachip é a forma como os roteadores estão conectados entre si.

Uma rede intrachip pode apresentar uma topologia regular ou irregular. Segundo Scherer [SCH07], a topologia de uma rede intrachip pode ser definida como um conjunto de nodos *N*, onde cada elemento de *N* está conectado a um conjunto de canais *C*. Um canal pode ser formalmente definido como o par ordenado como mostrado na Equação 1:

$$c=(x,y)\in C \quad (1)$$

Onde o nodo origem e o nodo destino pertencem a N. A Figura 18 apresenta algumas topologias regulares já propostas para uso em redes intrachip.

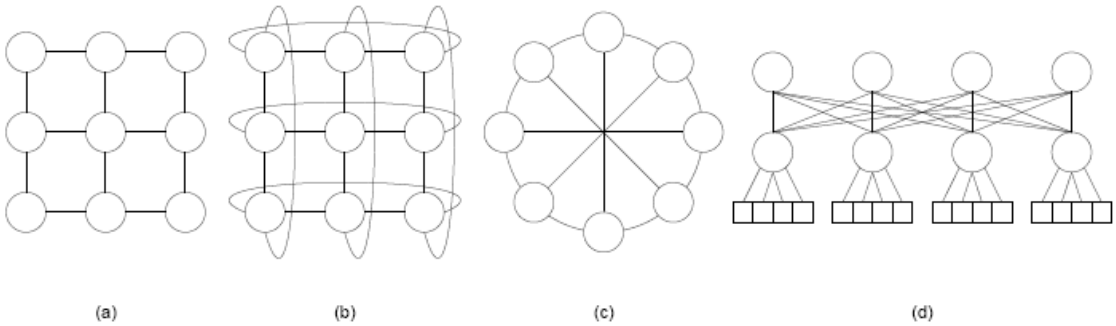

Figura 18 - Algumas topologias de redes intrachip [SCH07] (a) malha 2D; (b) toro 2D; (c) anel cordal; (d) árvore gorda.

2.2.2 Algoritmo de Roteamento

O algoritmo de roteamento define o caminho que será tomado por um pacote entre transitando entre uma fonte e um destino, ou seja, define a seqüência de canais e roteadores a serem percorridos pelo pacote entre os nodos fonte e destino [DAL04]. Em função do local onde é definido o roteamento pode ser classificado em duas categorias:

- Roteamento na origem: o remetente conhece a topologia da rede, determina qual caminho o pacote deverá seguir, e insere este no cabeçalho do pacote.
- Roteamento distribuído: cada roteador que recebe o pacote define, em função do endereço do destino, qual o próximo roteador para onde este pacote deve ser enviado.

2.2.3 Modos de Chaveamento

O modo de chaveamento de uma rede intrachip define a forma como os recursos da rede são alocados para a transmissão de um pacote [DAL04] [DUA02]. Uma primeira classificação dos modos de chaveamento distingue uma transmissão por comutação de circuito de uma transmissão por comutação de pacotes. A comutação de circuito é sempre orientada a conexão, isto é, o caminho dos dados é alocado antes de iniciar a transmissão e permanece alocado enquanto durar a conexão, havendo ou não transmissão de dados através deste caminho. Na comutação de pacotes, um enlace

entre dois roteadores é ocupado somente no instante da transmissão de um pacote entre estes roteadores, permitindo assim um melhor compartilhamento dos recursos da rede.

Redes intrachip que optam pelo modo de comutação de pacotes necessitam de armazenamento temporário (em inglês *buffers*), uma vez que é possível que nem todos os enlaces estejam disponíveis durante a transmissão do pacote. A alocação dos recursos de armazenamento temporário pode ser realizada de três maneiras (chamados modos de chaveamento na comutação de pacotes):

- *Store and forward*: o pacote é recebido e armazenado por completo no buffer do roteador antes de ser repassado para outro roteador. Esta abordagem necessita obrigatoriamente de *buffers* de armazenamento com capacidade pelo menos igual ao maior tamanho de pacote. O fato de o pacote ser armazenado antes de iniciar a transmissão para outro roteador pode inserir latências importantes na comunicação.
- *Virtual cut-through*: um roteador pode enviar um pacote adiante assim que o roteador seguinte der uma garantia que o pacote poderá ser aceito completamente [RIJ01].
- *Wormhole*: nesta abordagem, o pacote é dividido em pequenas unidades, denominadas *flits*. Aqui, a transmissão do pacote não requer o armazenamento completo deste em um roteador. Isto diminui a latência na transmissão uma vez que cada roteador pode ser visto por um *flit* como um estágio de um *pipeline*. Além da latência, o tamanho dos *buffers* pode ser reduzido, uma vez que não existe a necessidade de armazenar um pacote inteiro. Ao mesmo tempo, desacopla-se nesta estratégia o tamanho máximo de um pacote dos requisitos de armazenamento nos roteadores da rede. Como desvantagem deste método, a rede fica mais propensa a congestionamento, pois cada pacote circulando nesta pode alocar simultaneamente *buffers* de vários roteadores e múltiplos canais durante sua transmissão.

2.2.4 Canais Virtuais e Garantia de Serviço

Os roteadores de uma rede intrachip são interligados através de canais. Quando um pacote chega a um roteador pode ficar bloqueado quando o seu canal de destino está alocado para outro pacote. Quando isto ocorre em uma rede intrachip que utiliza o modo de chaveamento *wormhole*, o pacote pode estar distribuído por vários roteadores. Quando os *buffers* dos roteadores que hospedam este pacote enchem, este pacote, além dos *buffers*, também ocupará os canais intermediários que foram alocados para este pacote. Desta forma, a ocorrência do bloqueio de um pacote pode acarretar o bloqueio de diversos outros pacotes dentro da rede. Uma das formas de atenuar este efeito é através da utilização de canais virtuais. A adoção de canais virtuais permite que o canal físico seja

compartilhado por diversos canais virtuais. Cada canal virtual possui o seu próprio *buffer* e na transmissão deve se determinar qual dos canais virtuais deve ser utilizado para a alocação do pacote [MEL05]. Esta idéia é ilustrada na Figura 19.

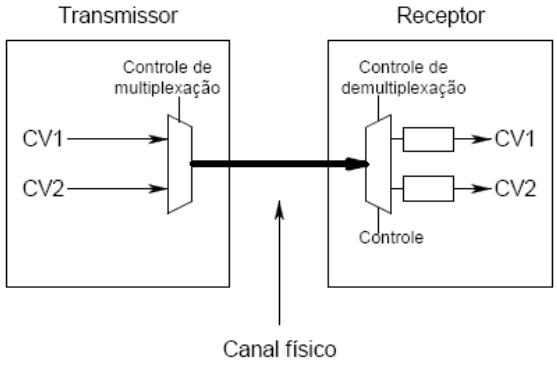

Figura 19 – Ilustração de implementação de 2 canais virtuais compartilhando 1 canal físico. Nota-se que os canais virtuais baseiam-se em *buffers* de entrada compartilhando um enlace físico e componentes de multiplexação e demultiplexação, além de uma lógica de controle associada [SCH07].

Quando uma rede intrachip não é capaz de prover garantias de serviço diz-se que esta oferece um serviço de transmissão de pacotes do tipo melhor esforço (do inglês, *best effort*). Este tipo de serviço não oferece nenhuma garantia temporal na transmissão do pacote.

Pode-se projetar redes que ofereçam alguma garantia de serviço (do inglês, *guaranteed services*) [BJE06]. Para alcançar garantias de serviço, existem dois métodos fundamentais: o uso de comutação por circuitos, ou a associação de níveis de prioridade aos pacotes que trafegam na rede. A associação de prioridade a canais virtuais, juntamente com mecanismos de escalonamento pode oferecer serviços garantidos, tais como vazão garantida ou latência máxima.

3 TRABALHOS RELACIONADOS

Existem diversos conceitos e técnicas que influenciam no processo de desenvolvimento de uma rede intrachip, alguns dos quais foram mostrados na Seção 2.2. Entre as considerações que devem ser feitas antes do desenvolvimento de uma NoC pode-se citar como importantes: a escolha da topologia, do algoritmo de roteamento, do formato do pacote, do tipo de chaveamento dos pacotes e do controle de fluxo. Outra decisão importante no projeto de redes intrachip é a escolha do estilo de projeto que será adotado para os módulos componentes desta: síncrono, assíncrono, GALS ou dessincronizante.

Um esquema simplificado da classificação de redes intrachip e de SoCs é mostrado na Tabela 6. Esta classificação parte do pressuposto de sistemas GALS, qual seja, que os núcleos IP são síncronos, e que a eles se acopla um roteador. A classificação considera o estilo de projeto adotado para implementar o roteador, o tipo de interface empregado entre roteadores e entre um roteador e um núcleo IP e deriva destas relações o tipo de NoC e o tipo de SoC do ponto de vista de sincronização de operação.

Uma NoC pode ser um sistema totalmente assíncrono, onde todos os roteadores são implementados como circuitos assíncronos e a comunicação com os núcleos IP síncronos é dotada de interfaces de comunicação responsáveis pela sincronização dos dados. Esse modelo equivale à última linha da Tabela 6. Note-se que ainda assim o SoC gerado é GALS, pois os núcleos IP sozinhos constituem um conjunto de ilhas síncronas e a NoC representa o conjunto de interfaces entre as ilhas síncronas assim constituídas.

Alternativamente uma NoC pode ser totalmente síncrona. Neste caso, existem ainda duas possíveis abordagens de sincronismo:

- A rede intrachip e os núcleos IP funcionam com o mesmo sinal de relógio, o que resulta em um SoC totalmente síncrono, primeira linha da Tabela 6. A maioria das primeiras propostas de NoC e.g. [KUM02] [RIJ01] assumiam este tipo de implementação.
- A rede intrachip possui um sinal de relógio que não possui necessariamente relação de freqüência ou fase com o sinal de relógio dos núcleos IP segunda linha da Tabela 6. A comunicação entre roteadores e núcleos IP é dotada de uma interface assíncrona de comunicação. Esta abordagem é adotada no trabalho de [KIM05] e apresenta um problema de escalabilidade. Tal problema deriva do fato que roteadores devem estar espalhados no chip ao lado de cada núcleo IP. Então, ainda é necessária uma árvore de distribuição de relógio para os roteadores, que no pior caso será tão extensa como

aquela usada em sistemas totalmente síncronos.

Outra abordagem é o desenvolvimento de roteadores síncronos, mas sem estabelecer uma relação de fase ou freqüência entre os roteadores. O uso dessa abordagem deriva mais duas possibilidades de implementação de SoCs:

- O roteador e o módulo IP conectado a ele formam um par síncrono, mas a comunicação entre roteadores é assíncrona, terceira linha da Tabela 6. Neste caso, devem existir interfaces assíncronas de comunicação entre os roteadores. Esta abordagem possui o inconveniente de aumentar a dependência entre a comunicação e a computação, indo contra a tendência moderna apontada e.g. por Keutzer et al. [KEU00] de separar estas preocupações durante o projeto de sistemas complexos, visando promover o reuso maciço de IPs.
- Cada roteador possui sinal de relógio independente, quarta linha da Tabela 6. A comunicação entre roteadores e a comunicação entre roteador e módulo IP são assíncronas. Neste modelo de rede intrachip, devem existir interfaces entre roteadores e entre roteador e núcleo IP. Esta é a forma mais flexível de sistema GALS com o mínimo emprego de estilos assíncronos de projeto e a adotada neste trabalho.

Tabela 6 – Classificação das NoCs e SoC baseados nestas, com base no tipo de projeto de roteador e na forma da comunicação entre roteadores e entre roteador e núcleo IP.

Roteador	Comunicação Roteador – Roteador	Comunicação Roteador – Núcleo IP	NoC	SoC
Síncrono	Síncrona	Síncrona	Síncrona	Síncrono
Síncrono	Síncrona	Assíncrona	Síncrona	GALS
Síncrono	Assíncrona	Síncrona	GALS	GALS
Síncrono	Assíncrona	Assíncrona	GALS	GALS
Assíncrono	Assíncrona	Assíncrona	Assíncrona	GALS

O restante deste Capítulo descreve algumas propostas de implementação de NoCs encontradas na literatura que dão suporte ao estilo GALS de projeto, bem como as alternativas adotadas para o desenvolvimento destas NoCs.

3.1 Redes com Roteadores Assíncronos

3.1.1 Proposta de Bainbridge e Furber [BAI02]

Bainbridge e Furber propõem uma NoC composta de roteadores totalmente assíncronos. O

enlace entre roteadores e entre roteador e núcleo IP é totalmente assíncrono e a etapa de sincronização dos dados provenientes do roteador no núcleo IP deve ser tratada no desenvolvimento do módulo IP. Os dados são representados utilizando codificação 1-de-4 para transportar dados e um bit sinaliza o fim do pacote. Utilizando os bits de fim de pacote os Autores afirmam que pacotes de tamanho variável e arbitrário podem trafegar pela rede. O roteamento dos pacotes através da rede é de responsabilidade do transmissor, caracterizando o uso de roteamento na origem. Desta forma, o transmissor deve conhecer a topologia da rede. O uso de roteamento especificado na origem elimina o módulo de controle para realizar roteamento e desta forma, simplifica o projeto do roteador em NoCs totalmente assíncronas, sendo uma das justificativas para o uso desse tipo de roteamento.

3.1.2 Proposta de Beigné et al. [BEI05] [BEI06]

Estes trabalhos propõem uma NoC dotada de interfaces capazes de dar suporte à comunicação interna ao chip, bem como à comunicação com circuitos externos ao chip. O roteador da NoC proposta é implementado de forma totalmente assíncrona. O roteador possui dois canais virtuais, visando prover garantia de serviço. O enlace responsável por prover comunicação entre a NoC e o mundo externo pode ser configurado para trabalhar de forma síncrona ou assíncrona. A comunicação entre roteadores utiliza a codificação insensível a atrasos 1-de-4, enquanto que a comunicação entre o roteador e o módulo IP é realizada através de uma fila bi-síncrona proposta em [CUM02].

3.1.3 Proposta de Bjerregaard e Sparsø [BJE05]

Bjerregaard e Sparsø apresentam a rede intrachip MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services through OCP interfaces*). A NoC MANGO possui canais virtuais que proporcionam serviços garantidos (em inglês, *Guaranteed Services* ou *GS*), orientados à conexão e roteamento sem conexão com algoritmo de melhor esforço (em inglês, *Best-effort* ou *BE*). A arquitetura da rede MANGO consiste de adaptadores de rede para dar suporte a transações OCP, roteadores e enlaces.

O roteador da NoC MANGO é implementado através de circuitos assíncronos. Este utiliza internamente um protocolo de quatro fases *bundled data*. Os enlaces entre roteadores utilizam codificação trilha dupla e protocolo de duas fases. A conexão entre o roteador assíncrono e o módulo IP síncrono é feita através de um adaptador de rede, o qual é responsável por sincronizar os sinais provenientes do roteador e fornecer os serviços pelo padrão OCP 2.0.

3.1.4 Proposta de Quartana et al. [QUA05]

Quartana et al. apresentam uma NoC cuja implementação tem como tecnologia alvo FPGAs. A proposta inclui o projeto de um roteador totalmente assíncrono. O desenvolvimento dos módulos assíncronos que compõem a NoC utilizou a ferramenta TAST [DIN02], que gera circuitos QDI a partir de uma descrição em CHP, uma linguagem baseada em CSP. A interface assíncrona-síncrona foi desenvolvida utilizando sincronizadores em série. Este surge como o principal inconveniente desta abordagem, pois a sincronização através de *latches* em série aumenta a latência significativamente, prejudicando o desempenho da NoC

3.1.5 Proposta de Rostislav et al. [ROS05]

Na proposta de Rostislav et al. é apresentada a estrutura de uma versão totalmente assíncrona da QNoC, que os Autores alegam poder garantir qualidade de serviço. A rede QNoC é baseada em uma arquitetura de roteador que suporta múltiplos níveis de serviço. Também é proposto nesta rede o uso de roteamento preemptivo, de acordo com a prioridade dos pacotes.

Os pacotes são particionados em flits, que são enviados através da rede utilizando o modo de chaveamento *wormhole*. O roteamento é especificado na origem e incluído nos pacotes. Desta forma, o transmissor define o caminho que o pacote deve percorrer até o destino, o que simplifica o desenvolvimento do roteador.

O controlador assíncrono utilizado no roteador foi desenvolvido utilizando a ferramenta Petrify [COR00], voltada para a geração automatizada de circuitos independentes de velocidade.

3.2 Redes com Roteadores Síncronos

3.2.1 Proposta de Zipf et al. [ZIP04]

Zipf et al. propõem uma rede intrachip com roteadores síncronos dotados de interfaces assíncronas e mecanismos de sincronização baseados em relógio pausável. Os serviços oferecidos pelo roteador são: sincronização de sinais de entrada, armazenamento temporário, envio de dados e conexão de unidades funcionais na rede.

O uso do mecanismo de relógio pausável garante a eliminação da possibilidade de metaestabilidade, se corretamente empregado. A idéia do trabalho envolve utilizar relógios pausáveis com geradores de relógio em anel nos canais de comunicação, sendo um gerador por roteador. No entanto, este mecanismo apresenta uma desvantagem importante quando utilizado para implementar roteadores em NoCs. Quando ocorre uma solicitação de transmissão por um canal no roteador a suspensão do sinal de relógio congela todo o roteador, se este for implementado como

um circuito síncrono. Isto elimina a principal vantagem de uma rede intrachip: o paralelismo na comunicação, que possibilita que um roteador trate simultaneamente várias comunicações entre portas de entrada e saída.

3.2.2 Proposta de Kim et al. [KIM05]

O roteador proposto por Kim et al. utiliza filas bi-síncronas como interface entre os roteadores. A estrutura utilizada na fila é a mesma proposta em [CUM02], porém neste trabalho os Autores propõem um mecanismo adicional para o controle de estouro da fila.

O roteador proposto por Kim et al. utiliza uma fila bi-síncrona em cada porta de entrada e de saída. O roteador possui quatro canais de comunicação bidirecionais, em cada canal são inseridas duas filas assíncronas, uma para a porta de entrada e outra para a porta de saída.

3.2.3 Proposta de Wang et al. [WAN06]

Neste trabalho, Wang et al. apresentam a versão assíncrona da NoC Proteo para FPGAs. O roteador proposto é implementado de maneira híbrida, ou seja, possui módulos síncronos e módulos assíncronos. A referência não especifica qual o mecanismo de sincronização adotado nas interfaces assíncrona-síncrona.

A NoC possui adaptadores de rede que a tornam capaz de se comunicar com núcleos IP desenvolvidos em conformidade com o padrão de interfaces VCI e OCP.

A transmissão de dados entre roteadores utiliza a codificação trilha dupla de quatro fases e a rede é prototipada em FPGAs comerciais.

3.2.4 Proposta de Panades et al. [PAN06]

A rede intrachip DSPIN é composta por duas sub-redes, uma para requisições e outra para respostas, para evitar *deadlock* na comunicação. A topologia da rede intrachip é organizada em uma malha bidimensional. Cada *flit* possui 34 bits, 32 para dados e 2 para controle (*begin of packet* BOP e *end of packet* EOP). A DSPIN utiliza a modo de chaveamento *wormhole*, roteamento *X-first* para os pacotes de requisição e *Y-first* para os pacotes de resposta (garantindo que o pacote de resposta utiliza o mesmo caminho do pacote de requisição).

A interface assíncrona utilizada é chamada de fila bi-síncrona e permite a escrita e a leitura utilizando freqüências distintas. A NoC consiste em um sistema mesócrono, isto é, um sistema onde todos os roteadores possuem o sinal de relógio com mesma freqüência, mas com relações de fase distintas. Desta forma a rede intrachip utiliza o mesmo relógio, mas a distribuição deste não necessita de uma árvore de relógio balanceada. Segundo os Autores, isto é suficiente para reduzir o

consumo de potência da rede intrachip.

A freqüência de operação dos núcleos IP pode ser distinta e totalmente independente da freqüência de operação do roteador. Para sincronização também são inseridas filas bi-síncronas na interface entre roteador e a interface de rede (entre IP e o roteador). Essa fila possibilita a transmissão de dados com relógios distintos, mas acrescenta uma latência maior se comparada à fila operando na interface mesócrona. Segundo os Autores, a fila apresenta latência de 1 a 2 ciclos de relógio entre módulos mesócronos e 2 a 3 ciclos entre relógios totalmente distintos.

A rede intrachip dá suporte a dois tipos de tráfego: melhor esforço e serviços garantidos, através do uso de dois canais virtuais, associados a um mecanismo de multiplexação por divisão do tempo (em inglês, *time division multiplexing*, ou TDM).

3.2.5 Proposta de Bjerregaard et al. [BJE07]

Os Autores apresentam a arquitetura de uma rede intrachip mesócrona. O sistema apresenta comportamento globalmente síncrono uma vez que todos os roteadores e núcleos IP utilizam o mesmo sinal de relógio. Entretanto não existe sincronização entre os roteadores, uma vez que o escorregamento de relógio entre os roteadores não é controlado. Isso garante um sistema modular, uma vez que a sincronização é requerida apenas dentro da ilha síncrona que engloba roteador e núcleo IP. Com essa abordagem são evitados os *buffers* do sinal de relógio utilizados para controlar o escorregamento de relógio. Isso colabora para a redução de potência uma vez que tais *buffers* posuem consumo elevado.

A topologia empregada na rede intrachip é a de árvore. Segundo os Autores essa topologia reduz a dissipação de potência que as redes com topologia malha. Na rede intrachip proposta cada roteador pode se comunicar com um nodo pai e dois nodos filhos (árvore binária). Assim cada roteador folha possui dois módulos IP conectados.

A comunicação usa um protocolo de *handshake* de quatro fases. O controle do protocolo é realizado utilizando as duas bordas do sinal de relógio, o que segundo os Autores permite a transmissão de um dado a cada dois ciclos de relógio.

A distribuição do sinal de relógio segue o sentido da árvore. Assim um nodo pai recebe o sinal de relógio antes dos seus dois filhos. Desta forma o escorregamento nos enlaces é determinado e elementos de atraso são inseridos nos sinais de dados conforme o escorregamento do enlace para que os tempos de *hold* e *setup* não sejam violados. Os Autores mostram o processo de determinação do valor desses atrasos para a tecnologia de 90nm. O principal problema dessa abordagem é a dependência a tecnologia, ao *place and route*, do SoC e a variações de temperatura, tensão e processo. Isso implica em um esforço de projeto para todo desenvolvimento de SoC que opte por utilizar esse sistema de comunicação para determinar os valores de atraso e inserir elementos de

atraso equivalentes nos sinais de dados.

O funcionamento do sistema foi validado através da construção de um MPSoC composto de 32 núcleos, onde cada núcleo compreende um processador e uma memória local. O sistema foi desenvolvido utilizando a tecnologia de 90nm.

3.3 Comparação das abordagens

A Tabela 7 apresenta um resumo das características das abordagens apresentadas. A Tabela apresenta a estratégia de implementação do roteador, a codificação utilizada para o enlace entre roteadores e o método de sincronização adotado pelas abordagens. Além das redes intrachip revisadas nesse Capítulo a Tabela 7 apresenta também as NoCs Hermes-G e Hermes-GLP [PON08], que são contribuições desse trabalho a serem descritas no Capítulo 7. Aqui, optou-se pela utilização de roteadores síncronos, podendo assim manter a utilização das ferramentas tradicionais de CAD, e filas bi-síncronas, devido a maior vazão de dados permitida por essa interface.

Tabela 7 – Comparação entre as abordagens. Quando a NoC possui um nome, este é usado na primeira coluna. Caso contrário, usa-se o nome dos autores da referência.

Abordagem	Roteador	Codificação (R←→ R)	Interface entre Roteadores
CHAIN [BAI02]	Assíncrono	1-de-4 mais bit de fim de pacote	Assíncrona
ANOC [BEI05] [BEI06]	Assíncrono	1-de-4	Assíncrona
MANGO [BJE05]	Assíncrono	Trilha dupla	Assíncrona
Quartana et al. [QUA05]	Assíncrono	Trilha dupla	Assíncrona - Sincronizadores
Rostislav et al. [ROS05]	Assíncrono	Trilha dupla	Assíncrona
Zipf et al. [ZIP04]	Síncrono	Bundled data	Assíncrona - Relógio pausável
Kim et al. [KIM05]	Síncrono	Binária	Síncrona
Wang et al. [WAN06]	Híbrido	Trilha dupla	Assíncrona
DSPIN [PAN06]	Síncrono	Binária	Fila Bi-Síncrona
Bjerregaard et al. [BJE07]	Síncrono	Bundled Data	Fila Bi-Síncrona
Hermes-G e Hermes-GLP [PON08]	Síncrono	Binária	Fila Bi-Síncrona

4 BIBLIOTECA DE MACROS - SUPORTE AO DESENVOLVIMENTO DE SISTEMAS NÃO-SÍNCRONOS EM FPGAS

Existem algumas funcionalidades necessárias à implementação de muitos circuitos não síncronos que são trivialmente implementadas ou irrelevantes ao se adotar o estilo síncrono de projeto. Exemplos são a arbitragem de acesso a recursos e a sincronização de dois fluxos de eventos.

Quando se abre mão do projeto síncrono, deve-se criar componentes especiais que implementem tais funcionalidades e/ou que apóiem a implementação das mesmas. Dispositivos do tipo FPGA comerciais não possuem tais componentes, mas alguns deles dispõem de recursos de projeto que podem ser usados para implementar esses componentes de forma eficiente. Essa Seção apresenta o desenvolvimento, projeto e validação de uma biblioteca de componentes não síncronos para uso em FPGAs da Xilinx [XIL06].

Componentes não-síncronos possuem restrições temporais associadas as suas funcionalidades. A natureza destas restrições está relacionada ao estilo de projeto em uso. Desta forma, embora seja possível desenvolver componentes não-síncronos utilizando linguagens de descrição de hardware e restrições temporais, como proposto em [HO02] [MOO98], a implementação destes pode ser realizada com vantagens caso se utilize estratégias de projeto de mais baixo nível de abstração.

Entre as vantagens de usar um nível de abstração mais baixo para o projeto de dispositivos assíncronos, pode-se citar que em FPGAs é possível obter implementações com alto grau de eficiência em temporização e área de componentes não-síncronos [LAN02]. Além disto, é possível controlar melhor os atrasos de componentes e de fios [PON07].

O sistema ISE de desenvolvimento de projeto para FPGAs da Xilinx habilita o projeto de componentes possuindo restrições temporais ou de posicionamento através do uso de *macros*. Esse trabalho aproveita essa funcionalidade das ferramentas da Xilinx para projetar componentes não-síncronos.

Uma *macro* é um módulo implementado com componentes primitivos do FPGA, tais como *look-up tables* (LUTs), *flip-flops*, e *latches*. Macros podem ser de três tipos: *soft*, com posicionamento relativo definido (em inglês, *Relationally Placed*, ou *RPM*) ou *hard*.

Uma *soft* macro consiste na instanciação de componentes fornecidos pela Xilinx. Este tipo de macro permite a manipulação dos recursos de hardware para o desenvolvimento de circuitos que possuem restrições de área e desempenho. Entretanto, não é possível controlar a distribuição de fios

através desse tipo de macro, o que dificulta o desenvolvimento de circuitos assíncronos, uma vez que algumas restrições temporais importantes em circuitos assíncronos são relacionadas aos atrasos de fio. A Figura 20 mostra um código VHDL que descreve um C-Element. A implementação utiliza uma LUT de quatro entradas. A LUT é o elemento básico em FPGAs da Xilinx, e tais elementos são disponibilizados pela Xilinx na biblioteca UNISIM. Além da instanciação da *soft* macro, o código VHDL possui também uma restrição temporal *MAXDELAY* associada ao sinal de realimentação, para indicar às ferramentas de *place and route* que se deve tentar garantir que o atraso deste sinal seja limitado, nesse caso, menor que o tempo de propagação de uma LUT. Essa restrição é aplicada ao C-element devido ao estilo independente de velocidade de implementação adotado. Desta forma os fios devem possuir atraso que possa ser considerado nulo.

```
entity C_element is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
              reset: in std_logic;
           z : out  STD_LOGIC);
end C_element;
architecture Behavioral of C_element is
signal s0: std_logic;
attribute maxdelay: string;
attribute maxdelay of s0: signal is "500ps";
begin
    z <= s0;
    LUT4_inst : LUT4
   generic map (
      INIT => X"00E8")
   port map (
      O => s0,
      I0 => s0,
      I1 => a,
      I2 => b,
      I3 => reset
   );
end Behavioral;
```

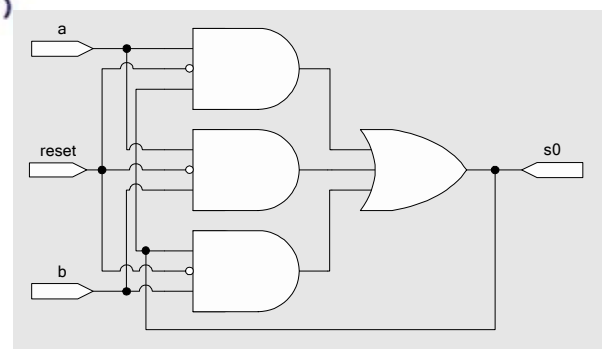

Figura 20 – *Soft macro* de um C-element implementado usando FPGAs da Xilinx. A determinação do valor de configuração da LUT (INIT => X"00E8") é feita através dos valores obtidos da tabela verdade do C-element, adicionado uma lógica de *reset* que leva a saída do C-element para 0 quando o sinal de reset = 1.

Macros do tipo RPM são *soft* macros com restrições de posicionamento relativo. Assim é possível determinar não só a funcionalidade, mas também a conformação do circuito no FPGA. Este tipo de macro também não possibilita controlar a distribuição dos fios no interior do circuito. O uso de RPMs, aliado a restrições temporais, para o desenvolvimento de circuitos assíncronos é proposto

por [MOO98]. Entretanto o uso de tais macros não garante a satisfação de todas as restrições temporais. A Figura 21 mostra o código de uma porta OR DIMS utilizando *soft macros* e restrições temporais para controlar o escorregamento nas derivações da porta.

```
entity OR_DIMS is
  Port ( Af : in STD_LOGIC;
         At : in STD_LOGIC;
         Bf : in STD_LOGIC;
         Bt : in STD_LOGIC;
         Sf : out STD_LOGIC;
        reset: in std_logic;
         St : out STD_LOGIC);
end OR_DIMS;
architecture Behavioral of OR_DIMS is
component c_element is
port(
        a: in std_logic;
        b: in std_logic;
        reset: in std_logic;
        z: out std_logic
);
end component;
signal s0t, s1t, s2t: std_logic;
attribute maxskew: string;
attribute maxskew of Af: signal is "500ps";
attribute maxskew of At: signal is "500ps";
attribute maxskew of Bf: signal is "500ps";
attribute maxskew of Bt: signal is "500ps";
begin
        CE1: c_element
        port map(a => Af,     b => Bf,        reset => reset, z => Sf);
        CE2: c_element
        port map(a => At,     b => Bt,        reset => reset, z => s0t);
        CE3: c_element
        port map(a => Af,     b => Bt,        reset => reset, z => s1t);
        CE4: c_element
        port map(a => At,     b => Bf,        reset => reset, z => s2t);
        St <= s0t or s1t or s2t;
end Behavioral;
```

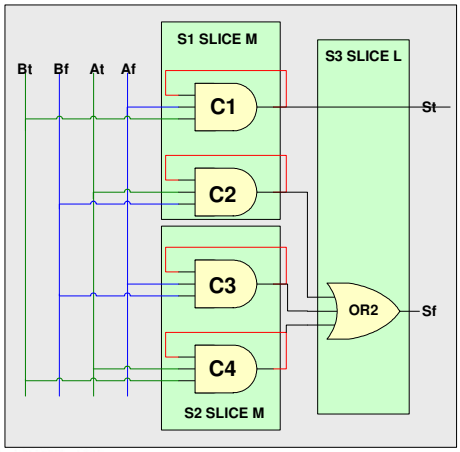

Figura 21 – Código de uma *soft macro* para uma porta OR DIMS. No código VHDL são especificadas as restrições temporais das derivações isócronas. Cada C-Element consiste em uma instância do componente da Figura 20.

O processo de criação de uma RPM pode ser feito utilizando a ferramenta de planejamento de planta baixa da Xilinx, o FloorPlanner. A Figura 22 mostra as restrições de posicionamento

relativo, RLOC, que posicionam as LUTs que compõem a porta OR DIMS relativamente dentro da matriz de elementos do FPGA, transformando a *soft* macro em uma RPM. Essa porta foi prototipada em um FPGA XC2V1000.

```
INST "St1" RLOC = "X1Y1" ;
INST "CE4/LUT4_inst" RLOC = "X1Y0" ;
INST "CE3/LUT4_inst" RLOC = "X1Y0" ;
INST "CE2/LUT4_inst" RLOC = "X0Y0" ;
INST "CE1/LUT4_inst" RLOC = "X0Y0" ;
```

Figura 22 – Restrições de posicionamento para transformar a porta OR DIMS da Figura 21 em RPM.

A Tabela 8 confronta as restrições de atraso máximo (MAXDELAY) especificadas na Figura 20 e as restrições de escorregamento nas derivações da porta OR DIMS (MAXSKEW) da Figura 21 com os resultados obtidos na síntese. Neste exemplo, os sinais de entrada da porta provêm dos pinos de entrada e saída do FPGA. Neste exemplo todas as restrições foram alcançadas, mas em circuitos mais complexos a ferramenta de roteamento nem sempre consegue satisfazer todas as restrições. As *soft macros* e as RPMs permitem que os circuitos compostos por elas possam ser verificados através de simulação funcional. Esses tipos de macros possuem maior independência de tecnologia do que as *hard* macros que serão apresentadas a seguir.

Tabela 8 – Comparação entre as restrições impostas e os valores obtidos após a síntese para um FPGA XC2V1000 da família Virtex II da Xilinx. Usou-se a ferramenta XST da Xilinx para realizar a síntese.

Restrição	Folga	Atraso
MAXDELAY = 0.5 ns	0.026ns	0.474ns
MAXDELAY = 0.5 ns	0.182ns	0.318ns
MAXDELAY = 0.5 ns	0.251ns	0.249ns
MAXDELAY = 0.5 ns	0.320ns	0.180ns
MAXSKEW = 0.5 ns	0.295ns	0.205ns
MAXSKEW = 0.5 ns	0.471ns	0.029ns
MAXSKEW = 0.5 ns	0.500ns	0.000ns
MAXSKEW = 0.5 ns	0.500ns	0.000ns

Macros do tipo *hard* são geradas a partir de um editor gráfico de *layout*, no caso da Xilinx a ferramenta FPGAEditor. Tais macros podem ser colocadas em uma de várias posições possíveis na planta baixa do projeto de um dado FPGA pelas ferramentas de síntese ou manualmente pelo projetista. *Hard macros* podem ser instanciadas em código VHDL como componentes de projeto quaisquer.

A implementação de *hard macros* em FPGAs possibilita que os atrasos de fios e de elementos lógicos sejam verificados durante a construção dos componentes. A implementação de

circuitos usando *hard* macros garante ainda que todas as instâncias de um módulo apresentem atrasos e comportamento previsíveis, o que é mais difícil de controlar utilizando restrições temporais sobre código HDL, *soft macros* e até mesmo nas *RPMs*.

A falta de flexibilidade de *hard* macros é um ponto negativo do seu uso em relação aos outros tipos de macros. Uma *hard* macro é desenvolvida exclusivamente para um FPGA específico (um dispositivo específico de uma dada família), e não pode ser reaproveitada para outros FPGAs. Devido a esta limitação, imposta ao uso de *hard macros*, este trabalho propõe uma biblioteca de dispositivos para facilitar o desenvolvimento de circuitos assíncronos eficientes e que respeitem os requisitos de desempenho de aplicações com facilidade.

4.1 O Processo de Projeto de *Hard Macros*

O processo de projeto de uma *hard macro* é similar a um trabalho de edição de esquemáticos sobre um editor gráfico. Os passos de criação de uma *hard macro* são:

Ao entrar na ferramenta FPGAEditor, escolhe-se criar uma nova *hard* macro para um dispositivo de uma dada família. A ferramenta mostra então o *layout* do FPGA escolhido.

Escolhe-se aleatoriamente uma região do FPGA e um bloco lógico configurável (em inglês, *configurable logic block* ou CLB), que vai servir de base para a implementação da *hard* macro. Isto pode ser feito porque o FPGA é constituído por uma grande matriz bidimensional quase totalmente regular de CLBs e outros blocos (blocos de memória, multiplicadores, pinos configuráveis de E/S, etc).

Cada CLB é formado por um conjunto de fatias (em inglês, *slices*) contendo várias tabelas-verdade configuráveis (em inglês, *look-up tables* ou LUTs), elementos de memórias configuráveis como *latches* ou *flip-flops* e várias conexões configuráveis entre estes elementos. O projeto se faz especificando a interconexão entre elementos de um CLB; escolhem-se dois ou mais pinos de elementos e solicita-se a criação de uma conexão entre os mesmos.

Os elementos configuráveis tais como LUTs, devem ter sua função especificada via equações Booleanas.

Depois de estabelecida a interconexão dos componentes e a função das LUTs define-se a interface externa da *hard macro*, especificando nome para pinos de entrada e saída da mesma. Os nomes dados serão mais tarde usados como nomes de portas de uma entidade VHDL usada em mais alto nível ao instanciar a macro.

Ao final do processo de definição da *hard macro*, deve-se escolher um *slice* de referência da macro, uma espécie de ponto de aplicação da macro, usado como referência para operações de posicionamento físico (manual ou automático) da macro.

Depois de concluída a macro, se salva essa como um arquivo binário. Este arquivo pode ser

então adicionado a projetos em linguagem HDL (VHDL ou Verilog).

O desenvolvimento de circuitos assíncronos através de *hard macros* necessita utilizar algum grau de tentativa e erro. Durante cada passo da implementação do circuito é feita a verificação das restrições temporais. Diferentes conformações são testadas e destas se escolhe a que melhor atende as restrições.

4.2 Exemplos de Projeto de *Hard Macros*

4.2.1 Portas DIMS

A biblioteca proposta dispõe de portas para construção de blocos funcionais trilha dupla. Essas portas possuem diversos C-elements na sua estrutura e cada C-Element possui um conjunto de restrições. Um exemplo desse tipo de *hard* macro é uma porta XOR DIMS apresentada na Figura 23.

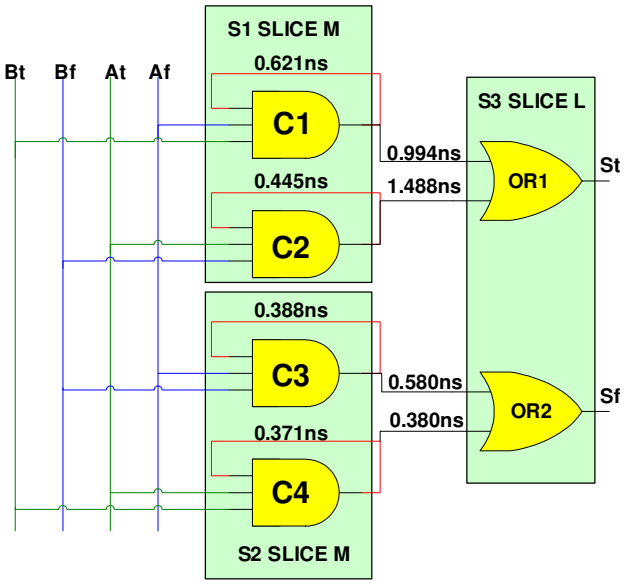

Figura 23 – Porta XOR implementada usando a técnica DIMS em *hard macro*.

Portas DIMS com *n* entradas possuem 2^n C-elements com *n* entradas cada. Todos os C-Elements devem possuir derivações isócronas assimétricas em suas saídas, isto é, o sinal de saída deve retornar ao C-element antes de chegar à porta OR. Essa restrição se deve ao estilo de implementação do C-Element, que é um circuito independente de velocidade, e não à porta DIMS que é QDI. A Figura 23 apresenta a temporização encontrada no desenvolvimento da macro desta porta para o FPGA XC3S200 da família Spartan 3. Esta *hard* macro ocupa seis das oito LUTs (4 *slices*) de um CLB e seu desenvolvimento requer várias tentativas na busca de uma conformação de fios que obedeça todas as restrições.

4.2.2 Elemento de Exclusão Mútua

Um elemento de exclusão mútua (ME) é um circuito que possui tipicamente duas entradas e duas saídas. Cada entrada corresponde a uma requisição de acesso a um recurso. Cada entrada está associada a uma saída. A função principal de um ME é seqüencializar os acessos a recursos controlados por suas saídas. Note que se ambas saídas controlam o acesso a um mesmo recurso, este acesso é concedido ora a uma ou a outra das entradas (requisições) do ME, nunca a ambas ao mesmo tempo. A Figura 24 ilustra a interface de entrada e saída de um ME.

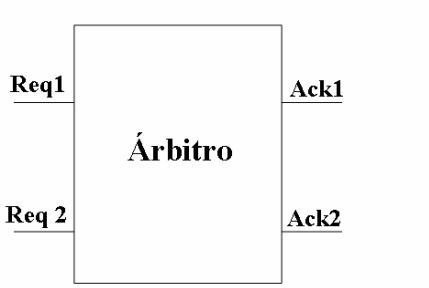

Figura 24 – Estrutura genérica de um árbitro com descrição do seu comportamento.

MEs são componentes de uso geral em sistemas não-síncronos. Em circuitos assíncronos toda acesso concorrente a um determinado recurso deve ser seqüenciado através de um ME para evitar corridas. Em sistemas GALS é usado, por exemplo, em geradores de relógio pausável para controlar se e quando o sinal de saída de relógio oscila. Uma implementação comum de um ME é através de um *latch* RS. Porém a utilização de apenas um *latch* como ME é propensa a metaestabilidade. Implementações em ASICs utilizam um filtro associado às saídas do *latch* para evitar esse fenômeno. A estrutura de um ME implementado através de um *latch* associado a um filtro de saída é mostrado na Figura 25. Os filtros garantem que enquanto não se resolver a metaestabilidade na saída do *latch* os sinais de saída serão mantidos em nível baixo. A biblioteca de macros conta com um ME composto por um *latch* RS sem filtro. Esse tipo de ME é utilizado no desenvolvimento de um sistema GALS em FPGA no trabalho de [NAJ05].

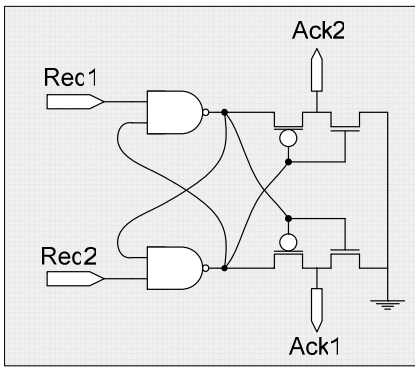

Figura 25 – Estruturas internas para árbitros empregados em projeto de ASICs [MOO98].

Implementações em FPGA, porém não contam com recursos como filtros. O

desenvolvimento de ME em FPGAs é o maior desafio para os projetos GALS que usam extensão do relógio. Implementações mais adequadas de ME para FPGAs são propostas em [MOO98] e [WAN06]. As implementações deste trabalho utilizaram o ME proposto por Moore e Robinson [MOO98] devido ao menor consumo de área e da maior robustez que este apresenta em relação ao proposto por Wang et al. [WAN06]. O circuito do ME é apresentado na Figura 26.

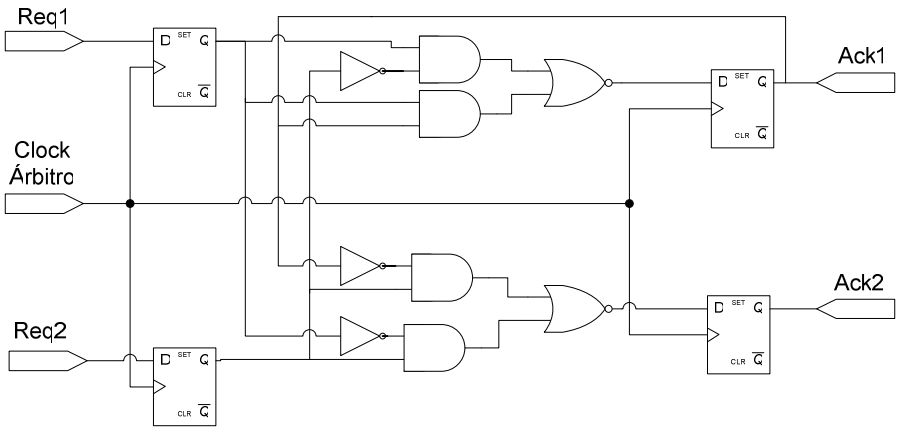

Figura 26 – Circuito do árbitro proposto por Moore e Robinson.

A análise estatística do tempo entre falhas do ME, utilizando os valores característicos fornecidos pela Xilinx, pelos Autores aponta para um tempo entre falhas de 240 anos. Esta abordagem é muito mais adequada para FPGAs que a abordagem utilizando um *latch* RS ao custo de maior área devido ao gerador de relógio introduzido para cada ME adicionado. Este ME foi implementado através de *hard* macros e ocupa dois *Slices*, um para os registradores de entrada e outro para a lógica combinacional e para os registradores de saída. A Figura 27 mostra a estrutura do *Slice* contendo dois *flip-flops* de entrada do ME proposto por Moore.

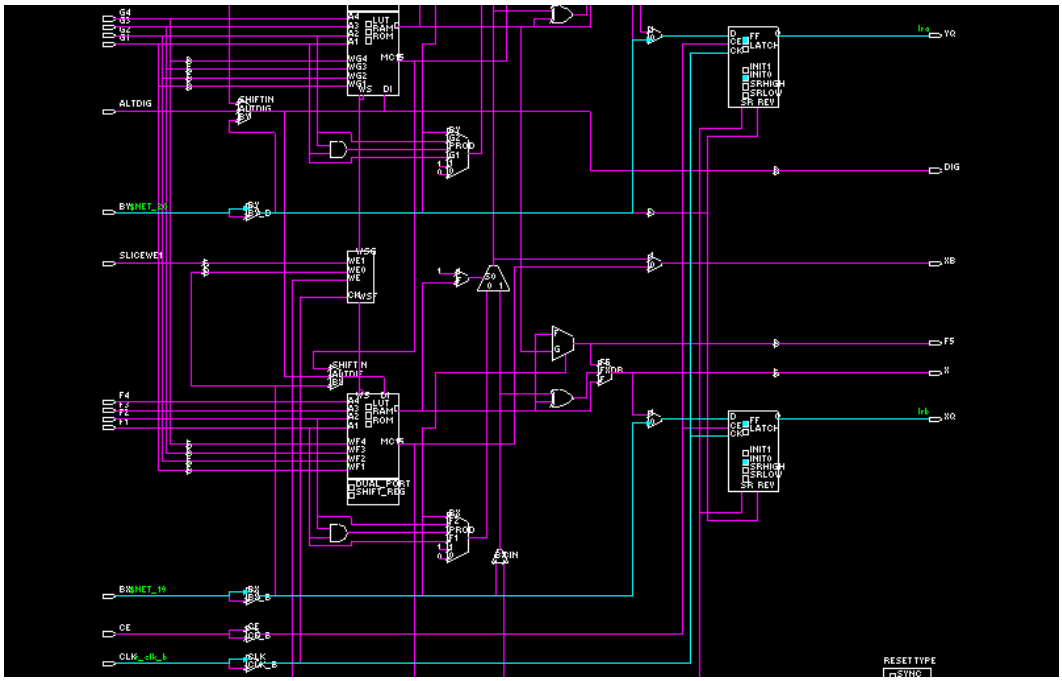

Figura 27 – Implementação dos *latches* de entrada do árbitro proposto por Moore e Robinson.

A Figura 28 mostra o *Slice* com a lógica combinacional e os *flip-flops* de saída da macro.

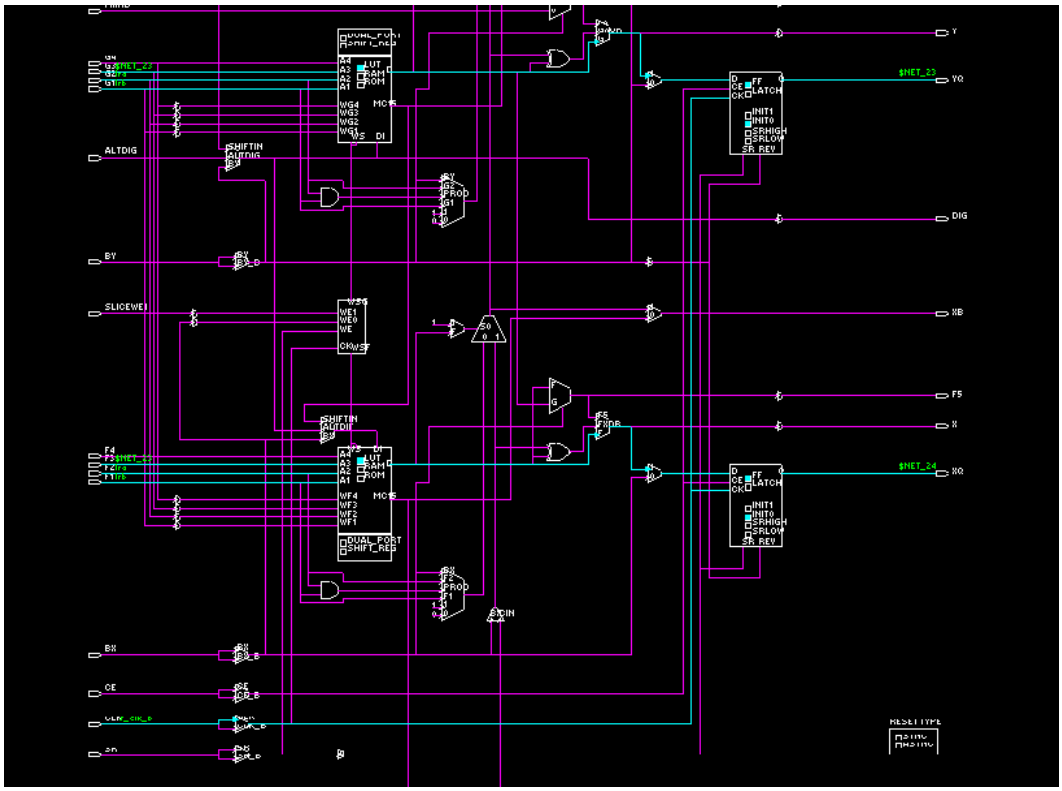

Figura 28 – Implementação da lógica combinacional e dos *latches* de saída da macro do árbitro proposto por Moore.

4.3 Conclusões do Capítulo

Esse Capítulo apresentou uma biblioteca de componentes assíncronos para FPGAs comerciais. Esta biblioteca de componentes permite a construção de circuitos assíncronos codificados em trilha dupla e *bundled data*. Essa biblioteca serve como base para projetos de circuitos assíncronos desenvolvidos dentro do grupo GAPH. Trabalhos futuros prevêem a expansão da biblioteca, inserindo novos componentes e portando os componentes pra outros FPGAs e a ligação de uma ferramenta de desenvolvimento de circuitos assíncronos como balsa, para a geração automática de circuitos assíncronos a partir de uma descrição de mais alto nível.

A Tabela 9 mostra todos os dispositivos que compõem a biblioteca proposta e o número de LUTs de cada uma ocupa.

Tabela 9 - Lista de componentes que compõem a biblioteca de hard macros e a área em LUTs para cada uma. ocupa. Biblioteca desenvolvida para os FPGAs XC3S200, XC2V1000, XC2V4000 das famílias Spartan 3 e Virtex II da Xilinx.

Dispositivo	Número de LUTs
Meio somador DIMS Strongly Indicating	7
Somador completo DIMS *Strongly Indicating*	12
Somador Completo DIMS Weakly Indicating	12
And DIMS de 2 entradas	5
Or DIMS de 2 entradas	5
Xor DIMS de duas entradas	6
C-element	1
Detector de Validade 8, 16 bits	15, 31
Portas SCAFFI	8 - 8
Geradores de Relógio em Anel	3 -4 -5
Stretcher	4
Árbitro – Moore	4
Conversor trilha dupla para trilha única 8, 16 bits	8, 16
Registrador trilha simples 8, 16 bits	16, 32
Elemento de atraso	1 - 2 – 4

5 INTERFACES DE COMUNICAÇÃO ASSÍNCRONAS

Diversas abordagens foram propostas na literatura para tentar solucionar o problema de comunicação entre módulos síncronos trabalhando em freqüências diferentes [CUM02] [GIN03] [MOO02] [NAJ05] [PON07] [MUT00]. As abordagens variam significativamente em função da forma de representação dos dados, da estratégia de sincronização utilizada, do protocolo de comunicação adotado e do estilo de circuito assíncrono adotado para as interfaces. Este Capítulo compara três destas propostas. As interfaces implementadas aqui foram escolhidas de forma a apresentar métodos com grau razoável de distinção na escolha da comunicação assíncrona.

5.1 Circuito de Teste e Projeto e Avaliação de Interfaces Assíncronas

A implementação utilizada para o projeto e avaliação de interfaces assíncronas consiste em um sistema produtor-consumidor com comunicação ponto a ponto, cujo diagrama de blocos é mostrado na Figura 29. O módulo produtor consiste de um contador que a cada novo valor de contagem produzido transmite este para o módulo consumidor. O módulo consumidor consiste de um decodificador que recebe o valor de contagem e o converte para então exibir o valor em um mostrador de sete segmentos de uma plataforma de prototipação baseada em FPGAs.

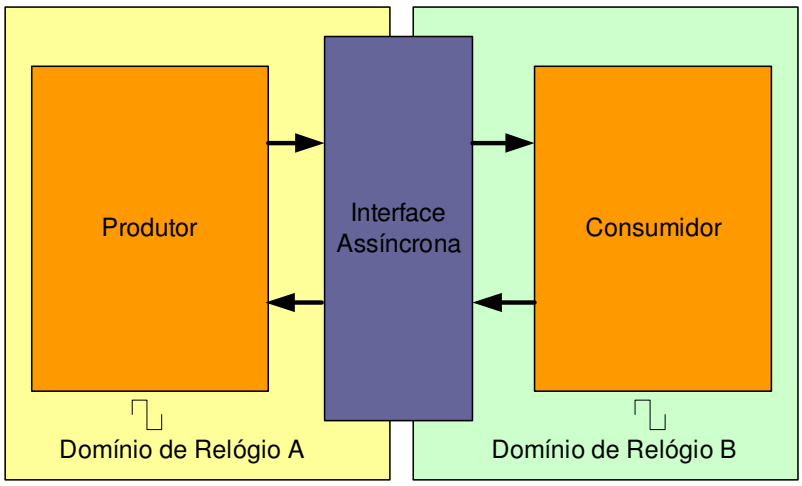

Figura 29 – Diagrama de blocos do sistema produtor-consumidor usado para validar as interfaces assíncronas implementadas.

Embora trivial, o sistema produtor-consumidor proposto permite a avaliação das interfaces de forma simples, e a velocidade de comunicação entre o produtor e o consumidor pode ser facilmente controlada através da máquina de estados do produtor e da freqüência de operação dos módulos. Todas as simulações deste Capítulo mostram os resultados gerados a partir da simulação com temporização usando valores de tempo de pior caso.

Os arquivos gerados para a simulação SDF possuem para cada elemento de atraso três

valores: o valor máximo (opção –sdfmax do Modelsim), típico (opção –sdftyp do Modelsim) e o mínimo(opção –sdfmin do Modelsim). A Figura 30 mostra a representação em SDF de um buffer de um IOB, com os tempos mínimo (= 1518ps), típico (=1554ps), e máximo (=1897ps).

```
(CELL (CELLTYPE "X_BUF")
(INSTANCE
prod/contador/prod_s_saida_5_DYMUX)
  (DELAY
    (ABSOLUTE
      (IOPATH I O (1518:1554:1897))
    )
  )
)
```

Figura 30 – Representação em SDF de um *buffer*.

5.1.1 Abordagem utilizando relógio pausável - baseada em Moore et al. [MOO02]

5.1.1.1 Descrição da abordagem

O modelo proposto em [MOO02] para a implementação de um sistema GALS utiliza a estratégia de extensão do relógio para a sincronização entre domínios distintos de relógio. Os dados são representados em codificação de trilha simples e o protocolo de comunicação utiliza *handshake* de duas fases. O árbitro proposto por Moore e Robinson [MOO98], já apresentado no Capítulo 4, foi uma contribuição importante para que sistemas GALS desse tipo sejam prototipáveis em FPGAs. Os circuitos de controle das portas de comunicação propostas por Moore et al. possuem também um diferencial importante. Enquanto abordagens anteriores, tais como a proposta existente em [NAJ05], pausavam o sinal de relógio durante todo o protocolo de comunicação a abordagem de Moore tenta evitar que a extensão do relógio ocorra, sempre que possível. O circuito da porta de entrada é mostrado na Figura 31, enquanto que o da porta de saída aparece na Figura 32.

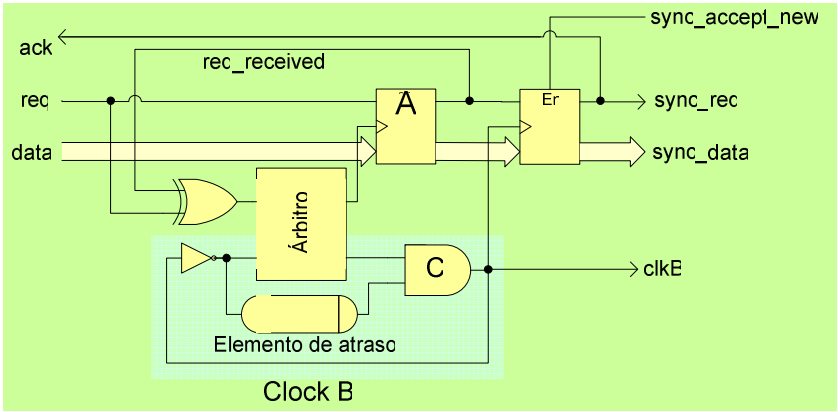

Figura 31 – Porta de entrada da interface com relógio pausável. Usada entre um produtor assíncrono e um consumidor síncrono.

Nota-se que cada uma das portas pode ser usada para conectar um domínio síncrono e um domínio assíncrono. Por outro lado, a conexão de uma porta de saída a uma porta de entrada permite interconectar dois domínio de relógio de forma unidirecional. Duplicando esses circuitos pode-se criar uma interface bidirecional entre dois domínios distintos de relógio.

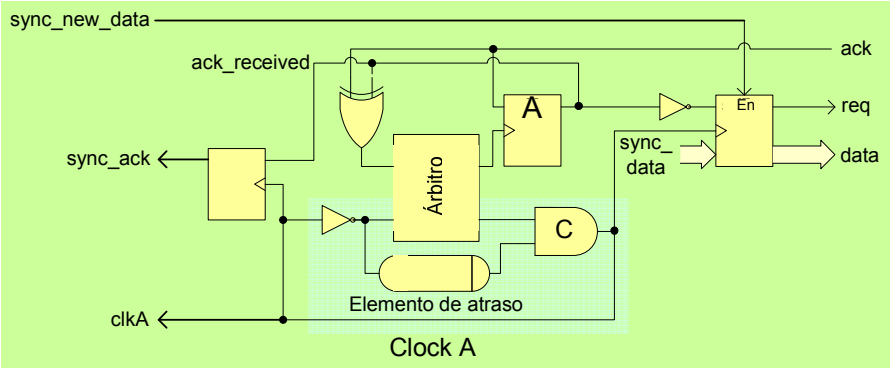

Figura 32 – Porta de saída da interface com relógio pausável. Usada entre um produtor síncrono e um consumidor assíncrono.

Embora o circuito da porta de entrada se assemelhe a um sincronizador, por possuir dois *flip-flops* em série, ele não se encaixa nesta classe de dispositivos. Apenas o segundo *flip-flop* é controlado pelo sinal de relógio pausável. O primeiro *flip-flop* é acionado pelo sinal de reconhecimento do árbitro, que ocorre quando uma requisição de transmissão foi atendida e o sinal de relógio foi pausado.

A janela de atendimento de requisição e o protocolo de comunicação são mostrados na Figura 33. O atendimento de uma requisição só acontece quando essa requisição chega dentro de uma janela, definida pelo nível alto do sinal de relógio. Se uma requisição acontece fora desta janela, ela precisa esperar até o próximo nível alto do pulso de relógio para ser atendida. O pior caso para esta espera dura meio ciclo de relógio e ocorre quando a requisição chega logo após a borda de descida do relógio.

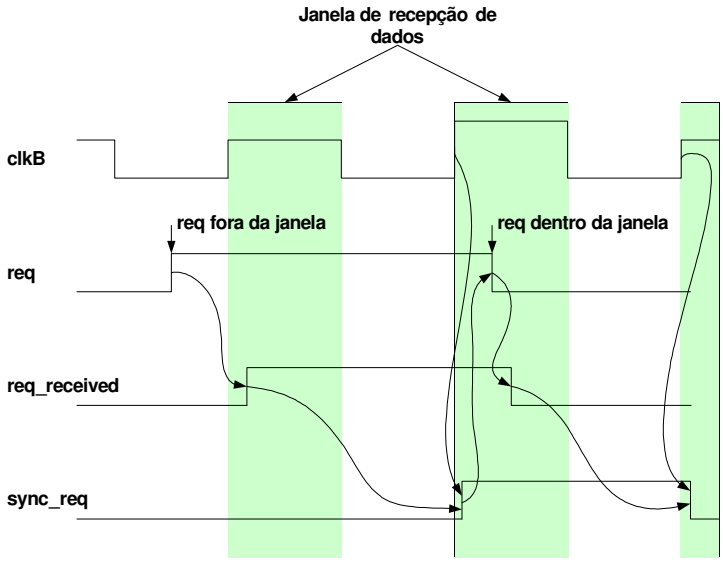

Figura 33 - Protocolo de comunicação para a interface assíncrona utilizando relógio pausável.

A Figura 31 mostra o circuito de interface entre um produtor assíncrono e um consumidor síncrono. Para iniciar o protocolo de transmissão o sinal de requisição deve ter o seu valor invertido, *req*↑ ou *req*↓. Quando isto ocorre, a porta *XOR* alimentada pelo sinal de requisição terá as suas entradas com valores distintos, uma vez que uma de suas entradas contém o valor antigo, de antes da transição da requisição. Isto ocorre devido ao registro do valor no *flip-flop A*. A outra entrada contém o valor atual. Com isso, a saída da porta *XOR* tem na sua saída uma transição *S*↑, gerando uma requisição de extensão do relógio. Se esta requisição acontece quando o sinal de relógio se encontra em nível alto, a requisição é atendida. Se o sinal de relógio se encontra em nível baixo, a requisição terá que esperar até a próxima borda de subida.

Quando atendida a requisição, o árbitro gera um sinal de reconhecimento, o qual é responsável por controlar a amostragem dos sinais de requisição e os dados no *flip-flop A*. Esta amostragem pode ocasionar metaestabilidade na saída do *flip-flop A*. A ocorrência de metaestabilidade neste estágio é pouco provável, contudo, uma vez que a geração do sinal de reconhecimento do árbitro, responsável por disparar a amostragem, acontece no mínimo *tp1* após o sinal de requisição chegar à entrada do *flip-flop A*. Aqui, *tp1* representa o tempo de propagação da porta *XOR*, somado ao tempo de propagação do árbitro. Quando a requisição não chega dentro do período em que o relógio está em nível alto, este tempo é acrescido do tempo que a requisição aguarda para ser atendida.

A metaestabilidade neste estágio pode ocorrer tanto no sinal de requisição quanto nos sinais do barramento de dados. Caso a metaestabilidade ocorra no sinal de requisição, pode acontecer uma falha de sincronização na saída do *flip-flop A*. Neste caso, este erro de sincronização não acarretará erro no funcionamento do circuito. O erro de sincronização mantém a porta *XOR* com suas entradas com valores complementares, uma em nível alto e outra em nível baixo, e a sua saída irá permanecer em nível alto, fazendo com que a requisição de pausa do relógio persista. Desta forma o sinal de relógio só será liberado quando a metaestabilidade for resolvida. Entretanto, não existe nenhuma garantia de que a metaestabilidade nos sinais de dados estará resolvida ao mesmo tempo em que a metaestabilidade do sinal de requisição. Para resolver este problema é necessário que a máquina de estados do circuito de controle responsável de gerar o sinal de requisição, gere-o somente após os dados estarem estáveis, ou seja, depois de um tempo maior que o tempo de *setup* do *flip-flop A*.

O sinal *sync_accept_new* funciona como um mecanismo de créditos. Enquanto este sinal estiver em nível alto, a interface gera sinais de requisição síncronos para o circuito consumidor, e gera também os sinais de reconhecimento *ack* para o produtor assíncrono.

O circuito de interface entre um produtor assíncrono e um consumidor síncrono, mostrado na Figura 32, possui funcionamento semelhante ao circuito descrito acima. Neste circuito, o sinal

que é assíncrono é o sinal de *ack*. Este sinal é responsável por gerar a requisição de extensão do sinal de relógio, e a resolução da metaestabilidade acontece da mesma forma que no circuito anterior. Entretanto a recomendação de gerar os sinais do barramentos de dados antes do sinal de requisição não é necessária nesta interface.

O sinal *sync_new_data* funciona de forma similar ao sinal *sync_accept_new* da interface descrita anteriormente, não permitindo que nenhuma requisição seja gerada se o circuito produtor síncrono não desejar transmitir.

5.1.1.2 Avaliação

A Figura 34 mostra a simulação SDF, gerada através do pior caso para os atrasos (-sdfmax), da abordagem proposta por Moore.

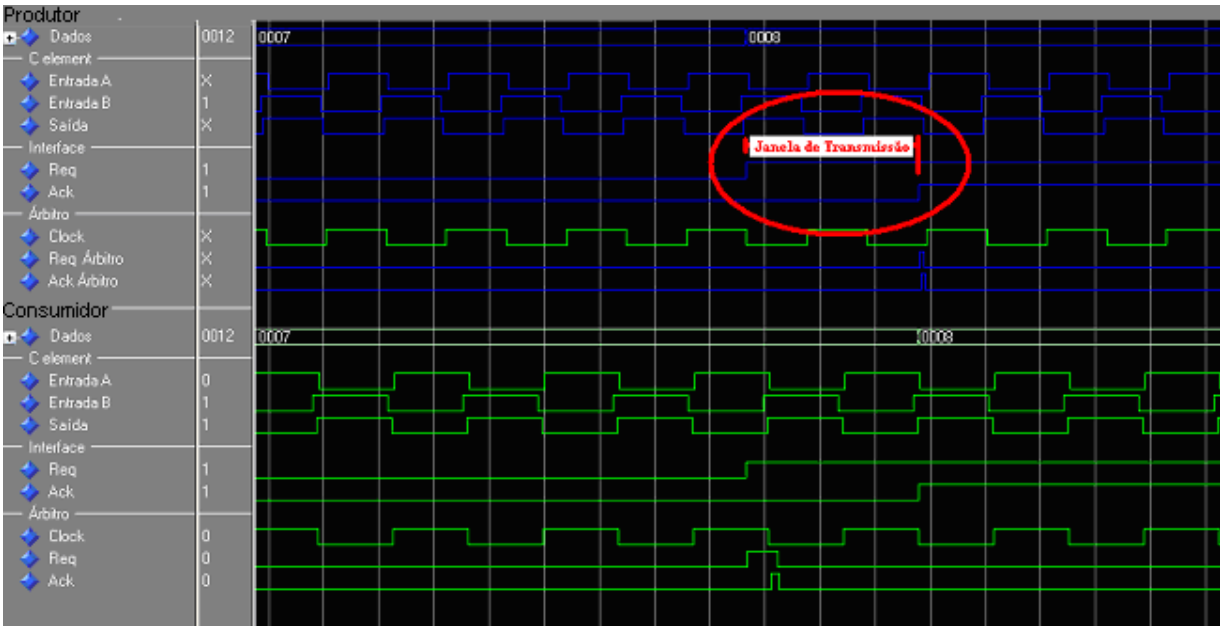

Figura 34 - Simulação SDF da abordagem proposta por Moore dividida em dois blocos: produtor e consumidor. Cada bloco é dividido ainda em C-element, Interface e Árbitro.

Nesta simulação é possível observar, a partir dos sinais de requisição e reconhecimento dos árbitros, que a abordagem proposta por Moore raramente provoca o alongamento do sinal de relógio. Verificando os sinais do C-element pode-se verificar a forma com que é realizada a sincronização do sinal que sai do elemento de atraso (Entrada B) e o sinal que sai do árbitro (Entrada A).

Os sinais referentes ao Árbitro mostram os sinais de requisição, reconhecimento e o sinal de relógio na entrada do árbitro, que é a negação do sinal de relógio do circuito síncrono.

Analisando o desempenho do circuito em função do sinal de relógio do circuito produtor, pode-se verificar que menos de dois ciclos depois de ter gerado o sinal de requisição, este já recebeu o sinal de reconhecimento.

O consumo de área desta abordagem é dependente da freqüência do relógio que a ilha assíncrona deve trabalhar. Quanto menor a freqüência, maior o número de elementos necessários para compor a linha de atraso. O árbitro proposto por Moore para FPGAs também consome uma quantia considerável de área, devido ao gerador de relógio pausável necessário ao seu funcionamento. Informações quantitativas são mostradas adiante, na Seção 5.1.4.

5.1.2 Abordagem utilizando sincronizadores

5.1.2.1 Descrição da abordagem

A forma mais direta de realizar a sincronização entre dois domínios de relógio distintos é através do uso de sincronizadores. Um sincronizador nada mais é, em sua forma mais simples, que uma cadeia de *flip flops* em série controlados pelo relógio do domínio local. A abordagem utilizada neste trabalho avaliou uma interface que funciona para o protocolo *bundle data* de duas fases. A Figura 35 mostra a estrutura geral da interface para realizar a comunicação entre produtor e consumidor.

Este tipo de interface não garante a eliminação da metaestabilidade, mas fornece uma medida probabilística que pode ser usada para determinar o tempo médio entre ocorrências de metaestabilidade na interface. De posse deste valor, pode-se dimensionar a interface para garantir que este tempo ultrapasse largamente a vida média dos componentes escolhidos.

As interfaces mostradas podem ser controladas por qualquer uma das bordas do sinal de relógio. Uma forma de diminuir a latência na interface é utilizar *flip-flops* alternando a borda de sensibilidade. Assim, o primeiro *flip-flop* da cadeia seria sensível à borda de subida, e o segundo à borda de descida. Essa abordagem diminui a latência, mas também a robustez do circuito, visto que o tempo entre a amostragem dos *flip-flops* é utilizado para resolver a metaestabilidade. Uma vantagem deste modelo é que ele pode ser totalmente implementado em HDLs.

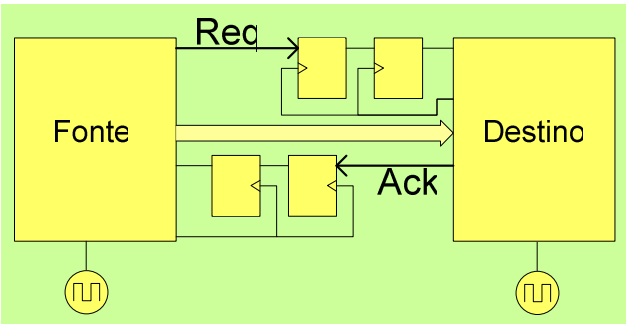

Figura 35 – Interface entre um produtor e um consumidor utilizando sincronizadores.

5.1.2.2 Avaliação

A Figura 36 e a Figura 37 mostram simulações dos sinais da interface de comunicação entre

o produtor e consumidor. Procede-se aqui a uma análise do desempenho da interface em função do sinal de relógio do produtor. Quando o sinal de relógio do produtor é maior que o do consumidor (Figura 36), verifica-se que a latência entre gerar o sinal de requisição (*req*) e obter o reconhecimento depois de sincronizado no produtor é muito grande. Ela é de sete ciclos de relógio, podendo variar para mais ou para menos, dependendo se a requisição é gerada imediatamente antes ou imediatamente depois do sinal de relógio do consumidor. Quando o sinal de relógio do produtor é menor que o do consumidor (Figura 37), a latência é reduzida para apenas quatro ciclos de relógio do produtor.

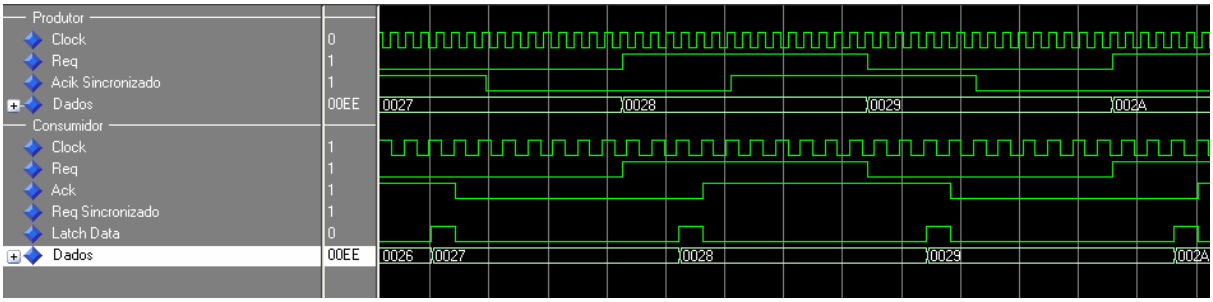

Figura 36 – Simulação SDF para período do produtor=26ns e período do consumidor=42 ns.

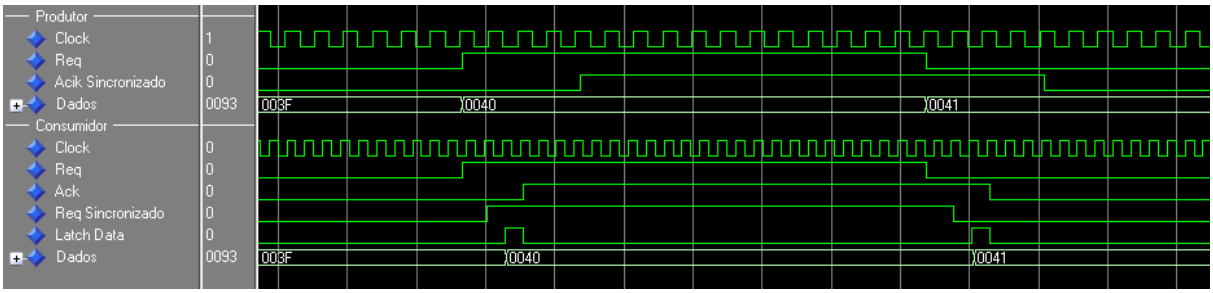

Figura 37 – Simulação SDF para período do produtor=42 ns e período do consumidor=26 ns.

Esta abordagem não adiciona muita área, percentualmente falando, assumindo que cada módulo síncrono associado à interface possui um alto grau de complexidade.

5.1.3 Abordagem utilizando fila bi-síncrona

5.1.3.1 Descrição da abordagem

Uma fila bi-síncrona consiste em uma fila onde a escrita na fila utiliza um sinal de relógio diferente do utilizado durante o processo de leitura da fila. A principal dificuldade de projeto de filas assíncronas reside na geração consistente dos sinais de fila cheia e fila vazia.

A implementação de uma fila necessita dois ponteiros, um para apontar a próxima escrita e outro para apontar a próxima leitura. Uma fila está cheia ou vazia quando os dois ponteiros apontam para o mesmo local. Em função desta condição para a geração dos sinais de fila cheia e vazia, observa-se a necessidade de realizar uma comparação entre os ponteiros. Este procedimento é trivial em sistemas síncronos, mas em uma fila bi-síncrona é problemático, uma vez que os ponteiros são

controlados por relógios diferentes, o ponteiro de escrita sendo controlado pelo relógio do produtor e o de leitura pelo relógio do consumidor.

A estratégia natural para resolver este problema é transferir o ponteiro de escrita para o domínio de relógio do consumidor onde será gerado o sinal de fila vazia, e transferir o ponteiro de leitura para o domínio de relógio do sinal de escrita onde é gerado o sinal de fila cheia.

A transferência dos ponteiros pode ser feita utilizando o método de sincronizadores ou o de relógio pausável. Porém, inserir um protocolo de *handshake* para controlar a transferência insere latência adicional no sistema.

Uma solução melhor é proposta em [CUM02][BEI06] é transferir os endereços através de sincronizadores, mas sem a inserção de um protocolo de *handshake*, passando os endereços diretamente por dois *latches*, como pode ser visualizado na Figura 38. Nesta, os sinais *wptr* e *rptr* correspondem respectivamente aos ponteiros de escrita e leitura e são sincronizados através de dois *latches*. O principal problema desta abordagem é que quanto maior o número de bits que são sincronizados, maior é a probabilidade de ocorrência de metaestabilidade em um desses bits. Para atenuar este problema os Autores propõem o uso de endereçamento baseados em código Gray.

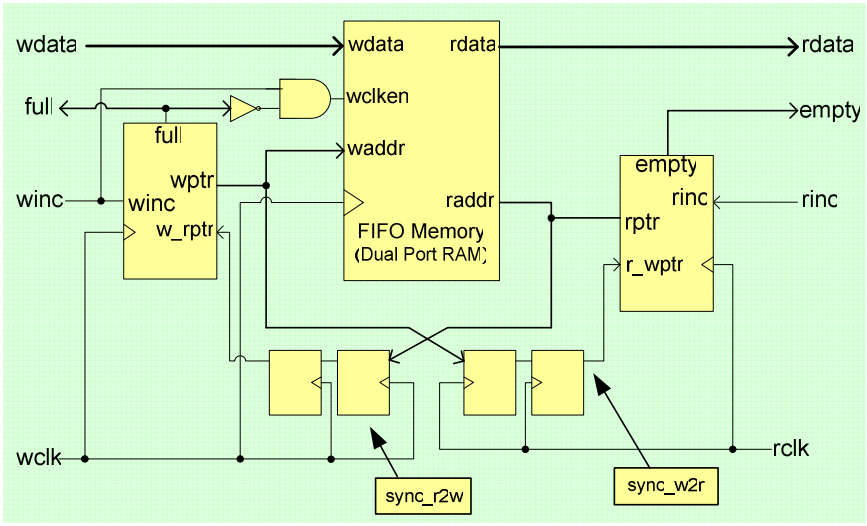

Figura 38 – Estrutura de uma interface utilizando uma fila bi-síncrona.

A codificação Gray possui a propriedade de que palavras consecutivas apresentam distância de Hamming igual a um, isto é, todos os bits de duas palavras consecutivas possuem o mesmo valor, com a exceção de um deles. Convertendo codificação binária para Gray reduz o risco de metaestabilidade na etapa de sincronização, uma vez que o endereço de escrita e leitura a cada nova operação tem seu valor incrementado, e somente um bit do endereço poderá causar riscos à sincronização.

A geração do sinalizador de fila cheia é realizada comparando o código Gray do endereço de escrita *wrptr* e o código Gray do endereço de leitura, obtido do sincronizador *wq2_rptr*. Da mesma forma, o sinalizador de fila vazia compara o endereço de leitura *rptr* com o endereço de escrita

sincronizado *rq2_wptr* para verificar se estes são iguais. Tanto para a geração da sinalização de fila cheia quanto para a de fila vazia os endereços devem ser iguais. Para diferenciar os estados de fila cheia e fila vazia, é necessário acrescentar mais um bit ao endereço. Este bit deve ser invertido toda vez que o ponteiro da fila passar da última posição do buffer para a primeira. Denomina-se este bit aqui de *marcador de página*. Utilizando este bit, pode-se diferenciar o estado de fila cheia do estado de fila vazia. Porém, para que isto ocorra de forma consistente, o bit marcador de página deve passar pelo estágio de sincronização juntamente com os apontadores.

Utilizando o bit marcador de página e o endereço da fila estado da fila é definido da seguinte forma:

A fila se encontra no estado cheia quando os endereços são iguais e o bit marcador de página de escrita é diferente do bit marcador do bit de leitura sincronizado.

A fila se encontra no estado vazio quando os endereços são iguais e o bit marcador de página de leitura é igual ao bit marcador do bit de escrita sincronizado.

Adicionando-se o bit marcador de página insere-se a possibilidade de dois bits alterarem o seu valor ao mesmo tempo. Isto acontece quando o apontador passa do topo da fila para o seu início. Neste caso, além da variação de um bit do apontador Gray o bit marcador de página também irá alterar seu valor.

Como a abordagem de sincronização entre os dois domínios de relógio é realizada através de sincronizadores, o sistema é passível de ocorrência de metaestabilidade nos apontadores Gray. A metaestabilidade pode ocorrer em um bit do apontador, o bit que teve seu valor alterado na última escrita ou leitura, ou no bit marcador de página.

A ocorrência de metaestabilidade no apontador não causa erros no funcionamento do circuito. A única conseqüência da ocorrência de falha de sincronização no apontador seria a geração prematura da sinalização de fila cheia ou fila vazia.

A ocorrência de metaestabilidade no bit marcador de falha pode levar a fila a um estado em que esta fica travada enquanto a metaestabilidade não se resolve. Neste estado, a fila não permite nem escritas e nem leituras pois sinaliza ao mesmo tempo que está cheia e vazia.

Considere-se uma fila de profundidade igual a oito. Quando inicializada, o apontador de leitura, o apontador de escrita e os dois marcadores de página estão zerados, e como tanto os apontadores como os marcadores de página possuem valores iguais a fila está vazia. Considere-se agora que a partir deste estado foram realizadas oito escritas na fila e nenhuma leitura, o apontador de escrita saiu da posição zero e, após as escritas, encontra-se na posição sete. Acontecendo uma nova escrita, o apontador é zerado e o marcador de página de escrita é invertido tendo seu valor alterado para ‘1’. Considere-se que neste momento ocorre uma falha de sincronização que afeta o valor do bit marcador de página. O módulo de leitura receberá dois ciclos de relógio de leitura,

seguido do apontador com valor zero e do marcador de página com valor também zero, pois ocorreu uma falha de sincronização, e seu valor foi invertido no sincronizador. Quando o módulo de leitura compara os valores recebidos do sincronizador com o apontador de leitura e o marcador de página de leitura observa que são iguais e sinaliza fila vazia. Ao mesmo tempo, o módulo de escrita estará sinalizando fila cheia, pois possui apontadores iguais e marcadores de página invertidos. Esta situação é restabelecida quando a metaestabilidade for resolvida na etapa de sincronização do sinal de escrita.

Para evitar que sejam realizadas escritas enquanto a fila esta cheia ou leituras enquanto a fila esta vazia os Autores dos trabalhos citados propõem que o ponteiro Gray seja gerado uma posição à frente da real posição da fila. Isto faz com que a abordagem seja mais robusta ao custo de uma posição da fila.

5.1.3.2 Avaliação

A principal vantagem da utilização da fila é a sua capacidade de permitir a realização de escrita e leitura em cada ciclo de relógio. Esta afirmação é verdadeira desde que a fila não se encontre em um estado onde ela está cheia ou vazia. Quando a fila está cheia e é realizada uma operação de leitura, a fila só deixará o estado de cheia dois ciclos de relógio depois da leitura. Isto ocorre porque o novo endereço de leitura passa pelo estágio de sincronização. O mesmo acontece quando a fila está vazia e ocorrer uma escrita. Desta forma, a fila não é vantajosa quando a velocidade de uma das operações é muito maior que a outra, pois quando isso acontece a fila freqüentemente estará cheia ou vazia e a latência na comunicação neste caso seria de no mínimo dois ciclos de relógio. A fila implementada neste trabalho possui um *buffer* de oito posições.

A maior desvantagem da fila é a área que esta consome. Os contadores e o *buffer* de armazenamento dos dados são os principais responsáveis pela grande demanda de área. Como citado anteriormente, para a fila ser vantajosa em termos de latência, deve-se evitar que esta entre em estados de Fila Cheia e Fila Vazia. Para alcançar este objetivo com módulos trabalhando em velocidades de comunicação muito diferentes pode ser necessário um *buffer* proporcional à diferença entre as velocidades.

A Figura 39 mostra as formas de onda para a simulação com temporização da fila bi-síncrona quando o produtor opera com um sinal de relógio de menor freqüência que o consumidor. Na Figura é possível observar os endereços de escrita no produtor e de leitura no consumidor. Além disso, no módulo consumidor é possível notar a geração do sinal de fila vazia, o qual provoca o atraso de leitura no endereço 3.

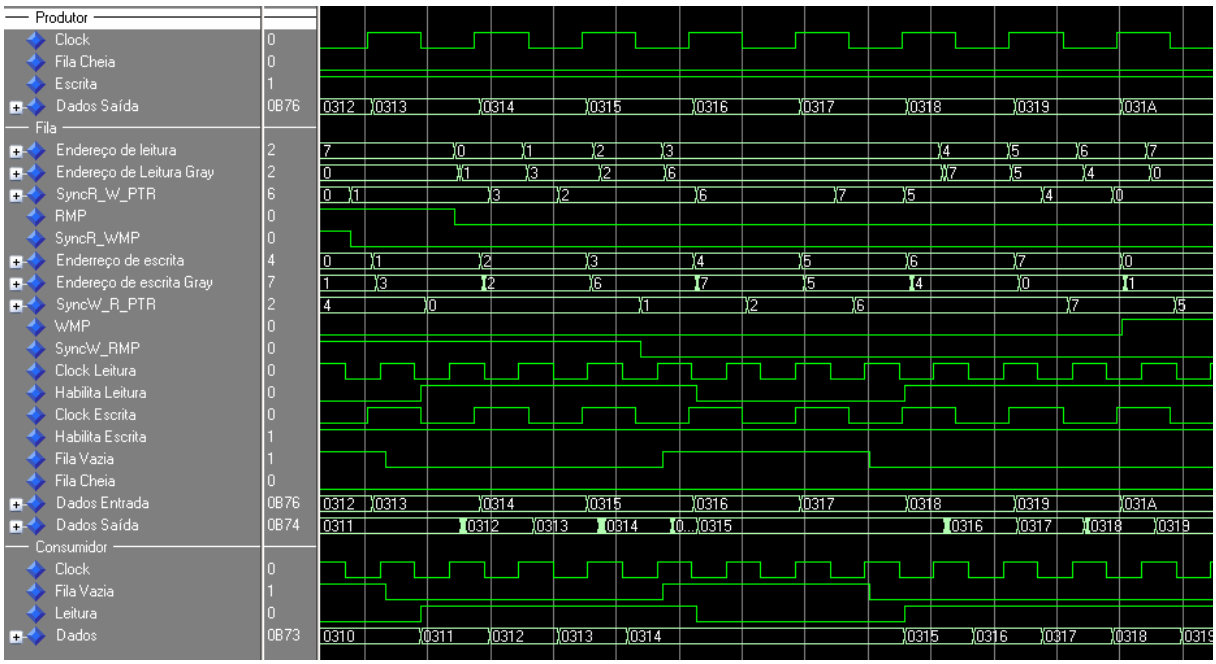

Figura 39 – Simulação SDF de uma fila bi-síncrona para Tcons = 22ns e Tprod = 34ns.

5.1.4 Comparação entre as abordagens

Esta Seção apresenta comparações quantitativas entre as abordagens estudadas em termos de área e desempenho. Os dados apresentados a seguir foram obtidos para o circuito de teste descrito na Seção 5.1. A comunicação entre o produtor e o consumidor se dá através da troca de dados de 16 bits. Na abordagem utilizando fila bi-síncrona, o *buffer* utilizado tem tamanho de 8 palavras de 16 bits.

A Tabela 10 mostra o consumo de área das três abordagens estudadas. Verifica-se nessa Tabela que a abordagem utilizando fila bi-síncrona consumiu mais área, como esperado. Isto ocorre porque esta possui circuitos de controle mais complexos. O total de portas equivalentes apresentado na Tabela não inclui as portas consumidas pelas *hard macros*. Mesmo assim, pode-se afirmar que em geral a abordagem de relógio pausável consome menos área que a abordagem utilizando filas.

Tabela 10 – Comparação entre as áreas consumidas. Dados obtidos para o dispositivo XC3S200, Família Spartan 3 da Xilinx.

Abordagem	Número de Flip Flops	Número de LUTs usados para lógica	Número de LUT RAMs	Número de hard macros	Total de portas equivalentes
Relógio Pausável	135	118	0	6	2096(*)
Sincronizadores	104	86	0	0	1656
Fila Bi-síncrona	90	152	32(**)	0	3988

(*) Não inclui as *hard macros*.

(**) LUTRAMs usadas como RAM de porta dupla.

A Tabela 11 e a Tabela 12 buscam avaliar o desempenho das interfaces. Para medir o desempenho de forma a possibilitar um estudo comparativo entre estas, foi adotado um método

onde o tempo entre 48 transmissões de dados entre produtor e consumidor foi verificado através dos resultados obtidos das simulações com atraso (48 é o mesmo que seis vezes a profundidade da fila empregada). Na Tabela 11 são apresentados os tempos referentes às medidas onde o sinal do relógio do consumidor é mantido com período constante e altera-se o valor do período do sinal de relógio do produtor. O mesmo método é utilizado na Tabela 12, onde o período do relógio do produtor foi mantido constante e o período do consumidor foi variado.

Tabela 11 – Comparação entre o desempenho das abordagens em função do sinal de relógio. Dados obtidos para o dispositivo XC3S200, Família Spartan 3 da Xilinx.

$T_{prod\ (ns)}$	$T_{cons\ (ns)}$	$t_{Sinc\ (ns)}$	$t_{RP\ (ns)}$	$t_{fila\ (ns)}$
30	30	12960	3660	1440
34	30	12330	4200	1650
38	30	13650	4620	1800
42	30	14130	5160	1920
46	30	15450	5610	2130
50	30	16800	7290	2310

Tabela 12 – Comparação entre o desempenho das abordagens em função do sinal de relógio do consumidor.

$T_{prod\ (ns)}$	$T_{cons\ (ns)}$	t Sinc (ns)	t RP (ns)	t fila (ns)
30	30	12960	3660	1440
30	34	12206	4148	1632
30	38	13642	4218	1824
30	42	14112	4410	2016
30	46	14468	4508	2208
30	50	14400	4700	2400

A Figura 40 e a Figura 41 apresentam gráficos contendo as curvas de transmissão em função do período do produtor e do consumidor, respectivamente. Na abordagem utilizando sincronizadores é interessante notar que o sistema funcionando com os dois módulos com período de 30ns possui tempo maior que quando um o produtor possui período de 40ns e o consumidor é mantido funcionando com 30ns. Isto acontece devido ao fato que quando os dois circuitos foram simulados com mesmo período, ambos possuíam a mesma fase nos respectivos sinais de relógio. Neste caso, o sinal de requisição, produzido na borda de subida do relógio do produtor chega ao produtor logo após a borda de subida do receptor. Este é percebido somente na próxima borda do sinal de relógio. Da mesma forma, o sinal de reconhecimento é gerado na borda de subida do consumidor. Esta configuração apresenta um desempenho muito próximo do pior caso, para sincronizadores funcionando nestes valores de freqüência.

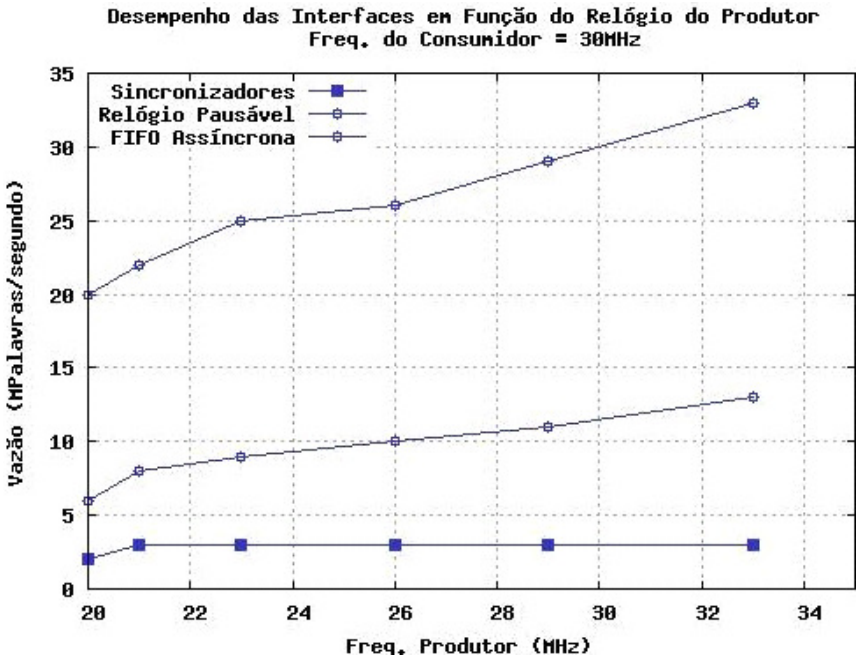

Figura 40 – Comparação entre o desempenho faz abordagens em função do sinal de relógio do produtor. O relógio do consumidor foi fixado com período Tcons = 30ns.

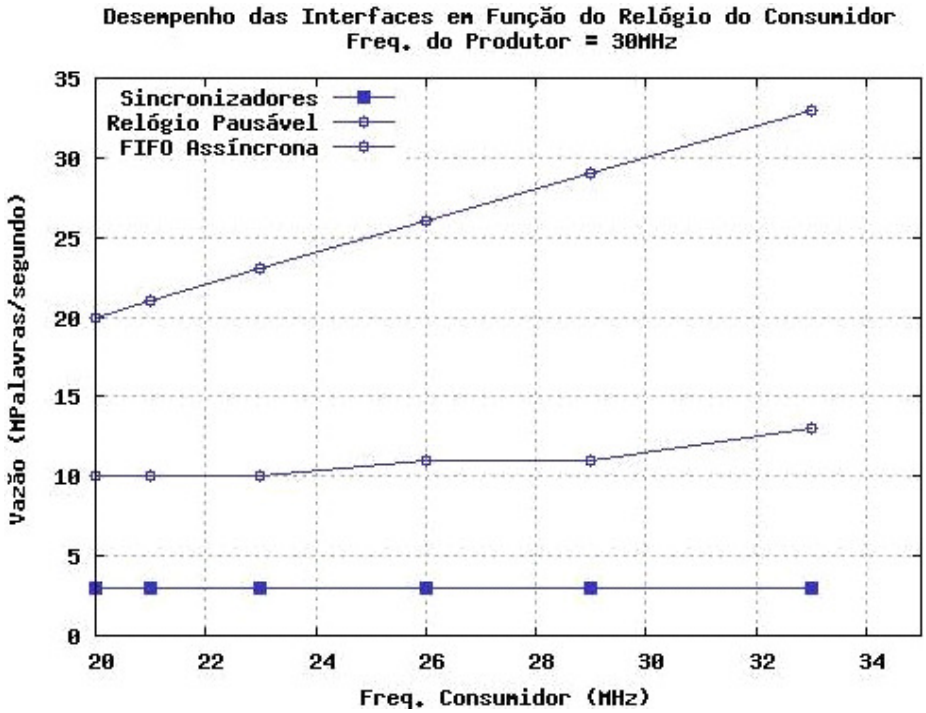

Figura 41 – Comparação de desempenho de interfaces em função do sinal de relógio do consumidor. O relógio do produtor foi fixado com período Tprod = 30ns.

5.2 Conclusões do Capítulo

Este Capítulo apresentou a comparação de interfaces assíncronas de comunicação propostas na literatura e a especificação. Entre as interfaces simuladas a abordagem de fila bi-síncrona foi a que apresentou melhor desempenho. Outra vantagem é que essa interface pode ser totalmente descrita em código RTL. O resultado do estudo das interfaces foi utilizado para selecionar qual interface assíncrona mais adequada para o uso nas redes intrachip Hermes-G e Hermes-GLP.

6 SCAFFI

Esse Capítulo descreve o desenvolvimento de uma interface assíncrona de comunicação ponto a ponto desde a especificação até a sua validação através de um caso de uso. Essa interface é uma contribuição desse trabalho de mestrado e foi desenvolvida com objetivo de suportar a interação entre módulos síncronos funcionando com sinais de *relógio* distintos ou a interação entre módulo síncrono e módulo assíncrono *bundled data* ou *dual rail* de quatro fases.

A interface SCAFFI (*Stretchable Clock Asynchronous Flexible FPGA Interface*) [PON07] utiliza o mecanismo de extensão de relógio para sincronização dos dados. Este mecanismo foi escolhido devido a sua capacidade de, desde que corretamente especificado, evitar a metaestabilidade. O esquema geral da interface proposta é apresentado na Figura 42.

A interface é composta de portas que utilizam um protocolo de duas fases entre porta e ilha síncrona e um protocolo de quatro fases entre porta e o sistema assíncrono. Maiores detalhes sobre a implementação das portas e os protocolos são apresentados na Seção 6.1.

O esquema geral do sistema de comunicação é mostrado na Figura 42. Este esquema é semelhante aos sistemas GALS propostos por [NAJ05] [MUT00]. Os diferenciais dessa proposta são: o mecanismo de extensão do relógio e o suporte a comunicação entre sistemas síncronos com sistemas assíncronos.

As propostas de [NAJ05] [MUT00] utilizam um elemento de exclusão mútua para decidir quando uma requisição deve ser atendida e o relógio deve ser pausado. Este trabalho propõe a utilização de um *Stretcher* como alternativa aos elementos de exclusão mútua. Quando uma interface utiliza um elemento de exclusão mútua para gerar a extensão do relógio, o relógio é pausado somente em um dos níveis, desta forma, a interface atende a requisições somente dentro de uma janela delimitada do sinal de relógio. Se a requisição ocorre fora desta janela, o sinal de relógio é estendido somente na próxima janela. Com o uso do *Stretcher* o sinal de relógio é estendido o mais rápido possível, isso faz com que a transmissão de dados ocorra de uma forma mais rápida evitando o trancamento prolongado do módulo transmissor enquanto a requisição é atendida pelo consumidor.

Outro diferencial dessa proposta de interface é a disponibilização de elementos que permitem a comunicação de módulos síncronos com módulos assíncronos *dual rail* de quatro fases. A utilização de comunicação *dual rail* é explorada mais detalhadamente na Seção 6.3.

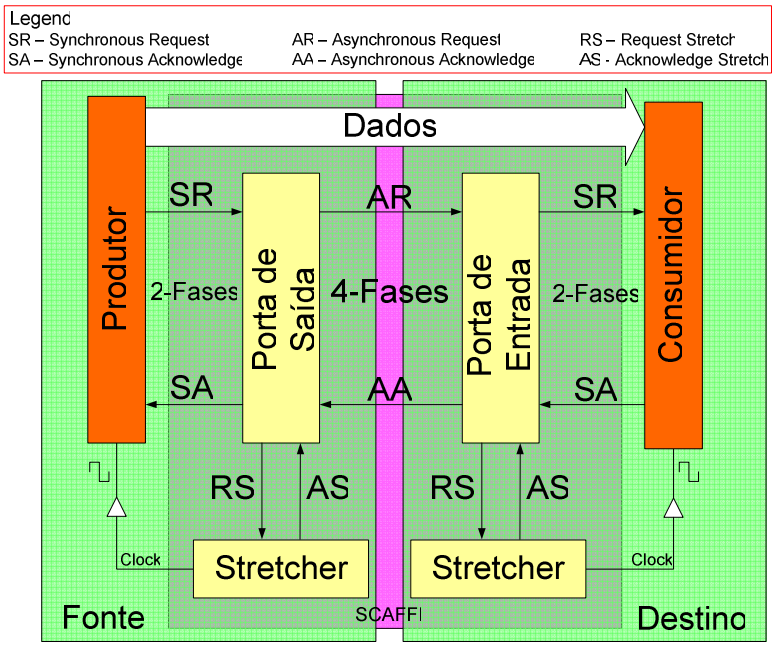

Figura 42 – Diagrama de blocos da interface SCAFFI, mostrando a comunicação entre um produtor e um consumidor de dados.

6.1 Módulo Extensor de Relógio (Stretcher)

Interfaces que realizam a extensão de relógio utilizam elementos de exclusão mútua para decidir quando o sinal de relógio irá oscilar e quando será pausado. A implementação mais comum desses elementos utilizam *latches* RS acoplados a um sistema de filtros que são encarregados de eliminar a metaestabilidade da saída do ME.

Dispositivos do tipo FPGA não permitem a construção de filtros, portanto a eliminação da metaestabilidade no árbitro não pode ser garantida.

Najibi [NAJ05] em seu trabalho propõe um sistema GALS que implementa o elemento de exclusão mútua utilizando um *latch* RS. Para evitar a metaestabilidade ele propõe que o sinal assíncrono de requisição de extensão do sinal do relógio passe por um *latch* sensível ao nível alto. Entretanto este método não evita metaestabilidade apenas transfere a possibilidade de metaestabilidade do árbitro para o *latch*.

O trabalho de Moore [MOO98] apresenta um circuito de árbitro para ser prototipado em FPGAs. Nesta abordagem o árbitro possui um sinal de relógio próprio. O sinal de requisição e o sinal de relógio devem ser sincronizados com o relógio do árbitro para então a arbitragem ser realizada. Este método embora quase elimine a probabilidade de metaestabilidade no árbitro introduz uma latência grande no mecanismo de arbitragem. A conseqüência desta latência é a inserção de um limite de freqüência que o gerador de relógio em anel pode gerar, uma vez que cada transição do sinal de relógio que chega ao árbitro leva dois ciclos de relógio do árbitro para aparecer na saída. Desta forma a freqüência do sinal de relógio que sai do gerador de relógio em anel é no

mínimo duas vezes maior que a freqüência do relógio do árbitro.

Neste trabalho é proposto um mecanismo para extensão do sinal de relógio que substitui o elemento de exclusão mútua por um *Stretcher*. O *Stretcher* é um componente que tem a finalidade de parar o sinal de relógio assim que uma requisição de extensão, não importando em que nível o sinal de relógio se encontra, chegue ao *Stretcher*.

O circuito do *Stretcher* é apresentado na Figura 43 – . O circuito é composto por um C-element, um multiplexador e elementos de atraso. O multiplexador é responsável por selecionar entre manter o sinal de relógio oscilando ou pausá-lo.

Os circuitos do multiplexador e do C-element foram projetados de forma que suas saídas sejam *hazard free*, sob o pressuposto do modo fundamental.

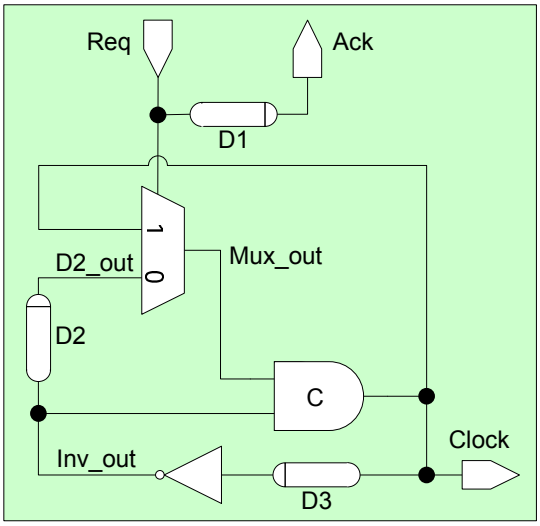

Figura 43 – Estrutura do *Stretcher*. O oscilador em anel compreende os elementos: C-element, o elemento de atraso D3 e o inversor.

Neste circuito, enquanto não houver uma requisição, o sinal de relógio de entrada passa pelo multiplexador e alimenta as duas entradas do C-element fazendo com que a saída deste copie as suas entradas.

Quando uma requisição for assinalada, o multiplexador terá como saída o sinal de relógio da saída, o qual alimentará uma das entradas do C-element fazendo com que nenhuma transição do sinal de relógio de entrada apareça na saída, assim o relógio é estendido.

O elemento de atraso D2 é inserido para garantir que não serão gerados *glitches* quando o sinal de requisição chegar simultaneamente com uma borda do sinal de relógio. Assim quando o sinal de relógio chegar ao multiplexador ao mesmo tempo em que uma requisição o elemento de atraso garante que o sinal de relógio já está estável na entrada do C-element. Desta forma o C-element então atua como um filtro, eliminando todos os *hazards* provenientes do multiplexador. Um exemplo de valor transitório gerado pela saída do multiplexador e filtrado pelo C-element é mostrado na Figura 44, a qual mostra o resultado de simulação SPICE do C-element para a tecnologia 0,35µm. O circuito prototipado em FPGA demonstra o mesmo comportamento.

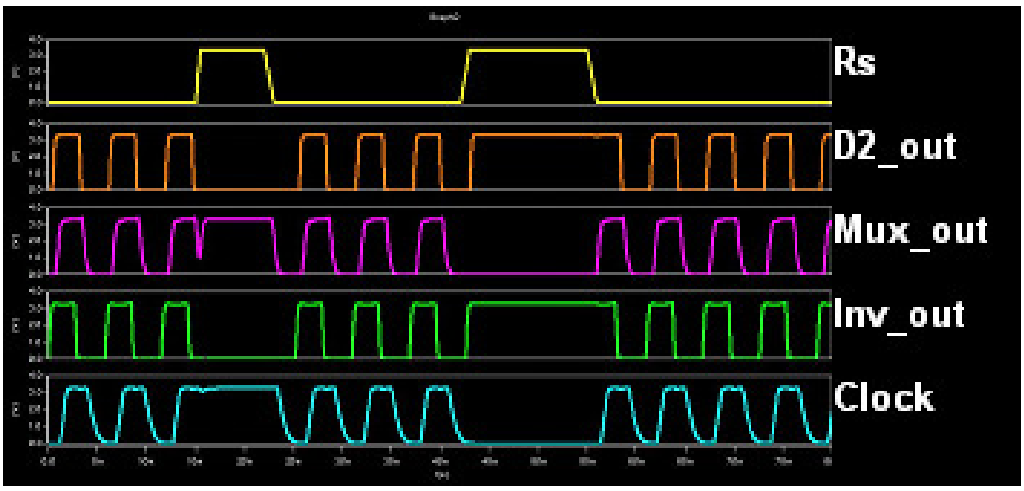

Figura 44 – Simulação SPICE do Stretcher.

6.2 Portas

A interface SCAFFI é dotada de portas de comunicação assíncronas responsáveis por converter o protocolo de duas fases, usado entre módulo síncrono e porta, para um protocolo de quatro fases e ainda controlar as solicitações de extensão do sinal de relógio. O protocolo de quatro fases foi escolhido para a comunicação assíncrona devido ao seu grande uso em circuitos assíncronos *dual rail* e *bundled data*.

A implementação das portas partiu de uma especificação modo rajada (do inglês, *burst mode*), do comportamento das portas. Esta especificação foi utilizada para alimentar a ferramenta MINIMALIST [FUH99], a qual gera circuitos de dois níveis e livres de *hazard*. O grafo da especificação *burst mode* é mostrado na Figura 45. Os circuitos gerados pela ferramenta MINIMALIST para as portas de saída e entrada são mostrados na Tabela 13.

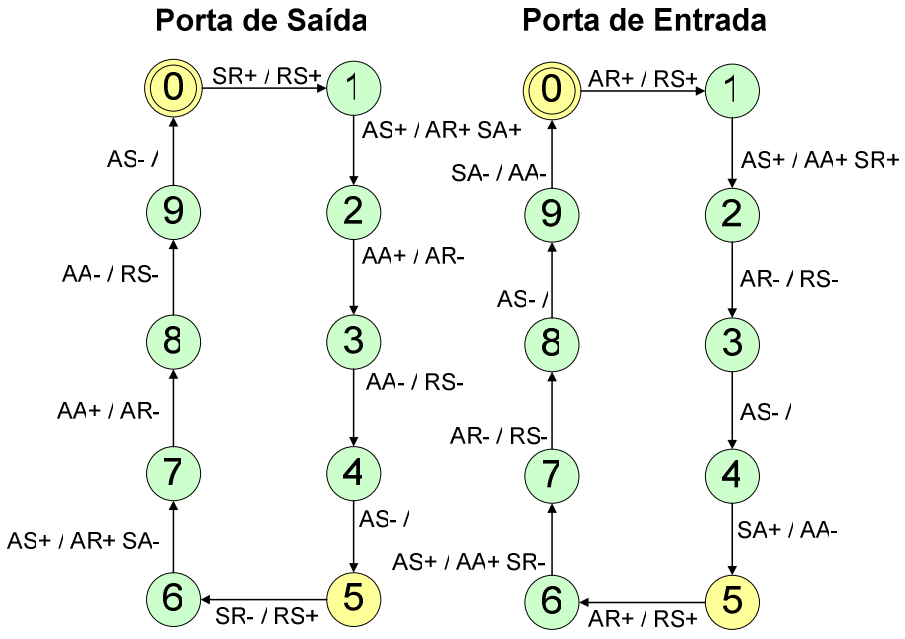

Figura 45 – Especificação *burst mode* das portas de entrada e de saída da interface SCAFFI.

Os circuitos gerados pela ferramenta MINIMALIST devem funcionar dentro do modo fundamental, isto é, é permitida a alteração de um sinal de entrada a cada instante de tempo e nenhuma outra alteração pode ocorrer até que o circuito se estabilize. Essa restrição se aplica também aos sinais de realimentação. Para que essas restrições sejam alcançadas em circuitos prototipados em FPGAs, as portas da interface SCAFFI foram desenvolvidas utilizando *hard macros* e inseridas na biblioteca proposta no Capítulo 4.

Tabela 13 – Equações lógicas das portas definidas pela ferramenta MINIMALIST.

Porta de Saída	Porta de Entrada
$RS = AA + \overline{SR} * Y_0 + SR * \overline{Y_0}$	$RS = AR$
$AR = SR * AS * \overline{AA} * \overline{Y_0} + \overline{SR} * AS * \overline{AA} * Y_0$	$SR = \overline{SA} * AS + Y_0 + SA * \overline{AS} * AR$
$SA = SR * Y_0 + \overline{AS} * Y_0 + SR * AS$	$AA = \overline{SA} * Y_0 + AS + SA * \overline{AR} * \overline{Y_0}$
$Y_0 = SR * AA + SR * Y_0 + \overline{AA} * Y_0$	$Y_0 = \overline{SA} * AS * \overline{AR} + \overline{AR} * Y_0$

As portas implementadas possuem um sinal de realimentação, Y0 (ver Tabela 13), que representa o estado da porta. Para atender a restrição de modo fundamental nos circuitos das portas foi inserida uma LUT no sinal de realimentação Y0 de ambas as portas, mostrado no canto inferior esquerdo da Figura 46. Essa LUT, além de inserir um atraso para a satisfação da restrição, também insere um mecanismo de *reset* na porta.

Uma segunda restrição importante para que as portas proposta não produzam valores transitórios diz respeito às derivações nos sinais de entrada da interface. Tais derivações devem ser isócronas simétricas. Para isso, as *hard macros* foram implementadas garantindo que todos os sinais de entrada possuem um único ponto de entrada na macro, incluindo o sinal de realimentação Y0. O objetivo dessa abordagem é que todas as derivações tivessem seu atraso especificado e validado no desenvolvimento da *hard macro*. Para tanto, foram inseridas LUTs transparentes nos sinais de entrada das portas conforme ilustrado na da Figura 46 onde as LUTs a esquerda são transparentes. Essa Figura representa a macro da porta de saída da interface SCAFFI para um FPGA Spartan 3 XC3S200. As derivações devem ter escorregamento menor que 0.21ns, que é o valor mínimo do tempo de propagação de uma LUT do FPGA citado. A inserção de LUTs transparentes adicionam atraso as portas e reduzem a máxima taxa de transferência na interface. Uma alternativa a essa abordagem seria a não utilização das LUTs transparentes, com exceção da LUT inserida para o sinal Y0, e a adição de restrições temporais de *MAXSKEW* mostradas na Figura 21. Entretanto, a adição de restrições não garante que essas serão atendidas pela ferramenta de roteamento. Desta forma, é necessária a validação temporal pós roteamento das portas, etapa não necessária com a macro proposta.

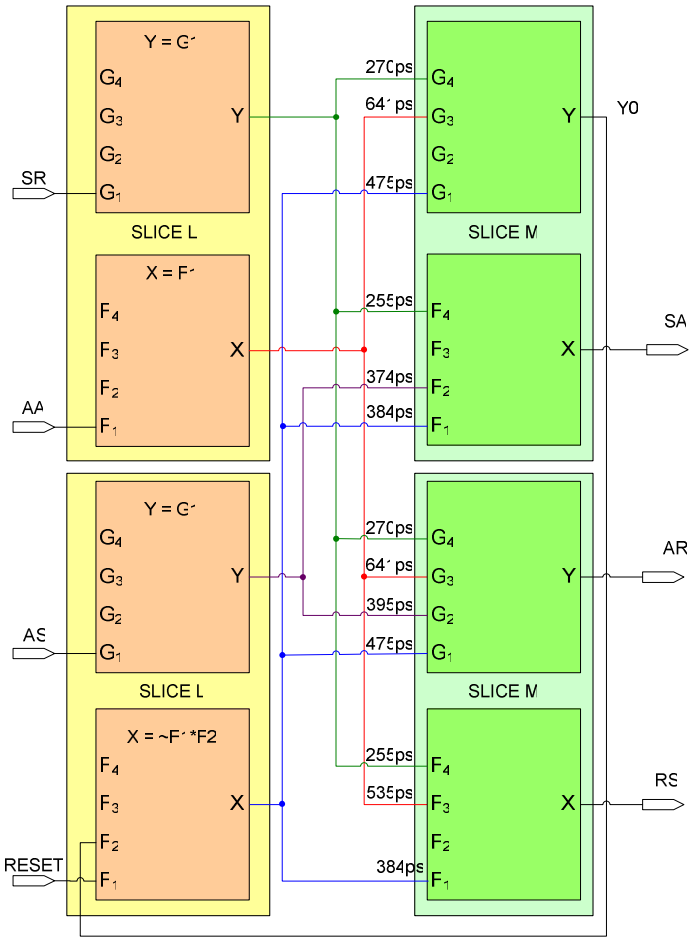

Figura 46 – Implementação de uma porta de saída da SCAFFI usando hard macros. As equações dos sinais Y0, AS, AR e RS são apresentadas na Tabela 13.

O protocolo de comunicação da interface SCAFFI apresentado na Figura 45 é ilustrado na Figura 47 e descrito a seguir.

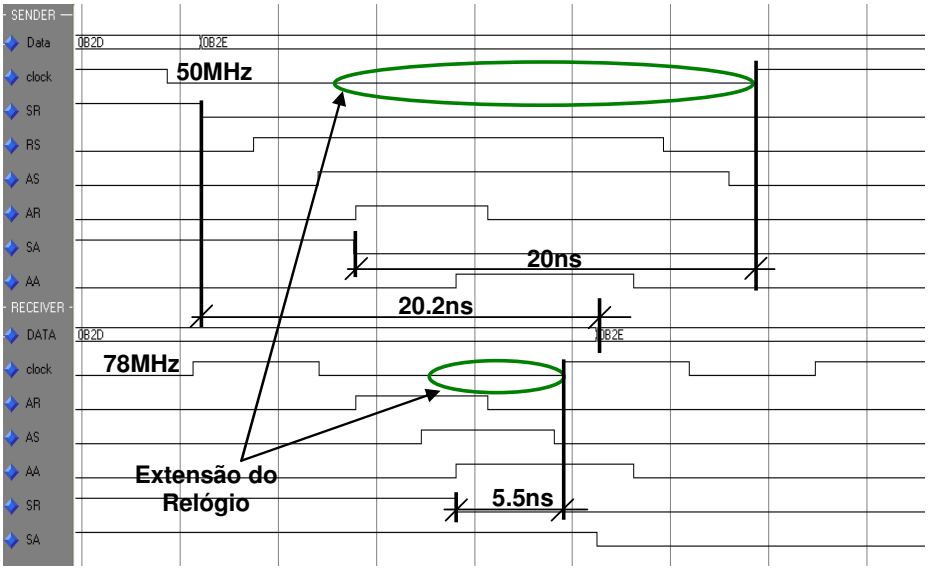

Figura 47 – Exemplo de uma transmissão de dados utilizando a SCAFFI extraído de uma simulação com temporização. Os sinais AR e AA são repetidos no transmissor e no receptor para facilitar a compreensão. O sinal RS do receptor é o mesmo sinal AR (ver Tabela 13). As duas portas partem do estado 5.

Uma transmissão de dados é iniciada quando o transmissor gera uma requisição síncrona, através de uma borda em *SR* (os sinais *SR* e AS pertencem ao protocolo de duas fases e são ativados por borda e não por nível), solicitando a porta de saída o início de uma transmissão de dados. Juntamente com o sinal *SR* o transmissor deve inserir os dados a serem transmitidos no canal de comunicação. Quando a porta de entrada recebe a requisição produz uma requisição de extensão do sinal de relógio *RS*↑ para o *Stretcher*.

Quando o sinal de relógio estiver garantidamente pausado o *Stretcher* gera um sinal de reconhecimento *AS*↑. Em função disto, a porta gera dois sinais de saída: uma requisição assíncrona *AR*↑ e um reconhecimento síncrono através de uma borda em *SA*. A geração do sinal *SA* ocorre neste estado para que este respeito o tempo de *setup* quando o sinal de relógio do transmissor voltar a funcionar.

Quando a requisição assíncrona gerada pela porta de saída é processada pela porta de entrada, esta sai do seu estado inicial e solicita a extensão do sinal de relógio através do sinal *RS*↑ ao *Stretcher*.

A porta de entrada quando recebe um reconhecimento *AS*↑, sinalizando que o sinal de relógio está parado, gera um sinal de reconhecimento assíncrono *AA*↑ e de requisição síncrona *SR* para que este esteja estável quando o sinal de relógio do receptor voltar a funcionar. Após esta seqüência de eventos a porta de entrada permanece no estado 2 esperando *AR*↓.

A porta de saída ao perceber o sinal *AA*↑ gera *AR*↓ e permanece esperando no estado 3 até que o sinal de reconhecimento volte ao estado inicial.

A porta de entrada ao perceber o sinal *AR*↓, solicita ao *Stretcher* que o sinal de relógio volte a funcionar *Rs*↓.

Quando o sinal de relógio retorna ao funcionamento, o sinal de reconhecimento *As*↓ é gerado pelo *Stretcher*. Quando a porta de entrada percebe este sinal de reconhecimento, passa para o estado 4 e fica esperando que a ilha síncrona processe o sinal de requisição síncrona, amostre os dados e gere o sinal de reconhecimento síncrono através de uma borda em SA.

Quando a porta de entrada percebe o sinal de reconhecimento síncrono, esta gera o sinal de reconhecimento assíncrono. O sinal de reconhecimento síncrono pode ter um elemento de atraso adicionado para garantir que o tempo de *hold* seja atendido. Na prática isto não é necessário porque o tempo de propagação do circuito das portas faz com que o tempo de *hold* seja alcançado antes do sinal de reconhecimento assíncrono ser inicializado.

A porta de entrada quando percebe que o sinal de reconhecimento assíncrono volta ao seu estado inicial solicita ao *Stretcher* que o sinal de relógio volte a oscilar, *Rs*↓. Quando o *Stretcher* responde, As↓, é finalizado o processo de transmissão e a porta de saída fica esperando a solicitação de transmissão um novo dado.

6.3 SCAFFI Trilha Dupla

Quando um SoC é composto de diversos módulos que utilizam comunicação ponto a ponto pode ocorrer que dois módulos interligados estejam distantes no chip. Quando isto acontece, se torna mais difícil de conseguir garantir em FPGAs a restrição imposta pelo modo *bundled data* de que o sinal de requisição tem um atraso maior que todos os sinais do canal de dados, principalmente quando o canal de dados é de grande largura.

Um meio de evitar esta requisição é utilizar um canal de comunicação onde os dados utilizam codificação insensível a atraso. A biblioteca de interfaces proposta possui componentes que permitem a conversão de uma comunicação *bundled data* para *dual*. Para isso conversores *single to dual* e *dual to single* podem ser inseridos junto às interfaces, como mostra a Figura 48, restringindo o pressuposto de temporização para a comunicação entre a interface e os conversores.

Os componentes DI da biblioteca permitem também a comunicação entre um sistema síncrono e um sistema QDI dual rail.

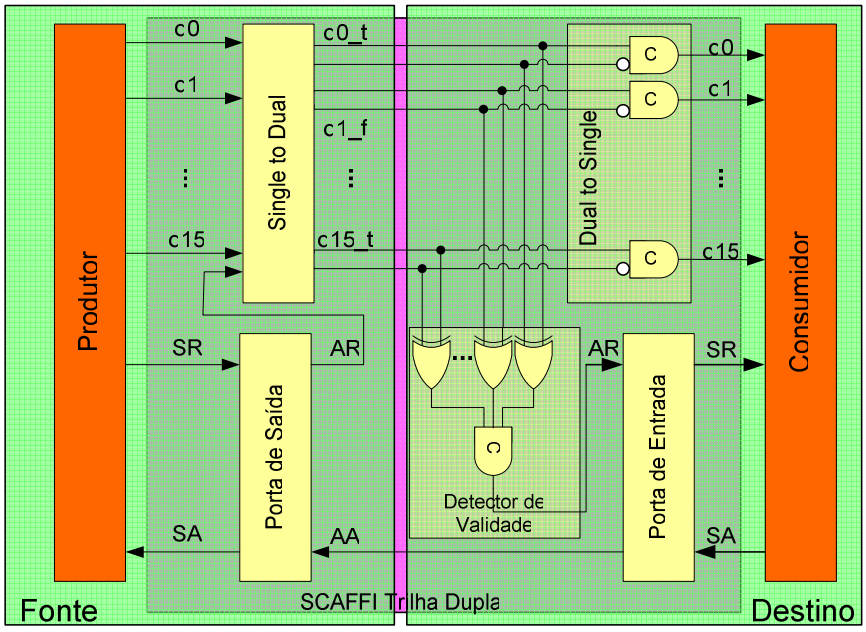

Figura 48 – Estrutura da versão trilha dupla da interface SCAFFI.

6.4 Caso de Uso

Nesta Seção um novo modelo de implementação do algoritmo de criptografia RSA é apresentado. A principal característica dessa implementação é seu potencial para redução da dissipação de potência e economia de energia.

O RSA consiste em uma operação de exponenciação modular. A operação de exponenciação modular, entretanto é computada como uma seqüência de multiplicações modulares.

A multiplicação modular é o módulo que mais executa e possui o caminho crítico do RSA. O módulo de exponenciação modular é responsável pelo controle de execução da multiplicação.

A implementação proposta divide o RSA em dois módulos - exponenciação modular e multiplicação modular – onde cada um opera em uma freqüência diferente. Durante a execução da multiplicação o módulo da exponenciação tem o seu relógio parado para diminuir o consumo de potência.

A implementação do GALS RSA segue o modelo apresentado na Figura 42, onde o *Sender* corresponde à Exponenciação Modular e o módulo *Receiver* corresponde à Multiplicação Modular.

A implementação GALS do RSA é comparada a uma implementação síncrona ambas com chaves de 128 bits. A Tabela 14 descreve a máxima freqüência de operação obtidas dos relatórios de temporização para o FPGA XC3S200 e o número de elementos de memória que são sincronizados a partir de cada relógio (*clock load*).

Tabela 14 – Freqüência máxima de operação e carregamento de cada sinal de relógio.

	Freqüência(MHz)	Carga do sinal de Relógio
Síncrono	41.748	1368
Exponenciação GALS	157.953	694
Multiplicação GALS	45.335	674

As duas abordagens foram validadas através de simulação SDF e prototipadas e verificadas em um FPGA XC3S200 da família Spartan 3 da Xilinx.

6.5 Resultados

A Tabela 15 mostra os resultados de ocupação de área e consumo de potência de cada abordagem. O consumo de potência foi extraído através de medições seguindo o modelo proposto em [BEC03]. Para a medição de potência foram comparadas à versão síncrona trabalhando na aproximadamente na sua freqüência máxima 40 MHz e a versão GALS com o módulo de multiplicação modular trabalhando aproximadamente na sua freqüência máxima 40MHz e o módulo de exponenciação modular trabalhando na freqüência de 72MHz. Este valor de freqüência da exponenciação foi escolhido por representar um valor ligeiramente superior a freqüência onde o desempenho da proposta GALS iguala o desempenho da versão síncrona. A partir deste valor de freqüência o sistema GALS possui um desempenho superior à versão síncrona, porém, este desempenho não chega a ser significativo uma vez que 98% do tempo de execução é destinada a multiplicação modular.

Tabela 15 – Resultados de consumo de área e dissipação de potência para implementações síncronas e GALS da aplicação RSA.

Abordagem	LUTs	Flip Flops	Portas Equivalentes	Potência (mW)
Síncrono	1521	1367	21294	27.07
GALS	1562	1367	21549	14.48

Os resultados mostram um acréscimo quase desprezível no consumo de área e uma redução acima de 46% no consumo de potência da versão GALS em relação à versão síncrona.

6.6 Conclusões do Capítulo

A interface SCAFFI provê um mecanismo de sincronização baseado na extensão do sinal de relógio, permitindo dessa forma a comunicação de módulos com domínios distintos de relógio em FPGAs. O desenvolvimento de sistemas que utilizam a interface SCAFFI pode ser feito utilizando a biblioteca de *hard macros* proposta no Capítulo 4. A validação da interface foi feita através de uma aplicação de criptografia RSA. Este módulo de criptografia foi particionado em dois sub-módulos, cada um com um domínio de freqüência. Medições demonstraram a redução de potência do sistema GALS quando comparado com a mesma implementação utilizando um sistema síncrono.

7 REDES INTRA-CHIP COM SUPORTE A SISTEMAS GALS

Este Capítulo descreve o desenvolvimento das redes intrachip Hermes GALS (Hermes-G) e Hermes GALS Low Power (Hermes-GLP) [PON08]. A NoC Hermes-G consiste de uma adaptação da rede intrachip Hermes para que esta possa suportar o desenvolvimento de SoCs GALS. A rede Hermes-GLP, além de suportar sistemas GALS, implementa mecanismos de controle de atividade nos roteadores visando a redução da dissipação de potência.

7.1 Hermes GALS

A rede intrachip Hermes possui: topologia de malha bi-dimensional, roteadores de até cinco portas, roteamento distribuído XY e chaveamento *wormhole* de comutação de pacotes.

A NoC Hermes é composta por 3 módulos principais: filas de entrada, módulos de controle de roteamento (*switchcontrol*), e um *crossbar*. A estrutura geral da NoC Hermes é mostrada na Figura 49. Os roteadores podem possuir entre três e cinco portas, dependendo da posição dos roteadores na NoC, cinco para roteadores no centro da NoC e três para os roteadores das pontas.

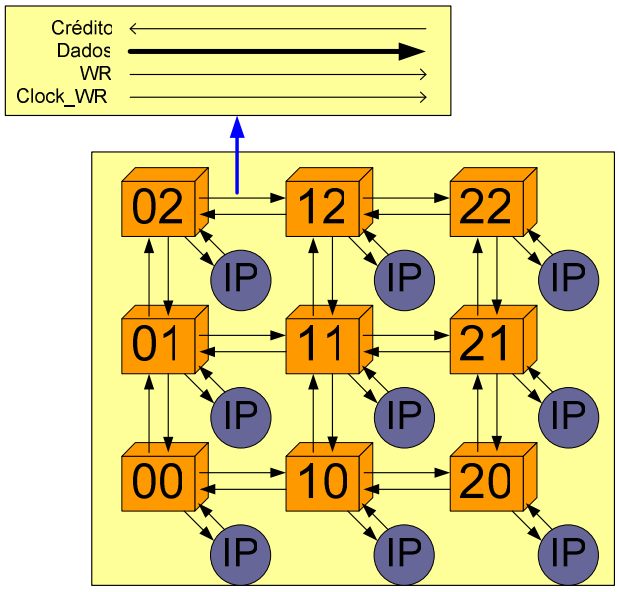

Figura 49 – Estrutura geral de uma rede intrachip Hermes 3x3. Cada roteador possui um endereço XY associado.

A rede Hermes emprega roteamento distribuído, isso é, em cada roteador é realizada uma parte do algoritmo de roteamento para cada pacote, em função do endereço destino e do endereço do roteador atual do pacote. O algoritmo usado nesse trabalho é o algoritmo XY, onde, um pacote percorre toda a sua rota primeiramente no eixo X da NoC (eixo horizontal da Figura 49), para depois percorrer a sua rota em Y. Este algoritmo é determinístico, isso é, dado um endereço destino e o endereço atual do roteador existe um único caminho através do qual o pacote pode ser roteado.

A estrutura adotada para o pacote da rede Hermes é mostrada na Figura 50. O primeiro *flit* do pacote contém o endereço do destino desse pacote. O segundo *flit* possui o tamanho do *payload* do pacote. O restante dos *flits* faz parte do *payload* do pacote.

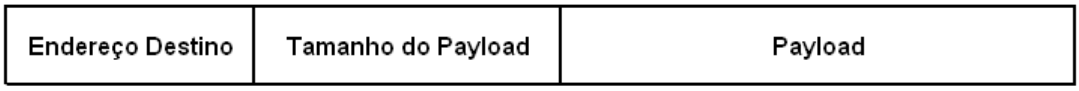

Figura 50 – Formato do pacote da rede Hermes.

Para permitir a comunicação entre módulos com freqüência de operação distintas a fila de entrada da NoC Hermes foi substituída. A fila proposta acrescenta um mecanismo de sincronização baseado na fila bi-síncrona abordada na Seção 5.1.3 e mostrada na Figura 51. Essas filas permitem que as escritas nas portas de entrada utilizem um sinal de relógio diferente do sinal de relógio de leitura que é utilizado internamente pelo roteador. Cada roteador transmite o seu sinal de relógio para a porta de entrada dos roteadores vizinhos.

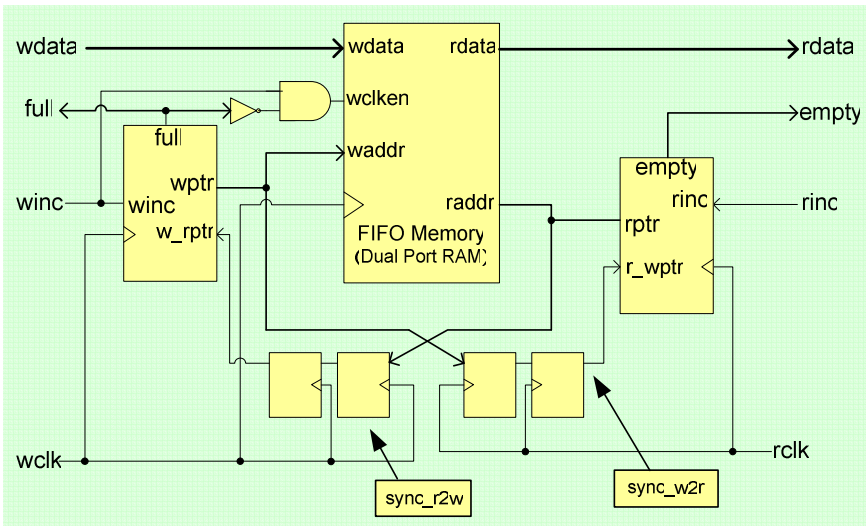

Figura 51 – Estrutura da fila bi-síncrona adicionada à rede Hermes para sincronização dos dados. [CUM02].

A fila bi-síncrona foi escolhida devido a sua capacidade de permitir à escrita e a leitura de dados em um ciclo do relógio. A fila bi-síncrona da HERMES-G consome mais área que a fila da NoC HERMES original, isso acarreta em um aumento de área nos roteadores

A inserção do mecanismo de sincronização nas portas de entrada dos roteadores permite que os roteadores funcionem cada um com uma freqüência distinta. Entretanto, dessa forma cada roteador deve trabalhar de forma sincronizada com o módulo IP associado. Essa estratégia fragmenta a árvore de relógio global em diversas árvores locais, responsáveis por distribuir o sinal de relógio em cada ilha síncrona, que compreende o roteador e seu núcleo IP. Esta fragmentação possibilita a eliminação de *buffers* grandes, reduzindo a dissipação de potência do sistema. Além disso, essa abordagem facilita o controle do escorregamento do sinal de relógio.

O principal problema da abordagem é a correlação da freqüência de operação do roteador

com o elemento de processamento, considerando que a máxima freqüência de operação dos roteadores costuma ser maior que dos núcleos IP associados. Isso torna a comunicação fortemente vinculada à computação e não explora plenamente a capacidade da rede intrachip.

Para desacoplar a freqüência de operação do roteador do núcleo IP é necessária a adição de uma fila para fazer a sincronização dos dados que chegam ao núcleo IP.

Além do mecanismo de sincronização, o *buffer* de entrada da NoC Hermes-G possui também um mecanismo de controle que solicita o roteamento de um pacote, quando o primeiro *flit* de um pacote é detectado. A detecção do primeiro *flit* ocorre quando uma porta de entrada que está no estado inativo percebe que a sua fila não está vazia. Quando isso acontece, a porta de entrada solicita o roteamento e o dado que está na primeira posição da fila é utilizado como endereço de destino. Quando o módulo *switchcontrol* confirma o roteamento do pacote, a porta de entrada incrementa o ponteiro de leitura e registra o tamanho do pacote. O estado seguinte realiza a transmissão do restante do pacote até que o tamanho seja alcançado. Quando isso ocorre, a porta entra em um estado onde solicita ao *switchcontrol* a liberação da porta de saída alocada para a transmissão.

A funcionalidade da rede intrachip Hermes foi validada através de simulação comportamental e simulação com temporização SDF. A próxima Seção apresenta dados de latência referentes à simulação da NoC Hermes-G.

7.1.1 Avaliação da NoC Hermes-G

Esta Seção apresenta alguns resultados referentes a simulações de uma rede Hermes-G 3x3. As simulações foram realizadas utilizando a ferramenta ModelSim. Os resultados mostrados foram obtidos de simulações funcionais. As simulações utilizaram um modelo em VHDL RTL da Hermes-G e um modelo SystemC para os geradores de tráfego e para módulos receptores, que juntos desempenham o papel de núcleos IP.

Cada gerador de tráfego é um módulo SystemC que opera em uma freqüência independente. O tráfego é gerado a partir da ferramenta ATLAS, a qual produz arquivos de texto que especificam o tempo da simulação em que o pacote deve ser inserido na rede, o endereço de destino, e o endereço de origem. Cada gerador de tráfego possui um arquivo contendo todos os pacotes que este deve inserir na rede.

Quando um módulo gerador de tráfego inicia a transmissão de um pacote, anexa ao pacote o tempo em que este pacote começou a ser transmitida uma informação denominada *timestamp*. O tempo é determinado como uma quantidade de pulsos de um sinal relógio de referência, utilizado somente pelos módulos SystemC como referência temporal, e não existe na descrição da NoC.

Além dos módulos de geração de tráfego, responsáveis por transmitir pacotes, são

adicionados também módulos SystemC responsáveis pela recepção dos pacotes. Os módulos receptores são responsáveis por receber os pacotes da rede e gerar dados de temporização dos tempos de chegada e a latência do pacote na NoC. Neste trabalho latência é a diferença entre o tempo de saída da NoC do último *flit* do pacote e o tempo de entrada do primeiro *flit* do pacote na rede. Os dois tempos são gerados em função do mesmo relógio de referência global.

Nas simulações, cada par módulo gerador-módulo receptor em uma posição da rede representa um núcleo IP. Cada um destes compõe uma ilha de freqüência, isso é, trabalham com o mesmo sinal de relógio. Adicionalmente, cada roteador também constitui uma ilha de freqüência. Entretanto, não existe sincronização entre os pares em posições diferentes da NoC. No restante desse trabalho, o termo de gerador de tráfego denotará o par responsável pela inserção e remoção de pacotes na rede.

Diversos cenários foram testados. Estes cenários variaram em função da freqüência de operação dos módulos geradores de tráfego, da freqüência de operação dos roteadores, tamanho das filas e número de *flits* dos pacotes. Nestes cenários, cada produtor envia dados para exatamente um consumidor. Os dados foram retirados de uma comunicação entre dois geradores de tráfegos distanciados por cinco roteadores em uma rede 3x3 (Endereço do Produtor = 22; Endereço do Consumidor = 00). O tamanho dos pacotes é de 100 *flits* e os *buffers* possuem capacidade para o armazenamento de 16 *flits*.

A Figura 52 mostra a variação da latência média dos pacotes na rede em função da freqüência de operação do produtor, quando a freqüência de operação do consumidor é fixada em 200MHz.

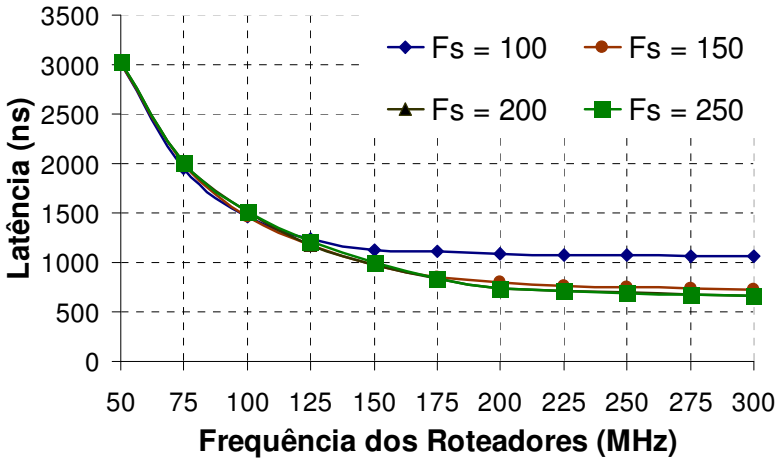

Figura 52 - Latência média em função da freqüência de operação dos roteadores. Fr = 200MHz.

A Figura 53 mostra a latência dos pacotes na rede em função da freqüência de operação do consumidor de dados, quando a freqüência de operação do produtor é fixada em 200MHz.

Os gráficos das duas Figuras mostram que a freqüência de operação dos roteadores influencia a latência de transmissão dos pacotes somente até um valor próximo do mínimo entre a

freqüência do produtor e a do consumidor. A partir desse valor, a latência estabiliza e não existe mais ganho de desempenho possível.

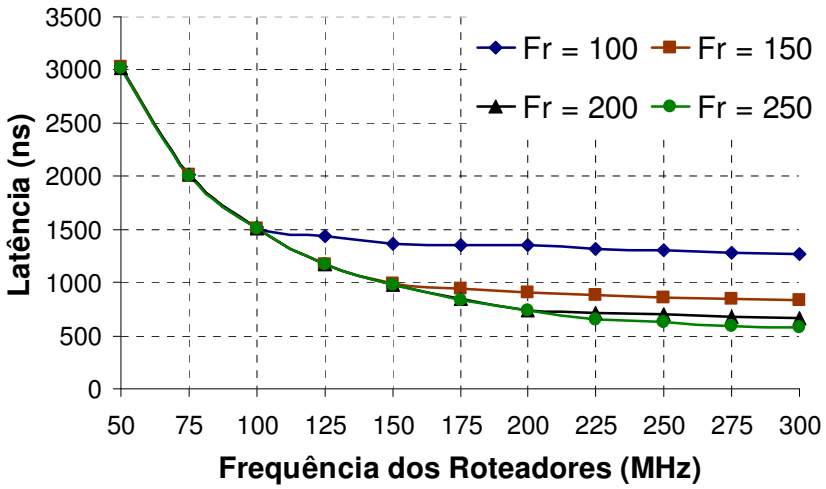

Figura 53 - Latência média em função da freqüência de operação dos roteadores. Fs = 200MHz.

7.2 Hermes GALS Low Power

As NoCs que dão suporte a sistemas GALS revisadas no Capítulo 3 permitem a redução da árvore de relógio, reduzindo assim a dissipação de potência. Entretanto, sistemas GALS podem ser aproveitados para a implementação de técnicas de controle de potência baseadas no controle dinâmico da freqüência de operação de cada ilha síncrona. Além disso, uma vez que a freqüência de operação pode ser controlada, mecanismos de controle dinâmico da tensão de alimentação também podem ser implementados. O controle de tensão depende do controle de freqüência, uma vez que ao reduzir a tensão de alimentação o tempo de propagação das portas é aumentado. Conseqüentemente, a máxima freqüência de operação é reduzida.

A Equação 2 descreve como calcular a dissipação de potência dinâmica média de um circuito síncrono composto de *n* portas lógicas, onde: C_i é a carga capacitiva da saída de cada porta, V_{dd} é a tensão de alimentação, T_c é o período do sinal de relógio e $P_t(x_i)$ é a probabilidade de transição da saída de uma porta quando uma alteração ocorre nas suas entradas.

$$P_{av} = \frac{1}{2T_c} V_{dd}^{\,2} \sum_{i=1}^{n} C_i P_t(x_i) \qquad (2)$$

Segundo essa Equação, a dissipação de potência dinâmica de uma porta lógica é proporcional ao quadrado da tensão de alimentação desta porta. Assim, a aplicação de técnicas de controle dinâmico de freqüência (em inglês *dynamic frequency scaling*, DFS) e controle dinâmico da tensão de alimentação (em inglês *dynamic voltage scaling*, DVS) podem reduzir consideravelmente a dissipação de potência de um SoC.

A próxima Seção faz uma breve revisão do estado da arte de propostas de controle de

potência em SoCs que usam NoCs, enquanto a Seção 7.2.2 apresenta a inserção de mecanismos de controle de potência dinâmica na rede intrachip Hermes-G, gerando a rede Hermes-GLP.

7.2.1 Propostas de Controle de Potência em SoCs com NoCs GALS

As propostas de redes intrachip que dão suporte a sistemas GALS descritas no Capítulo 3 não aproveitam o potencial dos sistemas GALS para o controle dinâmico de freqüência das ilhas síncronas. A implementação de tais controles é possível devido à possibilidade de cada ilha síncrona funcionar com freqüência totalmente distinta das demais ilhas. Desta forma, a freqüência de uma ilha pode ser alterada sem influenciar a freqüência de operação das demais. Este tipo de mecanismo é possível de ser implementado em todos os tipos de sistemas de comunicação GALS: ponto a ponto, barramento e NoCs. Esta Seção revisa as propostas de NoCs e/ou SoCs que habilitam esse tipo de controle.

O trabalho de Hsu et al. [HSU05] apresenta uma proposta de chaveamento do sinal de relógio (*frequency scaling low-power* - FSLP) para controlar a potência consumida em um SoC que utiliza uma rede intrachip como meio de comunicação. Para validar a abordagem uma aplicação de MPEG é prototipada em FPGA. Segundo os Autores, a abordagem proposta apresenta uma redução de aproximadamente 30% no consumo de potência. O controle dinâmico da freqüência é aplicado aos núcleos IP conectados à rede intrachip, e não aos roteadores. A freqüência de operação dos núcleos IP é alterada em função das taxas de comunicação nas portas de entrada e saída do roteador e nos requerimentos de comunicação.

Simunic et al. [SIM04] propõem um método semelhante ao de Hsu et al. Na abordagem dos primeiros, uma malha fechada de controle é responsável por determinar a melhor freqüência de operação dos núcleos IP conectados a rede intrachip com base nos requisitos de potência e QoS de cada núcleo IP.

O trabalho de Worm et al. [WOR05] propõe um mecanismo de auto calibração para empregar DVS em redes intrachip procurando manter um nível tolerável da taxa de erros, do inglês *bit error rate* (BER). O mecanismo proposto utiliza módulos que realizam a detecção de erro em cada palavra transmitida e solicitam a retransmissão quando ocorre um erro na transmissão. Entretanto, mecanismos de retransmissão podem gerar uma latência muito grande em um pacote e desta forma violar as restrições de QoS na comunicação.

O trabalho de Ogras et al. [OGR07] propõe um fluxo de particionamento de SoCs GALS que utilizam redes intrachip em ilhas de domínio de tensão e freqüência (VFIs). O principal objetivo é criar ilhas compostas de diversos roteadores e módulos IPs capazes de operar na mesma freqüência e tensão. Nesse trabalho, as ilhas são estáticas e não existe nenhum mecanismo de adaptação da rede intrachip a transmissão dos pacotes.

7.2.2 Mecanismos de controle de potência da NoC Hermes-GLP

A proposta de controle de freqüência apresentada nessa Seção se diferencia dos demais por oferecer uma estrutura de rede intrachip capaz de suportar o desenvolvimento de sistemas GALS e ainda oferecer mecanismos dinâmicos de redução de potência aplicados aos roteadores da NoC. A abordagem apresentada aqui cria dinamicamente ilhas de freqüência em função da prioridade de cada pacote. Os mecanismos de controle de potência adotados possuem baixa complexidade e, portanto não adicionam uma quantidade significativa de área, como será mostrado.

O primeiro mecanismo de controle implementado foi o de *clock gating*. O mecanismo de *clock gating* consiste em desabilitar o sinal de relógio de regiões inativas do sistema. O circuito utilizado para a implementação de *clock gating* é mostrada na Figura 54. Esse mecanismo pode ser aplicado em diferentes níveis de granularidade [PED05] [STR00]. Na Hermes-GLP o uso de *clock gating* é feito no nível de roteador. O roteador é considerado ocioso quando todas as suas portas estão ociosas, isto é, nenhuma possui pacote para transmitir. Para implementar a técnica de *clock gating*, foi inserido um módulo de controle do sinal de relógio no roteador.

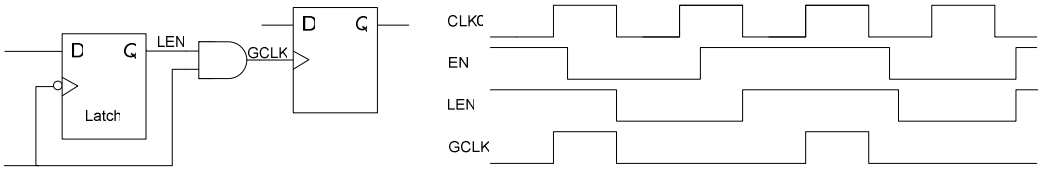

Figura 54 – Circuito e exemplo de funcionalidade para utilizar técnicas de *clock gating* [VLS08].

Todas as portas do roteador enviam para o *clock control* um sinal informando o seu estado. Quando todas as portas indicam que não possuem dados para transmitir o *clock control* para o sinal de relógio do roteador. O roteador permanece nesse estado até que uma escrita seja realizada em uma de suas portas, tirando essa porta do estado ocioso. Esse sistema é possível devido ao fato de que cada porta possui um sinal de relógio de escrita diferente, mas compartilham o mesmo relógio para leitura. O sinal de relógio de leitura também é utilizado internamente pelo roteador. A Figura 57 mostra a estrutura de controle do roteador da NoC Hermes-GLP.

Além de *clock gating* a Hermes-GLP implementa também o controle dinâmico da freqüência do roteador DFS. Nesse trabalho o mecanismo de DFS foi implementado através da seleção entre dois ou mais sinais de relógio em tempo de execução, porém é possível a adoção de sistemas mais elaborados de geração de relógio como em [PON07a]. A Figura 55 mostra o circuito utilizado para a implementação de *clock switching*. Esse circuito garante que o chaveamento entre os sinais de relógio é livre de *glitch*, restrição necessária para que essa seleção seja feita de forma robusta.

O uso dessa técnica permite que os roteadores pertencentes ao canal de comunicação estabelecido entre dois núcleos IP se adaptem à freqüência de operação requerida pelo pacote. Para

isso cada pacote deve sinalizar qual a freqüência com a qual deve ser transmitido. Este sinal é roteado juntamente com o pacote. Cada roteador recebe uma indicação de freqüência de cada porta de entrada e seleciona a maior freqüência das indicadas. Desta forma a transmissão obedece às restrições de latência de todos os pacotes e fornece um mecanismo de controle simples e que requer uma quantidade pequena de hardware adicional.

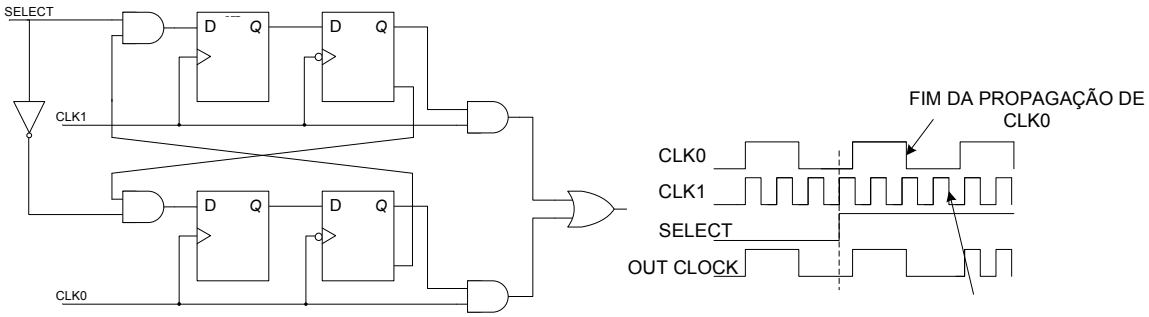

Figura 55 - Circuito que realiza a seleção entre dois sinais de relógio e exemplo de sua funcionalidade [MAH03].

A Figura 56 mostra a estrutura geral do roteador da NoC Hermes-GLP.

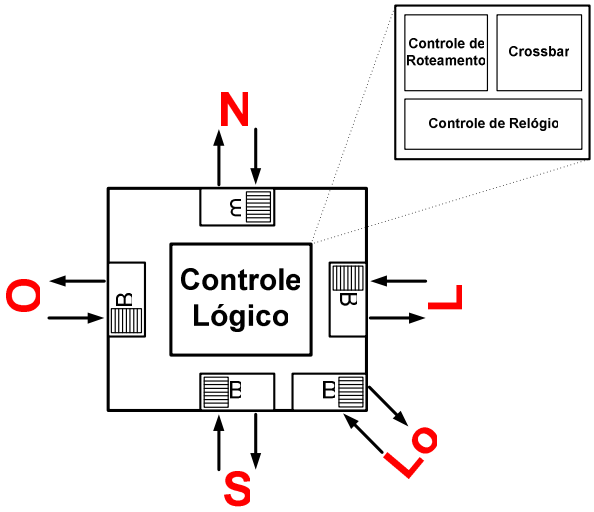

Figura 56 – Estrutura do roteador da NoC Hermes-GLP.

A Figura 57 mostra a estrutura do controle de freqüência do roteador da Hermes-GLP. Cada canal de comunicação possui, além dos sinais mostrados na Figura 49, dois sinais de controle *sideband*: *Sel_clk_in* e *Sel_clk_out*. O sinal *Sel_clk_in* é usado pelo módulo de *Clk_Ctrl* para o controle dinâmico da freqüência de operação (DFS). Este sinal é roteado para o sinal *Sel_clk_out* juntamente com o pacote. O sinal *Port_State* sinaliza o estado das portas e é utilizado para realizar o *clock gating* do roteador, e para determinar qual os sinais *Sel_clk_in* que o roteador deve utilizar para o *clock switching*, uma vez que são utilizados somente os *Sel_clk_in* das portas ativas.

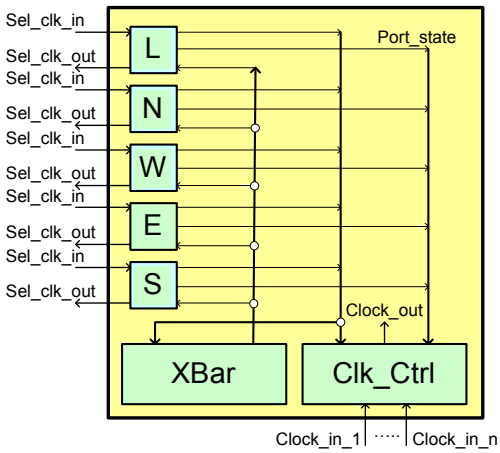

Figura 57 – Estrutura do controle de relógio do roteador da Hermes-GLP.

A Figura 58 mostra as formas de onda da simulação do controle de relógio com duas fontes de relógio.

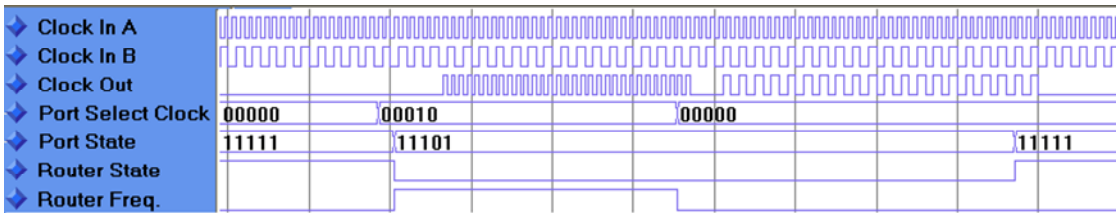

Figura 58 – Simulação do controlador de relógio.

A Figura 59 ilustra o controle de relógio dos roteadores em tempo de execução.

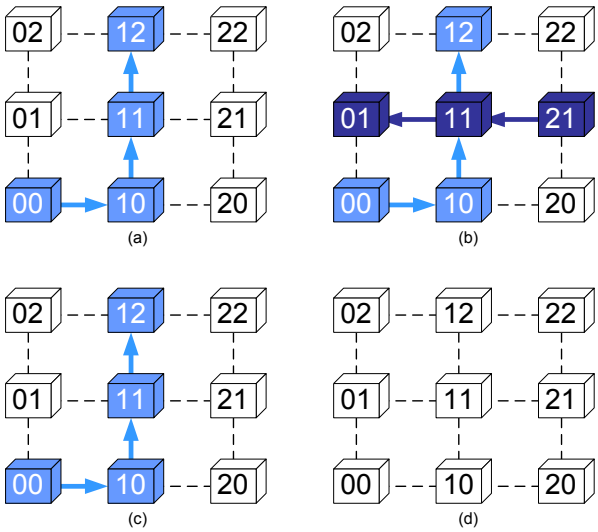

Figura 59 – Exemplo de controle de freqüência do sinal de relógio em uma NoC Hermes-GLP 3x3. Os roteadores mais escuros possuem freqüência de operação mais elevada.

No primeiro caso (a), a rede intrachip inicia com uma transmissão de baixa prioridade criando uma ilha de baixa freqüência $IF_1 = \{00, 10, 11, 12\}$. O segundo estado da NoC (b) mostra uma requisição com prioridade mais alta sendo iniciada, criando uma segunda ilha de freqüência $IF_2 = \{21, 11, 01\}$. Isto faz com que os roteadores que pertencem à intersecção dos dois caminhos de comunicação ($IF_1 \cap IF_2 = \{11\}$) funcionem na freqüência mais elevada. Tal situação permite que os requisitos de comunicação de mais alta prioridade sejam alcançados. Essa alteração não introduz nenhuma latência e não interfere funcionalmente na comunicação de baixa prioridade. Nesse

exemplo, a transmissão de maior freqüência é a primeira finalizada (c), fazendo com que os o roteador 11 reduza a sua freqüência de operação. Quando essa comunicação também é encerrada (d) os roteadores ficam ociosos e seus relógios são pausados.

7.2.3 Avaliação Comparativa das NoCs Hermes-GLP e Hermes-G

A Seção 7.1.1 mostrou a dependência da latência em função da freqüência de operação dos roteadores. Nessas simulações foi possível verificar que existe uma redução de latência somente até o momento em que a freqüência dos roteadores alcança *min* (F_p, F_c). Para freqüências superiores a este valor a latência se estabiliza. Com base nessa constatação, foi desenvolvido um mecanismo nos geradores de tráfego para que estes possam determinar a prioridade do pacote em função da freqüência dos módulos produtores e consumidores. Estes geradores de tráfego foram utilizados para as simulações utilizando a NoC Hermes-GLP. Não foi possível ao longo deste trabalho implementar montagens capazes de habilitar medidas físicas de potência das NoCs propostas. Assim, propõe-se aqui uma técnica para estimar os ganhos em termos de redução de dissipação de potência destas NoCs. A técnica é baseada no conceito de *taxa de ativação* definido a seguir.

A *taxa de ativação* instantânea de um roteador $A_x(t)$ é definida pela Equação 3.

$$A_x(t) = F_{op}(t) \Big/ F_{\max} \tag{3}$$

Nesta Equação, $F_{op}(t)$ é a freqüência de operação do roteador no instante *t,* e F_{max} é a maior freqüência de operação que o roteador pode selecionar.

As simulações na Hermes-GLP foram efetuadas com o objetivo de verificar a redução na taxa de ativação dos roteadores, e conseqüentemente da NoC. Para que a taxa de ativação de cada roteador pudesse ser especificada, foi desenvolvido um módulo em SystemC que fica atualizando a cada 1 ns o estado do roteador, que pode estar em situação de *clock gating* ou operando em uma de várias freqüências. O estado de cada roteador é registrado em um arquivo de texto que é utilizado após a simulação para determinar a taxa de ativação média de cada roteador. A taxa de ativação média de cada roteador é determinada a partir da Equação 4, onde n é o número total de atualizações registradas pelo módulo Observador durante a simulação e A_x é dado pela Equação 3.

$$A_{AV}(n) = \frac{\sum_{i=0}^{n} A_x(i)}{n} \tag{4}$$

A Equação 5 mostra como é calculada a taxa média de ativação da NoC, onde r denota o número de roteadores da NoC e A_{av} é dado pela Equação 4. A taxa de ativação média da NoC é calculada como a média das taxas dos roteadores.

$$A_{NoC} = \frac{\sum_{k=1}^{r} A_{av}(k)}{r} \quad (5)$$

A Tabela 16 e a Tabela 17 apresentam dois cenários de testes, (a) e (b) respectivamente, compostos pelos mesmos núcleos IP, mas em cada cenário o posicionamento desses núcleos IP é alterado. Esses cenários forma escolhidos para demonstrar o potencial de redução da taxa de ativação dos roteadores da NoC Hermes-GLP e a dependência da taxa de ativação média em função do posicionamento dos núcleos IP na NoC.

Tabela 16 – Cenário (a) de pares produtor-consumidor utilizados nas simulações.

Tráfego	Endereço Fonte	Endereço Destino	Freqüência Fonte (MHz)	Freqüência Destino (MHz)	Prioridade
T1	02	20	200	170	Alta
T2	22	00	120	150	Alta
T3	12	21	90	70	Baixa
T4	20	12	170	90	Baixa
T5	01	11	50	180	Baixa
T6	21	02	70	200	Baixa

Tabela 17 – Cenário (b) de pares produtor-consumidor utilizados nas simulações.

Tráfego	Endereço Fonte	Endereço Destino	Freqüência Fonte (MHz)	Freqüência Destino (MHz)	Prioridade
T1	02	22	200	170	Alta
T2	01	21	120	150	Alta
T3	20	00	90	70	Baixa
T4	22	20	170	90	Baixa
T5	12	10	50	180	Baixa
T6	00	02	70	200	Baixa

A Figura 60 apresenta a taxa média de ativação de cada roteador para o cenário (a) (Tabela 16). A taxa de ativação de todos os roteadores possui um ponto de saturação entre 75% e 80%, essa saturação corresponde ao ponto de saturação da NoC para esse tráfego.

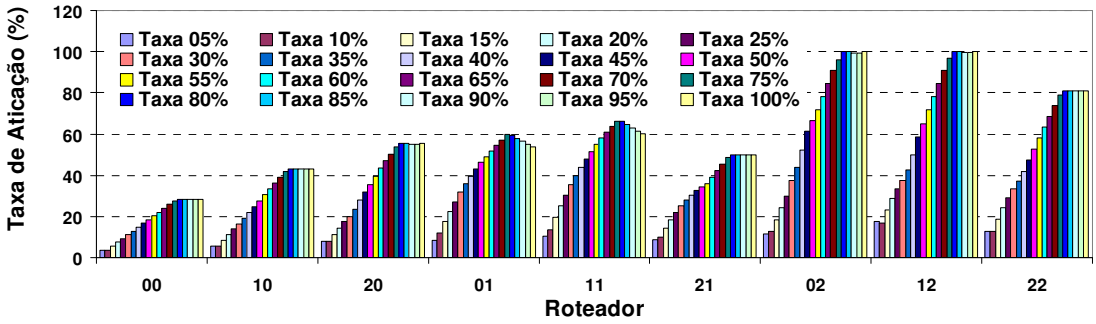

Figura 60 – Taxa de ativação por roteador do cenário (a) (Tabela 16), em função da taxa de injeção de dados.

A Figura 61 mostra a comparação dos dois cenários da Tabela 16 e da Tabela 17 (a e b). Os dois mapeamentos possuem taxas de ativação média distintas, sendo que o mapeamento (b) apresenta melhor aproveitamento dos recursos de controle de potência da NoC. Essa Figura mostra também que mesmo para taxas elevadas de injeção de tráfego, os mecanismos de controle de potência ainda provêem uma redução significativa da taxa de ativação da NoC. Um exemplo de redução que pode ser visualizado no gráfico é para a taxa de inserção de 50%, onde a NoC apresenta uma taxa de ativação inferior a 50% para os dois mapeamentos.

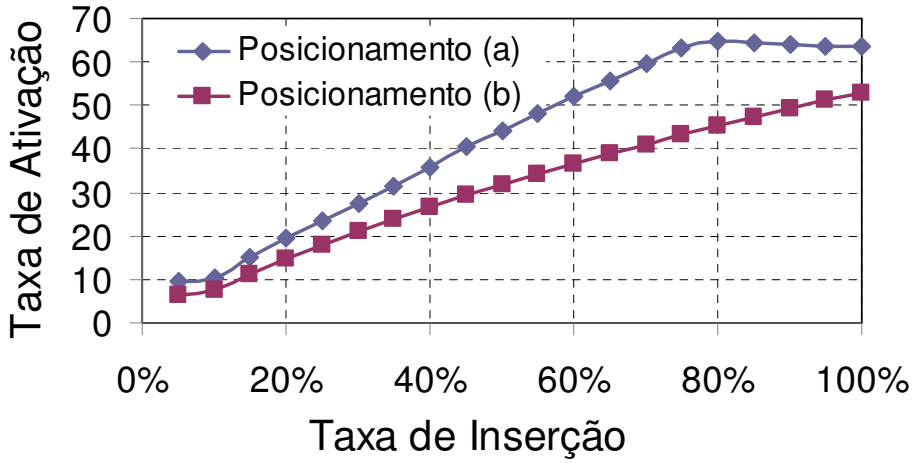

Figura 61 – Comparação da taxa de ativação dos mapeamentos da Tabela 16 e da Tabela 17, em função das taxas de inserção.

O tráfego (a) da Tabela 16 foi simulado também na NoC Hermes-G, onde os roteadores operam com freqüência de 200MHz. A latência média da NoC Hermes-G é mostrada na Tabela 18. Por outro lado, a Tabela 19 mostra a latência média da NoC Hermes-GLP. Ambas as tabelas mostram os resultados de latência média para o cenário (a) (Tabela 16).

Tabela 18 – Latência média dos tráfegos do cenário (a) descrito Tabela 16 para a NoC Hermes-G.

	Latência Média Hermes-G (ns)					
Taxa	T1	T2	T3	T4	T5	T6
100%	758.08	903.40	1740.49	1354.50	2061.12	1489.03
95%	757.94	903.03	1739.65	1354.50	2062.37	1488.99
90%	757.93	902.78	1739.65	1354.50	2060.75	1488.99
85%	757.90	902.38	1739.65	1354.50	2060.85	1489.02
80%	758.04	902.74	1739.65	1354.50	2061.10	1489.00
75%	740.37	903.14	1598.42	1354.50	2061.00	1488.95
70%	740.60	902.85	1597.38	1354.50	2062.36	1488.79
65%	740.19	902.90	1597.39	1354.53	2060.98	1488.88
60%	740.43	902.81	1598.01	1354.50	2061.06	1489.07
55%	739.29	902.94	1597.71	1354.50	2061.19	1488.83
50%	741.04	902.62	1596.33	1266.69	2061.12	1488.80
45%	739.18	902.89	1597.62	1266.70	2061.08	1488.99
40%	741.55	902.92	1597.97	1266.64	2061.12	1489.00
35%	740.27	902.93	1597.48	1266.36	2062.50	1488.88
30%	739.83	902.93	1597.62	1267.35	2062.08	1489.02
25%	741.08	903.03	1596.70	1265.96	2061.10	1488.80
20%	741.14	902.98	1596.67	1266.15	2061.23	1489.00
15%	739.56	903.25	1597.54	1265.90	2061.52	1489.02
10%	741.06	903.63	1596.63	1266.22	2061.48	1489.00
5%	741.02	903.99	1579.17	1265.86	2061.48	1468.02

Tabela 19 – Latência média dos tráfegos do cenário (a) descrito Tabela 16 para a NoC Hermes-GLP.

	Latência Média Hermes-GLP (ns)					
Taxa	T1	T2	T3	T4	T5	T6
100%	753.27	905.42	1781.19	1412.77	2090.82	1571.08
95%	752.57	909.86	1781.62	1411.47	2091.43	1565.88
90%	753.36	910.07	1782.58	1412.13	2092.76	1568.81
85%	752.83	911.85	1780.07	1410.74	2092.18	1570.23
80%	752.3	909.54	1781.85	1411.78	2093.18	1573.06
75%	745.36	909.22	1781.17	1412.25	2094.84	1573.91
70%	749.25	911.09	1738.75	1409.93	2093.83	1576.55
65%	749.42	908.98	1741.01	1417.18	2094.70	1577.44
60%	749.42	908.94	1740.23	1416.78	2094.31	1581.38
55%	749.50	908.83	1743.23	1418.78	2094.68	1581.79
50%	750.82	908.92	1742.80	1417.43	2098.76	1581.81
45%	749.45	908.77	1745.42	1356.25	2099.37	1586.55
40%	750.94	908.74	1748.03	1359.84	2099.29	1587.01
35%	751.86	909.24	1748.41	1363.34	2096.14	1589.24
30%	752.49	909.27	1748.79	1366.91	2097.23	1592.98
25%	752.75	909.04	1748.31	1369.16	2099.62	1595.31
20%	753.08	908.27	1753.47	1371.75	2099.94	1596.81
15%	752.78	909.62	1752.06	1376.39	2100.91	1600.63
10%	754.33	908.40	1754.19	1380.52	2100.09	1602.26
5%	754.82	908.42	1754.72	1383.94	2100.98	1601.65

A Figura 62, a Figura 63, a Figura 64 e a Figura 65 mostram comparações de latência média para as taxas de inserção de 30%, 50%, 70% e 90% respectivamente. Nestas Figuras é possível observar que não é inserida latência significativa na comunicação com a adição dos mecanismos de controle dinâmico de potência.

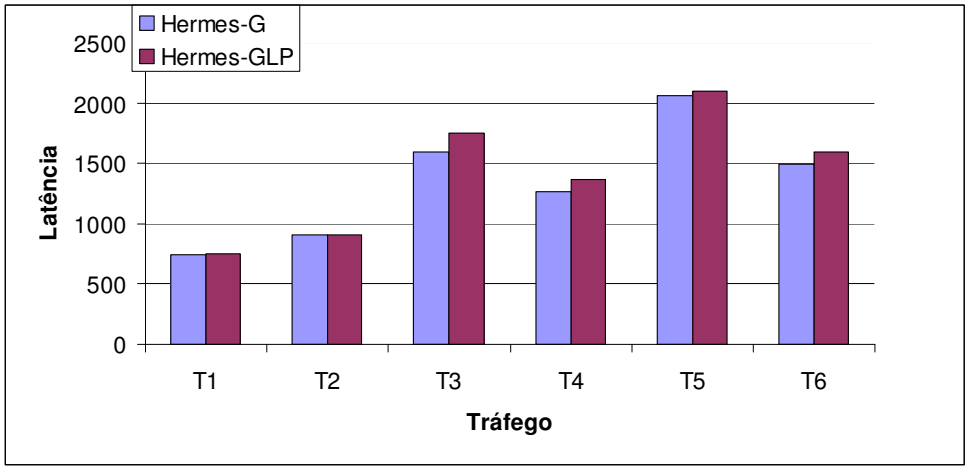

Figura 62 – Comparação da latência média entre a NoC Hermes-G e a NoC Hermes-GLP para os tráfegos da Tabela 16 com taxa de inserção 30%.

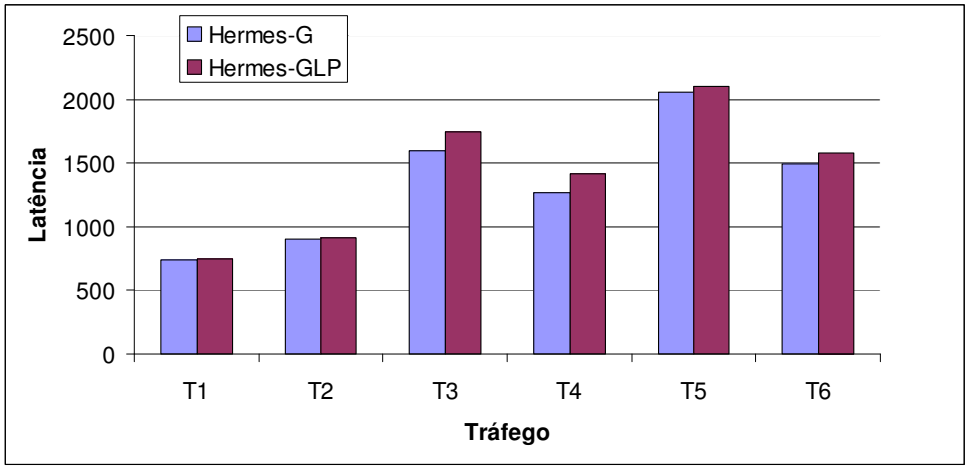

Figura 63 – Comparação da latência média entre a NoC Hermes-G e a NoC Hermes-GLP para os tráfegos da Tabela 16 com taxa de inserção 50%.

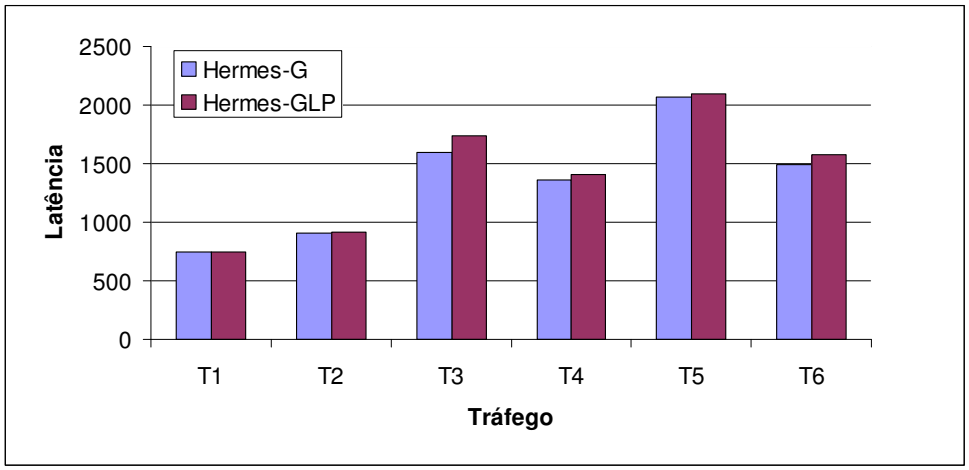

Figura 64 – Comparação da latência média entre a NoC Hermes-G e a NoC Hermes-GLP para os tráfegos da Tabela 16 com taxa de inserção 70%.

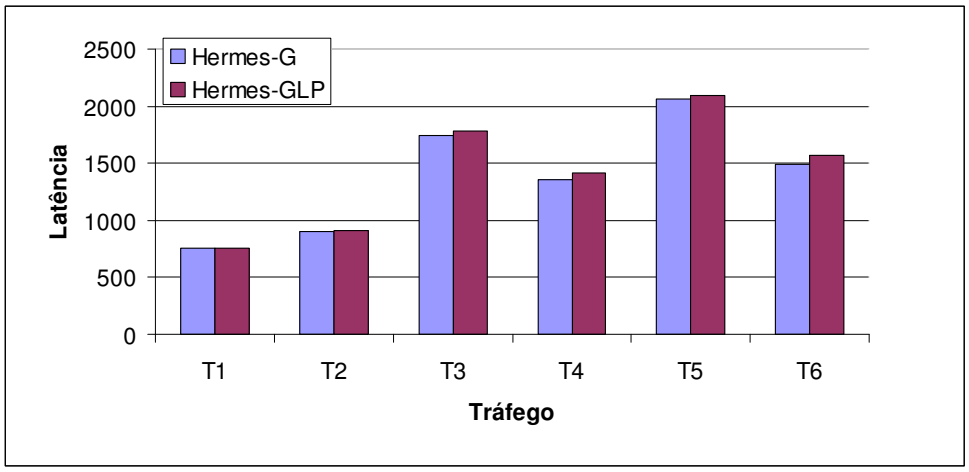

Figura 65 – Comparação da latência média entre a NoC Hermes-G e a NoC Hermes-GLP para os tráfegos da Tabela 16 com taxa de inserção 90%.

A Figura 66 compara a diferença média de latência das duas NoCs para todos os tráfegos em todas as taxas de inserção de dados. A mesma informação aparece na Tabela 20 sob a forma de dados numéricos.

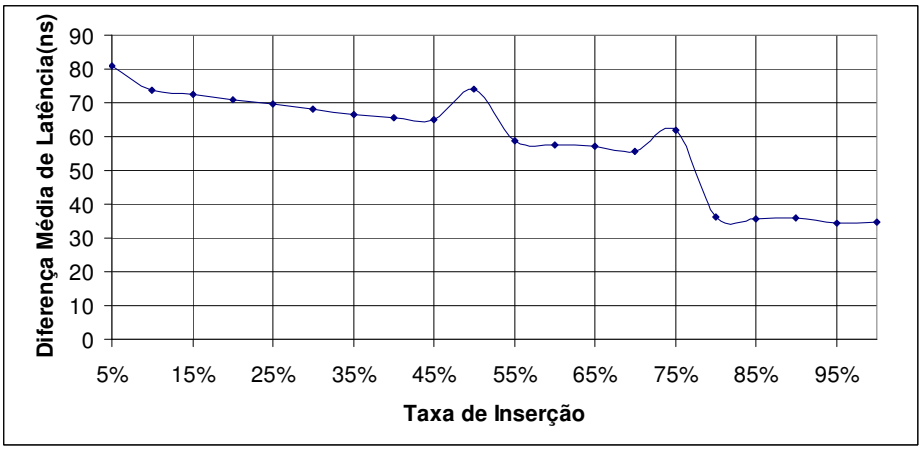

Figura 66 – Comparação da diferença média de latência entre a Hermes-G e a Hermes-GLP em função da taxa de inserção de dados.

A Figura 66 mostra uma que a diferença de latência entre as NoCs aumenta com a redução da taxa de inserção de dados. Esse resultado é esperado, uma vez que, quanto menor a taxa de inserção de dados na rede maior o número de pausas e chaveamentos de relógio nos roteadores. As mudanças de estado do roteador inserem uma pequena latência na transmissão do pacote.

Tabela 20 – Diferença entre as latências médias dos tráfegos descritos na Tabela 16 (a). L_{HG} denota a latência da Hermes-G e L_{HGLP} denota a latência da Hermes-GLP.

	Diferença entre as latências (ns) (L_{HGLP} - L_{HG})						
Taxa %	T1	T2	T3	T4	T5	T6	Diferença Média
100%	-4.81	2.02	40.7	58.27	29.7	82.05	34.66
95%	-5.37	6.83	41.97	56.97	29.06	76.89	34.39
90%	-4.57	7.29	42.93	57.63	32.01	79.82	35.85
85%	-5.07	9.47	40.42	56.24	31.33	81.21	35.60
80%	-5.74	6.8	42.2	57.28	32.08	84.06	36.11
75%	4.99	6.08	182.75	57.75	33.84	84.96	61.73
70%	8.65	8.24	141.37	55.43	31.47	87.76	55.49
65%	9.23	6.08	143.62	62.65	33.72	88.56	57.31
60%	8.99	6.13	142.22	62.28	33.25	92.31	57.53
55%	10.21	5.89	145.52	64.28	33.49	92.96	58.73
50%	9.78	6.3	146.47	150.74	37.64	93.01	73.99
45%	10.27	5.88	147.8	89.55	38.29	97.56	64.89
40%	9.39	5.82	150.06	93.2	38.17	98.01	65.78
35%	11.59	6.31	150.93	96.98	33.64	100.36	66.64
30%	12.66	6.34	151.17	99.56	35.15	103.96	68.14
25%	11.67	6.01	151.61	103.2	38.52	106.51	69.59
20%	11.94	5.29	156.8	105.6	38.71	107.81	71.03
15%	13.22	6.37	154.52	110.49	39.39	111.61	72.60
10%	13.27	4.77	157.56	114.3	38.61	113.26	73.63
5%	13.8	4.43	175.55	118.08	39.5	133.63	80.83

7.2.3.1 Comparação de Áreas

Para comparar o acréscimo de área resultante da adaptação da rede Hermes para dar suporte ao desenvolvimento de SoCs GALS, as três redes intrachip Hermes, Hermes-G e Hermes-GLP foram sintetizadas usando o FPGA XC2V1000 da família Virtex II da Xilinx. Todas as redes foram sintetizadas com a ferramenta de síntese XST da Xilinx, mantendo as opções de síntese idênticas para os três casos.

Os resultados de área são mostrados na Tabela 21. Foram comparados os consumos de *Flip Flops*, LUTs, S*lices* (cada slice é composto por 2 LUTs e 2 Flip Flops), e o consumo em função de portas equivalentes. O consumo de LUTs foi dividido entre consumo usado para lógica e LUTs usadas para *route-thru*. A ferramenta de síntese utiliza LUTs para *route-thru* quando um sinal é

amostrado por dois *flip-flops* em série. Para evitar que o tempo de *hold* seja violado no segundo *flip flop*, é adicionada uma LUT transparente (onde a saída copia a entrada), para adicionar um tempo de propagação maior que o tempo de propagação mínimo entre dois *flip flops*.

O acréscimo de área nas redes Hermes-G e Hermes-GLP se deve ao fato de que as filas empregadas utilizam *flip flops* em série para sincronizar o endereço de leitura e escrita. Dessa forma, o número de LUTs *route-thru* é bem maior que na NoC Hermes síncrona.

A Hermes-G ocupa mais s*lices* que a Hermes-GLP, mas esse acréscimo é devido somente ao fato da ferramenta de síntese ter distribuído a lógica de forma menos otimizada. Em nenhuma das três implementações forma inseridas restrições de tempo ou de área.

Tabela 21 – Comparação do consumo de área de um roteador com 5 portas prototipada em um FPGA XC2V1000 da família Virtex II da Xilinx, com *buffers* de 8 posições e palavra de 16 bits. A comparação de acréscimo de área é feita em relação à NoC Hermes. PE denota Portas Equivalentes.

Recurso	Hermes	Hermes-G	Hermes-GLP
Flip Flops	218	449	449
LUTs usadas para lógica	840	937	952
Slices	539	771	716
Dual Port RAM	160	160	160
LUTs route-thru	7	152	152
Total de LUTs	1007	1249	1264
Portas Equivalentes	27981	30990	31128
Acréscimo em LUTs (%)	-	24%	25,5%
Acréscimo em PEs (%)	-	10,7%	11,2%

7.3 Conclusões do Capítulo

Este Capítulo descreveu a implementação de duas redes intrachip, a Hermes-G e a rede Hermes-GLP.

A rede Hermes-G é capaz de suportar o desenvolvimento de sistemas GALS onde cada roteador e cada núcleo IP pode possuir uma fonte distinta do sinal de relógio. A interface de comunicação adotada foi comparada com outras interfaces propostas na literatura, desta forma o desempenho da NoC Hermes-G é projetada para atender aos requisitos de comunicação de SoCs com comunicação intensiva.

A rede Hermes-GLP é uma contribuição original desse trabalho, apresentando um sistema de controle de potência associado a cada roteador. Este sistema de controle permite a redução da taxa de ativação da NoC Hermes-GLP mesmo quando é submetida à taxas elevadas de inserção de tráfego. Outro ponto positivo dessa NoC é que o controle de potência implementado não acrescenta área e latência significativas se comparada a Hermes-G.

8 CONCLUSÃO E TRABALHOS FUTUROS

8.1 Conclusões

Esse trabalho apresentou o desenvolvimento de sistemas de comunicação em FPGAs com capacidade para dar suporte ao estilo GALS de projeto.

As principais contribuições desse trabalho foram o desenvolvimento de uma biblioteca de componentes básicos para dar suporte ao desenvolvimento de circuitos assíncronos em FPGAs; a proposta e implementação de uma nova interface de comunicação assíncrona de comunicação para FPGAs e o desenvolvimento de duas redes intrachip para sistemas GALS, a Hermes-G e a Hermes-GLP, sendo que essa última ainda oferece mecanismos de controle dinâmico de potência.

A biblioteca de *hard macros* simplifica o desenvolvimento de circuitos assíncronos em FPGAs, promovendo o reuso de hardware e reduzindo a verificação temporal dos circuitos assíncronos gerados. Entretanto ainda existem problemas de posicionamento e roteamento devidos a ferramenta de *place and route* ISE que dificultam o aproveitamento dos componentes da biblioteca.

A proposta de interface apresentada aqui utilizou um *Stretcher*, proposto durante o desenvolvimento desse trabalho, diferenciando essa abordagem das encontradas na literatura. A validação da interface através de um módulo GALS do *criptocore* RSA evidenciou o potencial de redução de potência de sistemas GALS.

A última e principal contribuição desse trabalho foi o desenvolvimento de duas redes intrachip capazes de suportar o estilo GALS. A primeira NoC, Hermes-G, é uma adaptação da Hermes síncrona diferenciando-se quase que somente pela fila empregada ns portas de entrada. As filas da Hermes original foram substituídas por uma fila capaz de prover mecanismos de sincronização. Essa interface foi cuidadosamente selecionada, sendo que a escolha abrangeu o estudo comparativo de diversas abordagens de interfaces assíncronas da literatura. A segunda NoC, denominada Hermes-GLP, consiste em uma extensão da Hermes-G. Essa extensão foi feita para aproveitar as características de sistemas GALS para reduzir a taxa de ativação dos roteadores. Através de simulações foi possível constatar que mesmo quando a NoC Hermes-GLP está sob intensa inserção de dados na rede, esta é capaz de reduzir significativamente a taxa de ativação de seus roteadores.

8.2 Trabalhos Futuros

A continuação desse trabalho prevê primeiramente o levantamento quantitativo, através de

medições e estimativas, de redução de potência que a Hermes-GLP é capaz de prover. Além disso, esses valores levantados serão confrontados com as taxas de ativação encontradas nas simulações. O principal objetivo é verificar a possibilidade de utilização das taxas de ativação como estimativas de dissipação de potência na comunicação, uma vez que a extração dessas taxas é possível em fases iniciais de projeto.

A implementação da NoC Hermes-GLP em ASIC é também uma das tarefas previstas para a continuação desse trabalho. Uma vez que, a Hermes-GLP seja transferida para ASIC serão implementadas técnicas de controle dinâmico de tensão nos núcleos IP conectados a NoC.

Tendo em vista que os resultados demonstraram a dependência da taxa de ativação da NoC em função do posicionamento dos núcleos IP na NoC, se verifica a necessidade de desenvolvimento de ferramentas que auxiliem a exploração correta do potencial da NoC.

Um segundo objetivo futuro com respeito a ferramentas de software consiste na adaptação do ambiente ATLAS, ferramenta que disponibiliza a geração automática de redes intrachip desenvolvidas pelo GAPH, para geração correta das duas redes intrachip visando o aproveitamento das NoCs pela comunidade acadêmica.

9 REFERÊNCIAS BIBLIOGRÁFICAS

[AMD05] Amde, M.; Felicijan, T.; Efthymiou, A.; Edwards, D.; Lavagno, L**.** "Asynchronous on-chip networks". IEE Proceedings - Computers and Digital Techniques, 152(2), pp. 273-283, March 2005.

[AND03] Anderson, R.; Moore, S.; Mullins, R.; Taylor, G.; Fournier, J. "Balanced Self-Checking Asynchronous Logic for Smart Card Applications". Microprocessors and Microsystems, 27(9), pp. 421-430, October 2003.

[BAI02] Bainbridge, J.; Furber, S. "Chain: a delay-insensitive chip area interconnect". IEEE Micro, 22(5), pp. 16-23, September/October, 2002.

[BEC03] Becker, J.; Huebner, M.; Ullman, M. "Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations". In: 16th Symposium on integrated Circuits and System Design (SBCCI03), pp. 283- 288, 2003.

[BEI05] Beigné, E.; Clermidy, F.; Vivet, P.; Clouard, A.; Renaudin, M.; "An Asynchronous NoC Architecture Providing Low Latency Service and its Multi-level Design Framework". In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC05), pp. 54-63, 2005.

[BEI06] Beigné, E.; Vivet, P. "Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture". In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC06), pp. 172-183, 2006.

[BER00] Bergamaschi, R. A.; Lee, R. W. "Designing Systems-on-chip Using Cores". In: AC/IEEE Design Automation Conference (DAC00), pp. 420-425, 2000.

[BER92] Van Berkel, K. "Beware the Isochronic Fork". Integration, the VLSI journal, 13(2), pp. 103-128, June 1992.

[BJE05] Bjerregaard, T.; Sparsø, J., "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip". In: Design, Automation and Test in Europe (DATE05), pp. 1226-1231, March 2005.

[BJE06] Bjerregaard, T.; Mahadevan, S. "A Survey of Research and Practices of Network-on-Chip". ACM Computing Surveys, 38(1), pp. 1-51, 2006.

[BJE07] Bjerregaard, T.; Stensgaard M.; Sparsø, J. “A Scalable, Timing-Safe, Network-on-Chip Architecture with an Integrated Clock Distribution Method”. In: Design, Automation and Test in Europe (DATE07), pp. 648-653, April 2007.

[BRE05] Brej, C. “Early Output Logic and Anti-Tokens” PhD Thesis, School of Computer Science, University of Manchester, Manchester, September 2005.

[CAL98] Calazans, N. L. V. “Automated Logic Design of Sequential Digital Circuits”. Rio de Janeiro:Imprinta, 1998. 342p.

[CHA84] Chapiro, D. M. “Globally-Asynchronous Locally Synchronous Systems”. PhD Thesis, Stanford University, October 1984, 134 p.

[COR00] Cortadella, J.; Kishinevsky, M.; Kondratyev, A.; Lavagno, L. “Introduction to asynchronous circuit design: specification and synthesis”. Tutorial. In: 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC00), April, 2000.

[COR06] Cortadella, J.; Kondratyev, A.; Lavagno, L.; Sotiriou, C.P., “Desynchronization: Synthesis of Asynchronous Circuits from Synchronous Specifications”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(10), pp. 1904-1921, October, 2006.

[CUM02] Cummings, C. E. “Simulation and Synthesis Techniques for Asynchronous Fifo Design”. In: Synopsys Users Group Conference (SNUG02), 2002.

[DAL04] Dally, W. J.; Towles, B. “Principles and Practices of Interconnection Networks”. San Francisco: Morgan Kaufmann, 2004, 550 p.

[DAV92] David, I.; Ginosar, R.; Yoeli, M. “An Efficient Implementation of Boolean Function as Self-Timed Circuits”. IEEE Transaction on Computers, 41(1), pp. 2-11, January 1992.

[DIN02] Dinh Duc, A.V.; Rigaud, J.B.; Rezzag, A.; Sirianni, A.; Fragoso , J.; Fesquet, L.; Renaudin, M. “TAST CAD Tools”. Proc. of the 2nd Asynchronous Circuit Design Workshop, 2002.

[DUA02] Duato, J.; Yalamanchili, S.; Ni, L. “Interconnection Networks”. San Francisco:Elsevier Science, 2002, 600 p.

[DIK99] Dike, C.; Burton, E. “Miller and Noise Effects in a Synchronizing Flip-Flop”. IEEE Journal of Solid-State Circuits, 34(6), pp. 849-855, June 1999.

[EDW97] Edwards, S.; Lavagno, L.; Lee, E. A.; Sangiovanni-Vincentelli, A. “Design of Embedded Systems: Formal Models, Validation and Synthesis”. Proceedings of the IEEE, 85(3), pp. 366-390, March 1997.

[FAN05] Fang, D.; Teifel, J.; Manohar, R. “A High-Performance Asynchronous FPGA: Test Results”. In: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM05), pp. 271- 272, April 2005.

[FUH99] Fuhrer, R.; Nowick, S.; Theobald, M. “MINIMALIST: An Environment for the Synthesis, Verification and Testability of Burst-Mode Asynchronous Machines”. Technical Report, Columbia University, Computer Science Department. Capturado em http://www1.cs.columbia.edu/~nowick . November 2007.

[GIN03] Ginosar, R. “Fourteen ways to fool your synchronizer”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC03), pp. 89-96, May 2003.

[HEM99] Hemani, A.; Meincke, T.; Kumar, S.; Postula, A.; Olsson, T.; Nilsson, P.; Oberg, J.; Ellervee, P.; Lundqvist, D. “Lowering power consumption in clock by using Globally Asynchronous Locally Synchronous design style”. In: ACM/IEEE Design Automation Conference (DAC99), pp. 873-878, 1999.

[HEN03] Henkel, J. “Closing the SoC Design Gap”. IEEE Computer, 36(9), pp. 119-121, September 2003.

[HO02] Ho, Q.; Rigaud, J.; Fesquet, L.; Renaudin, M.; Rolland, R. “Implementing asynchronous circuits on LUT based FPGAs”. In: International Conference on Field Programmable Logic and Applications (FPL02), pp. 36-45, 2002.

[HSU05] Hsu, C.; Wang, W.; Hong, Y. “Frequency-Scaling Approach for Managing Power Consumption in NoCs”. IEICE Transactions Fundamentals 88(12), pp. 3580-3583, December 2005.

[IEC04] IEC/IEEE. “Delay and power calculation standards – Part3: Standard Delay Format (SDF) for the electronic design process”. International Standard, IEC 61523-3/IEEE 1497, 2004.

[ITR05] International Technology Roadmap for Semiconductors. “ITRS 2005 Edition”. Capturado em http://www.itrs.net/Links/2005ITRS/Home2005.htm, December 2005.

[KEU00] Keutzer, K.; Malik, S.; Newton, A.; Rabaey, J.; Sangiovanni-Vincentelli, A. “System-Level Design: Orthogonalization and Plataform-Based Design”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19(12), pp. 1523-1543, December 2000.

[KIM05] Kim, D.; Kim, M.; Sobelman, G. “Asynchronous FIFO Interfaces for GALS On-Chip Switched Networks”. In: International SoC Design Conference (ISoCC05), pp. 186-189, 2005.

[KRS07] Krstic, M.; Grass, E.; Gurkaynak, F.; Vivet, P., “Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook”. IEEE Design & Test of Computers, 24(5), pp. 430-441, September-October 2007

[KUM02] Kumar, S.; Jantsch, A.; Soininen, J.-P.; Forsell, M.; Millberg, M.; Oberg, J.; Tiensyrja, K.; Hemani, A. “A network on chip architecture and design methodology”. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI02), pp. 105-112, 2002.

[LAN02] Langerwerf, M.; Reuter, J.; Kropp, C.; Pirsch, P. “Benefits of macro-based multi-FPGA partitioning for video processing applications”. In: 13th IEEE International Workshop on Rapid System Prototyping (RSP02), pp. 60-65, 2002.

[MAH03] Mahmud, R. “Techniques to make clock switching glitch free”. EETimes. Available at http://www.eetimes.com/story/OEG20030626S0035. Capturado em 11/11/2008.

[MAR06] Martin, A. J.; Nystrom, M. “Asynchronous techniques for system-on-chip design”. Proceedings of the IEEE, 94(6), pp. 1089-1120, June 2006.

[MEL05] Mello, A.; Tedesco, L.; Calazans, N.; Moraes, F. “Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC”. In: 18th Symposium on integrated Circuits and System Design (SBCCI05), pp. 178-183, 2005.

[MOO98] Moore, S.W.; Robinson, P. “Rapid Prototyping of Self-timed Circuits”. In: 17th IEEE International Conference on Computer Design (ICCD98), pp. 360-364, 1998.

[MOO02] Moore, S.; Taylor, G.; Mullins, R.; Robinson, P. “Point to point GALS interconnect”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC02), pp. 69-75, April 2002.

[MOR04] Moraes, F. G.; Calazans, N. L. V.; Mello, A. V.; Möller, L. H.; Ost, L. C. “Hermes: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip”. Integration the VLSI Journal, 38(1), pp. 69-93, Oct. 2004.

[MUT00] Muttersbach, J.; Villiger, T.; Fichtner, W. “Practical Design of Globally-Asynchronous Locally-Synchronous Systems”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC00), pp. 52-59, 2000.

[MYE01] Myers, C. “Asynchronous Circuit Design”. New York:Wiley-Interscience, 2001. 422 p.

[NAJ05] Najibi, M.; Saleh, N.; Naderi, M.; Pedram, H.; Sedighi, M. “Prototyping Globally Asynchronous Locally Synchronous Circuits on Commercial Synchronous FPGAs”. In: 16th IEEE International Workshop on Rapid System Prototyping (RSP05), pp. 63-69, 2005.

[NOW95] Nowick, S.; Dill, D., “Exact two-level minimization of hazard-free logic with multiple-input changes”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 14(8), pp.986-997, August 1995.

[NOW96] Nowick, S. “Design of a low latency asynchronous adder use speculative completion”. IEE Proceedings - Computer and Digital Techniques, 143(5), pp. 301-307, September 1996.

[OGR07] Ogras, U.; Marculescu, R.; Choudhary, P.; Marculescu, D. “Voltage-frequency island partitioning for GALS-based networks-on-chip”. In: ACM/IEEE Design Automation Conference (DAC07), pp. 110-115, June 2007.

[OZD02] Ozdag, R.; Beerel, P. “High-Speed QDI Asynchronous Pipelines”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC02), pp. 13-22, 2002.

[PAN06] Panades, M.; Greiner, A.; Sheibanyrad, A.; “A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach”. In: 1st International Conference on Nano-Networks and Workshops (NanoNet06), pp. 1-5, 2006.

[PAY96] Payne, R. “Asynchronous FPGA architectures”. IEE Proceedings - Computers and Digital Techniques, 143(5), pp. 282- 286, September 1996.

[PED05] Pedram, M. and Abdollahi, A. “Low power RT-level synthesis techniques: a tutorial”. IEE Proceedings - Computers and Digital Techniques, 152(3), pp. 333-343, May 2005.

[PON07] Pontes, J.; Soares, R.; Carvalho, E.; Moraes, F.; Calazans, N. “SCAFFI: An intrachip FPGA asynchronous interface based on hard macros”. In: 25th IEEE International Conference on Computer Design (ICCD07), pp. 541-546, 2007.

[PON07a] Pontikakis, B.; Bui, H. T.; Boyer, F.-R.; Savaria, Y. “A Low-Complexity High-Speed Clock Generator for Dynamic Frequency Scaling of FPGA and Standard-Cell Based Designs”. In: IEEE International Symposium on Circuits and Systems (ISCAS07), pp. 633-636, May 2007.

[PON08] Pontes, J.; Trevisan, M.; Soares, R.; Calazans, N. “Hermes-GLP: A GALS Network on Chip Router with Power Control Techniques”. IEEE Computer Society Annual Symposium on VLSI (ISVLSI08), pp. 347-352, April 2008.

[QUA05] Quartana, J.; Renane, S.; Baixas, A.; Fesquet, L.; Renaudin, M. “GALS systems prototyping using multiclock FPGAs and asynchronous network-on-chips”. In: International Conference on Field Programmable Logic and Applications (FPL05), pp. 299-304, 2005.

[RIJ01] Rijpkema, E.; Goossens, K.; Wielage, P. “A Router Architecture for Networks on Silicon”. In: 2nd Workshop PROGRESS on Embedded Systems (PROGRESS01), Netherlands, pp. 350-355, 2001.

[ROS05] Rostislav, D.; Vishnyakov, V.; Friedman, E.; Ginosar, R. “An asynchronous router for multiple service levels networks on chip”. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC05), pp. 44-53, March 2005.

[SAL06] Saleh, R.; Wilton, S.; Mirabbasi, S.; Hu, A.; Greenstreet, M.; Lemieux, G.; Pande, P.; Grecu, C.; Ivanov, A. “System-on-Chip: Reuse and Integration”. Proceedings of the IEEE, 94(6), pp. 1050-1069, June 2006.

[SCH07] Scherer, C. A. “Redes Intrachip com Topologia Toro e Modo de Chaveamento Wormhole: Projeto, Geração e Avaliação”. Dissertação de Mestrado, PPGCC-FACIN-PUCRS, Porto Alegre, Março 2007.

[SIM04] Simunic, T.; Boyd, S. P.; Glynn, P. “Managing power consumption in networks on chips”. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 12(1), pp. 96-107, January 2004.

[SPA02] Sparsø, J. and Furber, S. (Eds). “Principles of asynchronous circuit design - A systems perspective”. London:Springer, 2002, 360p.

[STR00] Strollo, A. G. M.; Napoli, E.; De Caro, D. “New clock-gating techniques for low-power flip-flops”. In: 2000 International Symposium on Low Power Electronics and Design, (ISLPED00), pp. 114-119, 2000.

[TEE07] Teehan, P.; Greenstreet, M.; Lemieux, G. “A Survey and Taxonomy of GALS Design Styles”. IEEE Design & Test of Computers, 24(5), pp.418-428, September-October 2007.

[TEI04] Teifel, J.; Manohar, R., “An asynchronous dataflow FPGA architecture”. IEEE Transactions on Computers, 53(11), pp. 1376-1392, November 2004.

[VLS08] VLSI Chip Design & Development. “Clock gating – clock on demand”. Available at http://www.vlsichipdesign/clockondemand.html. Capturado em 11/11/2008.

[WAN06] Wang, X.; Ahonen, T.; Nurmi, J. “Prototyping a Globally Asynchronous Locally Synchronous Network-on-Chip on a Conventional FPGA Device Using Synchronous Design Tools”. In: 16th International Conference on Field Programmable Logic and Applications (FPL06), pp. 657-662, August 2006.

[WES94] Weste, N.; Eshraghian, K. “Principles of CMOS VLSI Design”. Boston:Addison-Wesley, 2nd edition, 1994, 735 p.

[WOR05] Worm, F.; Ienne, P.; Thiran, P.; De Micheli, G., “A robust self-calibrating transmission scheme for on-chip networks”. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13(1), pp. 126-139, January 2005.

[XIL06] Xilinx, Inc. Capturado em: http://www.xilinx.com, November, 2006.

[YAH06] Yahya, E.; Renaudin, M. “Asynchronous Buffers Characteristics: Modeling and Design”. Research Report, TIMA, 2006.

[ZEF02] Zeferino, C. A.; Kreutz, M. E.; Carro, L.; Susin, A. A. “A study on communication issues for systems-on-chip”. In: 15th Symposium on Integrated Circuits and Systems Design (SBCCI02), pp. 121-126, 2002.

[ZIP04] Zipf, P.; Hinkelmann, H.; Ashraf, A.; Glesner, M. “A Switch Architecture and Signal Synchronization for GALS System-on-Chips”. In: 17th Symposium on Integrated Circuits and System Design (SBCCI04), pp. 210-215, 2004.