



**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL**

**FACULDADE DE INFORMÁTICA**

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**UM MPSOC GALS BASEADO EM REDE INTRACHIP  
COM GERAÇÃO LOCAL DE RELÓGIO**

**GUILHERME HECK**

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

Orientador: Prof. Dr. Ney Laert Vilar Calazans

Porto Alegre

Agosto de 2012



H448m Heck, Guilherme  
Um MPSOC GALS baseado em rede intrachip com geração local de relógio / Guilherme Heck. – Porto Alegre, 2012.  
123 f.

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

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

CDD 004.6

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

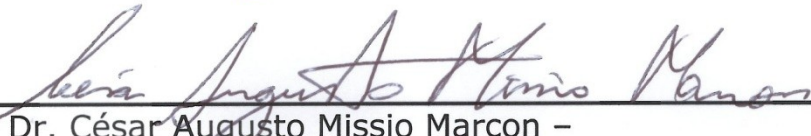




## TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Um MPSoC GALS Baseado em Rede Intrachip com Geração Local de Relógio", apresentada por Guilherme Heck como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas Embarcados e Sistemas Digitais, aprovada em 27/08/2012 pela Comissão Examinadora:

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

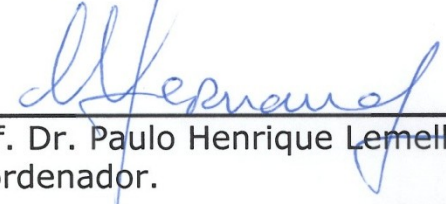
  
Prof. Dr. César Augusto Missio Marcon - PPGCC/PUCRS

  
Prof. Dr. Fernando Gehm Moraes - PPGCC/PUCRS

  
Prof. Dr. Jader Alves de Lima Filho - UFSC

  
Prof. Dr. Renato Perez Ribas - UFRGS

Homologada em <sup>24</sup>...../...../.....<sup>01</sup> 2013, conforme Ata No. <sup>002</sup>..... pela Comissão Coordenadora.

  
Prof. Dr. Paulo Henrique Lemelle-Fernandes  
Coordenador.

**PUCRS**

**Campus Central**

Av. Ipiranga, 6681 - P32- sala 507 - CEP: 90619-900

Fone: (51) 3320-3611 - Fax (51) 3320-3621

E-mail: [ppgcc@pucrs.br](mailto:ppgcc@pucrs.br)

[www.pucrs.br/facin/pos](http://www.pucrs.br/facin/pos)



## DEDICATÓRIA

Dedico este trabalho à minha família, principalmente a meus pais.

“A vingança nunca é plena. Mata a alma e a envenena”.  
(Ramón Gómez Valdés y Castillo, seriado Chaves).





## AGRADECIMENTOS

Gostaria de agradecer principalmente a Deus e também a toda a minha família, que me ofereceu suporte e acreditou na finalização desse trabalho. Agradeço aos meus pais: Reinaldo Heck e Ida Acélia Heck, por todo o apoio e empenho para educar-me. Aos meus irmãos: Marcos Heck e Bárbara Heck juntamente com seus cônjuges, Sônia Asquidamini e Sérgio Farina Ucoski, e meus sobrinhos Caroline Asquidamini Heck e Eduardo Asquidamini Heck que me apoiaram e me motivaram para que esse trabalho fosse concluído. Gostaria de agradecer a todos os meus amigos e colegas de trabalho do GAPH, em especial ao Ricardo Guazzelli, Matheus Gibiluka, Leandro Sehnem Heck, Matheus Trevisan, Thiago Raupp da Rosa, Guilherme Castilhos e Eduardo Wächter por toda a ajuda e amizade durante esses dois anos. Ainda agradeço a Henrique Mamoru Hayasaka, Anderson Barrionuevo e Vanessa Espinosa que auxiliaram e cooperaram com este trabalho. Gostaria de agradecer ao CNPq pelo apoio financeiro e ao serviço Circuit Multi – Projects (CMP) de Grenoble/França pela disponibilização das bibliotecas de semicondutores utilizadas neste trabalho. 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. Finalizando, agradeço a todos que de alguma maneira colaboraram e contribuíram para este trabalho ser o que é.

Muito Obrigado.



# UM MPSOC GALS BASEADO EM REDE INTRACHIP COM GERAÇÃO LOCAL DE RELÓGIO

## RESUMO

Devido à evolução das tecnologias nanométricas profundas em semicondutores, hoje é possível a fabricação de sistemas cada vez mais complexos em um único chip. Entretanto, esta evolução está inviabilizando, em alguns casos, práticas de projeto tradicionais. O desenvolvimento de sistemas complexos puramente síncronos começa a ser influenciado por distâncias intrachip relativamente longas, bem como por efeitos parasitas em fios com áreas de secção reta cada vez menores. Adicionalmente, ganha destaque em pesquisa e na indústria a necessidade de projetar dispositivos com elevada capacidade de processamento para atender a demanda de múltiplas aplicações, enquanto aprimoram-se os níveis de eficiência energética. Isto é motivado pelo significativo aumento da procura por equipamentos portáteis multifunções como *tablets* e celulares inteligentes mais velozes e com durabilidade de bateria razoável. À luz destes fatos, novos paradigmas de projeto de sistemas globalmente assíncronos e localmente síncronos (GALS) ganham destaque para construir sistemas multiprocessados em chip (MPSoCs). Este trabalho tem como principal objetivo estratégico explorar arquiteturas GALS para MPSoCs com alvo no controle da potência dissipada. Escolhe-se trabalhar sobre MPSoCs devido ao aumento significativo de módulos de processamento em projetos atuais como uma forma de tirar vantagem plena da evolução das tecnologias de fabricação baseadas em silício. Ao longo das atividades, cinco contribuições podem ser destacadas oriundas cada uma de um conjunto de trabalhos práticos desenvolvidos. Primeiro, propôs-se um conjunto de correções e modificações nas arquiteturas do roteador da NoC Hermes-GLP e do MPSoC HeMPS, visando transformar estes em um melhor suporte à implementação de sistemas GALS. Isto produziu uma nova arquitetura de MPSoC, denominado HeMPS-GLP. Segundo, alterações na estrutura do *microkernel* embarcado dos processadores do MPSoC HeMPS possibilitaram a interconexão e configuração corretas de novas estruturas em hardware aos processadores em questão. Terceiro, disponibilizou-se um ambiente de verificação em linguagem de alto nível para o MPSoC HeMPS-GLP, com suporte a até 256 níveis distintos de frequência para a rede, bem como a definição do relógio de cada IP de processamento de forma independente. Em quarto lugar, foram realizados o estudo e o projeto de um gerador local de relógio visando obter área mínima, baixa dissipação de potência, estabilidade em frequência e insensibilidade a variações de processo, tensão de alimentação e temperatura. Quinto e último, foi desenvolvido um ambiente de simulação e geração de código sintetizável em silício para o MPSoC HeMPS-GLP. Este provê a emulação do sistema de geração local de relógio, baseado no gerador local projetado.

**Palavras Chave:** GALS, NoCs, MPSoCs, geração local de relógio, osciladores controlados digitalmente, inibição de relógio.



# A NETWORK ON CHIP-BASED GALS MPSoC WITH LOCAL CLOCK GENERATION

## ABSTRACT

Due to the evolution of deep submicron technologies for semiconductor fabrication, it is possible nowadays to manufacture increasingly complex systems inside a single silicon die. However, this evolution in some cases mandates the abandonment of traditional design techniques. The development of purely synchronous complex systems begins to be influenced by relatively long intrachip distances as well as by parasitic effects in wires with growingly small cross-sections. Besides, it is important to enable the design of devices with enhanced processing capabilities to fulfill the demand for multiple applications in research and industry environments, while at the same time improving energy efficiency. This is motivated by the significant increase on the demand for multifunctional portable equipments like *tablets* and smart phones that must everyday become faster and yet present reasonable battery life. In view of these facts, new paradigms for the design of globally asynchronous locally synchronous (GALS) systems come to the forefront in the construction of multiprocessor systems on chip (MPSoCs). This work has as main strategic objective to explore GALS MPSoC architectures that target the control of power dissipation. The decision to work with MPSoCs comes from the natural need to increase the number of processing elements in current designs, as a way to take full advantage of the silicon technological evolution. During the development of this work five distinct contributions are worth mentioning. First, the architectures of the Hermes-GLP NoC router and of the HeMPS MPSoC were subject to a set of corrections and modifications, to provide these modules with better support to the implementation of GALS systems. This allowed the proposition of a new MPSoCs, called HeMPS-GLP. Second, a set of changes in the embedded processor *microkernel* of the HeMPS MPSoC enabled the smooth interconnection and configuration of new hardware structures to the system processors. Third, a new high-level language verification environment for the HeMPS-GLP MPSoC was made available, which supports up to 256 distinct operating frequencies for the NoC, together with the independent definition of each processing element's clock. Fourth, there is the proposition of a new local clock generator targeting minimum area, low power dissipation, operating frequency stability and insensitivity to process, voltage and temperature variations. Finally, this work provides a simulation and code generation environment for silicon implementations of the HeMPS-GLP MPSoC. This environment emulates the local clock generators, based on the designed local clock generator.

**Keywords:** GALS, NoCs, MPSoCs, local clock generation, digitally controlled oscillators, clock gating.



## LISTA DE FIGURAS

<i>Figura 1. MPSoC baseado em NoC (Legenda: R → Roteador, IP núcleo de propriedade intelectual ou elemento de processamento).</i> .....	29
<i>Figura 2: MPSoC com VFIs e NoC mesócrona [LUD11].</i> .....	35
<i>Figura 3. Diagrama de blocos do sistema de DFS do módulo IP proposto por Almeida et al. [ALM11].</i> .....	36
<i>Figura 4. MPSoC com VFIs desenvolvido por [OGR09].</i> .....	38
<i>Figura 5. Arquitetura de um nodo de rede do MPSoC proposto por Beigné e colaboradores em [BEI08].</i> .....	39
<i>Figura 6. Oscilador de relaxamento com CTS [KLA08].</i> .....	42
<i>Figura 7. Oscilador em anel com variação de número de estágios [SOB08].</i> .....	43
<i>Figura 8. Atuação do gerador de relógio com DVFS variando-se a tensão de alimentação [YAD12].</i> .....	44
<i>Figura 9. Gerador de relógio ADPLL [HOP12].</i> .....	45
<i>Figura 10. Oscilador em anel utilizado no ADPLL [HOP12].</i> .....	45
<i>Figura 11. Diagrama de blocos do roteador da rede Hermes-GLP.</i> .....	49
<i>Figura 12: Interconexões utilizadas dentro do roteador Hermes-GLP para a seleção de frequência nos roteadores. Os módulos empilhados à esquerda do desenho representam as portas de entrada e saída. O crossbar é mostrado de forma simplificada apenas detalhando o fluxo de sinais de controle da frequência de relógio das diferentes portas.</i> .....	50
<i>Figura 13: Estrutura de uma instância típica do MPSoC HeMPS, no caso uma implementação 2x2 da mesma.</i> .....	51
<i>Figura 14. Detalhe dos módulos internos ao IP de Processamento da HeMPS.</i> ....	52
<i>Figura 15: Interface gráfica do Gerador HeMPS.</i> .....	53
<i>Figura 16. Diagrama de blocos da fila bissíncrona implementada na rede Hermes-GLP.</i> .....	55
<i>Figura 17. Nova organização dos ponteiros de leitura e escrita. Situações: (a) fila vazia; (b) fila com dados; (c) fila cheia.</i> .....	56
<i>Figura 18. Codificação livre de glitches para a lógica de incremento para ponteiros Gray [ALT12A].</i> .....	57
<i>Figura 19. Circuito conversor de codificação Johnson para binário.</i> .....	58
<i>Figura 20. Seleção de frequência empregando uma árvore binária simples como circuito equivalente.</i> .....	60
<i>Figura 21. Diagrama de conexões do módulo BUFGCTRL [XIL10].</i> .....	61
<i>Figura 22. Trecho do código VHDL que gera a seleção de frequência.</i> .....	62

Figura 23. Circuito simplificado de votação para seleção de frequência.....	63
Figura 24. Forma de onda extraída da simulação do IP 11 da NoC Hermes-GLP executando no módulo HardNoC prototipado em FPGA Xilinx.....	64
Figura 25. Diagrama de blocos do MPSoC HeMPS-GLP. ....	65
Figura 26. A interface do Gerador HeMPS-GLP. ....	66
Figura 27. Destaque da caixa de seleção de NoC no Gerador HeMPS-GLP. ....	67
Figura 28. Destaque dos campos de números de relógios e da lista de suas configurações. ....	68
Figura 29. Destaque da configuração de relógio dos IPs de processamento. ....	68
Figura 30. Detalhe da Interface entre um IP de Processamento e a porta local de um roteador Hermes-GLP.....	72
Figura 31. Trecho do arquivo de saída da simulação do MPSoC HeMPS-GLP 3x3 (a) e 16x16 (b). ....	74
Figura 32. Formas de onda para identificar variação da prioridade e da frequência. ....	75
Figura 33. Diagrama de blocos genérico de um gerador local de relógio. ....	78
Figura 34. Diferentes modelos de topologias de osciladores construídos como circuitos eletrônicos. (a) osciladores sintonizados, (b) osciladores de relaxamento, e (b) osciladores em anel.....	78
Figura 35. Oscilador em anel com controle de corrente.....	79
Figura 36. Conversor digital analógico com saída em modo corrente com codificação de termômetro [BAK10]. ....	81
Figura 37. Conversor digital analógico binário em modo corrente [BAK10]. ....	82
Figura 38. Representação simplificada do transistor NMOS.....	82
Figura 39. Associação de transistores formando um espelho de corrente.....	83
Figura 40. Topologia W-2W para conversores DA modo corrente [BAK10].....	84
Figura 41. Topologia de multiplicação direta de corrente.....	84
Figura 42. Oscilador controlado por corrente de carga. ....	86
Figura 43. Oscilador em anel mostrando a estrutura simplificada do circuito de inibição de relógio.....	87
Figura 44. Oscilador em anel com circuito de inibição de relógio. ....	87
Figura 45. Oscilador em anel com controle de corrente com inibição de relógio (versão final).....	88
Figura 46. Interface externa do Oscilador com suas respectivas entradas e saídas. ....	89
Figura 47. Conversor digital analógico com saída em modo corrente.....	90



<i>Figura 48. Interface externa do Conversor DA com suas respectivas entradas e saídas.</i>	91
<i>Figura 49. Diagrama de blocos do DCO com Conversor DA e Oscilador, versão preliminar.</i>	92
<i>Figura 50. Variação de frequência conforme valores de entrada digital do DCO, considerando caso típico e condições de contorno de variação de parâmetros dos transistores que compõem o DCO.</i>	93
<i>Figura 51. Circuito do compensador de variações de PVT.</i>	95
<i>Figura 52. Diagrama de blocos final do DCO.</i>	96
<i>Figura 53. Frequência de saída do DCO conforme incremento no compensador de PVT (comp).</i>	97
<i>Figura 54. Detalhe das frequências geradas conforme variação do compensador comp.</i>	98
<i>Figura 55. Escala de frequências, conforme entrada fornecida ao DCO (sel) com compensação de PVT.</i>	99
<i>Figura 56. Resultado da simulação Monte Carlo para o melhor caso (5000 amostras).</i>	100
<i>Figura 57. Resultado da simulação Monte Carlo para o pior caso (5000 amostras)</i>	101
<i>Figura 58. Ação do sinal Inibir_relogio em ponto de conflito no nodo N do DCO.</i>	101
<i>Figura 59. Ação do sinal Inibir_relogio em ponto de conflito no nodo N do DCO 0.1ps antes do visto na Figura 58.</i>	102
<i>Figura 60. O MPSoC HeMPS-GLP com módulos de geração local de relógio (GLR).</i>	105
<i>Figura 61. Gerador HeMPS-GLP com gerador local de relógio.</i>	109
<i>Figura 62. Forma de onda do roteador 22 da HeMPS-GLP com GLR.</i>	110



## LISTA DE TABELAS

<i>Tabela 1 - Política do controlador DFS de [ROS12]. ↓↑ implicam diminuir/aumentar a frequência em um ponto, ↑↑ significa aumentar a frequência em dois pontos, = significa manter a frequência inalterada, e – significa valor ignorado.....</i>	<i>38</i>
<i>Tabela 2 - Definição dos estados possíveis de DVFS.....</i>	<i>40</i>
<i>Tabela 3 - Comparação entre MPSoCs que permitem manipulação de frequência e/ou tensão (Legenda: NC – Nada Consta).....</i>	<i>41</i>
<i>Tabela 4 - Comparativo entre geradores locais de relógio (Legenda: NC - nada consta).....</i>	<i>46</i>
<i>Tabela 5 - Comparativo de ocupação de área (em número de LUTs de quatro entradas) para códigos Johnson e Gray.....</i>	<i>58</i>
<i>Tabela 6 - Comparativo de ocupação de área (em número de LUTs de quatro entradas) para códigos Johnson, Gray e Johnson com conversão direta para binário. ....</i>	<i>59</i>
<i>Tabela 7 - Relação entre entrada fornecida ao DCO e a frequência de saída.....</i>	<i>93</i>
<i>Tabela 8 - Configuração do ambiente de simulação para pior, melhor e caso típico de simulação.....</i>	<i>94</i>
<i>Tabela 9 - Relação entre entrada fornecida ao DCO com compensador de PVT e a saída de frequência do oscilador. ....</i>	<i>100</i>
<i>Tabela 10 - Dissipação de potência do DCO conforme entrada sel e caso testado. ....</i>	<i>102</i>
<i>Tabela 11 - Relatório simplificado de módulos do MPSoC HeMPS-GLP em nível de porta.....</i>	<i>111</i>



## LISTA DE SIGLAS

<i>ASIC</i>	<i>Application-Specific Integrated Circuit</i>
<i>CI</i>	<i>Circuito Integrado</i>
<i>CMOS</i>	<i>Complementary Metal-Oxide-Semiconductor</i>
<i>CPU</i>	<i>Central Processing Unit</i>
<i>CTS</i>	<i>Comprehensive Transponder System</i>
<i>DA</i>	<i>Digital Analógico</i>
<i>DAC</i>	<i>Digital Analog Converter</i>
<i>DCO</i>	<i>Digitally Controlled Oscillator</i>
<i>DFS</i>	<i>Dynamic Frequency Scaling</i>
<i>DLL</i>	<i>Delay Locked Loop</i>
<i>DMA</i>	<i>Direct Memory Access</i>
<i>DVFS</i>	<i>Dynamic Voltage and Frequency Scaling</i>
<i>DVS</i>	<i>Dynamic Voltage Scaling</i>
<i>EP</i>	<i>Elemento de Processamento</i>
<i>FF</i>	<i>Fast-Fast</i>
<i>FIR</i>	<i>Finite Impulse Response</i>
<i>FPGA</i>	<i>Field-Programmable Gate Array</i>
<i>FSL</i>	<i>Fast Simplex Link</i>
<i>GALS</i>	<i>Globally Asynchronous Locally Synchronous</i>
<i>GAPH</i>	<i>Grupo de Apoio ao Projeto de Hardware</i>
<i>GLP</i>	<i>GALS Low-Power</i>
<i>GLR</i>	<i>Gerador Local de Relógio</i>
<i>GP</i>	<i>General Purpose</i>
<i>GRLS</i>	<i>Globally Ratiochronous Locally Synchronous</i>
<i>HardNoC</i>	<i>Hardware Network on-a-Chip</i>
<i>HDL</i>	<i>Hardware Description Language</i>
<i>HeMPS</i>	<i>Hermes MPSoC</i>
<i>HeMPS-GLP</i>	<i>Hermes MPSoC GALS Low-Power</i>
<i>IP</i>	<i>Intellectual Property</i>
<i>ISS</i>	<i>Instruction set Simulator</i>

<i>JTAG</i>	<i>Joint Test Action Group</i>
<i>LCG</i>	<i>Local Clock Generator</i>
<i>LP</i>	<i>Low-Power</i>
<i>LUT</i>	<i>Look-Up Table</i>
<i>MBLite</i>	<i>MicroBlaze Lite</i>
<i>MJPEG</i>	<i>Motion Joint Photographic Experts Group</i>
<i>MOS</i>	<i>Metal Oxide Semiconductor</i>
<i>MPEG</i>	<i>Moving Picture Experts Group</i>
<i>MPSoC</i>	<i>Multiprocessor System on-a-Chip</i>
<i>NA</i>	<i>Não se Aplica</i>
<i>NI</i>	<i>Network Interface</i>
<i>NoC</i>	<i>Network-on-a-chip</i>
<i>NPU</i>	<i>Network Processing Unit</i>
<i>PE</i>	<i>Processing Element</i>
<i>PID</i>	<i>Proporcional Integral Derivativo</i>
<i>PLL</i>	<i>Phase Locked Loop</i>
<i>PMU</i>	<i>Power Management Unit</i>
<i>PVT</i>	<i>Process, Voltage and Temperature</i>
<i>RFID</i>	<i>Radio Frequency Identification</i>
<i>RTL</i>	<i>Register Transfer Level</i>
<i>SPL</i>	<i>Southern Programmable Logic Conference</i>
<i>SS</i>	<i>Slow-Slow</i>
<i>TT</i>	<i>Typical-Typical</i>
<i>VFI</i>	<i>Voltage and Frequency Island</i>
<i>VHDL</i>	<i>Very High Speed Integrated Circuit Hardware Description Language</i>
<i>VOPD</i>	<i>Video Object Plane Decoder</i>

## LISTA DE SÍMBOLOS

$\alpha$	<i>Constante derivada experimentalmente</i>
$\mu$	<i>Mobilidade dos elétrons</i>
$A$	<i>Fração de portas que estão em atividade</i>
$C$	<i>Capacitância total de todas as portas do circuito</i>
$C'_{ox}$	<i>Capacitância de óxido de silício por área</i>
$C_{in}$	<i>Capacitância de entrada do próximo inversor</i>
$C_{out}$	<i>Capacitância de saída do inversor anterior</i>
$C_{ox}$	<i>Capacitância do óxido de silício da tecnologia</i>
$C_{tot}$	<i>Capacitância total de um nodo</i>
$f_{osc}$	<i>Frequência de oscilação</i>
$GND$	<i>Terra</i>
$HP$	<i>High-power</i>
$Hv_T$	<i>Tensão de limiar alta</i>
$i_D$	<i>Corrente de dreno</i>
$I_{leak}$	<i>Corrente de fuga</i>
$I_{MN}$	<i>Corrente de descarga da porta inversora do oscilador em anel</i>
$I_{MP}$	<i>Corrente de carga da porta inversora do oscilador em anel</i>
$I_{REF}$	<i>Corrente de referência</i>
$L$	<i>Comprimento do transistor</i>
$L_n$	<i>Comprimento do transistor NMOS</i>
$L_p$	<i>Comprimento do transistor PMOS</i>
$LP-Sv_T$	<i>Transistor de baixa potência e tensão de limiar padrão</i>
$Lv_T$	<i>Tensão de limiar baixa</i>
$N$	<i>Número de estágios do oscilador em anel</i>
$P$	<i>Potência dissipada</i>
$Sv_T$	<i>Tensão de limiar padrão</i>
$t_1$	<i>Constante de tempo de subida</i>
$t_2$	<i>Constante de tempo de descida</i>
$Vctrl$	<i>Tensão de controle do oscilador em anel com controle de corrente</i>
$VDD$	<i>Tensão de alimentação do circuito</i>

$v_{DS}$	<i>Tensão de dreno-fonte</i>
$v_{GS}$	<i>Tensão de porta-fonte</i>
$V_{SP}$	<i>Tensão que indica a transição lógica</i>
$v_T$	<i>Tensão de limiar do transistor</i>
$W$	<i>Largura do transistor</i>
$W_n$	<i>Largura do transistor NMOS</i>
$W_p$	<i>Largura do transistor PMOS</i>



# SUMÁRIO

<b>1.</b>	<b>INTRODUÇÃO .....</b>	<b>29</b>
<b>1.1.</b>	<b>Limitações em MPSoCs .....</b>	<b>30</b>
<b>1.2.</b>	<b>Novos paradigmas de sincronização .....</b>	<b>30</b>
<b>1.3.</b>	<b>Redução na dissipação de potência.....</b>	<b>31</b>
<b>1.4.</b>	<b>Objetivos do trabalho.....</b>	<b>32</b>
<b>1.5.</b>	<b>Contribuições .....</b>	<b>32</b>
<b>1.6.</b>	<b>Organização do restante do documento .....</b>	<b>33</b>
<b>2.</b>	<b>TRABALHOS RELACIONADOS .....</b>	<b>35</b>
<b>2.1.</b>	<b>MPSoCs com baixa dissipação de potência .....</b>	<b>35</b>
2.1.1.	Ludovici et al. [LUD11] .....	35
2.1.2.	Almeida et al. [ALM11] .....	36
2.1.3.	Rosa et al. [ROS12].....	37
2.1.4.	Ogras et al. [OGR09].....	38
2.1.5.	Beigné et al. [BEI08].....	39
2.1.6.	Discussão dos trabalhos apresentados.....	40
<b>2.2.</b>	<b>Geradores locais de relógio .....</b>	<b>42</b>
2.2.1.	Klapf et al. [KLA08].....	42
2.2.2.	Sobczyk et al. [SOB08].....	43
2.2.3.	Yadav et al. [YAD12] .....	43
2.2.4.	Höppner et al. [HOP12] .....	44
2.2.5.	Discussão dos trabalhos apresentados.....	45
<b>3.</b>	<b>ARQUITETURA ALVO .....</b>	<b>49</b>
<b>3.1.</b>	<b>NoC Hermes-GLP .....</b>	<b>49</b>
<b>3.2.</b>	<b>MPSoC HeMPS.....</b>	<b>51</b>
3.2.1.	Gerador HeMPS.....	53
<b>4.</b>	<b>ADAPTAÇÕES REALIZADAS SOBRE A REDE HERMES-GLP .....</b>	<b>55</b>
<b>4.1.</b>	<b>Ponteiros de filas bissíncronas.....</b>	<b>55</b>
<b>4.2.</b>	<b>Inibição e liberação de sinal de relógio .....</b>	<b>59</b>
<b>4.3.</b>	<b>Memorização e distribuição de seleção de frequência .....</b>	<b>59</b>
<b>4.4.</b>	<b>Extensão para “n” relógios .....</b>	<b>60</b>

4.5.	Votação para seleção de frequência.....	62
4.6.	Simulação e prototipação da nova NoC Hermes-GLP .....	64
5.	O MPSOC HEMPS-GLP .....	65
5.1.	Alterações na interface gráfica .....	66
5.2.	Alterações na estrutura de cenários.....	69
5.3.	Definição de valor de seleção de máxima frequência.....	70
5.4.	Alterações no <i>microkernel</i> .....	70
5.5.	Alteração da estrutura da interface de rede.....	71
5.6.	Estrutura do hardware e interconexão de módulos HDL.....	72
5.7.	Simulações .....	73
6.	GERAÇÃO LOCAL DE RELÓGIO .....	77
6.1.	Bloco oscilador.....	78
6.1.1.	Oscilador realimentado com elementos de atraso .....	79
6.2.	Bloco atuador .....	81
6.2.1.	Conversores digitais analógicos .....	81
7.	PROJETO DO OSCILADOR CONTROLADO DIGITALMENTE .....	85
7.1.	Considerações iniciais do projeto .....	85
7.2.	Análise do oscilador em anel com controle de corrente .....	86
7.2.1.	Inibição de sinal de relógio .....	86
7.3.	Conversor digital analógico .....	89
7.4.	Oscilador controlado digitalmente .....	91
7.5.	Simulações e análises preliminares .....	92
7.6.	Compensador de variações PVT.....	93
7.7.	Operação e características do DCO.....	96
8.	HEMPS-GLP COM GLR .....	105
8.1.	Modelamento do GLR .....	105
8.2.	Integração do modelo do GLR ao MPSoC HeMPS-GLP .....	108
8.3.	Gerador HeMPS-GLP com geração local de relógio .....	108
8.4.	Gerador HeMPS-GLP sintetizável .....	109
9.	CONCLUSÕES E TRABALHOS FUTUROS .....	113
9.1.	Conclusões .....	113
9.2.	Trabalhos futuros .....	114

<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>117</b>
<b>APÊNDICE A .....</b>	<b>121</b>
<b>A.1. Codificação de ponteiros Johnson.....</b>	<b>121</b>
<b>A.2. Codificação de ponteiros <i>Gray</i>.....</b>	<b>122</b>
<b>A.3. Codificação de ponteiros <i>Johnson</i> com conversor binário .....</b>	<b>123</b>



## 1. INTRODUÇÃO

Com o avanço de tecnologias nanométricas empregadas na indústria de semicondutores acentua-se a disponibilidade de projetos de circuitos integrados (CIs) de baixa potência (do inglês, *low-power* ou LP). Este avanço contribui para o desenvolvimento de equipamentos multifuncionais portáteis, tais como telefones inteligentes (do inglês, *smart-phones*), onde não somente a capacidade de processamento, mas também a durabilidade de bateria tornam-se palavras-chave.

Em função da demanda por tais produtos e da grande capacidade de integração atual, aliado a restrições para operar com processadores monolíticos nas tecnologias atuais, sistemas multiprocessados em chip vem sendo usados de forma crescente, e constituem um tema de pesquisa de alta relevância.

Um sistema multiprocessado em chip (do inglês *multiprocessor system on-a-chip* ou MPSoC), como pode ser observado na Figura 1, é um agrupamento de módulos de propriedade intelectual (do inglês, *intellectual property core*, abreviado para IP core ou simplesmente IP), interconectados por uma arquitetura de comunicação, e que conta com múltiplos processadores integrados em uma mesma pastilha de silício (em inglês, *die*) [JER05]. Para interconectar IPs modernamente, escolhe-se em muitos casos uma rede intrachip (do inglês *network on-a-chip* ou NoC) para implementar a arquitetura de comunicação do MPSoC. Redes intrachip propiciam a troca de informações simultânea entre os múltiplos IPs de um MPSoC. Tipicamente, NoCs são formadas por roteadores com conexões ponto a ponto com seus vizinhos, que podem ser outros roteadores ou IPs, podendo formar uma rede malha-2D como apresentado na Figura 1, ou outra topologia. Um roteador com frequência conecta-se a um único IP, denominado IP local do roteador. Este último utiliza a arquitetura de comunicação para receber/enviar dados de/para outros IPs.

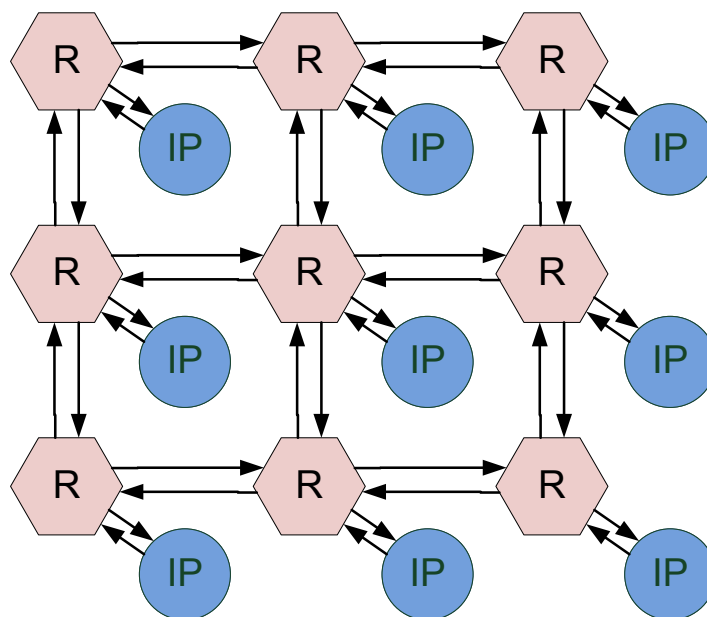


Figura 1. MPSoC baseado em NoC (Legenda: R → Roteador, IP núcleo de propriedade intelectual ou elemento de processamento).

Para MPSoCs tradicionais, todos os módulos internos são síncronos entre si, ou seja, possuem um mesmo sinal de relógio. Entretanto, com o advento de tecnologias na-

nométricas e a grande capacidade de integração proporcionada, novos paradigmas de sincronização começam a ser empregados.

### 1.1. Limitações em MPSoCs

Atualmente, diversos projetos envolvendo MPSoCs com centenas de IPs estão apresentando dificuldades em relação à sincronização. Como as capacitâncias de porta dos transistores de tecnologias atuais, como 65nm, por exemplo, são na casa dos *femto-Farads* (fF), os efeitos parasitas dos fios começam a ser mais perceptíveis [HO01]. Em função da resistência parasita das dimensões mínimas e das resistências, capacitâncias e indutâncias dos fios, a constante de tempo dos fios está aumentando e reduzindo a eficiência de propagação de sinais.

Estes efeitos já começam a ser percebidos em projetos como aquele descrito por Vangal et al. em [VAN08]. Estes autores identificam problemas de sincronização entre seus módulos IPs devido ao escorregamento de relógio (do inglês, *clock skew*). O escorregamento de relógio é um efeito de retardo na chegada deste sinal em um determinado módulo não alterando sua frequência, mas sua fase em relação a pontos distintos do sistema. Uma das técnicas utilizadas para contornar este problema é compensar o atraso das fases por meio de amplificadores ou células de atraso. Assim, as bordas voltam a ser coincidentes, mas o aumento de área e de potência dissipada é facilmente percebido.

Outra limitação de MPSoCs com centenas de módulos de processamento é a potência consumida por estes dispositivos. Apesar das tecnologias nanométricas reduzirem a tensão de alimentação, a grande quantidade de módulos IP agrega área considerável. Por sua vez, o aumento de área resulta em elevação na potência dissipada, contrariando a perspectiva atual de redução de potência.

Tendo em vista os problemas relacionados anteriormente, novos paradigmas de sincronização de módulos IP e a dissipação de potência destes dispositivos vem sendo estudados de forma intensa.

### 1.2. Novos paradigmas de sincronização

Segundo Calazans [CAL98], um sistema digital permite abstrair sinais de entrada ou saída a partir de grandezas físicas como tensão e corrente em sinais discretos considerados níveis lógicos ou binários. Assim permite-se trabalhar com sinais discretos ao longo do tempo facilitando sua manipulação. Além deste nível de abstração dos valores das grandezas físicas, pode-se acrescentar uma camada de abstração com relação ao tempo. Se o tempo também é considerado como composto de um conjunto de momentos discretos, o sinal digital é dito síncrono. A maneira mais comum de implementar a discretização do tempo é utilizar um sinal de controle periódico que comanda todas as ações do sistema digital simultaneamente. Tal sinal normalmente é denominado *relógio* (em inglês *clock*). Quando o sinal digital não trabalha com tempo discretizado, este sinal digital é dito assíncrono.

Mais profundamente, um sistema digital síncrono é um sistema digital assíncrono onde todos os laços de realimentação passam através de um elemento de memória controlado por um sinal de relógio. Para permitir a correta memorização dos valores, devem-

se observar os tempos de estabelecimento (do inglês, *setup time*) e o tempo de manutenção (do inglês, *hold time*) onde o sinal de entrada do circuito de memória deve permanecer estável para garantir sua correta retenção [CAL98].

Assim, em sistemas síncronos o problema consiste em fazer com que os tempos necessários para a correta memorização sejam cumpridos. Atendendo estes períodos mínimos, obtêm-se a máxima frequência de operação do circuito síncrono. Qualquer frequência abaixo desta atenderá os requisitos de tempo de *setup* e *hold*.

Como dito anteriormente, manter a operação de circuitos digitais síncronos em dispositivos com quantidade massiva de módulos começa a causar dificuldade ao projeto de sistemas. Desta forma, sistemas assíncronos que não dependem deste sinal global de discretização no tempo se apresentam como possíveis soluções.

Por outro lado, sistemas assíncronos são sistemas que assumem sinais binários, mas não assumem o pressuposto de discretização do tempo, isto é, em sistemas assíncronos o tempo é tratado como uma variável contínua [SPA02]. Este tipo de circuito elimina os problemas de escorregamento citados anteriormente. 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 abordados anteriormente, a falta de ferramentas para dar suporte a circuitos totalmente assíncronos, e a grande popularidade do estilo síncrono, alternativas vem sendo pesquisadas como soluções intermediárias entre o projeto síncrono e assíncrono. O principal objetivo é manter as ferramentas do projeto síncrono e reduzir o uso de sincronização através do sinal de relógio. Entre essas propostas, pode-se destacar o uso de sistemas globalmente assíncronos localmente síncronos (em inglês *globally asynchronous locally synchronous* ou GALS).

Sistemas GALS visam unir a facilidade de projetos síncronos com as vantagens da comunicação assíncrona. Para tanto, localmente cada módulo possui um sinal de relógio que sincroniza internamente todas suas operações. Entretanto, ao se comunicar com módulos externos, utilizam-se circuitos assíncronos especiais (os *sincronizadores*), que permitem a troca de dados entre dois ou mais domínios distintos de relógio. Assim, o desenvolvimento de cada módulo é facilitado pelo uso de paradigmas de projeto síncrono, ficando a parte assíncrona restrita apenas às interfaces entre módulos do sistema.

### 1.3. Redução na dissipação de potência

Com o desenvolvimento de MPSoCs de grandes dimensões (na casa das centenas de módulos), a potência dissipada tende a ser uma das principais preocupações com estes projetos podendo chegar à casa das centenas de Watts. A potência dissipada por circuitos digitais pode ser estimada pela Equação (1) [KIM03] onde se observa tanto a influência da frequência quanto da tensão de alimentação. Nesta Equação,  $P$  corresponde à potência,  $A$  é a fração de portas que estão em atividade,  $C$  corresponde à capacitância total de todas as portas do circuito, enquanto que  $VDD$  representa a tensão de alimentação,  $f_{osc}$  é a frequência de oscilação e  $I_{leak}$  a corrente de fuga. Por estes motivos, atualmente pesquisas para reduzir esta dissipação vêm sendo realizadas envolvendo a manipulação de tensão de alimentação e de frequência de operação.

$$P = A \cdot C \cdot VDD^2 \cdot f_{osc} + VDD \cdot I_{leak} \quad (1)$$

Como a Equação (1) apresenta, ao se aplicarem técnicas de seleção dinâmica de tensão (em inglês *dynamic voltage scaling* ou DVS), poder-se-ia reduzir o consumo do circuito quadraticamente. Entretanto, observando a Equação (2) [KIM03], onde  $V_t$  é a tensão de limiar (em inglês *threshold*) e  $\alpha$  é uma constante derivada experimentalmente, ao manipular-se esta tensão, reduz-se a frequência de operação, implicando a necessidade de uma seleção dinâmica de frequência e tensão (em inglês *dynamic voltage and frequency scaling* ou DVFS). Esta dependência pode gerar comportamento inesperados como a perda de prazos de aplicações (em inglês *deadlines*) ou até o desrespeito dos tempos de estabelecimento ou de manutenção necessários para a correta memorização dos dados em circuitos digitais.

$$f_{osc} \propto \frac{(VDD - v_t)^\alpha}{VDD} \quad (2)$$

Por outro lado, ao se aplicarem técnicas de seleção dinâmica de frequência (em inglês *dynamic frequency scaling* ou DFS), as variações nos tempos de *setup* e *hold* causados pela tensão não ocorrem, podendo-se variar livremente a frequência entre mínima e máxima suportadas pelo circuito. Desta forma, pode-se ter certeza da velocidade do circuito o que contribui para cumprir os prazos requeridos pelas aplicações.

#### 1.4. Objetivos do trabalho

Este trabalho tem como objetivo desenvolver uma plataforma que permita manipular a dissipação de potência de um dispositivo MPSoCs por meio de técnicas de DFS focando-se sua aplicação individual em cada módulo IP. Ainda propõe-se um novo oscilador controlado digitalmente (em inglês *digitally controlled oscillator* ou DCO) para geração local de relógio (em inglês *local clock generator* ou LCG) com suporte a DFS e inibição de relógio sem geração de espúrios, visando integração com os módulos IP do MPSoC, bem como com roteadores da NoC deste. Como se busca integrar este LCG a todos os IPs, deve-se observar quesitos como área ocupada, dissipação de potência e compensação de variações de processo de fabricação (P), de tensão (V) e de temperatura (T), ou seja, variações de PVT.

#### 1.5. Contribuições

As principais contribuições deste trabalho são focadas em:

- Desenvolvimento de um MPSoC com suporte a DFS e capacidade de inibição de relógio;
- Desenvolvimento de ambiente de simulação comportamental com frequências ajustáveis para o MPSoC projetado;
- Desenvolvimento de Oscilador Controlado Digitalmente para um módulo Gerador Local de Relógio de baixo custo em área e reduzido consumo de energia;



- Desenvolvimento de ambiente de geração de código em linguagem de descrição de hardware (em inglês *hardware description language* ou HDL) para síntese em circuito integrado de aplicação específica (em inglês *application-specific integrated circuit* ou ASIC) do MPSoC com suporte à LCG.

## 1.6. Organização do restante do documento

O restante desse documento está organizado em oito Capítulos, com os conteúdos descritos a seguir.

No Capítulo 2 discutem-se trabalhos relacionados com o apresentado aqui. Neste mesmo Capítulo analisa-se cada trabalho, ponderando prós e contras de cada abordagem e posiciona-se este trabalho em relação ao estado da arte.

Após, no Capítulo 3, apresenta-se a arquitetura utilizada como base pelo Autor para o desenvolvimento das atividades dedicadas a atingir os objetivos descritos na Seção 1.4.

O Capítulo 4 descreve as alterações conduzidas na rede Hermes-GLP (Hermes GALS *low-power*) e as primeiras contribuições deste trabalho. O objetivo destas modificações é propiciar a integração correta desta nova rede com o restante da infraestrutura do MPSoC HeMPS. Demonstrou-se a operação correta do roteador após as alterações na plataforma de testes HardNoC (*hardware network on-a-chip*) [HEC12].

Posteriormente, o Capítulo 5 discute as modificações em hardware e software no MPSoC HeMPS visando a implementação de técnicas de DFS em seus módulos. Também se expõe as alterações no ambiente de geração do MPSoC e apresentam-se simulações que validam as novas funcionalidades

A seguir, o Capítulo 6, propõe e justifica as características básicas do novo gerador local de frequência proposto, um dos principais módulos alvo deste trabalho. Além disto, listam-se as bases de eletrônica analógica necessárias para o dito projeto, os circuitos de base, sua análise e definições necessárias aos conteúdos descritos no Capítulo 7.

No Capítulo 7, detalha-se o projeto do oscilador controlado digitalmente proposto. Expõem-se as especificações do projeto, sua concepção e resultados preliminares, apresentam-se e justificam-se as adaptações sugeridas e implementadas e as mostra-se as simulações finais do oscilador.

O Capítulo 8 expõe o modelamento do gerador local de relógio e sua integração com o MPSoC HeMPS-GLP. Desenvolve-se um ambiente de simulação do MPSoC em versão prototipável em ASIC, utilizando bibliotecas *standard-cell* STMicroelectronics 65nm, constrói-se o circuito do MPSoC para extração de figuras de dissipação de potência e área preliminares e compara-se estes dados com dados do oscilador controlado digitalmente anteriormente desenvolvido.

Por fim, o Capítulo 9 apresenta as conclusões gerais desse trabalho, juntamente com rumos interessantes para a continuação do mesmo.



## 2. TRABALHOS RELACIONADOS

Este Capítulo revisa o estado da arte em dois temas principais relacionados a este trabalho: MPSoCs que visam proporcionar baixa dissipação de potência e sistemas de geração de relógio para módulos digitais em sistemas integrados.

### 2.1. MPSoCs com baixa dissipação de potência

Esta Seção abordará pesquisas recentes sobre o desenvolvimento de MPSoCs insensíveis à distribuição de sinal de relógio e que, de alguma maneira, manipulam ou permitem a manipulação da frequência de operação de seus módulos, com o objetivo de obter redução na potência dissipada. Para tanto, foram consultadas bases de conhecimento de diversas conferências nas áreas de MPSoCs, redução de potência e técnicas GALS com alvo em redução de potência.

#### 2.1.1. Ludovici et al. [LUD11]

Ludovici et al. comparam os ganhos obtidos ao utilizar topologia de MPSoCs GALS que empregam NoCs mesócronas (aquelas que admitem atraso arbitrários nas linhas de distribuição de relógio entre seus roteadores, ver Figura 2), em relação a abordagens de uso de NoCs síncronas e/ou multissíncronas. A intenção é comprovar a redução de dissipação de potência ao se aplicar técnicas GALS a NoCs mesócronas quando comparado a sistemas completamente síncronos ou multissíncronos. Para tanto, os autores utilizam MPSoCs baseados em ilhas de frequência e tensão (em inglês *voltage and frequency islands* ou VFIs). O conceito de VFI remete à divisão de um MPSoC em regiões distintas chamadas *ilhas síncronas*, que possuem cada uma independência em relação às demais em relação à tensão de alimentação e à frequência de operação.

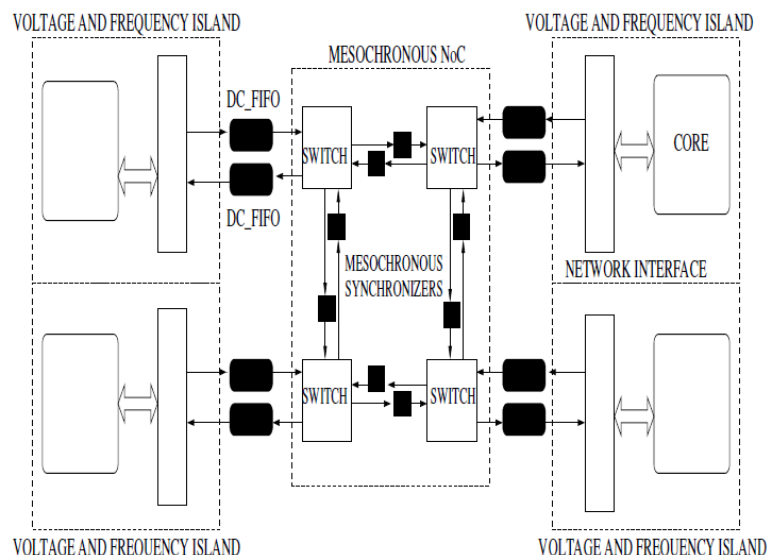


Figura 2: MPSoC com VFIs e NoC mesócrona [LUD11].

O artigo compara diferentes topologias GALS e diferentes estratégias de sincronização. A intenção é reduzir o consumo de energia causado pela árvore de distribuição de relógio e o número e tamanho das filas bissíncronas entre roteadores. A primeira topologia analisada integra o roteador (aqui chamado de *switch*) na VFI e utiliza filas bissíncro-

nas (aqui denominadas *DC\_FIFO*) para se conectar com as demais ilhas. Entretanto, ao transferir um pacote pela rede, a velocidade de transmissão será limitada à da ilha mais lenta em seu caminho. A segunda topologia utiliza uma rede síncrona onde todos os roteadores estão isolados em uma única ilha e se comunicam com as demais ilhas (PEs) via filas bissíncronas. Neste caso, os autores apontam que o problema encontra-se no custo necessário para que distribuir o relógio a todos os roteadores sem escorregamento de relógio.

A solução apresentada pelos autores aparece na Figura 2, e utiliza módulos sincronizadores mesócronos híbridos nos canais de comunicação entre os roteadores da NoC. Desta forma, admite-se o escorregamento de relógio entre os roteadores, constituindo uma rede mesócrona. Estes sincronizadores possuem custo em área muito inferior a filas bissíncronas tradicionais, pelo fato das frequências de operação serem nominalmente idênticas, apenas diferindo em fase.

Para avaliar os benefícios, comparam-se gastos de dissipação de potência com relação ao tráfego injetado na rede. Também se avalia a redução na dissipação de potência proporcionada pela admissão de escorregamento de relógio frente à árvore de distribuição de relógio. Segundo os Autores, quando comparada às dissipações de potência de duas redes 4x4 em malha, a rede mesócrona dissipa cerca de 20% a mais que a rede síncrona quando o sistema está ocioso. Entretanto, ao simular tráfego uniforme, há redução na dissipação de potência da rede mesócrona de 10% em relação à outra. Em relação à dissipação de potência da árvore de relógio devida à rede, esta técnica apresenta reduções de até 40%. Comparado com o gasto total da árvore global, os autores citam reduções de potência de até 20%.

### 2.1.2. Almeida et al. [ALM11]

Almeida e colaboradores apresentam um sistema de DFS para módulos IP em um MPSoC GALS baseado em NoC. O relógio é acionado através de um controlador de ganho proporcional, integral e derivativo (PID) que age baseado no monitoramento da vazão de dados, conforme apresenta a Figura 3.

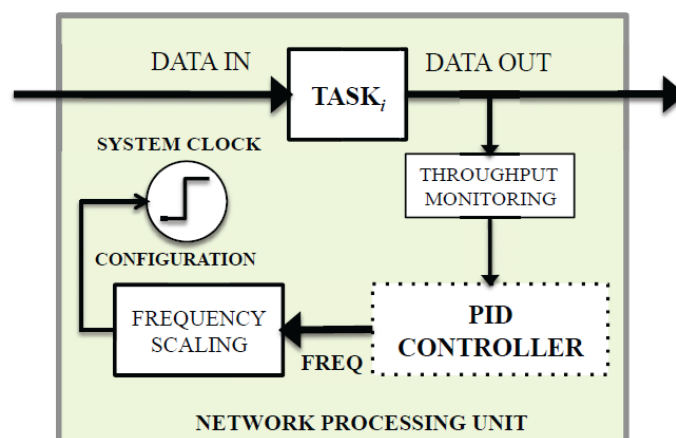


Figura 3. Diagrama de blocos do sistema de DFS do módulo IP proposto por Almeida et al. [ALM11].

Os módulos IP denominados pelo autor como NPU (*Network Processing Unit*) são formados por memória, controlador de relógio e o processador Plasma. A rede é baseada

na NoC Hermes [MOR04]. O controlador de relógio é formado por um monitor de vazão de dados (em inglês *throughput monitoring*), por um controlador PID (em inglês *PID controller*), por um seletor de frequência (em inglês *frequency scaling*) e por um configurador de relógio do sistema (em inglês *system clock configuration*). Por estes blocos, dispostos na Figura 3, o controlador de relógio monitora a vazão de dados que alimenta o bloco de controle PID que por sua vez comanda o sistema de seleção de frequência.

A intenção dos autores é controlar, por intermédio da seleção de frequência, a velocidade necessária para atender as aplicações da NPU. Para tanto, utiliza-se um sistema que monitora a vazão de dados necessária e a disponibilizada pela NPU para uma dada aplicação. Ao identificar uma incoerência entre estes valores, o controle PID atua e seleciona uma nova frequência de operação. Apesar de ser utilizada para garantir a vazão de determinada aplicação, segundo os autores, pode-se configurar o controlador de relógio para reduzir a potência dissipada pela NPU.

Foram implementadas três aplicações reais executadas simultaneamente no MP-SoC desenvolvido e validadas por simulação: um decodificador ADCMP (áudio), um filtro de resposta finita ao impulso (em inglês *finite impulse response* ou FIR) e um decodificador de vídeo MJPEG (*Motion Joint Photographic Experts Group*). Como resultado final, o sistema com controlador PID apresenta um ganho de quase 50% em relação ao cenário sem tal controle.

### 2.1.3. Rosa et al. [ROS12]

Rosa e colaboradores propõem um MPSoC baseado em NoC com DFS, usando um subsistema alimentado por dois valores inteiros que representam uma razão (numerador e denominador da mesma), ou seja, um número racional qualquer. Por agir a partir da liberação e inibição de determinados ciclos de um único relógio de referência, este MP-SoC pode ser considerado como um sistema globalmente ratiócrono, localmente síncrono (em inglês *globally ratiochronous locally synchronous* ou GRLS), visto como um caso especial de sistema GALS.

O controlador DFS, responsável por manipular o relógio global, desenvolvido para os módulos IP, utiliza, para determinar os valores de numerador e denominador, a taxa de ocupação da memória de transmissão/recepção de dados através da rede (os chamados “*buffers*” de rede). Seguindo a política apresentada na Tabela 1, o numerador e o denominador são alterados a fim de aumentar ou diminuir a frequência de atuação do IP. O módulo de geração de relógio que trabalha com valores de *num* e *den* de quatro bits, gasta somente 12 flip-flops, 31 LUTs (*Look-Up-Tables*) e dois *buffers* de relógio (BUFG). Este controlador DFS permite gerar até 120 multiplicadores distintos para a frequência de base.

Os roteadores, assim como os módulos IPs, possuem internamente um controlador DFS. Para determinar os valores de numerador e denominador, o controlador DFS do roteador consulta a metade superior do primeiro *flit* do pacote recebido em alguma de suas portas e coleta os valores de numerador e denominador. Este valor é então comparado aos demais recebidos em outras portas e assume-se a maior frequência dentre estas.

Estes valores de numerador e denominador são determinados a partir da maior frequência entre o módulo IP de origem e o de destino do pacote.

Tabela 1 - Política do controlador DFS de [ROS12]. ↓/↑ implicam diminuir/aumentar a frequência em um ponto, ↑↑ significa aumentar a frequência em dois pontos, = significa manter a frequência inalterada, e – significa valor ignorado.

Ação na Frequência	Mensagens Pendentes	Ocupação Atual do Pipe	Ocupação Anterior do Pipe	Uso da CPU
1 - ↓	0	alto	-	-
2 - ↓	0	médio	baixo	-
3 - ↓	0	baixo	-	Baixo
4 - =	0	médio	médio	-
5 - =	0	baixo	-	Médio
6 - =	1	-	-	Baixo
7 - ↑↑	1	-	-	médio/alto
8 - ↑	0	baixo	-	Alto
9 - ↑	0	médio	alto	-

Para validar o sistema e obter resultados, foram escritas três aplicações: *Pipeline*, *VOPD (Video Object Plane Decoder)* e filtro parcial MPEG. A primeira gera tráfego sintético e as duas últimas geram tráfego real. Segundo os autores, em simulação de tráfego sintético há uma redução de 20,85% na potência dissipada pelos módulos IP e de 72,58% nos roteadores. Em aplicações reais, esta redução chega a 25,89% para os IPs e 75,29% para os roteadores.

#### 2.1.4. Ogras et al. [OGR09]

Ogras e colaboradores abordam técnicas de DVFS aplicadas a ilhas de tensão e frequência (VFIs). Para permitir a comunicação entre os diferentes domínios de tensão e frequência, são utilizadas filas bissíncronas (aqui chamadas de *mixed frequency and mixed voltage FIFOs*) conforme pode ser observado na Figura 4 (a). Inicialmente, as aplicações são analisadas e agrupadas em ilhas conforme necessidades de desempenho e de hardware (módulos heterogêneos) e armazenadas como referência de utilização (*Reference Utilization*). Posteriormente, em tempo de execução, o controle estabelece as seleções dinâmicas de tensão e frequência pelo controle de tensão-frequência baseado na carga de processamento e no menor consumo de energia.

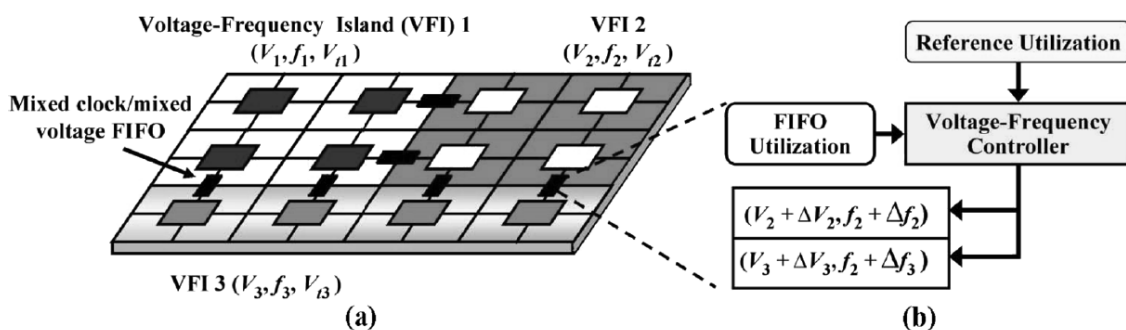


Figura 4. MPSoC com VFIs desenvolvido por [OGR09].

A intenção desta técnica é variar a tensão e frequência para melhorar a dissipação de potência, diminuindo o tempo ocioso dos processadores de uma determinada ilha. Assim, busca-se utilizar apenas a energia necessária para executar as aplicações no prazo (*deadline*) previamente estabelecido.

Os experimentos realizados foram baseados em quatro aplicações diferentes sendo elas sintéticas (produtor-consumidor) e reais (dos domínios de comunicação, indústria automotiva e telecomunicações). Segundo simulações realizadas com o conjunto de *benchmarks* E3S, foram observados reduções tanto no consumo de energia na aplicação sintética (aproximadamente 36%) quanto nas aplicações reais (chegando a 78%) dependendo da quantidade de ilhas de tensão e frequência.

### 2.1.5. Beigné et al. [BEI08]

Beigné et al. desenvolveram um esquema DVFS para MPSoCs GALS que usam NoCs assíncronas (em inglês Asynchronous NoC ou ANoC) como arquitetura de interconexão e com um nodo de rede tendo a estrutura apresentada na Figura 5. Cada módulo IP utiliza uma unidade de suprimento de potência (em inglês *power supply unit*) responsável por multiplexar duas linhas de tensão, um programador de linha de atraso (em inglês *delay line programmer*) que programa a frequência de um gerador de relógio pausável (em inglês *pausable clock generator*), um inibidor de relógio (em inglês, *gated clock*) e um gerenciador para baixa potência (em inglês *low power manager*) que comanda todos estes módulos.

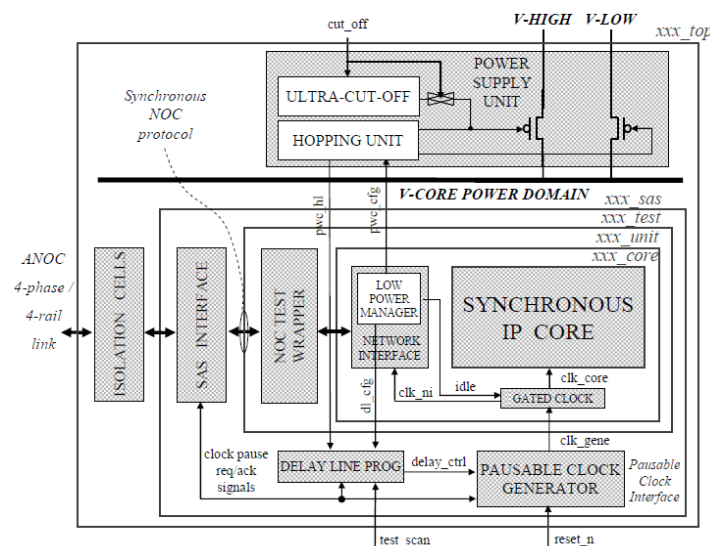


Figura 5. Arquitetura de um nodo de rede do MPSoC proposto por Beigné e colaboradores em [BEI08].

Para o DVS, utiliza-se um módulo chamado unidade de fornecimento de energia. Utilizando o sinal provido pelo gerenciador de energia da interface de rede para comandar o chaveamento de tensões disponíveis (somente dois: alto e baixo), o módulo IP varia sua tensão de operação. Para o DFS, utiliza-se um módulo chamado interface de relógio pausável. Este módulo possui dois submódulos que comandam uma linha de atraso programável e o gerador de relógio pausável. A linha de atraso é utilizada para definir o relógio utilizado no módulo IP. Para estas linhas, utilizam-se osciladores em anel baseados em células padrão (em inglês *standard cells*). A cada alteração na tensão de alimentação, o relógio é pausado e somente liberado após a confirmação de configuração correta em sua frequência. O gerador de relógio pausável utiliza a linha de atraso configurada no módulo responsável quando existe ou dado sendo recebido, dado sendo enviado, ou dado sendo processado. Existe ainda um controle dedicado para inibir o relógio quando o IP esteja inativo.

Tabela 2 - Definição dos estados possíveis de DVFS.

Estado	Tensão	Frequência
INIT	Tensão alta	Inibe o relógio
HIGH	Tensão alta	Frequência nominal (pré-programada)
LOW	Tensão baixa	Frequência abaixo da nominal (pré-programada)
HOPPING	Tensão chaveia entre alta e baixa (DVFS)	Frequência configurável (DVFS)
IDLE	Tensão baixa	Inibe o relógio
OFF	Desliga alimentação	Inibe o relógio

O sistema de DVFS tem seis estados possíveis, detalhados na Tabela 2. Ao estabelecer-se o estado *HOPPING* o sistema de DVFS age dinamicamente conforme programação do módulo IP. Entretanto, o autor não se refere, em momento algum no artigo, como o módulo IP define ou programa estes estados. Segundo os autores, é possível atingir eficiências na dissipação de potência de até 95% utilizando suas técnicas, mas não informa em que situação isto ocorre. Aqui o Autor enfatiza que esta situação pode ser inclusive o estado *OFF* onde desliga-se completamente o módulo IP.

#### 2.1.6. Discussão dos trabalhos apresentados

A Tabela 3 estabelece um paralelo entre o conjunto de trabalhos revisados na Seção 2.1. Os trabalhos anteriormente discutidos são analisados quanto à arquitetura, às técnicas de sincronização, à flexibilização na frequência e/ou tensão, ao controle de DFS ou DVFS e quanto à geração local de relógio. Com estes dados busca-se comparar os diferentes trabalhos que buscam a redução na potência dissipada de um MPSoC sem causar aumento considerável em sua área em silício.

Todos os trabalhos analisados são baseados em NoCs diferenciado-se apenas os trabalhos de Ludovici et al., que permite o escorregamento de relógio em sua NoC mesó-crona e o de Beigné et al., que emprega uma NoC completamente assíncrona. Neste ponto, o trabalho aqui descrito segue a mesma tendência dos demais, e utiliza uma NoC com topologia malha-2D devido a escalabilidade proporcionada por este tipo de rede e sua facilidade de implementação usando tecnologias de fabricação MOS planares.

Com relação à comunicação entre os diferentes domínios de frequência, observa-se a tendência do uso de filas bissíncronas em vários trabalhos, tais como o de Ludovici et al., Rosa et al. e Ogras et al. Este tipo de sincronização obtém melhores resultados em relação à latência de comunicação quando comparado ao uso de *handshake* como propõem Beigné et al. Outro ponto que desfavorece o sistema assíncrono de comunicação proposto por Beigné et al. é sua dificuldade de elaboração e a falta de ferramentas comerciais compatíveis. Por estes motivos, opta-se aqui por utilizar em todas as portas do roteador filas bissíncronas, com o objetivo de reduzir a complexidade do projeto e a latência de comunicação.

Quanto à flexibilização na frequência e na tensão de alimentação, os estudos sobre uso de VFIs, como Ludovici et al. e Ogras et al. apresentam reduções consideráveis no consumo. Entretanto, a diminuição no grão de aplicação de DFS ou DVFS possibilitaria uma redução ainda maior na potência dinâmica dissipada pelos módulos do MPSoC. Outro ponto que contribui diminuindo o grão de aplicação de DFS ou DVFS é observado na flexibilidade que o sistema obtém com relação à migração de tarefas. Assim, podem-se alterar as políticas de DFS/DVFS independentemente por módulo IP. Por este motivo,



opta-se por instanciar filas bissíncronas em todas as portas dos roteadores com o alvo de permitir que cada módulo opere com uma frequência distinta.

Tabela 3 - Comparação entre MPSoCs que permitem manipulação de frequência e/ou tensão (Legenda: NC – Nada Consta).

Autor	Arquitetura	Técnica de sincronização	Flexibilização na frequência e/ou na tensão	Controle de DFS ou DVFS	Geração local de Relógio
Ludovici et al. [LUD11]	MPSoC baseado em NoC mesócrona com topologia malha-2D	Filas bissíncronas entre VFI e NoC e sincronizadores específicos entre roteadores	Entre VFIs	NC	NC
Almeida et al. [ALM11]	MPSoC baseado em NoC com topologia malha-2D	NC	Entre módulos IP locais	Vazão de dados	NC
Rosa et al. [ROS12]	MPSoC baseado em NoC com topologia malha-2D	Filas bissíncronas entre IPs locais e entre roteadores	Entre módulos IP locais e entre roteadores	Ocupação do pipe, mensagens pendentes e uso da CPU	Manipulação de relógio global
Ogras et al. [OGR09]	MPSoC baseado em NoC com topologia malha-2D	Filas de frequência mista e tensão mista entre VFIs	Entre VFIs	Carga de processamento e utilização das filas	Relógios Externos
Beigné et al. [BE108]	MPSoC Baseado em NoC assíncrona com topologia malha-2D	Handshake	Entre módulos IP locais	NC	Linhas de atraso com dois níveis de frequência
<b>Este trabalho</b>	MPSoC baseado em NoC com topologia malha-2D	Filas bissíncronas entre IP locais e entre roteadores	Entre módulos IP locais e entre roteadores	Programável	Linhas de atraso e inibição de relógio

Comparando-se o controle de DFS/DVFS, praticamente todos os trabalhos discutidos utilizam um conjunto predefinido de dados para atuar na seleção dinâmica de tensão e frequência. A intenção aqui se concentra em propor um MPSoC com seleção dinâmica de frequência baseado em software. Assim, tanto o programador das tarefas quanto o do *microkernel* poderão selecionar a melhor frequência conforme suas necessidades. Isto viabiliza a usuários desenvolver em alto nível sistemas de gestão de frequência e potência dissipada.

Por fim, compara-se o modo como se dá a geração local de frequência utilizada para DFS ou DVFS. Rosa et al. desenvolveram um sistema de geração local de frequência baseado na inibição de determinados ciclos de relógio, o que permite gerar frequências inferiores a uma dada referência global. Entretanto, os pulsos de relógio apresentam todos a mesma duração e, no pior caso, a distância entre duas bordas de subida consecutivas do relógio é exatamente igual ao período do relógio de referência. Isso restringe a possibilidade de ação do DFS, pois todos os módulos do sistema devem ser dimensionados para operar corretamente na frequência de referência do MPSoC. Ogras et al., por outro lado, propõem um sistema de seleção de frequências providas externamente. Entretanto, esta topologia requer quantidade considerável de área e energia devido ao acúmulo de múltiplas árvores de distribuição de relógio no circuito. Já Beigné et al. disponibilizam um sistema de geração local de frequência baseado em linhas de atraso. Entretanto, para permitir o ajuste tanto de tensão quanto de frequência, o sistema exige a paralisação do relógio disponibilizado ao módulo IP. Deste modo, pode-se perder um tempo considerável aguardando o ajuste de frequência ou tensão. Tanto Ogras et al. quanto Rosa et al. preveem o uso de um gerador para cada módulo IP ou roteador dependente de frequência. Aqui, propõe-se o desenvolvimento de um gerador local de frequência que

ocupe uma área consideravelmente reduzida quando comparado ao menor bloco individualmente dependente de frequência. Outro ponto importante da proposta deste trabalho é a construção de um mecanismo que permita a troca de frequências ou a inibição de relógio sem a geração de espúrios (em inglês *glitches*) de forma a possibilitar a atuação sem a necessidade de paralisação do relógio.

## 2.2. Geradores locais de relógio

Nesta Seção examinam-se pesquisas recentes realizadas para prover geração local de relógio. Neste ponto, coloca-se o foco na procura por soluções de baixa potência e pequena área de silício. Para tanto, foram consultadas bases de conhecimento de diversas conferências nas áreas de osciladores locais tanto para circuitos digitais quanto para rádio frequência.

### 2.2.1. Klapf et al. [KLA08]

Klapf e colaboradores analisam quatro topologias de geradores de baixo consumo de energia integráveis em etiquetas de identificação por rádio frequência (em inglês *radio frequency identification* ou RFID). Foram analisados osciladores em anel com atraso RC, oscilador em anel controlado por corrente, oscilador em anel com estágios diferenciais e a proposta dos autores, o oscilador de relaxamento com sistema de *transponder* abrangente (em inglês *comprehensive transponder systems* ou CTS), detalhado na Figura 6. Segundo os autores, este último oscilador é menos suscetível a variações de fonte de alimentação e de fonte de corrente.

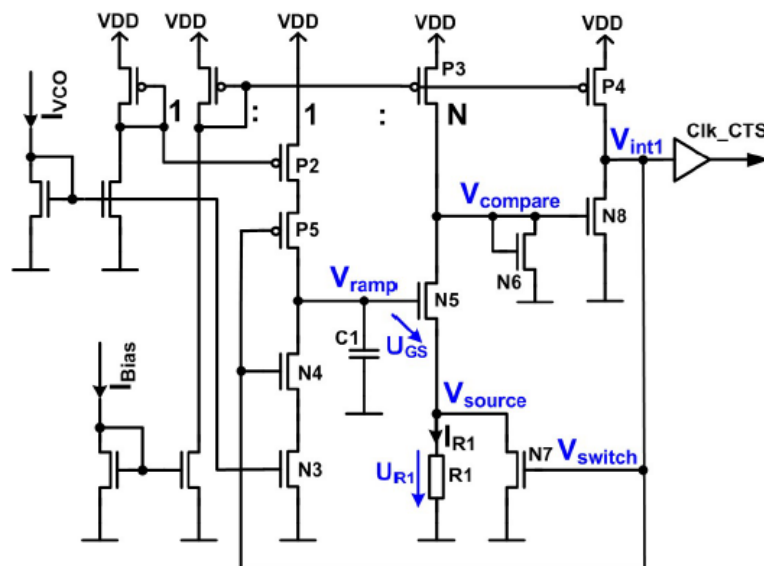


Figura 6. Oscilador de relaxamento com CTS [KLA08].

Este tipo de oscilador requer duas fontes de corrente distintas para controlar a oscilação. Uma corrente controlada fixa  $I_{Bias}$  responsável por polarizar e ajustar a corrente dos nodos do oscilador e uma corrente de controle  $I_{VCO}$ , que permite a seleção de frequência do oscilador. Este oscilador foi projetado utilizando tecnologia de 120nm CMOS.

Os resultados apontam que este projeto é menos suscetível a variações de tensão, de temperatura e da corrente de referência em relação aos demais. A potência dissipada

para gerar a frequência de 3 MHz a uma tensão de alimentação de 1,5V é aproximadamente  $0,9\mu\text{W}$ .

### 2.2.2. Sobczyk et al. [SOB08]

Em [SOB08], Sobczyk et al. apresentam um gerador local de sinal de relógio controlável para arquiteturas GALS. O circuito apresentado possui uma unidade de controle, o oscilador em anel configurável e um estágio de saída. A Figura 7 apresenta o oscilador em anel proposto.

A unidade de controle ativa ou desativa o anel (*ON\_OFF\_RING*), baseado em um sinal de inibição disponibilizado pelo módulo IP ao qual está conectado. Esta unidade de controle sincroniza o sinal de inibição e disponibiliza-o para o oscilador presente na Figura 7. A unidade de controle ainda providencia um sinal para o estágio de saída que libera o relógio para o módulo IP.

O oscilador em anel é constituído de portas *NAND*, *NOT* e *buffers tristate*. Assim, por meio de programação de dois bits ( $S<1:0>$ ), podem-se chavear entre cinco, sete, nove e onze estágios de atraso.

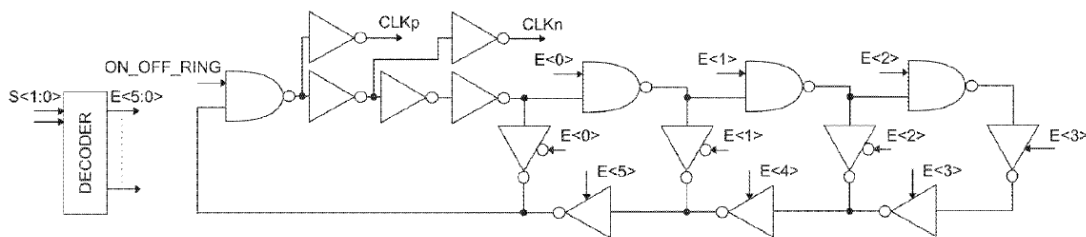


Figura 7. Oscilador em anel com variação de número de estágios [SOB08].

O estágio de saída é constituído por uma porta *NAND* que recebe o sinal de relógio do oscilador em anel e o sinal negado de liberação do relógio proveniente da unidade de controle. A saída ainda possui um inversor.

Os resultados apontam que se podem gerar frequências com 22% de incerteza quando se varia tensão e temperatura. Entretanto, os autores informam no mesmo artigo que ao analisar-se os casos extremos (melhor e pior casos), pode-se ter diferenças de até 70% nas frequências geradas.

### 2.2.3. Yadav et al. [YAD12]

Em sua pesquisa, Yadav e colaboradores desenvolvem um sistema de DVFS que permite manipular a tensão de operação de um circuito e sua frequência de operação inibindo ciclos de relógio de uma frequência globalmente distribuída, algo muito semelhante ao desenvolvido na proposta de Rosa et al. [ROS12]. Segundo os autores, este tipo de técnica de manipulação de frequência dispensa o uso de PLL (*phase locked loop*) ou DLL (*delay locked loop*), e facilita a geração de frequências a partir de uma frequência global. Os autores utilizam uma tabela de seleção de relógios que, por meio de um registrador de escalonamento de 4 bits gera 16 níveis diferentes de relógio, incluindo a inibição completa do relógio. Por meio desta tabela, determinam-se quantos ciclos do relógio global serão liberados e quantos serão inibidos.

Ainda segundo os autores, esta técnica pode ser empregada juntamente com manipulação da tensão de alimentação. Para tanto, deve-se restringir a entrada no registrador de escalonamento a apenas valores inferiores à metade da frequência do relógio global, conforme apresenta a Figura 8. Como ao alterar-se a tensão, altera-se também a velocidade com que a lógica combinacional é executada, o tempo entre as amostragens deverá também ser maior. Com isso, os autores recomendam usar no mínimo a metade da frequência global.

Em uma das simulações utilizando um roteador para NoCs com tecnologia 45nm, foram disponibilizadas duas tensões de alimentação distintas: 1,15V e 0,75V. Os resultados apontam que a área ocupada pelo controlador DVFS ocupa apenas 1,3% da área do roteador.

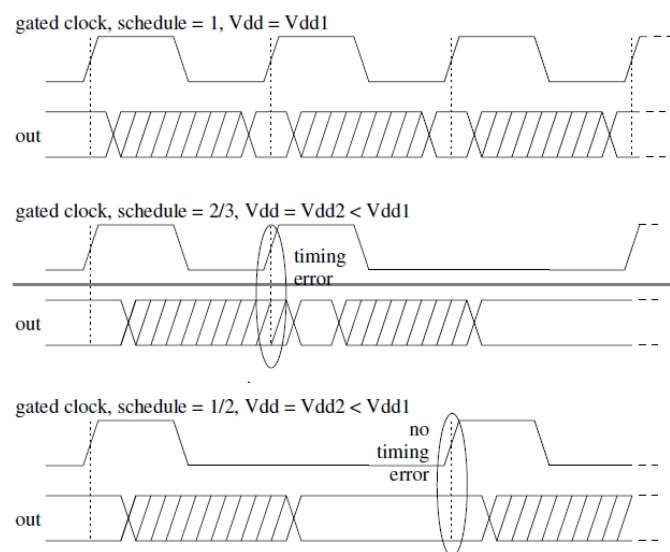


Figura 8. Atuação do gerador de relógio com DVFS variando-se a tensão de alimentação [YAD12].

#### 2.2.4. Höppner et al. [HOP12]

Höppner e colaboradores apresentam um gerador de relógio de laço fechado com fase fixa completamente digital (em inglês *all-digital phase locked loop* ou ADPLL) para geração e seleção rápida de frequências para roteadores e/ou módulos IP. Os autores enfatizam que diversos MPSoCs utilizam múltiplos relógios em seu interior devido às necessidades dos seus diferentes módulos (memórias, processadores e redes intrachip), cada um exigindo uma frequência distinta. A estrutura geral do gerador proposto aparece na Figura 9.

Como é possível observar na Figura 9, o gerador de relógio proposto apresenta um módulo oscilador controlado digitalmente (DCO) comandado por um ajuste grosso (na Figura denominado *coarse ctrl*) e um ajuste fino (que gera o sinal *fine*). A intenção destes ajustes é compensar a variação de processo e, assim garantir uma oscilação constante de 2 GHz. A estrutura do oscilador utilizado (OSC na Figura 9) aparece na Figura 10. Ele possui oito saídas de frequência, defasadas em intervalos de 45 graus.

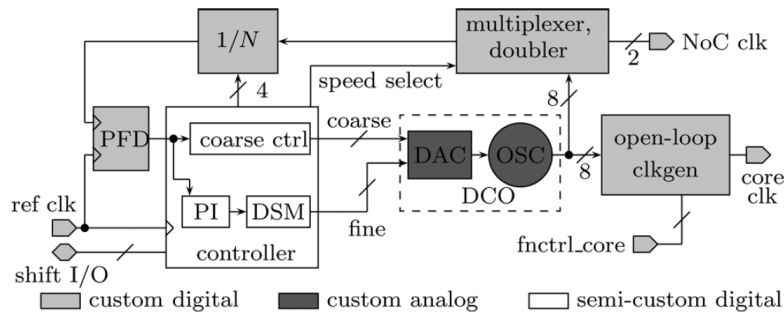


Figura 9. Gerador de relógio ADPLL [HOP12].

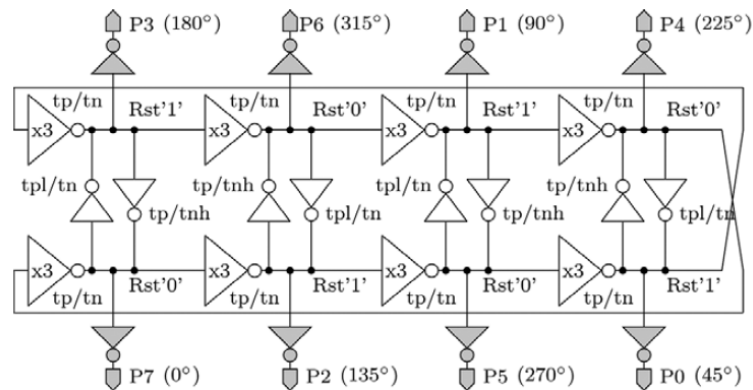


Figura 10. Oscilador em anel utilizado no ADPLL [HOP12].

Por meio de comparadores de sinais defasados, este circuito permite o uso de uma entre duas frequências distintas para os roteadores da rede (2 GHz e 4 GHz). Além desta saída de frequência para o roteador, o circuito ainda possui mais uma dedicada ao módulo IP. Utilizando divisores e multiplexadores que ficam comutando uma das oito saídas defasadas do mesmo oscilador, o gerador ainda pode produzir 33 frequências de relógio distintas, entre 83 MHz e 666 MHz, sempre com ciclo de serviço de 50% (em inglês, *duty cycle*).

Os testes foram realizados em um MPSoC com uma unidade de gerenciamento de energia (em inglês *power management unit* ou PMU) que ainda propicia o uso de seleção de tensão de alimentação. A tabela que comanda o gerador de frequências a partir da PMU é alimentada externamente por uma interface JTAG. Segundo os autores, selecionando a menor frequência possível tanto para o módulo IP (83 MHz) quanto para a rede (2 GHz) e com a menor seleção de tensão disponível, o gerador consome apenas 2,7mW.

### 2.2.5. Discussão dos trabalhos apresentados

A Tabela 4 resume o estado da arte revisado em geradores locais de relógios e posiciona o estudo desenvolvido neste trabalho em relação aos mesmos. Aqui se relaciona circuitos aos quais os geradores são aplicados. Pondera-se também a quantidade de frequências distintas fornecidas em cada estudo e a existência ou não de um sistema de inibição de relógio. Por fim, compara-se o foco do trabalho e os objetivos de cada um dos projetos.

Com relação às aplicações, apenas Klappf et al. desenvolveram seus estudos com foco na geração de relógio para rádio frequência, ou seja, ondas senoidais. Para tanto, exige-se muito da estabilidade, para garantir a portadora para a comunicação da etiqueta

com o sensor. Os demais estudos fixam como alvo aplicações digitais e arquiteturas GALS. Entretanto, Yadav et al. e Höppner et al. endereçam suas atividades para circuitos de correção de fase em seus desenvolvimentos. O interessante é que arquiteturas GALS, em sua grande maioria, já provêm defasagem entre relógios e já utilizam técnicas como paralisação de relógio ou o uso de filas bissíncronas para compensar este efeito. Assim, o Autor aqui sugere abstrair a necessidade de exatidão de frequência para os roteadores da NoC e para os módulos IPs, desde que a frequência oferecida não seja inferior à necessária para atender os prazos de tarefas (*deadlines*). Neste caso sugere-se o uso do relógio de referência para garantir o instante de execução.

Tabela 4 - Comparativo entre geradores locais de relógio (Legenda: NC - nada consta).

Autor	Aplicação	Objetivo	Número de frequências disponibilizadas	Inibição de relógio
Klapf et al. [KLA08]	RFID (senoidal) com PLL	Precisão de frequência e baixo consumo energético	1	NC
Sobczyk et al. [SOB08]	Arquiteturas GALS	Uso de células padrão	4	Sim (abertura do anel de oscilação)
Yadav et al. [YAD12]	Arquiteturas GALS com PLL/DLL (externo)	Minimização de área e consumo energético, e precisão de frequência	16	Sim
Höppner et al. [HOP12]	MPSoCs GALS com DLL	Precisão de frequência, com 50% de <i>duty cycle</i>	33	Não
<b>Este trabalho</b>	Arquiteturas GALS	Minimização de área e dissipação de potência com 50% de <i>duty cycle</i>	Pelo menos 16	Sim (abertura do anel de oscilação)

Em seguida, pode-se comparar os objetivos de cada trabalho. Klapf et al. desenvolveram um gerador de frequência para um RFID. Por este motivo, buscaram um oscilador que fosse para baixa frequência (característica do RFID), preciso e com baixíssimo consumo energético. Isso é necessário, considerando que este tipo de dispositivo se alimenta das próprias ondas eletromagnéticas emitidas pelo seu leitor e armazenadas em uma célula capacitiva ou microbateria. Já Sobczyk et al. buscavam projetar um gerador local de frequências para dispositivos digitais que fosse projetado apenas com células padrão (*standard cells*) para aplicações digitais. Este tipo de circuito gera áreas relativamente pequenas, mas não possui grande fidelidade quanto a variações de PVT, o que prejudica sua utilização. Yadav et al. pesquisaram um meio termo entre uso de células digitais e manipulação de frequências, objetivando área reduzida e baixa potência dissipada e com precisão de fase, mesmo alterando sua tensão de operação. Entretanto, como se baseia em uma frequência global e, devido ao processo de geração de relógio por supressão de pulsos desta, a dita frequência tem valor limitado pelo máximo valor do módulo mais lento do dispositivo. Já Höppner et al. desenvolveram seu projeto objetivando o número de frequências distintas e alguma preocupação com precisão das frequências, além de garantir o *duty cycle* de 50% para todas as frequências geradas. Entretanto, como utiliza DLLs e uma lógica considerável para manipular as oito diferentes fases geradas, este gerador consome energia considerável chegando a 2,7 mW para a frequência mais baixa. Aqui se busca um gerador local de relógio que seja relativamente pequeno quando comparado ao menor módulo dependente de relógio a fim de permitir sua integração individualizada em cada um destes. Privilegia-se também a pesquisa por um gerador de muito baixo consumo de forma que seu uso em cada módulo digital do MPSoC GALS não produza impacto significativo nem na área nem no consumo do circuito final.

Agora, com relação ao número de frequências disponibilizadas, Klapf et al. em sua aplicação de etiquetas RFID apenas necessitam de uma frequência, o que elimina a necessidade de manipulação de outras variáveis que não seja para garantir a exatidão desta. Já o trabalho de Sobczyk et al. gera apenas quatro frequências distintas entre si, pois seu oscilador em anel pode selecionar 5, 7, 9 ou 11 estágios, que fornecem uma incerteza de 22%. Entretanto, os autores apenas consideram a variação do próprio nos limites de contorno da simulação (em inglês *corners*). Quando comparam melhor caso e pior caso, a variação chega a 70% nas frequências o que inviabilizaria seu uso em um circuito comercial. Yadav et al. geram até 16 níveis de frequência, pela eliminação de ciclos de uma frequência distribuída globalmente. Eles os fazem através do uso de uma lógica simples de atuação, o que resulta em área e dissipação de potência diminutas. Entretanto, assim como Rosa et al. em [ROS12], sua frequência global está limitada à máxima frequência de operação do módulo mais lento dependente deste sinal. Outro problema nesta abordagem é a sugestão de que apenas utilizando a metade da frequência global possa-se manipular a tensão de alimentação sem afetar a amostragem do sinal. Isso é possível desde que a lógica combinacional entre as barreiras de memorização seja maior que a desta barreira. Outro ponto falho é que os autores preveem que o circuito apenas atuará em uma das bordas do relógio. Já Höppner et al. desenvolveram um módulo de geração de frequência que disponibiliza 2 frequências distintas para os roteadores e 33 frequências distintas para os módulos IP. Entretanto, as frequências geradas para a rede são pelo menos três vezes superiores à frequência dos IPs. O Autor não consegue identificar um ganho nesta diferença, tendo em vista que será necessário pelo menos um ciclo de relógio do módulo IP local para produzir um dado para a rede e, na outra ponta, um outro módulo semelhante também necessitará um ciclo para coletá-lo. Assim, a rede ficaria subutilizada em grande parte do tempo desperdiçando energia. Sugere-se aqui o desenvolvimento de um gerador de frequência que disponibiliza pelo menos 16 níveis distintos de frequência tanto para o roteador, quanto para o módulo IP, ou seja, um gerador local de frequência de pelo menos 16 níveis para cada um destes módulos.

Por fim, analisando-se a inibição de relógio, Sobczyk et al. utilizam uma simples porta *NAND* que desabilita a realimentação do oscilador em anel. Entretanto, deve-se tomar cuidado para sincronizar este sinal com a frequência do anel de forma a não gerar-se espúrios no sinal de relógio. Já Yadav et al. por já utilizarem uma técnica que inibe ciclos de relógio para gerar frequências menores, facilmente permite a inibição completa deste sinal. Entretanto, o consumo causado por uma árvore global de relógio para MPSoCs de grandes dimensões também deve ser levado em consideração e este não será inibido pelo gerador local de relógio. Outro ponto ainda é que a lógica deste gerador, apesar de parar seu relógio de saída, permanece ativa e consumindo energia. Aqui, sugere-se um sistema que inibe o sinal de relógio sem geração de espúrios em sua saída independente do instante de ação do sinal que seleciona esta opção, além de paralisar por completo o circuito de geração de relógio reduzindo seu consumo.





### 3. ARQUITETURA ALVO

Este Capítulo visa apresentar a arquitetura alvo das atividades desenvolvidas neste trabalho. Na próxima Seção será apresentada a NoC Hermes-GLP, sua operação e suas funcionalidades, que servem aos propósitos deste trabalho. Já, na Seção 3.2, será exposta a arquitetura do MPSoC HeMPS, alvo das atividades de pesquisa e desenvolvimento.

#### 3.1. NoC Hermes-GLP

Com o surgimento de diversas abordagens de comunicação globalmente assíncronas e localmente síncronas, o grupo de apoio ao projeto de hardware (GAPH) veio a desenvolver a rede Hermes-G, uma NoC que, por meio de filas bissíncronas permite a comunicação entre diferentes domínios de relógio [PON08A]. Após este desenvolvimento, o GAPH propôs outra versão de NoC, chamada Hermes-GLP, com a intenção de reduzir a dissipação de potência na rede por meio de utilização de sinais de relógio de frequências diferenciadas e aplicação de técnicas de inibição de sinal de relógio (em inglês, *clock gating*). O diagrama de blocos do roteador Hermes-GLP é detalhado na Figura 11.

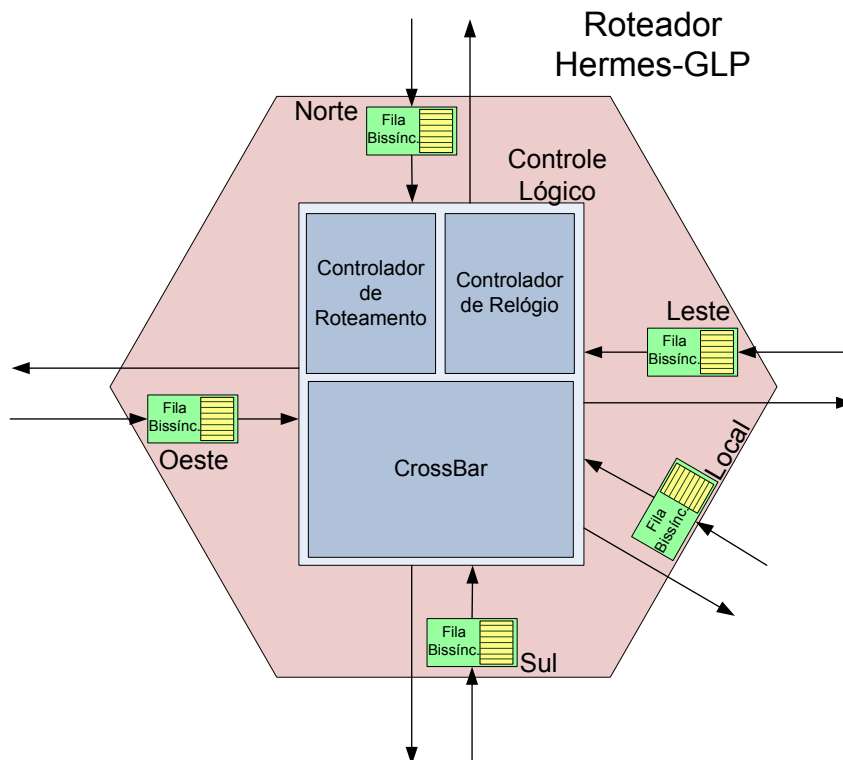


Figura 11. Diagrama de blocos do roteador da rede Hermes-GLP.

O roteador da NoC Hermes original possui em suas portas de entrada filas circulares (em inglês *buffers*) que armazenam temporariamente cada dado recebido, em unidades conhecidas pela denominação *flit* (tipicamente, trata-se de uma subdivisão de um pacote que trafega na rede). Para sua correta operação, ambas as portas devem se comunicar com sinais de relógio em bordas idênticas. Para a utilização de sinais de relógios distintos nos lados interno e externo da NoC, são necessárias filas bissíncronas que possuem, além da qualidade de operar independente das características do relógio em cada

porta (e em cada lado da porta), maior vazão de dados quando comparadas com outras opções de sincronização de domínios de relógio como o uso de sincronizadores ou esquemas de inibição de relógio [PON08B]. Assim, a escrita pode ser realizada em uma determinada frequência enquanto a leitura é realizada em outra. O problema nesta operação é identificar quando a fila está vazia ou cheia e, para isso, são realizadas comparações entre os ponteiros de leitura e escrita usando operações assíncronas. Para se garantir a consistência desta comparação, são utilizados internamente sincronizadores “two flops”, pois com a utilização de bordas de relógios distintas, podem ser gerados efeitos de meta-estabilidade nestes sinais. O uso de filas bissíncronas permite um acoplamento próximo de ótimo entre domínios distintos de relógio.

Nos roteadores Hermes-GLP todas as filas de entrada são bissíncronas. Além disso, conforme se observa na Figura 11, existe o módulo Controlador de Relógio (*clock\_control*) que é responsável pela correta geração de sinal de relógio para o roteador. Este módulo recebe um conjunto de sinais de relógios disponibilizados pelo sistema e conecta ao circuito do roteador um deles (ou nenhum, no caso de *clock gating*) conforme o sinal de seleção de frequência (*entrada\_sel\_freq*) e o estado de cada porta (*estado\_porta*). Após a definição da interconexão entre portas de entrada e saída do roteador (via a rede de multiplexadores e circuitos de controle que constitui o chamado *crossbar* do roteador), o sinal de seleção de frequência é retransmitido da porta de entrada à porta de saída (*saida\_sel\_freq*) do roteador. A Figura 12 apresenta o esquema geral das interconexões dos módulos no interior do roteador.

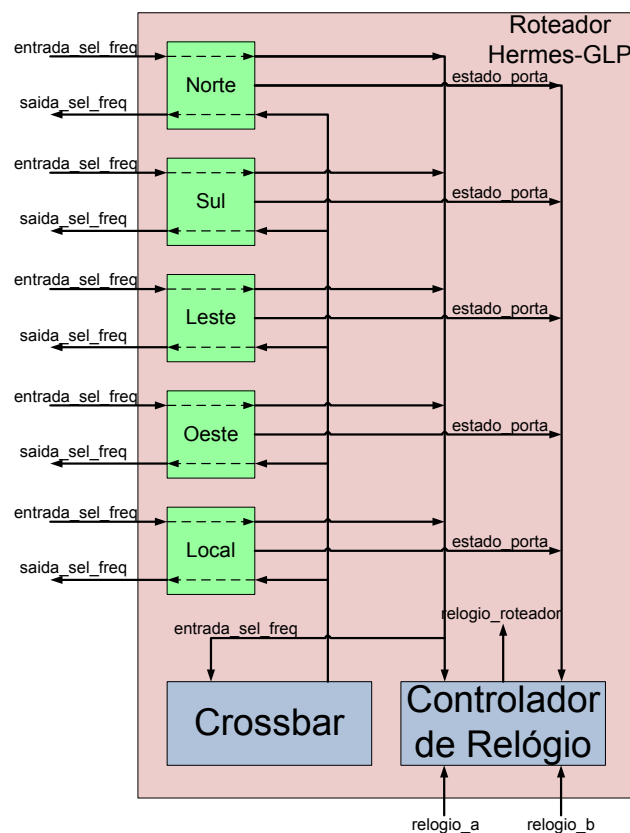


Figura 12: Interconexões utilizadas dentro do roteador Hermes-GLP para a seleção de frequência nos roteadores. Os módulos empilhados à esquerda do desenho representam as portas de entrada e saída. O crossbar é mostrado de forma simplificada apenas detalhando o fluxo de sinais de controle da frequência de relógio das diferentes portas.

O bloco Controlador de Relógio recebe duas frequências distintas (“*relogio\_a*” e “*relogio\_b*”). Note-se que o número de frequência pode ser maior do que 2, o que pode ser visto como um parâmetro para a geração da rede. Os sinais “*entrada\_sel\_freq*” recebidos por este bloco de cada porta de entrada determinam qual destas frequências será utilizada pelo roteador, resultando em um esquema de prioridade de tráfego para pacotes que podem trafegar por este roteador com maior ou menor velocidade, dependendo dos valores dos sinais “*entrada\_sel\_freq*”. Neste caso simples com duas opções de frequência apenas, se existem “*entrada\_sel\_freq*” diferentes, o bloco Controlador de Relógio faz uma eleição levando em conta este sinal somente das portas que possuem algum dado a ser transmitido (identificados pelo sinal “*estado\_porta*”). Assim, a porta que necessita de maior velocidade e possui dado para a transmissão é a determinante da velocidade de operação do roteador. Entretanto, quando nenhuma das portas possui dado a ser transmitido, todos os sinais de “*estado\_porta*” estarão desativados e o Controlador de Relógio atuará inibindo o relógio provocando o bloqueio do relógio (“*clock gating*”) deste roteador.

### 3.2. MPSoC HeMPS

HeMPS é um MPSoC com multiprocessamento homogêneo formado por elementos de processamento (EP) constituídos de uma instância do roteador Hermes [MOR04] e um IP de processamento. A seguir aborda-se em detalhe os módulos de hardware que constituem este MPSoC e a organização do software e comunicação entre programas que executa na HeMPS. A estrutura geral de uma instância da HeMPS é apresentada na Figura 13.

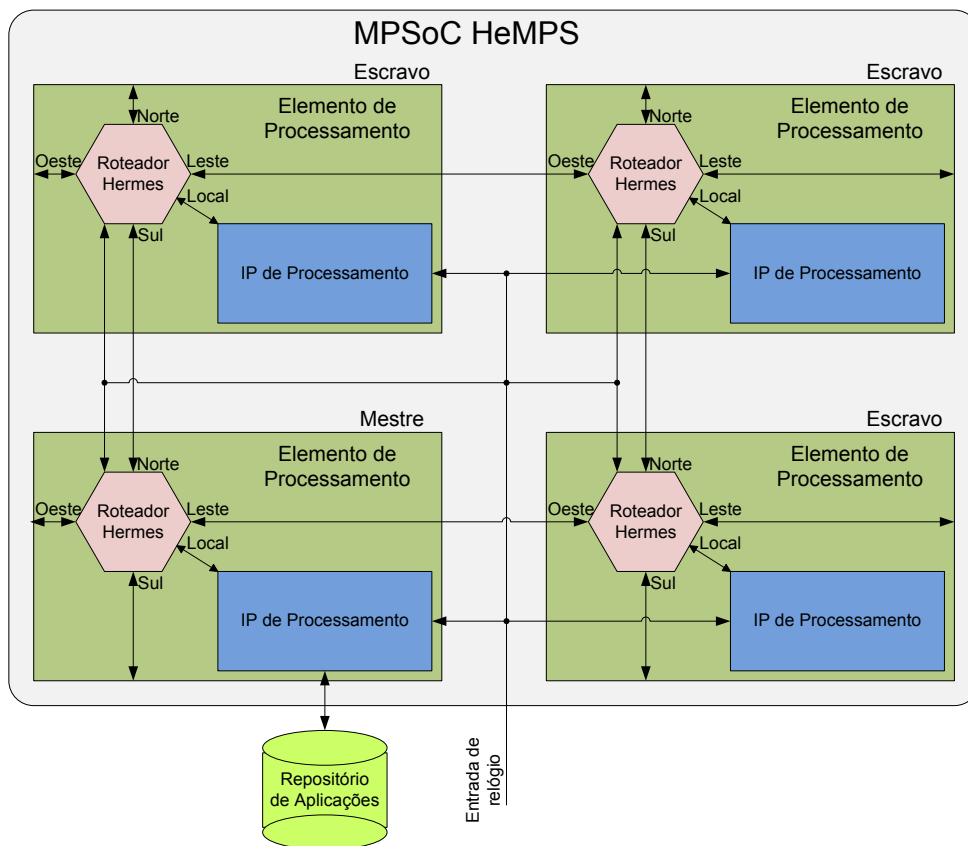


Figura 13: Estrutura de uma instância típica do MPSoC HeMPS, no caso uma implementação 2x2 da mesma.

A NoC Hermes é uma rede com topologia malha 2D, que permite a comunicação entre diversos módulos IP conectados cada um a alguma porta local de um roteador da rede. Conforme nota-se na Figura 14, o IP de Processamento é constituído por: uma interface de rede (em inglês *network interface* ou NI), um controlador de acesso direto à memória (em inglês *direct memory access* ou DMA), uma memória local ao processador, e um processador. A interface de comunicação é responsável pela interconexão dos módulos internos do IP de Processamento com a porta local do roteador Hermes. O DMA permite o acesso à memória pela interface de rede (para carga de novos programas, por exemplo). A memória armazena as instruções e dados em uso pelo processador. O módulo processador, para a HeMPS 4.0, pode ser um de dois processadores: MIPS-Lite e MicroBlaze Lite, ambos disponíveis no sítio Opencores (<http://www.opencores.org>). Quando o IP de processamento utiliza o módulo processador MIPS-Lite, este recebe o nome Plasma. Já quando recebe um MicroBlaze Lite, recebe o nome de MBLite. Na plataforma existem dois tipos de EPs: mestre (atualmente existe apenas um EP mestre em cada instância do MPSoC - ele coordena a distribuição das tarefas que estão em repositório externo apenas acessível a ele, além de viabilizar a comunicação com o mundo externo ao MPSoC) e escravo (demais EPs - apenas processam as tarefas que estão em sua respectiva memória, se comunicando com outros processadores se/quando isso é necessário). A Figura 13 apresenta de maneira simplificada um exemplo de MPSoC HeMPS 2x2, com um mestre e três escravos.

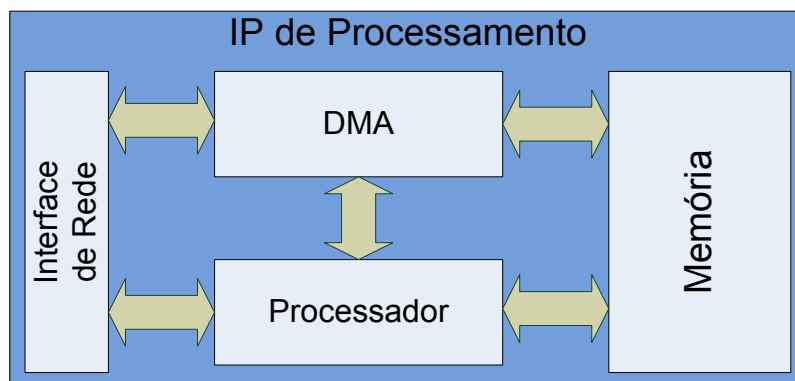


Figura 14. Detalhe dos módulos internos ao IP de Processamento da HeMPS.

Com relação à organização de software do MPSoC HeMPS, cada IP de Processamento suporta o uso de execução multitarefa, exceto o EP mestre que apenas executa uma tarefa (gerenciar as demais tarefas pelo MPSoC). Cada IP de processamento possui uma cópia de um núcleo de sistema operacional denominado *microkernel* (incluindo o mestre). A cópia presente no EP mestre é dedicada a distribuir as tarefas presentes em seu repositório, informar aos elementos de processamento as suas localizações e fornecer a saída de dados do MPSoC. Já a cópia do *microkernel* dos escravos é responsável por gerenciar as tarefas internas de cada IP de Processamento (usando escalonamento de tarefas que segue uma estratégia *round robin*), armazenar a localização das demais tarefas com as quais suas tarefas interagem e promover a comunicação entre as tarefas existentes neste ou em outro EP.

Para realizar a troca de mensagens, o *microkernel* possui canais de comunicação concretizados como filas circulares denominadas "*pipes*". As funções de envio e recebimento de mensagens ("*send*" e "*recv*", respectivamente) são realizadas por meio de cha-

madras do *microkernel* que possui um subsistema de controle de escrita e leitura nestes *pipes* (“*WritePipe*” e “*ReadPipe*”). Nestes, a escrita é não-bloqueante e a leitura é bloqueante. Caso a tarefa alvo da comunicação não esteja presente na tabela de localização do IP de Processamento local, é solicitada ao EP mestre sua localização. Caso esta ainda não esteja alocada em nenhum EP do MPSoC, o EP mestre envia a mesma a um EP escravo para execução e após informa ao solicitante de comunicação sua posição atual.

### 3.2.1. Gerador HeMPS

O Gerador HeMPS [CAR09] é a ferramenta desenvolvida pelo grupo de pesquisa GAPH que propicia a geração automatizada de dispositivos MPSoC do tipo HeMPS, bem como sua simulação e emulação. A Figura 15 mostra a interface gráfica de partida do Gerador HeMPS. Por meio de parâmetros é possível determinar o número de IPs de processamento conectados à NoC Hermes, o número máximo de tarefas que podem ser executadas simultaneamente em um mesmo EP escravo e o tamanho da memória destes IPs. Pode-se também selecionar o tipo de módulo processador que será utilizado em cada EP escravo (Plasma ou MBLite). Permite-se ainda selecionar o modelo de descrição dos processadores entre simulação com precisão em nível de instruções (em inglês, *instruction set simulator* ou ISS), modelos C/SystemC ou simulação em nível de transferência de registradores (em inglês *register transfer level* ou RTL).

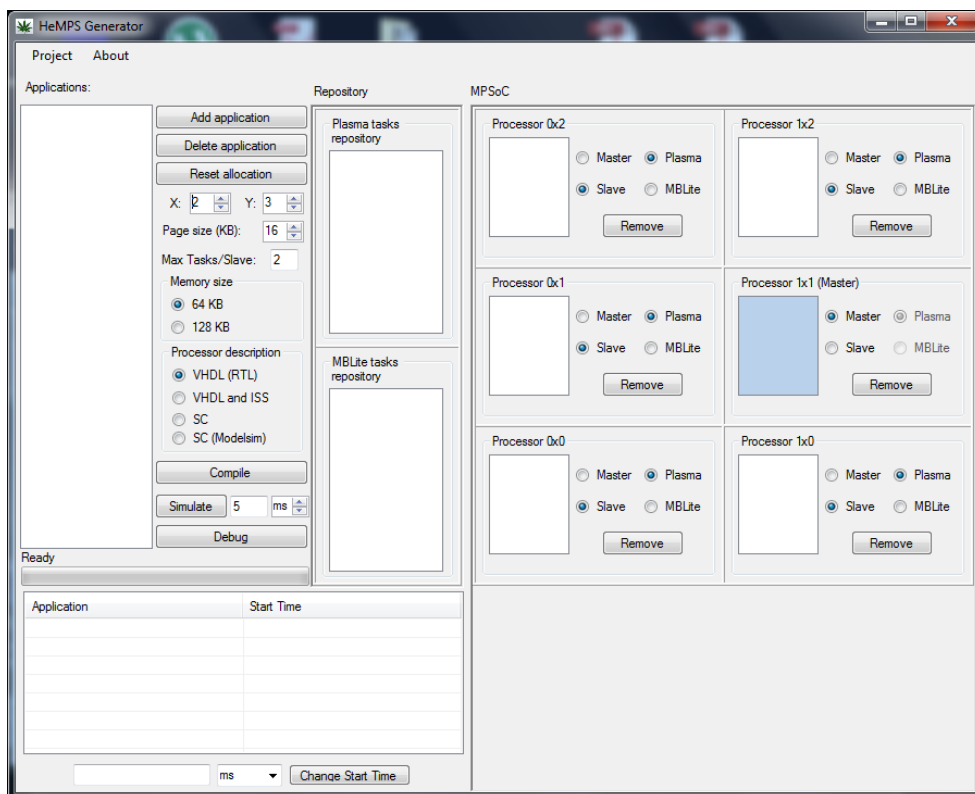


Figura 15: Interface gráfica do Gerador HeMPS.

Além dos parâmetros já apresentados, permite-se ainda determinar qual dos módulos do MPSoC será considerado o mestre e quais estão habilitados para receber tarefas dinamicamente distribuídas por ele. Assim, as tarefas existentes neste módulo são distribuídas pelos módulos habilitados. Ainda permite-se determinar individualmente qual módulo receberá determinada tarefa, colocando-a diretamente em sua lista de tarefas, habilitando o uso de alocação inicial estática de tarefas.

Ao clicar no botão “*Compile*”, a ferramenta constrói o hardware conforme os parâmetros definidos e executa a integração entre os módulos em hardware e software permitindo a simulação da instância produzida do MPSoC HeMPS. Esta tarefa está previamente configurada para atuar com o simulador Modelsim da Mentor Graphics, mas pode em princípio ser realizada com qualquer outro simulador de hardware compatível. Após a simulação, ao clicar no botão “*Debug*” pode-se observar as saídas programadas de cada tarefa em cada IP de processamento, o que provê um ambiente gráfico de depuração interativa do MPSoC.

## 4. ADAPTAÇÕES REALIZADAS SOBRE A REDE HERMES-GLP

Originalmente desenvolvida como parte de um trabalho de mestrado [PON08A], a rede Hermes-GLP possuía algumas deficiências funcionais que necessitavam ser sanadas para integração com o MPSoC HeMPS. Conseqüentemente, o presente trabalho propôs e produziu as alterações necessárias na estrutura, funcionalidade e capacidade de parametrização da rede. Estas alterações são apresentadas neste Capítulo.

### 4.1. Ponteiros de filas bissíncronas

A primeira contribuição do presente trabalho reside no esquema de identificação de fila cheia ou fila vazia das filas bissíncronas, presentes nas portas de entrada de cada roteador da NoC Hermes-GLP. Estas alterações foram propostas a fim de corrigir alguns erros de comparação de ponteiros da solução original, bem como facilitar a compreensão da funcionalidade da fila como um todo e do comportamento dos ponteiros da fila circular em particular.

Para permitir a troca de informações entre domínios distintos de relógio, a rede Hermes-GLP emprega filas bissíncronas. Estas filas permitem o uso de relógios diferentes para ler e gravar dados. Para evitar a sobrescrita ou releitura de informações constantes na fila, módulos de comparação entre os ponteiros de escrita e leitura são inseridos na fila. Entretanto, como se trabalha com domínios de frequência distintos, é necessário o uso de sincronizadores entre o valor dos ponteiros e seus módulos comparadores complementares. A Figura 16 apresenta o circuito da fila bissíncrona da Hermes-GLP. Ao centro há uma fila circular, principal componente da estrutura, ladeada de duas máquinas de estado que controlam a geração dos ponteiros de fila cheia e fila vazia. Abaixo da fila circular existem os sincronizadores, que adaptam sinais gerados de um lado da fila bissíncrona ao domínio de relógio do outro lado da fila. Note-se que a fila circular recebe apenas o relógio de escrita (*wclk*), pois a escrita é síncrona, mas a leitura é assíncrona, gatilhada apenas pela atualização do ponteiro de leitura sincronizado pelo relógio de leitura (*rclk*).

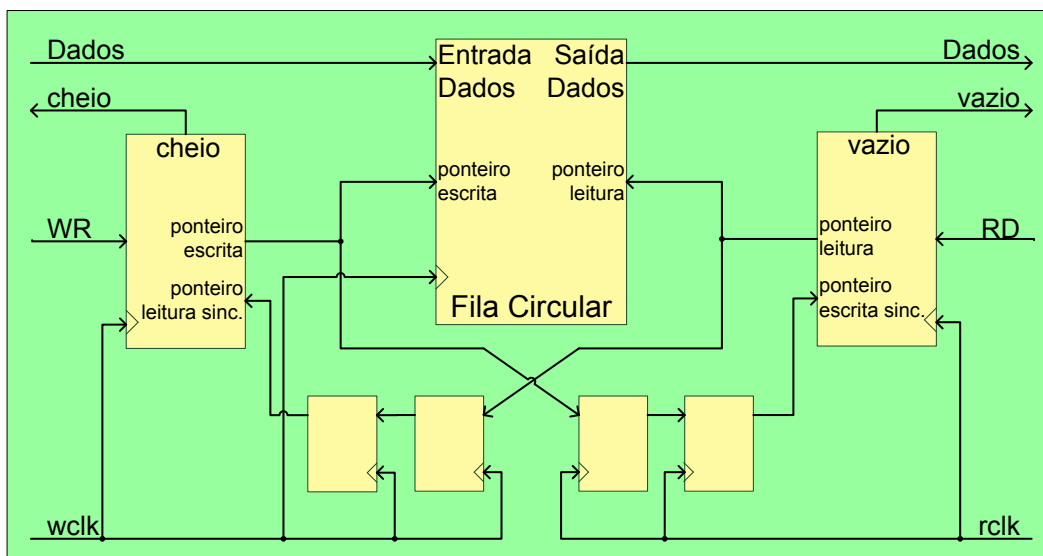


Figura 16. Diagrama de blocos da fila bissíncrona implementada na rede Hermes-GLP.

Na versão original das filas bissíncronas da Hermes-GLP, existiam apenas dois ponteiros cujos valores eram usados em comparações a fim de determinar se a fila estava vazia ou cheia. Entretanto, o modo como a comparação era realizada era logicamente incoerente, apesar de funcional por usar particularidades de um tamanho fixo de fila (8 posições). Para oito endereços de fila, quando o bit mais significativo dos ponteiros de leitura e escrita era idêntico, comparavam-se os dois bits menos significativos de cada um e, quando fossem completamente diferentes entre si, haveria um endereço entre os dois. Isso garantiria a existência de uma posição entre os endereços de escrita e leitura. Essa comparação é funcional graças a uma exceção do comportamento dos ponteiros Gray de três bits, mas logicamente não existe nexu em sua utilização para filas de tamanho qualquer.

A versão atual proposta aqui utiliza três ponteiros: leitura, escrita e escrita anterior. O esquema é ilustrado na Figura 17. O ponteiro de leitura sempre aponta para o último endereço lido da fila. O endereço de escrita aponta o próximo endereço disponível para gravar um novo dado. O ponteiro de escrita anterior se refere à posição do ponteiro de escrita um ciclo de relógio antes. A Figura 17 ilustra os casos de fila vazia (a), fila com dados (b) e fila cheia (c). A lógica opera da seguinte maneira: quando o ponteiro de leitura é igual ao ponteiro de escrita anterior, a fila está vazia. Quando o ponteiro de escrita é igual ao ponteiro de leitura, significa que todas as posições da fila estão ocupadas e a fila está cheia. Com esta alteração, os ponteiros devem ser inicializados de tal maneira que o ponteiro de leitura esteja uma posição atrás do ponteiro de escrita. Já o ponteiro de escrita anterior deve estar uma posição atrás do de escrita, como esperado. Com esta modificação, necessitou-se adaptar a máquina de estados responsável pela leitura que, ao identificar que a fila não está mais vazia, incrementa o ponteiro de leitura antes de enviar o dado para a porta de saída.

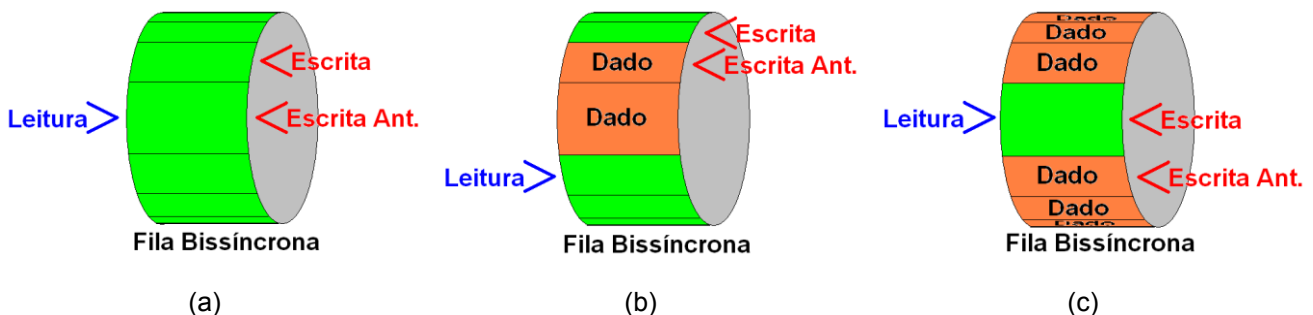


Figura 17. Nova organização dos ponteiros de leitura e escrita. Situações: (a) fila vazia; (b) fila com dados; (c) fila cheia.

Para minimizar os efeitos de *glitches* nos sinais de fila vazia ou fila cheia, decidiu-se alterar a maneira como os ponteiros são incrementados e comparados no roteador da rede Hermes-GLP. Em particular alterou-se a estrutura do *buffer* das portas de entrada do roteador. Originalmente, utilizava-se de codificação Gray para, em teoria, eliminar o efeito de *glitches* nas comparações entre os ponteiros de leitura e escrita. Entretanto, estes endereços em codificação Gray eram anteriormente gerados em binário e convertidos por meio de deslocamento de bits e portas “ou exclusivo” (XOR). Com esta conversão acabava-se por se criar a possibilidade de gerar *glitches* nestes sinais, que somente podem ser percebidos em simulações com atrasos. Tais *glitches* são provocados por portas e fios e não foram identificados na versão original da NoC. Para solucionar este problema, pro-



pôs-se a segunda contribuição deste trabalho: a substituição da codificação original por uma codificação *Johnson*.

De maneira simplificada, o incremento de um ponteiro com codificação *Johnson* é realizado utilizando-se um registrador de deslocamento e um inversor de realimentação o que gera uma maneira simples, porém eficaz de produzir valores de ponteiros sem a possibilidade de *glitches*. A principal desvantagem da codificação *Johnson* é seu crescimento linear ( $O(n)$ ) enquanto o da codificação *Gray* cresce apenas logaritmicamente ( $O(\log(n))$ ). Além do contador *Johnson*, inicialmente utilizou-se um contador binário auxiliar para determinar o endereço físico da gravação ou leitura na memória. Maiores detalhes sobre a codificação VHDL do gerador de ponteiro *Johnson* podem ser visualizados no Apêndice A.1. A codificação *Gray*, por ocupar o mesmo número de bits que um contador binário de mesmo módulo, permite utilizar o mesmo ponteiro para as duas operações.

Assim, surgiu o interesse de averiguar o crescimento em área de controle de ponteiros, comparando as codificações *Gray* e *Johnson*. Procurou-se substituir o antigo sistema *Gray* por um que permitisse tamanho parametrizável de fila e que não gerasse *glitches*. Para realizar um comparativo entre o crescimento em área das codificações de *Johnson* e de *Gray*, buscou-se um método simplificado de incremento para tamanho parametrizável de ponteiro *Gray* sem a geração de *glitches*.

Uma solução para a codificação *Gray* que atende as necessidades aqui impostas e que foi implementada aparece na Figura 18, adaptada para tamanho parametrizável do circuito. Esta se baseia em uma solução proposta pela Altera [ALT12A]. A cada incremento realizado em um ponteiro *Gray*, uma determinada posição deve ter seu bit invertido. Desse modo, pode-se comprovar que a cada passo de incremento a paridade do ponteiro se altera. Por este motivo, utiliza-se o bit menos significativo para armazenar a paridade do endereço. A partir deste valor, pode-se identificar qual bit necessita ser alterado fazendo uma simples pesquisa. Percorre-se o registrador do bit menos significativo ao mais significativo em busca de um bit de valor '1'. Verifica-se se antes desta posição existe algum outro bit com valor '1' e, caso não exista, inverte-se o próximo bit. Como a atualização do endereço somente se dá nas bordas de relógio de escrita/leitura, esta codificação não apresenta *glitches* e sua estrutura permite a parametrização do tamanho do ponteiro. Maiores detalhes sobre a codificação VHDL do gerador de ponteiro *Gray* podem ser visualizados no Apêndice A.2.

```

process
begin
  wgray(0) <= not(wgray(0));
  for i in 1 to (TAM_POINTER) loop
    if ((i = (TAM_POINTER) or wgray(i-1) = \ '1') and
no_ones_below_wgray(i-1) = '0') then
      wgray(i) <= not(wgray(i));
    end if;
  end loop;
end process;

no_ones_below_wgray(0) <= '0';
no_ones_below_wgray(TAM_POINTER downto 1) <= \
wgray(TAM_POINTER-1 downto 0) or \
no_ones_below_wgray(TAM_POINTER-1 downto 0);

```

Figura 18. Codificação livre de glitches para a lógica de incremento para ponteiros *Gray* [ALT12A].

Tendo estes dois modelos de codificação corretos em mãos, realizou-se um estudo comparativo a fim de determinar a razão de crescimento da área de controle dos *buffers* quando se varia a profundidade das filas. Esta tarefa foi realizada prototipando e extraíndo a área ocupada pelo *buffer* assíncrono. Para tanto, utilizou-se do o ambiente ISE Versão13 da Xilinx, definindo-se como alvo dispositivo FPGA da família Virtex-5. Os resultados coletados aparecem na Tabela 5, onde cada campo representa a quantidade total de LUTs de quatro entradas usadas variando-se a codificação do ponteiro e a profundidade da fila de entrada. Observa-se considerável vantagem no uso da codificação *Gray* quando a profundidade das filas é maior que oito posições. Ao analisar os dados, cogitou-se que o contador binário presente na codificação *Johnson* poderia estar contribuindo para o aumento da área ocupada. Assim, sugeriu-se uma conversão direta do ponteiro *Johnson* para binário para analisar a utilização de apenas um registrador.

Tabela 5 - Comparativo de ocupação de área (em número de LUTs de quatro entradas) para códigos Johnson e Gray.

Profundidade da fila →	4	8	16	32	64	128
Código Johnson	156	170	184	253	369	600
Código Gray	161	168	175	226	316	486

Para realizar a conversão de registrador *Johnson* para binário utilizou-se a lógica apresentada na Figura 19. Maiores detalhes sobre a codificação VHDL do gerador de ponteiro *Johnson* com conversor binário podem ser visualizados no Apêndice A.3. Após implementar esta solução, executou-se novamente a prototipação e, variando os mesmos parâmetros, obtiveram-se os dados da Tabela 6.

```

XORs_rd : for i in 0 to TAM_JOHNSON_POINTER-2 \
generate
  xor_vect_rd(i) <= rjohn(i) xor rjohn(i+1);
end generate XORs_rd;

ORs_rd : for j in 0 to TAM_POINTER-2 generate
  ORs2_rd : for k in 0 to \
(TAM_JOHNSON_POINTER/2)-1 generate
    or_vect_rd(j)(k) <= xor_vect_rd(k + \
((k/(2**j))*(2**j)));
  end generate ORs2_rd;

  ORed_rd : for l in 1 to
(TAM_JOHNSON_POINTER/2)-2 generate
    ored_vect_rd(j)(l) <= ored_vect_rd(j)(l- \
1) or or_vect_rd(j)(l+1);
  end generate ORed_rd;
  ored_vect_rd(j)(0) <= or_vect_rd(j)(0) or \
or_vect_rd(j)(1);
end generate ORs_rd;

encoderOut_rd : for m in 0 to TAM_POINTER-2 \
generate
  raddr(m) <= \
ored_vect_rd(m)((TAM_JOHNSON_POINTER/2)-2);
end generate encoderOut_rd;
raddr(TAM_POINTER-1) <= rjohn(0);

```

Figura 19. Circuito conversor de codificação Johnson para binário.

Pelos resultados é possível observar que a solução que emprega codificação Johnson ocupa menor área quando comparado com a que implementa a conversão direta de ponteiro. Ainda é visível que, para casos de filas de quatro posições, o controle de

ponteiros de codificação *Johnson* ocupa menor área quando comparada com codificação *Gray*. Isso se deve à utilização direta do ponteiro *Johnson* para acesso a posições de memória. Entretanto, após oito posições na fila, é visível a economia de área utilizando codificação *Gray*.

Tabela 6 - Comparativo de ocupação de área (em número de LUTs de quatro entradas) para códigos Johnson, Gray e Johnson com conversão direta para binário.

Profundidade da fila →	4	8	16	32	64	128
Código Johnson	156	170	184	253	369	600
Código Gray	161	168	175	226	316	486
Código Johnson c/ conversor	156	169	185	272	445	759

#### 4.2. Inibição e liberação de sinal de relógio

A terceira contribuição deste trabalho é voltada ao sistema de inibição e liberação de sinal de relógio. O circuito original presente na Hermes-GLP utilizava três sinais para determinar quando o sinal de relógio deveria ser liberado: enquanto estivesse recebendo dados (sinal *WR* alto); enquanto estivesse transmitindo pacotes para o próximo roteador; e quando a fila bissíncrona indicasse dado presente na fila (sinal *empty='0'*). Entretanto, identificou-se que em casos extremos de variação de frequência e de pacotes muito pequenos (menores que o tamanho da fila) o sinal de *WR* era desativado antes do sinal de *empty* sê-lo, impedindo a liberação do sinal de relógio e trancando pacotes no roteador. Para resolver este problema, a lógica de liberação de relógio foi alterada. Assim, quando o *buffer* está com a fila vazia e recebe um dado (borda de subida de escrita) a liberação do relógio é ativada, e somente se altera quando a fila fica vazia, garantindo o envio de todos os dados.

#### 4.3. Memorização e distribuição de seleção de frequência

A quarta contribuição deste trabalho é percebida nas alterações da propagação do sinal de seleção de frequência dos pacotes ao longo dos roteadores da rede Hermes-GLP. Inicialmente, esta rede foi projetada baseando-se em uma plataforma fixa onde não se variava o valor da seleção de frequência imposto nas portas locais de cada roteador. Assim, os pacotes de mesma origem sempre mantinham a mesma frequência de operação. Ao permitir a alteração deste valor para pacotes diferentes, mas com mesma origem, ocorria do sinal de seleção ser perdido ou inapropriadamente enviado ao longo do caminho.

Durante a implementação desta rede em um MPSoC real, o autor identificou que, quando o pacote era demasiadamente pequeno ou muitos pacotes eram enviados simultaneamente, o sinal de seleção de frequência podia ser alterado ao longo do caminho. Este erro se deve ao modo como este valor era transferido entre a porta de entrada e a de saída do roteador. Originalmente, este sinal era transferido diretamente da porta de entrada para a saída através do *crossbar*.

Para determinar o roteamento de um pacote, o buffer assíncrono gasta em média até três ciclos de relógio (dois ciclos do sincronizador e um para a máquina de leitura identificar a origem). Agora, imaginem-se dois pacotes de três *flits* cada, onde o primeiro utiliza frequência "X" e o segundo, "Y". Ambos os pacotes são transferidos em sequência. O primeiro pacote é recebido completamente pela fila em três ciclos e o segundo já está à disposição. Ao determinar a porta que será utilizada para o primeiro pacote, a entrada de

seleção de frequência que está disponível na entrada é a do segundo. Este erro pode ser propagado aos demais roteadores, produzindo operação incorreta da rede.

Outro comportamento anômalo e semelhante na rede original acontece durante o chaveamento da frequência de operação pelo controlador de relógio. O valor de seleção de frequência é disponibilizado diretamente à porta de saída. No exemplo anterior, ao invés de disponibilizar a frequência de seleção “X”, acaba-se disponibilizando aquela definida pela seleção “Y”. Isto pode, além de propagar o erro, causar comportamento anômalo no relógio do roteador. Para solucionar ambos os problemas, optou-se por armazenar temporariamente a seleção de frequência de comunicação em um registrador, assim que a recepção do segundo *flit* de algum pacote ocorre. Assim, disponibiliza-se ao controlador de relógio o valor correto de seleção de frequência para sua configuração. Para resolver o problema da propagação deste valor, o *crossbar* recebe o mesmo sinal de seleção de frequência armazenado temporariamente no buffer de entrada, garantindo que o valor correto será propagado.

#### 4.4. Extensão para “n” relógios

A quinta contribuição deste trabalho consiste na ampliação da capacidade de relógios disponibilizados na entrada do roteador da rede Hermes-GLP. Originalmente, a rede foi descrita para utilizar exclusivamente dois sinais de relógio por roteador, caracterizando uma implementação de prova de conceito para redes GALS. Existem duas estruturas que podem ser utilizadas em FPGAs para controlar o chaveamento destas frequências, a primeira baseada apenas em portas lógicas convencionais e a segunda em um módulo específico da Xilinx chamado “BUFCTRL” [XIL10]. Obviamente, esta última opção somente pode ser usada quando se prototipa redes Hermes-GLP em FPGAs da Xilinx. Para viabilizar o uso de número parametrizável de relógios, optou-se por utilizar uma árvore binária simples de multiplexadores 2:1, ilustrada na Figura 20.

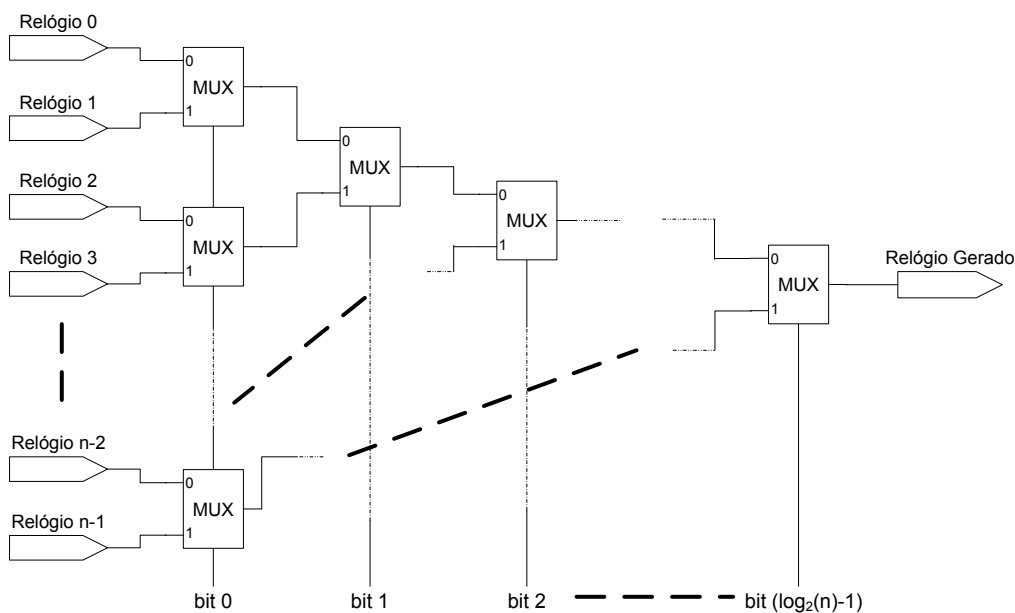


Figura 20. Seleção de frequência empregando uma árvore binária simples como circuito equivalente.

Assim, cada elemento possui duas frequências de entrada. Se o valor de seleção for “0”, chaveia-se a da porta “0”. Se for “1”, chaveia-se a da porta “1”. A Figura 20 mostra uma visão simplificada desta estrutura de seleção de frequência. Para montar esta árvore,

escolheu-se o uso de módulos dedicados da Xilinx, pois estes garantem chaveamento sem *glitches* [XIL10].

O módulo “BUFGCTRL”, ilustrado na Figura 21, é dedicado a permitir o chaveamento assíncrono entre dois sinais de relógio. Segundo sua documentação, para selecionar uma entre duas frequências de entrada (“I0” e “I1”), os pinos “S0” e “S1” devem ser comutados. Por meio dos sinais CE é possível habilitar ou desabilitar o sinal de relógio (não recomendáveis para determinadas transições, segundo a documentação). Sendo assim, o chaveamento de frequências é feito pelos valores de “S0” e “S1” e para bloquear o relógio utiliza-se os sinais de “CEi” interligados entre si para evitar comportamento indesejado.

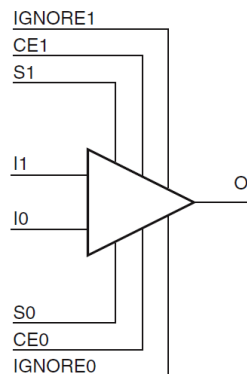


Figura 21. Diagrama de conexões do módulo BUFGCTRL [XIL10].

Para construir o circuito de seleção de frequência baseado em árvore binária, cada “BUFGCTRL” recebe dois sinais distintos de relógio. Suas saídas são ligadas à entrada dos “BUFGCTRL” do próximo nível e assim sucessivamente até o último nível (topo da árvore). Cada nível utiliza um bit da prioridade para chavear a seleção do seu nível. Na base, utiliza-se o bit menos significativo e, no topo, o mais significativo. “S1” recebe diretamente o valor do bit de prioridade referente ao seu nível enquanto “S0” recebe esse dado negado. Os sinais “CE” de todos os níveis, com exceção do último, recebem “1” para propagar sua frequência pela árvore. O último nível recebe o valor de liberação do relógio para ativar ou não sua saída.

É importante salientar que relógios vizinhos entre si em cada nível da árvore binária devem estar ordenados em valores de frequência. Isso implica que se o relógio de um determinado ponto da árvore tiver período  $x$ , um de seus lados apenas apresentará períodos superiores (ou iguais) a  $x$  e do outro, apenas inferiores (ou iguais) a  $x$ . Isso garante o chaveamento correto de frequência conforme o valor da seleção, que corresponde a prioridade de um pacote que trafega na NoC Hermes-GLP.

Durante a geração parametrizável da rede, a quantidade de relógios que será disponibilizada a cada roteador deve ser informada no arquivo “*Hermes\_package.vhd*” no campo designado para esta operação (*NUM\_CLOCKS*). Os demais campos referentes ao controle de ponteiros e tamanho dos registradores e barramentos de prioridade serão automaticamente configurados durante a geração. Por se tratar de uma árvore binária, é necessário que os  $n$  relógios sejam sempre em números que correspondem a potências de 2. A Figura 22 apresenta o trecho de código que constrói o circuito de chaveamento de relógios usado neste trabalho.

```

Bufgctrl_check1: if NUM_CLOCKS > 1 generate
  BUFGCTRL_inst0 : BUFGCTRL
  generic map (
    INIT_OUT => 0,
    PRESELECT_I0 => FALSE,
    PRESELECT_I1 => TRUE)
  port map (
    O => clock_temp(0),
    CE0 => not_clockgating_router,
    CE1 => not_clockgating_router,
    I0 => clock_temp(1),
    I1 => clock_temp(2),
    IGNORE0 => '0',
    IGNORE1 => '0',
    S0 => vote((TAM_SEL_FREQ-1)),
    S1 => not_vote((TAM_SEL_FREQ-1)));
end generate Bufgctrl_check1;

Bufgctrl_check2: if NUM_CLOCKS > 2 generate
  Bufgctrl_inst: for i in 1 to (NUM_CLOCKS-2) \
generate
  BUFGCTRL_inst0 : BUFGCTRL
  generic map (
    INIT_OUT => 0,
    PRESELECT_I0 => FALSE,
    PRESELECT_I1 => TRUE)
  port map (
    O => clock_temp(i),
    CE0 => '1',
    CE1 => '1',
    I0 => clock_temp(((2*i)+1)),
    I1 => clock_temp(((2*i)+2)),
    IGNORE0 => '0',
    IGNORE1 => '0',
    S0 => vote((TAM_SEL_FREQ-1)- \
(integer(FLOOR(LOG2(REAL(i+1)))))),
    S1 => not_vote((TAM_SEL_FREQ-1)- \
(integer(FLOOR(LOG2(REAL(i+1)))))));
  end generate Bufgctrl_inst;
end generate Bufgctrl_check2;

```

Figura 22. Trecho do código VHDL que gera a seleção de frequência.

#### 4.5. Votação para seleção de frequência

A sexta e última contribuição realizada sobre a estrutura da rede Hermes-GLP encontra-se no circuito de votação de seleção de frequências. O circuito original utilizava uma simples porta *OR* para identificar a maior seleção, pois se tratava de controlar apenas um bit. O novo circuito tem por objetivo identificar quais portas possuem dados a serem transmitidos e identificar qual das entradas de seleção de frequência destas prevalece sobre as demais. Primeiramente, deve-se decidir qual dos valores de seleção de frequência definirá a maior frequência de operação. Esta configuração implica também na maneira como as frequências devem ser codificada e inseridas na rede.

Ao configurar a rede para que o menor valor de seleção selecione a maior frequência, o relógio de índice “0” deverá conter o menor período dentre os demais relógios. Caso o maior valor de seleção defina a maior frequência, o relógio de índice  $n-1$  ( $NUM\_CLOCKS-1$ ) deverá possuir o menor período dentre os demais. Isso permite que a lógica de programação seja simplificada a ponto do valor introduzido na seleção de frequência chavear o relógio de mesmo índice, além de garantir que o voto de seleção de

um prevaleça sobre outro. Esta definição também é parte do arquivo “*Hermes\_package.vhd*” pela definição do valor da nova definição deste, “*HSELECT\_HIGHEST*”. Se este receber *true*, quanto maior o valor de seleção de frequência, maior será a frequência associada. Quando configurado com *false*, o menor valor de seleção de frequência representa a maior frequência. Esta definição configura qual dos circuitos de votação será utilizado pelo controlador de relógio.

O circuito de votação, apresentado na Figura 23, possui três multiplexadores de três entradas configurados em dois níveis. No primeiro nível, dois multiplexadores recebem os valores das portas Norte/Sul e Leste/Oeste, juntamente com o valor de seleção que escolhe a menor frequência (*menor\_freq*). Já o multiplexador do segundo nível, recebe as duas respostas provenientes dos dois anteriores (*voto\_NS* e *voto\_LO*) mais o da porta Local. A lógica para a votação entre os valores foi simplificada: Se as duas portas estiverem ativas, o multiplexador seleciona a que escolherá a maior frequência. Caso somente uma esteja ativa, propaga-se o valor desta porta. Se nenhuma está ativa, disponibiliza-se o valor que escolhe a menor frequência possível. O último multiplexador define qual o valor final de seleção de frequência a ser utilizado pelo controlador de relógio. Assim, caso exista dado na porta Local, propaga-se qual dos três valores é majoritário, ou seja, o que escolhe a maior frequência, e, caso contrário, disponibiliza-se qual das duas oriundas do nível inferior prevalece. O circuito que inibe ou libera o relógio usa uma porta *OR* que recebe os sinais de inibição de cada porta do roteador.

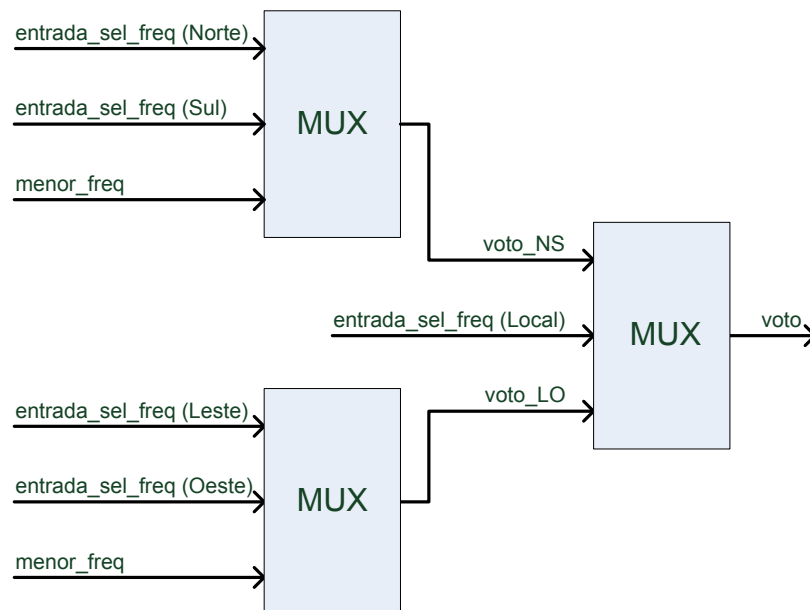


Figura 23. Circuito simplificado de votação para seleção de frequência.

Como os módulos “*BUFCTRL*” não geram *glitches*, pode-se utilizar valores binários para a seleção de frequência sem comprometer a funcionalidade do circuito. Entretanto, momentaneamente pode-se ter um valor de frequência inesperado, devido às comparações das prioridades que são feitas assincronamente. Contudo, desde que garantida apenas a disponibilidade de frequências abaixo da máxima suportada pelo circuito, esta variação momentânea de frequência não apresentará transtornos maiores, logo passando para seu valor de regime.

#### 4.6. Simulação e prototipação da nova NoC Hermes-GLP

Para simular e validar a nova NoC Hermes-GLP, utilizou-se a plataforma HardNoC. A HardNoC (*hardware Network on a Chip*) é um ambiente de testes desenvolvido pelo GAPH para validar as NoCs em FPGAs de forma simplificada. Entretanto, a versão original da HardNoC não operava corretamente devido a falhas na lógica de envio e recebimento de pacotes. Reformulou-se as lógicas de envio e recebimento dos pacotes, corrigindo-se os problemas existentes na plataforma. Além do mais a HardNoC operava originalmente com a rede Hermes-CV dotada de dois canais virtuais por canal físico.

Para validar a rede Hermes-GLP, foram criados circuitos em FPGA para distribuir dois sinais distintos de relógio: um de 66 MHz e outro de 200 MHz. Definiu-se uma rede 3x3 executando um cenário onde o IP 12 envia pacotes ao IP 10 em baixa frequência (prioridade baixa), e o IP 21 envia pacotes para o IP 01 em alta frequência (prioridade alta). Deste modo, o roteador 11 pode, de acordo com a temporização do cenário elaborado passar por três possíveis casos de operação: frequência inibida (por falta de tráfego disponível), baixa frequência (porque conduz apenas um fluxo de baixa prioridade) e alta frequência (quando fluxos de alta prioridade apenas ou fluxos de alta e baixa prioridade cruzam o roteador). A Figura 24 ilustra exatamente os momentos onde se pode observar os três casos de operação. Este trabalho foi alvo de escrita de artigo submetido, aprovado e apresentado pelo Autor no VIII *Southern Programmable Logic Conference* ou SPL. O dito artigo foi indicado entre os melhores artigos da dita conferência [HEC12].

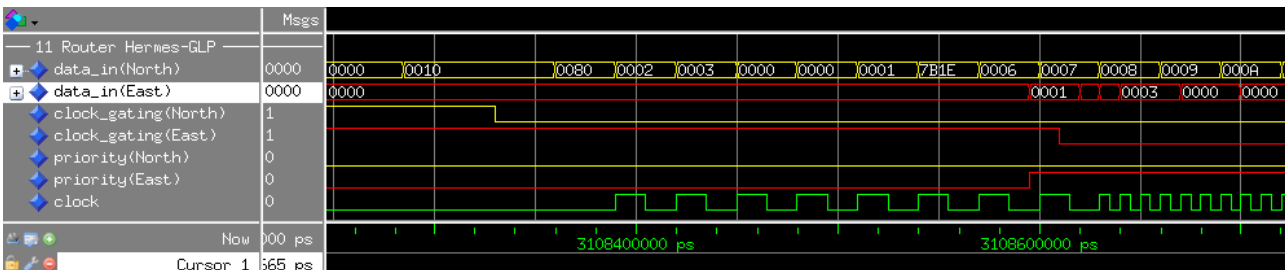


Figura 24. Forma de onda extraída da simulação do IP 11 da NoC Hermes-GLP executando no módulo HardNoC prototipado em FPGA Xilinx.



## 5. O MPSOC HEMPS-GLP

O Gerador HeMPS é uma ferramenta desenvolvida originalmente por Everton Carara em linguagem C# [CAR09]. Este aplicativo tem como objetivo facilitar a geração de instâncias de um MPSoC de dimensões parametrizáveis, baseado na rede Hermes, bem como auxiliar sua simulação e depuração. Em sua última versão (v4.0), o Gerador HeMPS sofreu diversas alterações, sendo a mais radical em sua estrutura de arquivos.

Ao longo deste trabalho surgiu o interesse em desenvolver um MPSoC baseado na rede Hermes-GLP, construindo-se a HeMPS-GLP, ilustrado na Figura 25.

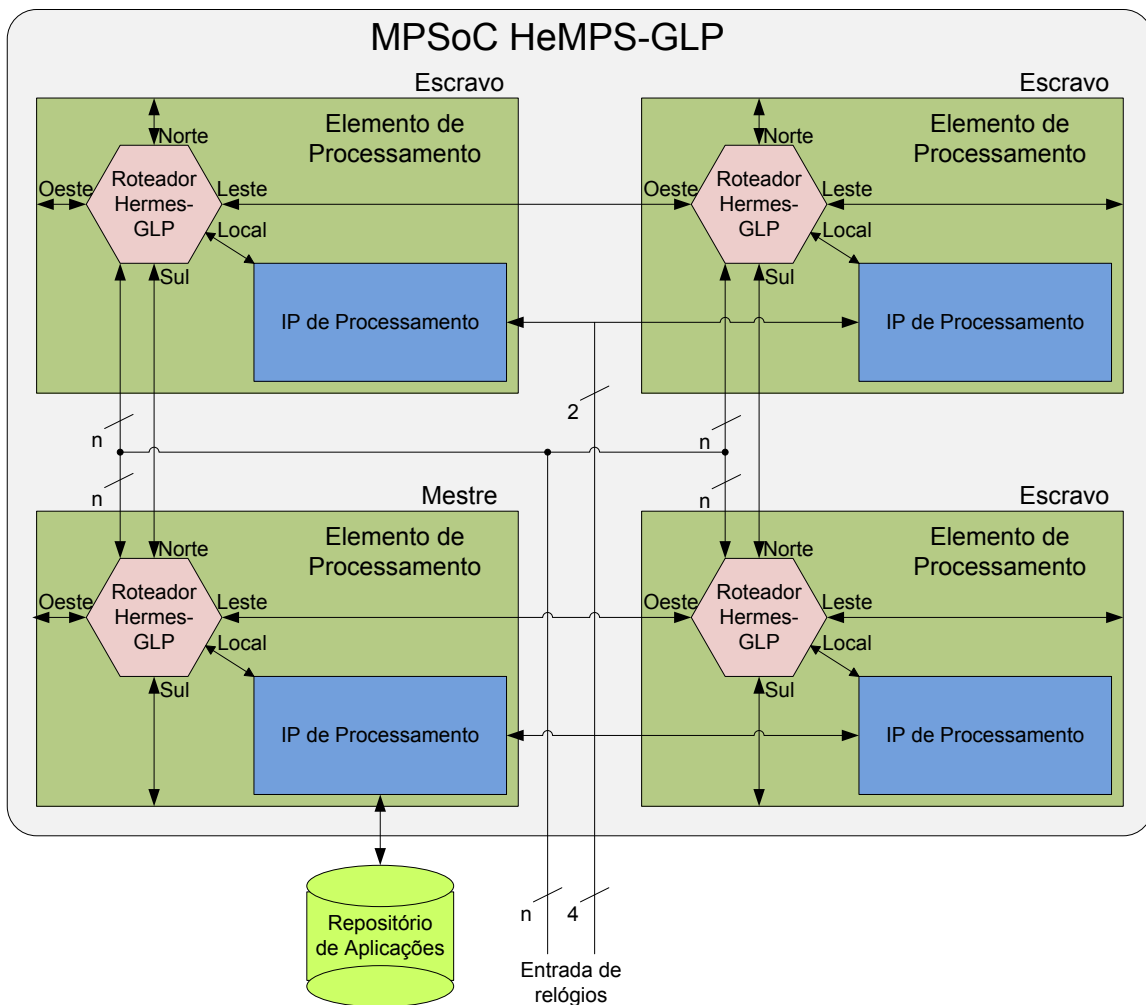


Figura 25. Diagrama de blocos do MPSoC HeMPS-GLP.

Para permitir a utilização de diferentes relógios em seus módulos, incorporou-se a rede Hermes-GLP ao ambiente do Gerador HeMPS. Diferenciando-se das demais políticas de VFI, aqui se propõe permitir o controle de frequência em cada módulo reduzindo-se assim o grão de aplicação de DFS. A ideia inicial consiste em disponibilizar a infraestrutura para definir em software este ajuste de frequência, possibilitando o desenvolvimento de aplicações de gestão de potência para MPSoCs em mais alto nível. Estas alterações no MPSoC HeMPS acabam por constituir outra das contribuições deste trabalho. As adaptações foram realizadas em diferentes estruturas do MPSoC HeMPS e no Gerador HeMPS para disponibilizar o Gerador HeMPS-GLP, apresentado na Figura 25. Estas alterações estão presentes na interface gráfica do ambiente de geração e simulação do MP-

SoC, na estrutura de arquivos de cenários para simulação, na definição dos valores de prioridade de comunicação, no núcleo de sistema operacional embarcado (o *microkernel*), na interface de rede e na estrutura dos arquivos de descrição de hardware. Vale ressaltar que todas as mudanças aqui propostas e realizadas permitem tanto o uso da rede Hermes quanto da Hermes-GLP no mesmo ambiente. Com isso, o Gerador HeMPS-GLP é completamente compatível com a última versão disponibilizada do Gerador HeMPS, permitindo geração e simulações de qualquer dos MPSoCs, HeMPS ou HeMPS-GLP. Apresentam-se ao final deste Capítulo algumas simulações para comprovar a operação correta do novo MPSoC desenvolvido.

### 5.1. Alterações na interface gráfica

A aplicação gráfica conhecida como Gerador HeMPS é a ferramenta utilizada para organizar, compilar, simular e apresentar os resultados de diversos casos de teste do MPSoC HeMPS. Para permitir ao usuário escolher qual rede utilizar, o autor propôs e realizou diversas alterações que permitem ao usuário manipular a rede, informar diversas frequências usadas pela rede Hermes-GLP, além de disponibilizar a possibilidade de alteração do relógio de cada um dos IPs de processamento do MPSoC. Para permitir ao usuário esta interação, foi necessário realizar diversas alterações no ambiente Gerador HeMPS. Note-se que o Gerador HeMPS-GLP é na realidade um superconjunto do Gerador HeMPS.

Ao abrir a interface gráfica principal do Gerador HeMPS-GLP, apresentada na Figura 26, observa-se a existência de um novo campo, de seleção de NoC.

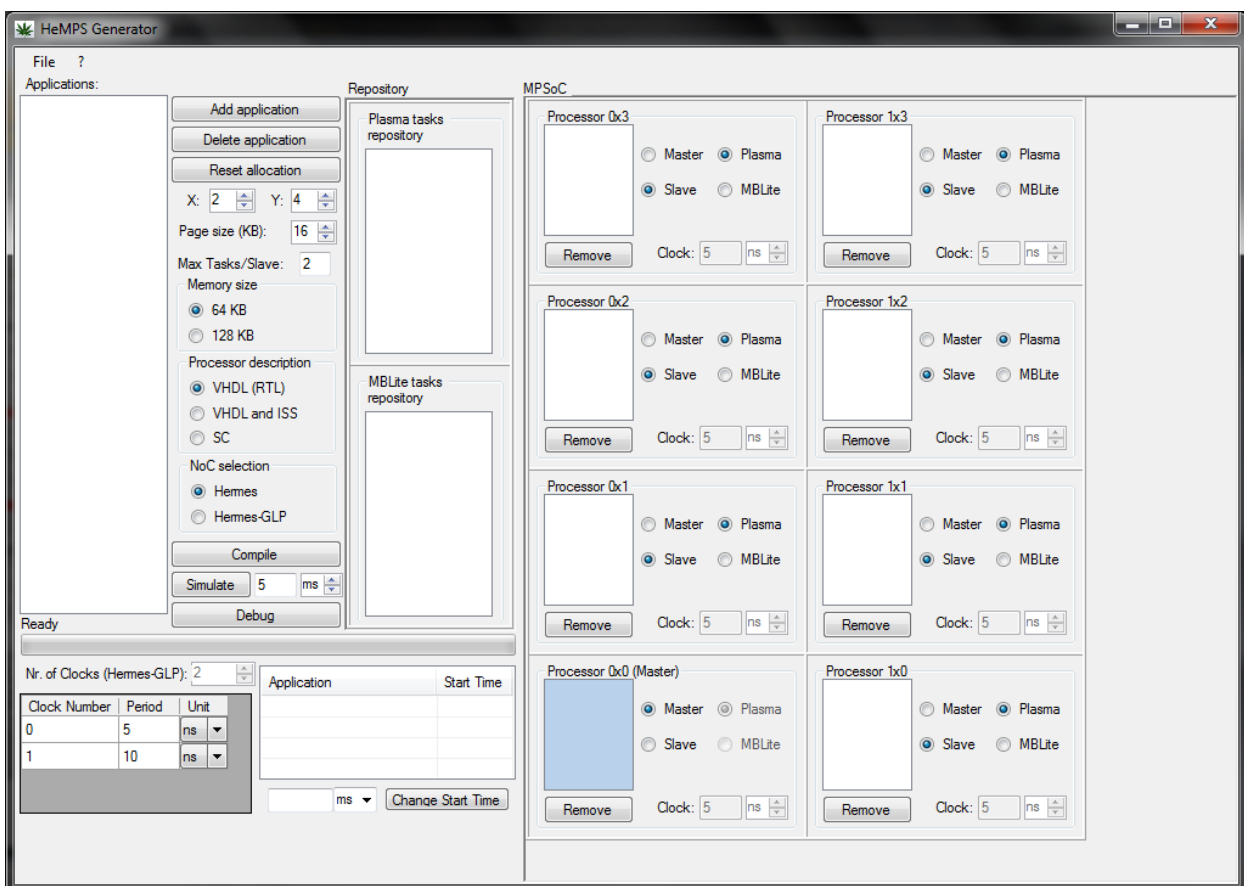


Figura 26. A interface do Gerador HeMPS-GLP.

A Figura 27 apresenta em destaque a área de seleção da NoC.

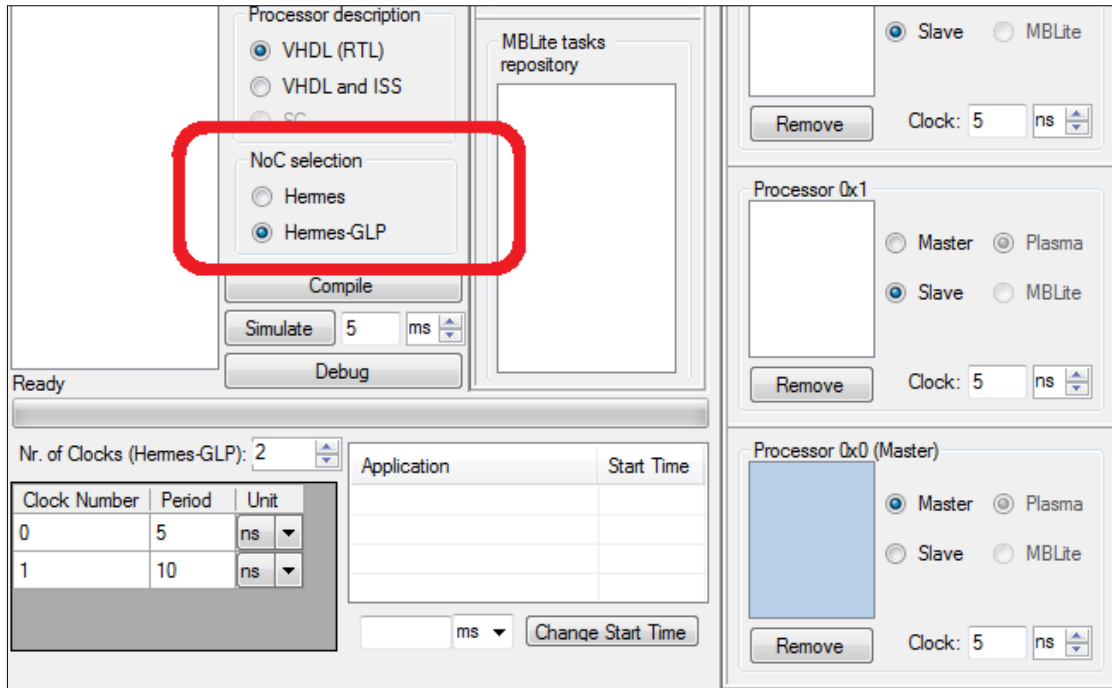


Figura 27. Destaque da caixa de seleção de NoC no Gerador HeMPS-GLP.

Quando se seleciona a NoC Hermes, os campos de número de relógios da rede, lista de relógios e os relógios dos IPs são bloqueados, não se permitindo sua alteração. Neste caso, a ferramenta opera como o Gerador HeMPS.

Ao optar-se pela rede Hermes-GLP, todos os IPs de processamento são automaticamente convertidos para operar com processadores Plasma (denominados MIPS-Lite no ambiente). Bloqueia-se a possibilidade de troca de processador, pois a presente proposta apenas dá suporte tão somente ao Plasma, no momento. Os campos que definem a quantidade de relógios da rede e seus respectivos períodos são liberados para edição. Além disso, viabiliza-se a alteração do período de oscilação para cada IP de processamento de maneira independente.

Na versão atual do software, o campo de número de relógios, em destaque na Figura 28, permite seleções de valores pré-definidos pelo Autor. Limitou-se este campo, pois a lista de relógios necessita ser preenchida pelo usuário o que poderia demandar um tempo elevado, dependendo da quantidade selecionada. A lista abaixo do campo em questão permite ao usuário configurar os períodos de cada relógio disponibilizado aos roteadores da rede. Cada índice de frequência da rede desta lista refere-se diretamente ao campo de seleção de frequência da função responsável pelo envio de mensagem. Assim, se a maior seleção de frequência em um roteador tiver valor igual à "5", será selecionado o relógio de índice "5". Desse modo, para o sistema de seleção de frequência operar corretamente, o período de relógio de uma linha da lista deve ser sempre superior ao de sua antecessora e inferior ao de sua sucessora. Para facilitar seu preenchimento, esta lista adota um sistema que completa automaticamente os novos períodos baseando-se na razão dos últimos dois relógios configurados.

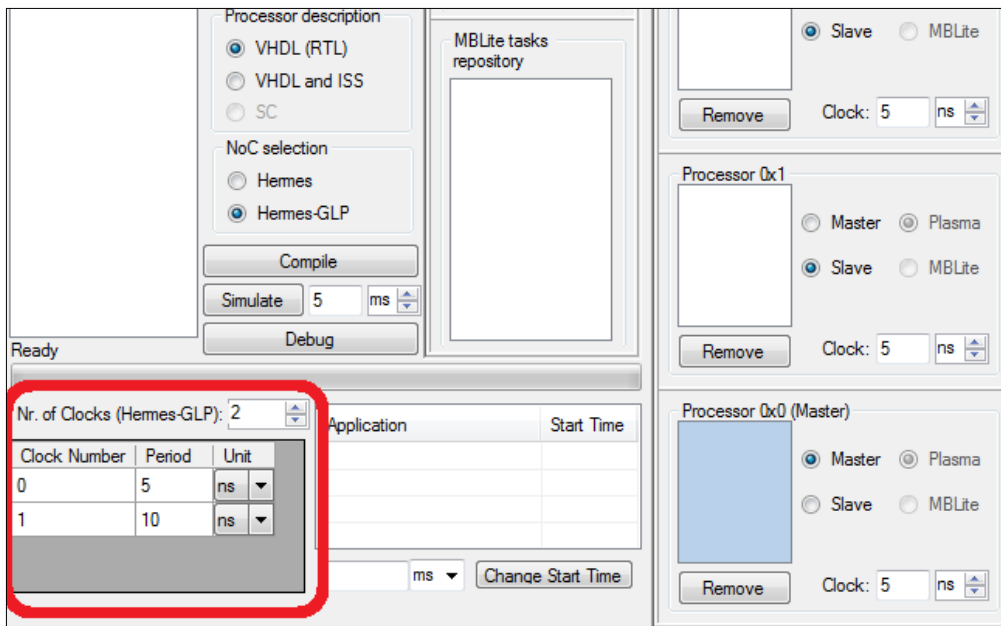


Figura 28. Destaque dos campos de números de relógios e da lista de suas configurações.

Como a rede Hermes-GLP permite o uso de qualquer frequência para os módulos conectados às suas portas locais, é disponibilizada ao usuário a manipulação do relógio para cada IP de processamento, em destaque na Figura 29. Isso viabiliza ao usuário selecionar a frequência que melhor atende sua aplicação para cada processador, permitindo realizar um melhor dimensionamento de dissipação de potência do MPSoC. Este sistema pode ser convertido para operar dinamicamente, assim como acontece com o roteador da Hermes-GLP apenas instanciando um registrador mapeado em memória para atuar no gerador de relógio e uma chamada de sistema para configurá-lo. Entretanto, por este não ser o foco desta dissertação, essa tarefa é deixada como trabalho futuro. Por este motivo, recomenda-se o uso de mapeamento estático para aplicações em função de suas requisições temporais de execução.

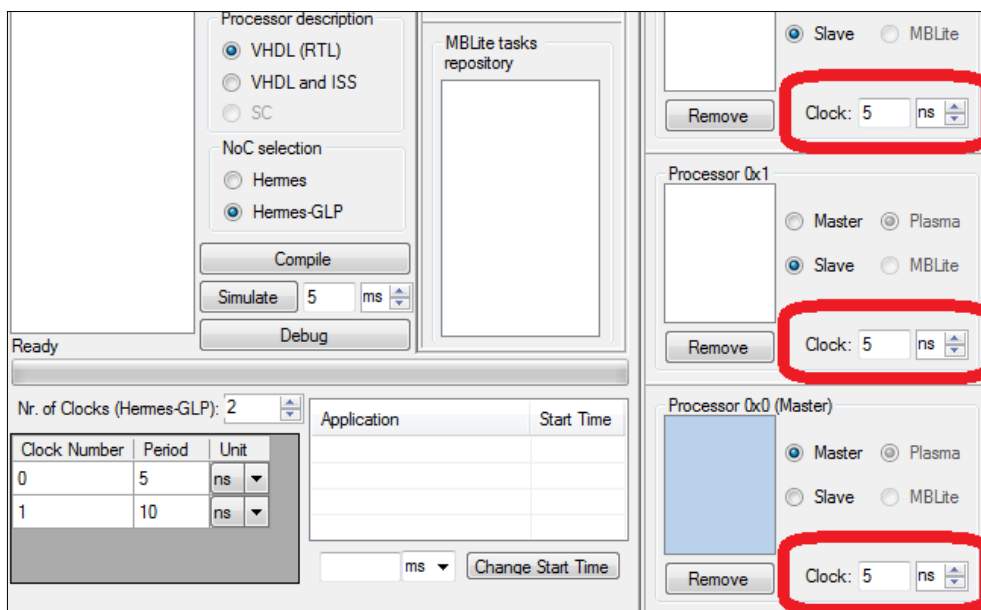


Figura 29. Destaque da configuração de relógio dos IPs de processamento.

Disponibilizam-se juntamente com a ferramenta as aplicações clássicas já presentes na versão original da HeMPS, agora adaptadas para a nova rede. Assim, basta aces-

sar a pasta “*applications\_glp*” e selecionar um ou mais exemplos dentre os vários existentes para executar testes e verificar a correta operação do MPSoC.

## 5.2. Alterações na estrutura de cenários

A versão do Gerador HeMPS utilizada pelo Autor como base para a adaptação foi reestruturada para separar os arquivos de ambiente/simulação gerados pelo usuário, dos arquivos de geração do hardware e do *microkernel*. Assim, os arquivos de uma determinada simulação são armazenados na pasta de seu cenário, facilitando a identificação de problemas e permitindo a abertura simultânea de diversas simulações, o que é garantido por sua modularidade. Assim, como o Gerador HeMPS utilizava um arquivo fixo de inserção de sinais para teste (em inglês *test bench*), necessitou-se empreender algumas alterações na estrutura de geração de cenários para o Gerador HeMPS-GLP.

A nova estrutura da NoC Hermes-GLP permite a parametrização da quantidade de relógios que serão disponibilizados para os controladores de relógio de cada roteador. Este valor também precisa ser informado para que a estrutura de hardware construa corretamente o MPSoC.

Como a NoC Hermes-GLP necessita alguns dados a mais para ser gerada, algumas mudanças nas estruturas do cenário tiveram de ser realizadas. Estas alterações visam manter a coesão da proposta de cenários, garantindo flexibilidade e modularidade dos arquivos de teste.

Ao se informar a quantidade de relógios disponibilizadas à rede (informação crucial para a geração desta), o arquivo de definições da HeMPS (denominados “*HeMPS\_PKG.vhd*” e “*HeMPS\_PKG.h*”) necessita receber esta informação. Assim, a constante *NUM\_CLOCKS* é definida com esse valor em ambos os arquivos para que essa informação esteja disponível para consulta ao compilar seus arquivos de hardware e software.

Para a simulação correta do MPSoC HeMPS-GLP, além de informar quantos relógios serão utilizados, necessita-se injetar os diversos sinais de relógio configurados pelo usuário no Gerador HeMPS-GLP. Para tanto, a proposta utilizada foi gerar um novo *testbench* para cada novo cenário de teste. Assim, ao clicar-se no botão de compilação, o Gerador HeMPS-GLP provê um arquivo de *testbench* utilizando os períodos definidos pelo usuário para a rede e para os IPs de processamento. Este arquivo é armazenado na raiz da pasta do cenário em questão e será compilado e conectado ao MPSoC gerado durante sua simulação.

A última alteração nos arquivos do cenário constituiu-se na mudança dos scripts de compilação do software e do hardware originais para os utilizados no MPSoC HeMPS-GLP. O script de software sofreu as mudanças mais drásticas, pois se necessita informar a quantidade de relógios existentes na rede. Essa informação é essencial para definir algumas prioridades de comunicação utilizadas pelo *microkernel* de cada EP. A utilização de prioridades pelo *microkernel* altera uma série de questões, que serão abordadas na próxima Seção, mas adianta-se que existem versões distintas para cada modelo de roteador. Assim, o script é manipulado pelo Gerador HeMPS-GLP para garantir que somente a versão correta será compilada, com base nas definições presentes no arquivo “*HeMPS\_PKG.h*”. O script de hardware é mais genérico ao ponto de ser gerado exata-

mente o mesmo código nas duas versões de redes disponíveis. Isso ocorre porque os módulos de hardware apenas se interconectam durante a simulação e seguem as definições presentes no arquivo “*HeMPS\_PKG.vhd*”. As alterações aqui encontradas são apenas a inclusão dos novos módulos de hardware que serão descritos nas Seções 5.5 e 5.6 e a nova localização do arquivo de *testbench*.

### 5.3. Definição de valor de seleção de máxima frequência

Esta Seção esclarece as escolhas adotadas pelo Autor com relação à seleção de frequências utilizada pelos pacotes durante seu trajeto ao longo da NoC Hermes-GLP. O objetivo desta ação é facilitar o desenvolvimento de aplicações para o MPSoC HeMPS-GLP.

Como foi possível observar na Seção 4.5, a definição de qual valor de seleção designa a maior frequência inviabiliza a outra. Se ambas as opções fossem disponibilizadas ao usuário, aumentar-se-ia a complexidade do processo, necessitando-se maior atenção para não misturar aplicações com seleções invertidas. Caso isto ocorresse, acarretaria em execução errônea no MPSoC, caso o ambiente esteja configurado com o outro padrão.

Por se tratar de envio de mensagem e a seleção de frequência levar à manipulação da velocidade de propagação dos pacotes pelos roteadores, pode-se assumir este valor como a *prioridade* da mensagem. Ao enviar-se uma mensagem com alta prioridade, os roteadores do caminho pelo qual seus pacotes trafegarão irão selecionar o relógio de maior frequência. Assim, para o MPSoC HeMPS-GLP, a prioridade está diretamente associada à seleção de frequência dos roteadores Hermes-GLP por onde a mensagem passa.

A solução implementada pelo Autor utiliza o menor valor binário de prioridade para representar a maior frequência de relógio, ou seja, o valor de prioridade ‘0’ corresponde à maior prioridade. Esta definição tem por objetivo dar certeza de qual valor de prioridade corresponde à maior frequência. Assim, independente de quantos relógios forem definidos para os roteadores, o valor zero indicará sempre a maior frequência de operação. Se fosse adotada a outra política, ao se adicionar mais relógios, o valor de prioridade para a máxima frequência seria alterado implicando em, além de recálculo para identificar esse valor, reprogramação das aplicações por parte do usuário.

### 5.4. Alterações no *microkernel*

Para enviar uma mensagem para outra tarefa pela NoC Hermes-GLP, deve-se informar ao *microkernel* qual valor de prioridade ela terá (ou equivalentemente qual a frequência selecionada). Para informar este valor, alterou-se a função de envio (*Send*) de mensagens para receber mais um parâmetro, a prioridade.

Em sua versão original, a função *Send* aceita dois parâmetros: a tarefa de destino e o ponteiro para o início da mensagem. Por definição, sua chamada é convertida em uma chamada de sistema interpretada pelo *microkernel*. Assim, o *microkernel* decodifica a operação e envia a mensagem para a interface de rede.

Para se passar a informação de prioridade na função de *Send*, adicionou-se um terceiro parâmetro para esta função. Assim, o envio de uma mensagem *M* para uma tarefa *T* com uma prioridade *P* deve seguir o código em (3).

$$\text{Send}(T, M, P) \quad (3)$$

Para mudar a chamada de sistema original a fim de incluir o terceiro parâmetro, alterou-se a codificação de definição da função *Send* encontrada no arquivo “*task.h*” no diretório *software/include* da HeMPS. Para evitar conflitos, a nova versão recebeu o nome “*task\_glp.h*” que deve obrigatoriamente ser incluída nas aplicações a serem executadas na HeMPS-GLP e que utilizem envio de mensagens.

A chamada de sistema no *microkernel* recebe os três parâmetros e insere-os em uma estrutura de dados dedicada ao envio da mensagem para a interface de rede do IP de processamento. Assim, como a prioridade deve ser uma das primeiras informações a chegar à rede, modificou-se a estrutura *Slot* existente no arquivo “*kernel.h*” para ter o valor inteiro de prioridade como primeiro dado. Desse modo, garante-se que o primeiro dado a chegar à interface de rede será a informação de prioridade.

Ao inserir-se mais um campo no pacote a ser enviado para a interface de rede, é necessário aumentar o tamanho da área de memória que será armazenada e enviada (tamanho do *Slot*). Essa operação não altera a quantidade de flits que trafegarão na rede, pois o campo de prioridade será transferido por um barramento dedicado na interface de rede (ou seja um sinal de canal lateral dedicado, do inglês *side channel signal*).

Com relação às demais mensagens trocadas entre os *microkernels* do MPSoC, definidas aqui como mensagens de serviço, estas também necessitam de um valor de prioridade. Como consenso entre os demais desenvolvedores do MPSoC HeMPS, estas mensagens devem circular pela rede com a menor prioridade possível para evitar interferência nas mensagens do usuário. Para definir este valor, o *microkernel* utiliza a quantidade de relógios da rede presente no arquivo “*HeMPS\_PKG.h*” na pasta do cenário. Assim, é automaticamente definida a prioridade de serviço das mensagens sem a interferência do usuário. Tendo esse dado, também se controla o maior valor de prioridade possível que, se for maior que o número de relógios menos um, a prioridade é convertida para este valor. Essa operação remove do hardware o controle para detectar tal violação.

Como o código do *microkernel* e de seu arquivo de definições são muito diferentes dos originais, os que são utilizados pelo MPSoC HeMPS-GLP foram renomeados para “*kernel\_glp.h*” e “*kernel\_glp.c*” tanto para os EPs escravos como para o EP mestre.

### 5.5. Alteração da estrutura da interface de rede

A interface de rede tem por objetivo prover a comunicação entre o processador e/ou o DMA do IP de Processamento e a porta local do roteador ao qual o IP de Processamento está conectado. Como se alterou tanto a rede Hermes-GLP quanto a sequência de dados que são enviados pelo processador, mudanças são necessárias para a comunicação correta entre estes módulos. A Hermes-GLP necessita receber juntamente com os dados a seleção de frequência definida pela prioridade de comunicação para permitir ao roteador selecionar corretamente sua frequência de operação. Como visto na Seção 5.4, a primeira informação que é enviada pelo processador à interface de rede é a prioridade.

Assim, a máquina de estados de envio deste módulo de hardware foi alterada para manipular este novo dado.

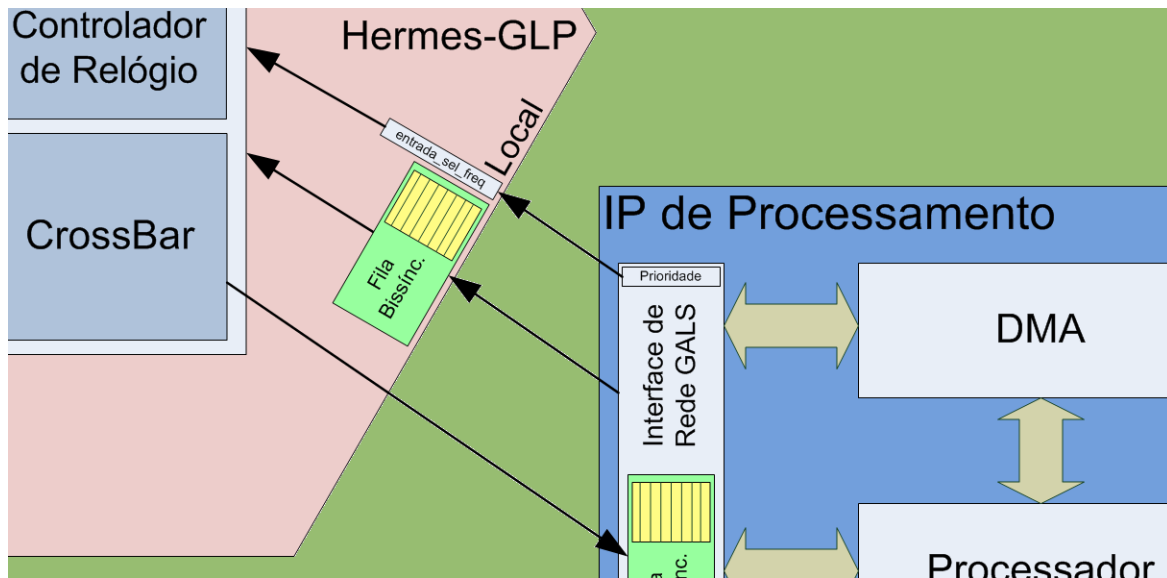


Figura 30. Detalhe da Interface entre um IP de Processamento e a porta local de um roteador Hermes-GLP.

Ao detectar um novo dado a enviar, a máquina de estados pertinente da interface de rede captura a prioridade e a armazena em um registrador dedicado. Como é possível observar na Figura 30, este registrador está ligado diretamente ao barramento de seleção de frequência da porta local do roteador Hermes-GLP, garantindo que esteja disponível antes mesmo do primeiro dado ser enviado. Além desta alteração, a interface de rede precisa receber dados do roteador ao qual está conectada. Entretanto, o roteador Hermes-GLP pode assumir qualquer frequência disponibilizada, sendo necessária uma interface especial para esta comunicação entre domínios distintos de relógio.

O uso de filas bissíncronas na interface de rede já havia sido abordado por trabalho desenvolvido anteriormente [ROS12]. Apesar de tratar-se de uma interface raciócrona, foi possível adaptá-la para operar assincronamente com a Hermes-GLP. Para tanto, incorporou-se o mesmo modelo de controle baseado em ponteiros das filas bissíncronas presente nas portas dos roteadores da Hermes-GLP e descritos na Seção 4.1.

## 5.6. Estrutura do hardware e interconexão de módulos HDL

Como a nova versão da HeMPS (Versão 4.0) organizou e separou os módulos de hardware por área de atuação, facilitou-se a adaptação e inclusão de novas estruturas. Assim, os arquivos de descrição de hardware do MPSoC HeMPS-GLP que foram modificados para incorporar a rede Hermes-GLP têm seu nome modificado para “*\_glp.vhd*”. Assim, identificam-se facilmente mudanças ao se navegar nas pastas de hardware do sistema.

Ao alterar-se a interface de rede para permitir a conexão com a Hermes-GLP criou-se um novo IP de Processamento, que se chamou de Plasma-GLP, constituído pela nova interface de rede GALS conectada a um módulo DMA, uma memória e um processador MIPS-Lite. Para ser reconhecido a partir do arquivo de definições “*HeMPS\_PKG.vhd*” do cenário definido pelo usuário, esta estrutura recebeu o nome de *plmglp* que juntamente com *plasma* e *mblite* podem constituir diferentes IPs de Processamento e, por conseguinte, diferentes EPs.



Entretanto, o elemento de processamento foi o que sofreu as maiores modificações para dar suporte ao MPSoC HeMPS-GLP. Para compatibilizar a conexão de rede também com prioridade, adicionou-se mais um barramento de seleção de frequência em cada porta deste módulo (Norte, Sul, Leste e Oeste) herdado do roteador Hermes-GLP. Apesar de serem necessárias para este novo IP de Processamento, estas conexões não são necessárias para as outras versões destes IPs. Não conectar estas portas não resulta em comportamento errôneo nas demais soluções. Inclusive, durante a síntese, essas portas sequer são criadas quando não possuem conexões com módulos internos. Assim, o elemento de processamento é o mesmo entre todas as três versões possíveis de IP de processamento, garantindo plena compatibilidade entre estas diferentes arquiteturas.

### 5.7. Simulações

Ao realizar as simulações do MPSoC HeMPS-GLP, após diversas alterações no buffer assíncrono responsável por grande parte da operação da rede Hermes-GLP obteve-se um comportamento estável e condizente com o desejado pelo usuário. Assim, as aplicações-exemplo disponíveis ao usuário funcionam corretamente, gerando respostas solicitadas mesmo em ambientes adversos de simulação. Por ambientes adversos entende-se aqui os casos em que as frequências de cada IP de processamento recebem uma frequência diferente da do seu vizinho e que também é diferente da utilizada na rede. Para ter certeza de que as bordas de relógio raramente coincidem entre os diferentes módulos da rede, utilizam-se frequências relacionadas por números primos. Para gerá-las, basta configurar os períodos de oscilação com números primos. Como estes números são divisíveis apenas por um e por eles mesmos, o seu inverso, ou seja, a frequência, também não possuirá múltiplos em relação às demais frequências.

Mesmo configurando este ambiente adverso, as simulações ainda indicaram a operação correta tanto do tráfego dos pacotes quanto da seleção de frequência baseada na prioridade de transmissão. Entretanto, estes testes foram inicialmente realizados em um MPSoC de pequenas dimensões (3x3). O arquivo de saída de dados disponibilizado pelo processador mestre da aplicação exemplo de comunicação está apresentado de maneira resumida na Figura 31(a). Entretanto, ao alterar-se este dimensionamento para o extremo suportado pela rede Hermes, observou-se comportamentos anômalos.

Constatou-se correta operação do MPSoC HeMPS para redes de até 3x3 EPs. Entretanto, ao utilizar a função ECHO para distâncias maiores que 3x3 nodos, a operação retornava apenas algumas das mensagens. Inicialmente, pensou-se haver um erro na integração da rede Hermes-GLP. Entretanto, após várias simulações e constatando a chegada de todos os flits dos pacotes ao destino, observou-se que esta falha já existia na versão original do MPSoC HeMPS. Este erro ocorria devido a uma falha na programação do *microkernel* dos EPs escravos onde, ao verificar que o DMA estava ocupado, simplesmente se abandonava o envio da mensagem, retornando ao programa principal sem realizá-lo. Atualmente, o *microkernel* da HeMPS aguarda a liberação do DMA para concluir o envio da mensagem, garantindo a sua operação correta.

Verificou-se outros comportamentos anômalos ao extrapolar as dimensões do MP-SoC além de 10x10 EPs. As tarefas eram erroneamente mapeadas em outros processadores. Isso se deve a um erro na ferramenta que gera o repositório do processador mestre. As coordenadas dos IPs de processamento onde uma determinada tarefa deveria e-

xecutar eram baseadas em números decimais. Como o formato padrão de endereçamento da rede Hermes é baseado em hexadecimal, alterou-se a forma de envio destes endereços para esta codificação. Esta mesma correção foi realizada sobre o endereçamento do EP mestre definido em cada EP escravo, justamente pelo mesmo motivo aqui apresentado. Estas alterações garantem plena compatibilidade com a rede permitindo a geração correta do repositório de tarefas e a definição do endereço do processador mestre, o que findou por garantir a operação correta no MPSoC em todos os testes realizados.

<pre># Master strikes again. \$,32,0,Communication task C started. \$,32,0,5962 \$,1,1,Communication task A started. \$,1,1,7530 \$,1,1,11163 \$,17,2,Communication task B started. \$,1,1,Communication task A finished. \$,17,2,8731 # Static mapped tasks allocated. Chama o PREMAP do PE 32 tarefa 0 Chama o PREMAP do PE 1 tarefa 1 Chama o PREMAP do PE 17 tarefa 2 Chama o PREMAP do PE 34 tarefa 3 \$,34,3,Communication task D started. \$,34,3,4794 \$,17,2,14019 \$,17,2,Communication task B finished. \$,32,0,64018 \$,32,0,Communication task C finished. \$,34,3,10 \$,34,3,11 ... \$,34,3,728 \$,34,3,729 \$,34,3,6 \$,34,3,112593 \$,34,3,Communication task D finished.</pre>	<pre># Master strikes again. \$,0,0,Communication task A started. \$,0,0,139629 # Static mapped tasks allocated. Chama o PREMAP do PE 0 tarefa 0 Chama o PREMAP do PE 240 tarefa 1 Chama o PREMAP do PE 15 tarefa 2 Chama o PREMAP do PE 255 tarefa 3 \$,0,0,158977 \$,15,2,Communication task B started. \$,0,0,Communication task A finished. \$,15,2,107246 \$,15,2,135631 \$,15,2,Communication task B finished. \$,240,1,Communication task D started. \$,240,1,25263 \$,255,3,Communication task C started. \$,255,3,77824 \$,255,3,115367 \$,255,3,Communication task C finished. \$,240,1,10 \$,240,1,11 ... \$,240,1,728 \$,240,1,729 \$,240,1,6 \$,240,1,203304 \$,240,1,Communication task D finished.</pre>
(a)	(b)

Figura 31. Trecho do arquivo de saída da simulação do MPSoC HeMPS-GLP 3x3 (a) e 16x16 (b).

Assim, realizou-se uma simulação de até 16x16 EPs, limite atualmente suportado pela rede Hermes e comprovou-se a operação correta do MPSoC. A saída proporcionada pelo processador mestre da aplicação exemplo chamada “communication” é, de forma resumida, apresentada na Figura 31 (b).

Para comprovar a correta operação lógica da seleção dinâmica de prioridades, a Figura 32 mostra a ativação do processo de inibição de relógio e uma troca da prioridades durante o recebimento de pacotes.

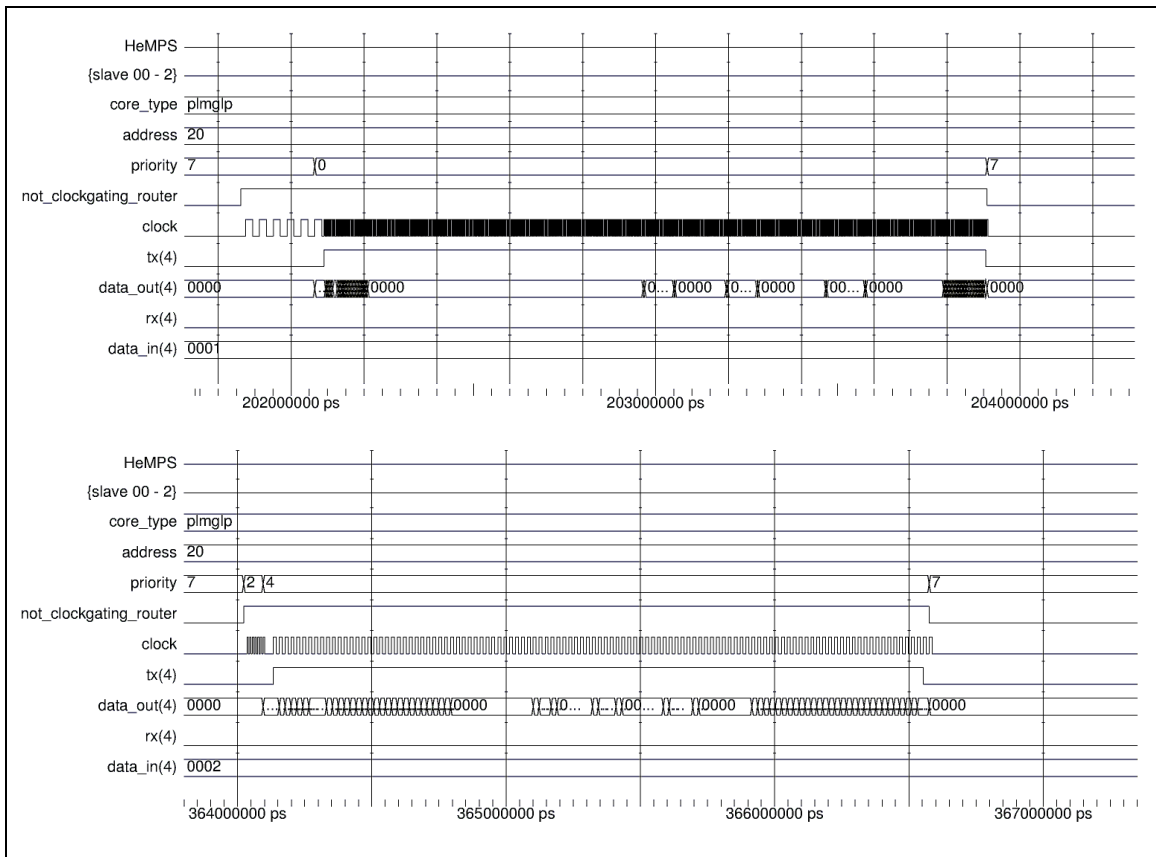


Figura 32. Formas de onda para identificar variação da prioridade e da frequência.

Assim, construiu-se até esta Seção um MPSoC que permite a utilização de uma nova rede (Hermes-GLP) para verificar as vantagens de seu uso em aplicações reais e em um ambiente real de simulação.



## 6. GERAÇÃO LOCAL DE RELÓGIO

O MPSoC HeMPS-GLP, como dito anteriormente, utiliza como arquitetura de interconexão a rede intrachip ou NoC Hermes-GLP. Entretanto, os roteadores desta, na sua versão original, somente viabilizam a seleção de uma frequência de operação dentre várias que devem ser disponibilizadas externamente. Por exemplo, se o projetista necessitar 16 frequências diferentes de operação, seria necessário o dispor de 16 árvores de distribuição distintas para toda a NoC. Isto, além de agregar área ao circuito integrado, devido a grande quantidade de árvores de distribuição de relógio, pode elevar sobremaneira a dissipação de potência do sistema.

O foco inicial da proposta da NoC Hermes-GLP foi disponibilizar um roteador de NoC com dissipação de potência ajustável via DFS. A questão da geração de frequências de operação deste foi deixada como trabalho futuro. Contudo, da discussão acima depreende-se que a economia de potência auferidas com o uso de DFS pode ser perdida e mesmo transformada em perda devido à multiplicação de redes de distribuição de relógio. Uma forma de evitar tais perdas é dispor de geradores locais de relógio para cada roteador. Em um MPSoC, este paradigma pode ser estendido, assumindo que cada elemento de processamento (PE ou IP) e cada roteador serão dotados de um gerador local próprio.

Em um primeiro trabalho nesta direção, Toffolo [TOF10] investigou e propôs uma topologia de oscilador controlado por tensão. O presente trabalho partiu dos resultados preliminares daquele trabalho, estendeu e modificou seu projeto para definir um gerador local de baixo custo, bom nível de precisão e tolerância a variações de processo (em inglês *process* ou P), tensão (em inglês *voltage* ou V) e temperatura (em inglês *temperature* ou T) ou PVT. Estes resultados originais constituem uma das principais contribuições deste mestrado. A conclusão do gerador ainda deverá, contudo contar com a realização de três etapas (todas previstas para serem realizadas no escopo de doutorado a ser iniciado em breve):

1. Geração do leiaute do oscilador na tecnologia selecionada (ST-65nm);
2. Projeto e implementação da fonte de corrente analógica que o gerador requer;
3. Prototipação em chip do gerador e sua validação.

Aqui, um Gerador Local de Relógio (GLR) é uma fonte de sinal de relógio interna ao MPSoC para alimentar um ou mais módulos IP deste. Geralmente, como é possível observar na Figura 33, este gerador de relógio possui um oscilador que produz uma frequência de base, um circuito atuador que permite alterar a frequência conforme parametrização (possivelmente externa) e um controlador para garantir operação do módulo dentro de determinados limites. Quando o atuador possui como entrada informação digital, o conjunto conversor associado ao oscilador pode receber o nome de *oscilador controlado digitalmente* ou DCO.

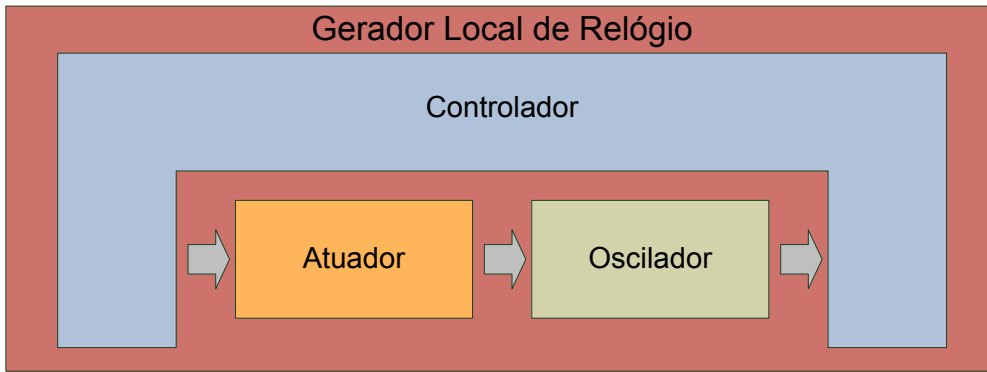


Figura 33. Diagrama de blocos genérico de um gerador local de relógio.

Este trabalho foca suas atividades sobre os blocos Atuador e Oscilador, deixando o Controlador como trabalho paralelo [HEC11]. Para facilitar o projeto do controlador, decidiu-se disponibilizar entradas binárias ao Atuador e uma saída binária para o oscilador. Enfatiza-se aqui a necessidade do Bloco Controlador para compensar variações de processo, tensão e temperatura e garantir a frequência solicitada pelo(s) subsistema(s) conectado(s) ao GLR.

Este Capítulo aborda nas Seções 6.1 (Oscilador) e 6.2 (Atuador) de forma mais genérica as características de cada um dos módulos desenvolvidos neste trabalho. O próximo Capítulo apresenta o projeto detalhado do oscilador controlado digitalmente e os resultados de sua simulação.

### 6.1. Bloco oscilador

Um oscilador tem como objetivo gerar uma variação repetitiva de um determinado sinal a cada período pré-determinado de tempo. Circuitos osciladores podem apresentar diversas topologias, como ilustram os exemplos vistos na Figura 34.

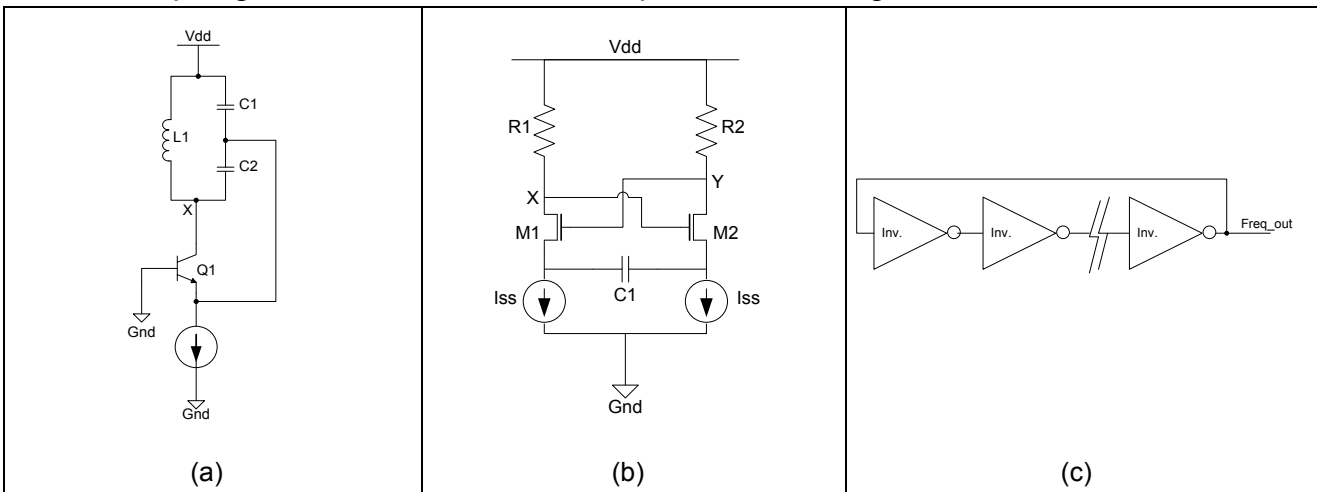


Figura 34. Diferentes modelos de topologias de osciladores construídos como circuitos eletrônicos. (a) osciladores sintonizados, (b) osciladores de relaxamento, e (c) osciladores em anel.

As topologias usuais de osciladores controlados em silício são: osciladores sintonizados (a), osciladores de relaxamento (b) ou osciladores em anel (c) [RAZ98]. Osciladores em anel são constituídos de um número ímpar de elementos inversores realimentados. Osciladores de relaxamento carregam e descarregam alternadamente um capacitor com uma corrente constante entre dois níveis de limiar. Osciladores sintonizados contêm

um ressonador passivo como tanque indutor-capacitor (LC), linha de transmissão ressonante, ou cristal, que servem como um elemento de configuração da frequência.

Como este trabalho visa projetos de circuitos integrados e o uso de células capacitivas ou indutivas agrega muita área e é suscetível a variações de processo em tecnologias nanométricas [BAK10], concentra-se aqui esforços no oscilador em anel, também conhecido como oscilador realimentado com elementos de atraso. Para este tipo de oscilador, a constante de tempo depende diretamente da capacitância de cada nodo e da corrente fornecida a estas. À medida que a corrente carrega ou descarrega cada capacitor, a porta inversora correspondente muda o valor de sua saída, alternando entre “0” lógico e “1” lógico.

### 6.1.1. Oscilador realimentado com elementos de atraso

Osciladores em anel normalmente baseiam-se na manipulação de elementos de atraso. Quando possuem entradas simples (não diferenciais) estes osciladores sempre são formados por um número ímpar de componentes de atraso (inversores ou componentes equivalentes) a fim de garantir a oscilação. Estes osciladores podem manipular seja sua capacitância seja sua corrente de carga para alterar a constante de tempo.

Um oscilador em anel clássico pode gerar diferentes frequências conectando ou desconectando pares de inversores para variar a quantidade de cargas do oscilador e, conseqüentemente, o tempo de propagação do laço. A principal vantagem deste tipo de oscilador é a possibilidade de projeto completamente digital, pois utiliza apenas portas lógicas convencionais como inversores e/ou portas “E” negadas (em inglês *not and* ou *NAND*). Entretanto, a capacitância de porta de transistores de tecnologias nanométricas é significativamente pequena, na casa dos *femtoFarads* (fF). Neste caso, este tipo de oscilador agrega área considerável para gerar frequências mais baixas, pois necessita de muitos estágios capacitivos.

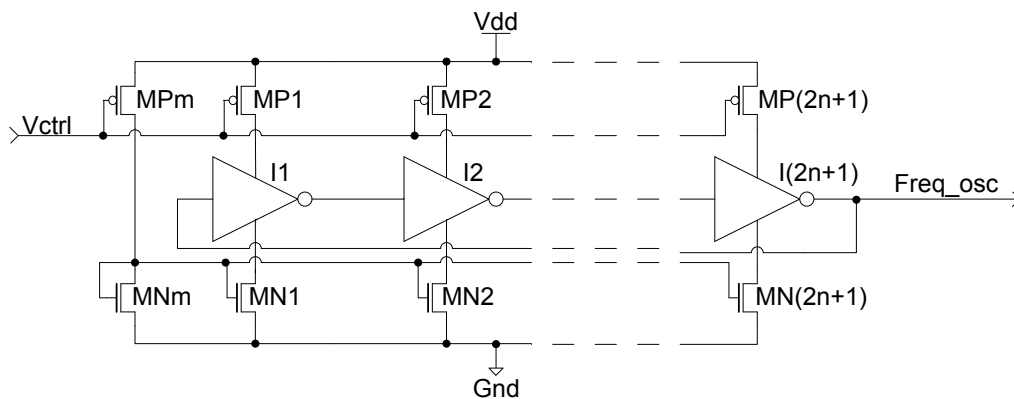


Figura 35. Oscilador em anel com controle de corrente.

O oscilador em anel com controle de corrente (em inglês, *current-starved ring oscillator*) [BAK10] se baseia em manipulação de corrente. Para tanto, utiliza um transistor em cada nodo de alimentação da porta inversora, como mostra a Figura 35, para permitir a manipulação da corrente elétrica que passa por esta. Ao aplicar-se uma tensão de controle ( $V_{ctrl}$ ), varia-se a corrente incidente no inversor alterando-se a constante de tempo que, por sua vez, faz variar a frequência de oscilação. Assim, pode-se dizer que, comparado ao oscilador em anel, este modelo deverá ocupar menor área, pois acrescenta apenas

dois transistores a cada porta inversora. Entretanto, este tipo de circuito é totalmente personalizado (em inglês *full custom*) sendo necessário empregar técnicas de projeto analógico para seu desenvolvimento.

Para calcular a frequência de oscilação deste circuito, necessita-se primeiramente calcular as capacitâncias envolvidas em cada nodo do oscilador. Segundo a Equação (4), a capacitância de um determinado nodo ( $C_{tot}$ ) é dada pela soma das capacitâncias de saída do inversor anterior ( $C_{out}$ ) e de entrada do próximo inversor ( $C_{in}$ ). Segundo [BAK10], é possível aproximar a capacitância de entrada do inversor como sendo 3/2 da capacitância de saída do inversor considerando as capacitâncias de ambos os transistores operando entre triodo e corte e a capacitância de Miller. Garantindo-se que todos os inversores possuem o mesmo dimensionamento, a capacitância total de um nodo é dada pela Equação (5), onde  $C'_{ox}$  é a capacitância do óxido de silício por área,  $W_p$  e  $L_p$  são as dimensões do transistor PMOS e  $W_n$  e  $L_n$  são as do NMOS [BAK10].

$$C_{tot} = C_{in} + C_{out} = \frac{3}{2}C_{out} + C_{out} = \frac{5}{2}C_{out} \quad (4)$$

$$C_{tot} = \frac{5}{2}C'_{ox}(W_pL_p + W_nL_n) \quad (5)$$

Ao calcular a capacitância do nodo, é possível obter a constante de tempo de subida ( $t_1$ ) e de descida ( $t_2$ ) da tensão neste nodo. Para se calcular estes valores, necessita-se saber a resistência equivalente da constante de tempo. Para determinar este valor, utiliza-se como referência a tensão  $V_{SP}$  indicando a transição do valor lógico (alto ou baixo) na saída da porta inversora. Assim, obtêm-se as Equações (6) e (7) conforme as correntes  $I_{MP}$  e  $I_{MN}$ , que são respectivamente a corrente de carga e descarga da capacitância  $C_{tot}$  e  $VDD$  é a tensão de alimentação do circuito.

$$t_1 = C_{tot} \cdot \frac{V_{SP}}{I_{MP}} \quad (6)$$

$$t_2 = C_{tot} \cdot \frac{VDD - V_{SP}}{I_{MN}} \quad (7)$$

Por fim, a frequência de oscilação ( $f_{osc}$ ) é dada pelo inverso da soma das constantes de tempo de cada nodo do oscilador em anel com controle de corrente. Assim, determina-se a frequência de oscilação pela Equação (8), onde  $N$  é o número de estágios do oscilador em anel e  $I_D$  é a corrente manipulada para variar a constante de tempo ( $I_D = I_{MN} = I_{MP}$  para igualar as constantes de tempo de subida e descida na tensão de saída do inversor).

$$f_{osc} = \frac{1}{N(t_1 + t_2)} = \frac{I_D}{N \cdot C_{tot} \cdot VDD} \quad (8)$$

Como é possível observar na Equação (8), a frequência de oscilação é dependente direta da corrente ajustada para os inversores. Caso o ajuste disponibilizado à  $I_D$  seja fino o suficiente, por meio desta variável podem-se compensar alterações em  $C_{tot}$  e em  $VDD$  causadas por variações no processo de fabricação, na tensão de alimentação e na temperatura. Entretanto, como o MPSoC HeMPS-GLP não se necessita de exatidão em suas frequências de operação devido as suas filas bissíncronas, tais osciladores podem ser



muito bem utilizados neste caso. A única questão pertinente envolve casos de aplicações de tempo real. Nestes casos, deve-se garantir pelo menos os requisitos mínimos de prazos (em inglês *deadlines*) da aplicação [JER05]. Neste caso, qualquer frequência superior à necessária atenderá este quesito.

Entretanto, aqui o que se explora é a possibilidade de utilizar este oscilador para gerar diferentes níveis de frequência ajustando-se a corrente  $I_D$ . Deste modo, foge-se um pouco do foco original de aplicação deste oscilador, conforme dito anteriormente.

## 6.2. Bloco atuador

O circuito atuador tem como objetivo produzir um estímulo elétrico suficientemente conforme para ser utilizado na entrada do bloco oscilador. Este estímulo é gerado a partir de uma determinada entrada, possivelmente não adaptada ao que o oscilador pressupõe. Como dito anteriormente, para facilitar o desenvolvimento do bloco Controlador, pressupõe-se aqui que o Atuador deverá receber na sua entrada um dado binário e convertê-lo em estímulo elétrico para manipular a corrente do oscilador em anel com controle de corrente. Assim, busca-se um conversor digital analógico (em inglês *digital to analog converter* ou DAC) que atue no controle de corrente do DCO.

### 6.2.1. Conversores digitais analógicos

Como o próprio nome já deixa claro, os conversores digitais analógicos convertem sinais binários discretos em sinais elétricos contínuos. Estes sinais elétricos podem ser tanto configurar-se como valores de tensão, designando-se neste caso conversores “modo tensão”, quanto em valores de corrente, quando são designados conversores “modo corrente”. Como o controle do oscilador em anel escolhido já é baseado em corrente concentram-se aqui os estudos em conversores digitais analógicos em modo corrente [JOH97].

#### 6.2.1.1. Codificação

O princípio básico de conversores digitais analógicos em modo corrente é a soma de diversas correntes oriundas de fontes de correntes conectadas em paralelo, como é possível observar na Figura 36.

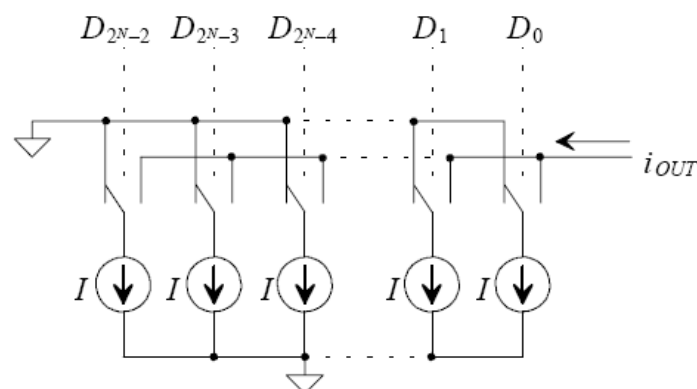


Figura 36. Conversor digital analógico com saída em modo corrente com codificação de termômetro [BAK10].

Ao ligar-se uma das chaves  $D$ , a fonte de corrente correspondente é conectada à saída. Como as fontes de corrente utilizadas neste conversor possuem peso semelhante ( $I$ ), cada bit acionado na entrada corresponde diretamente ao incremento de uma fonte de corrente  $I$ . Esta relação direta entre entrada/saída com correspondência um para um (1:1) é característica da codificação de termômetro (em inglês *thermometer code*) [BAK10], também denominada unária por corresponder a um código unário. Assim, para representar  $255I$  em corrente, seriam necessárias 255 chaves e, por consequência, 255 bits de entrada.

Já na codificação binária, cada fonte de corrente possui peso diferenciado em relação à outra, como é visível na Figura 37. A fonte de um determinado bit terá sempre o dobro de corrente da sua anterior e a metade de sua posterior gerando contribuições menos significativas para os primeiros bits ( $D_0$ ) e mais significativas para os últimos bits ( $D_{N-1}$ ). Utilizando-se deste arranjo de incrementos, pode-se afirmar que a codificação utilizada é binária. Levando em conta o mesmo caso anterior de representar-se  $255I$  em corrente de saída, seriam necessárias apenas 8 chaves ( $I+2I+4I+8I+16I+32I+64I+128I=255I$ ) e, logicamente, 8 bits de entrada.

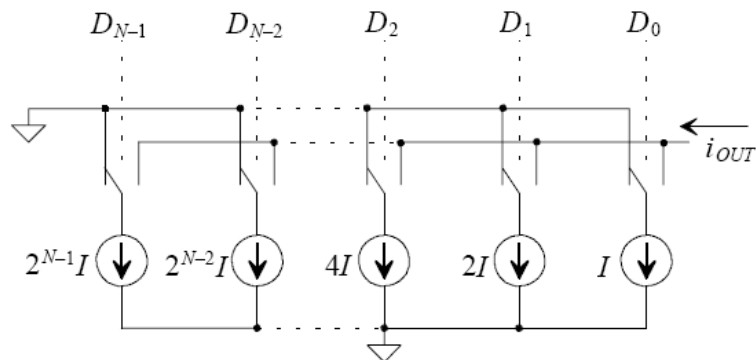


Figura 37. Conversor digital analógico binário em modo corrente [BAK10].

### 6.2.1.2. Topologias

Antes de analisarem-se as topologias de fontes de corrente, necessita-se esclarecer alguns princípios básicos de operações de transistores. Assim, os próximos três parágrafos visam esclarecer alguns conceitos fundamentais de microeletrônica na utilização de transistores CMOS para manipulação de corrente elétrica.

Basicamente, um transistor MOSFET, representado de maneira simplificada na Figura 38, pode ser configurado como uma fonte de corrente alimentada por tensão e fonte de tensão alimentada por corrente, dependendo de como seus terminais estão conectados.

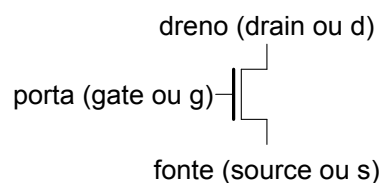


Figura 38. Representação simplificada do transistor NMOS.

No caso mais usual, o transistor é uma fonte de corrente alimentada (9)

por tensão. Neste caso, ao garantir-se que a tensão de dreno-fonte ( $v_{DS}$ ) seja igual ou superior a tensão de porta-fonte ( $v_{GS}$ ) diminuída da tensão de limiar do transistor ( $v_T$ ), tem-se o transistor na região de saturação. Nesta região, é mantida a relação que consta na equação de segunda ordem do transistor na região de saturação correspondendo aqui à Equação (9) [RAZ02].

$$i_D = \frac{1}{2} \mu C_{OX} \frac{W}{L} (v_{GS} - v_T)^2$$

Nesta Equação, a corrente de dreno ( $i_D$ ) é diretamente proporcional à mobilidade dos elétrons ( $\mu$ ), à capacitância do óxido da tecnologia ( $C_{OX}$ ), ambas dependentes da tecnologia. Além disso,  $i_D$  é proporcional à largura do transistor ( $W$ ), ao quadrado da diferença entre a tensão de porta-fonte ( $v_{GS}$ ) e à tensão de limiar do transistor ( $v_T$ ). A corrente  $i_D$  também é inversamente proporcional ao comprimento do canal ( $L$ ).

Caso os terminais de porta e de dreno estejam interligados, formando a configuração chamada de diodo (conforme se pode observar na Figura 39 para o transistor M1), garante-se o pressuposto de manutenção do transistor na região de saturação. Neste caso, à medida que se altera a corrente de referência  $I_{REF}$ , altera-se a tensão no nodo  $N$ . Quando esta porta é interligada a um transistor de dimensões idênticas, ou seja,  $M1=M2$ , copia-se a corrente que incide no transistor M1 para M2 (desde que esteja saturado) formando um espelho de corrente.

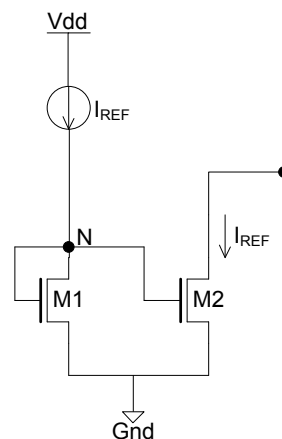


Figura 39. Associação de transistores formando um espelho de corrente.

Neste trabalho foram analisadas duas topologias para gerar as fontes de corrente do conversor digital analógico (DA). Ambas utilizam espelhos de corrente e dimensionamento de transistores para manipular uma fonte de corrente de referência e disponibilizar uma saída proporcional a esta.

A primeira topologia analisada utiliza a mesma ideia existente nas redes de resistores R-2R dos DACs em modo tensão. Esta estrutura recebe o nome de rede W-2W justamente porque manipula a largura do canal do transistor ( $W$ ) gerando as relações binárias entre as correntes de saída baseando-se na corrente de referência  $I_{REF}$ , como é possível observar na Figura 40.

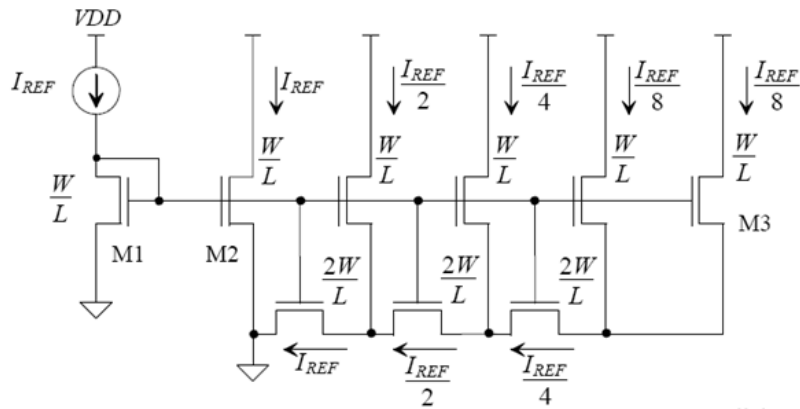


Figura 40. Topologia W-2W para conversores DA modo corrente [BAK10].

Por meio de associações série/paralelo e manipulando o dimensionamento dos transistores, permite-se a geração das fontes necessárias com área consideravelmente pequena. Entretanto, esta topologia pode não ser viável, caso se necessite de muitos bits de resolução, devido às associações em série de seus transistores. Como esta relação é um para um (1:1) serão necessários 8 transistores em série para representar 8 bits. Em tecnologias mais atuais, a tensão de alimentação é muito baixa, sendo impraticável deixar os oito transistores em série saturados.

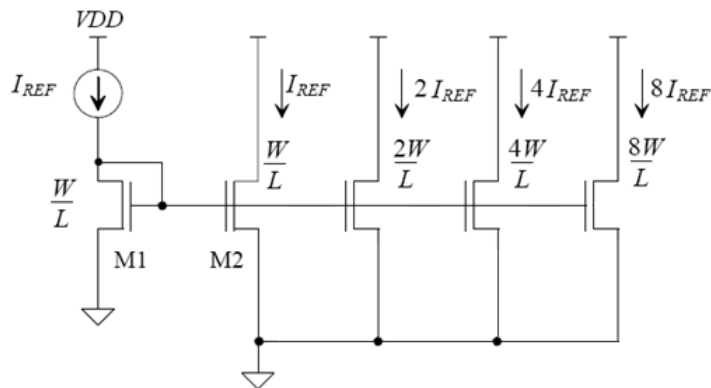


Figura 41. Topologia de multiplicação direta de corrente.

A segunda topologia analisada também manipula a corrente pela soma binária ponderada de fontes de corrente. Esta técnica utiliza como base a codificação de termômetro apresentada na Seção 6.2.1.1, modificando-se apenas o peso que cada chave possui. Para tanto, uma determinada chave possui o dobro de peso de sua antecessora e a metade de sua sucessora. Esta manipulação de corrente pode ser obtida ou alterando-se a largura do canal do transistor ( $W$ ) conforme indica a Equação (9), ou simplesmente conectando mais transistores em paralelo. Entretanto, esta técnica apresenta incremento exponencial em área onde a cada novo bit associado ao conversor dobra-se a área ocupada. Por outro lado, pode-se empregá-la em tecnologias mais atuais, pois não possui associações de transistores em série. Entretanto, deve-se atentar o uso de baixas correntes nestes circuitos para reduzir a dissipação de potência causada pela multiplicação da corrente de referência.

## 7. PROJETO DO OSCILADOR CONTROLADO DIGITALMENTE

Tendo como base as informações apresentadas nos Capítulos anteriores, descreve-se neste o projeto do oscilador controlado digitalmente. Inicialmente, apresentam-se considerações que delimitam a necessidade do projeto e o foco de atuação do oscilador, na Seção 7.1. Após, a Seção 7.2 analisa a estrutura do oscilador escolhido e a viabilidade de se obter uma faixa de variação de frequências com o mesmo. Nesta Seção também sugere-se algumas adaptações, visando aplicações em circuitos de baixa dissipação de potência e possibilidades de redução na dissipação do próprio oscilador. Posteriormente, a Seção 7.3 discute o conversor DA utilizado, sua implementação inicial e seus resultados preliminares nas Seções 7.4 e 7.5. Depois, analisam-se os resultados obtidos e propõem-se alterações, visando solucionar pontos falhos do projeto inicial na Seção 7.6. Por fim, a Seção 7.7 apresenta os resultados do circuito final e algumas comparações de área e dissipação de potência do oscilador, comparado ao roteador Hermes-GLP.

### 7.1. Considerações iniciais do projeto

O oscilador controlado digitalmente pesquisado e desenvolvido tem como objetivo gerar uma determinada frequência para algum bloco do MPSoC HeMPS-GLP. Este gerador deve possuir dimensões reduzidas quando comparado ao menor módulo do MPSoC, normalmente definido como sendo o próprio roteador da NoC do MPSoC. Deve-se utilizar alguma tecnologia recente disponível. No caso, optamos pela tecnologia CMOS 65nm da STMicroelectronics (STM). Ainda, o oscilador deve poder gerar frequências de até 1 GHz. O módulo deve sempre operar associado diretamente a componentes digitais, devendo então operar com a mesma tensão de alimentação dos blocos digitais acionados pelo oscilador. Por fim, o oscilador deve apresentar baixa dissipação de potência.

Deve-se ressaltar, entretanto, que este bloco será empregado unicamente em circuitos digitais. Assim, deve-se garantir um processo de intercâmbio de frequências livre de espúrios. Isto visa permitir aos circuitos dependentes deste operar mesmo durante a transição entre frequências. Por fim, a tensão de alimentação incidente no oscilador deve ser idêntica à tensão de alimentação do MPSoC ao qual estará ligado, no caso da tecnologia escolhida, a uma fonte de 1,2V.

Antes de iniciar o projeto analógico, deve-se determinar o tipo de transistor a empregar. As bibliotecas STM 65nm disponibilizam três tipos diferentes de transistores: de propósito geral (em inglês *general purpose* ou GP), de baixa potência (em inglês *low power* ou LP) e de alta potência (em inglês *high power* ou HP). Neste caso opta-se pelos transistores LP visando atender uma das especificações fundamentais do projeto (baixo consumo). Os transistores LP ainda podem possuir diferentes níveis de tensão de limiar ( $v_T$ ): baixa ( $Lv_T$ ), padrão ( $Sv_T$ ) e alta ( $Hv_T$ ). Neste trabalho escolhe-se usar transistores de tensão de limiar padrão ( $Sv_T$ ). Os transistores de tensão de limiar baixa tendem a apresentar maior dissipação de potência estática, devido ao excesso de correntes de fuga [BHA09]. Já os transistores de tensão de limiar alta tendem a reduzir as frequências máximas de operação alcançáveis, dificultando a manipulação das correntes no oscilador

escolhido. Finaliza-se enfatizando o projeto em área reduzida. Para tanto, sempre que possível, utilizam-se transistores de tamanho mínimo.

## 7.2. Análise do oscilador em anel com controle de corrente

Como discutido na Seção 6.1, o oscilador escolhido para este trabalho baseia-se na técnica de controle de corrente dos elementos de atraso, devido ao potencial desta para redução da área utilizada e para a redução na dissipação de potência. Devido à tensão de alimentação disponibilizada na biblioteca de 65nm ser 1,2V, idealizou-se inicialmente o controle de corrente sobre apenas uma das constantes de tempo, usando apenas um transistor PMOS que controlaria a corrente de carga da capacitância dos nodos dos inversores, conforme observado na Figura 42. Esta escolha visava ampliar o número de níveis de frequência disponibilizados, pois seria apenas necessário manter um transistor na região de saturação; o PMOS de controle. Entretanto, esta topologia necessita de correção no ciclo de trabalho do relógio gerado (em inglês, *duty cycle*), visto que o tempo de subida da onda quadrada será diferente do tempo de descida. Mesmo ajustando manualmente o ciclo de trabalho do relógio, ao alterar-se a frequência, este valor é alterado novamente. Por este motivo, retomou-se a utilização de controle em ambas constantes de tempo com dois transistores de controle por elemento de atraso, um PMOS e um NMOS.

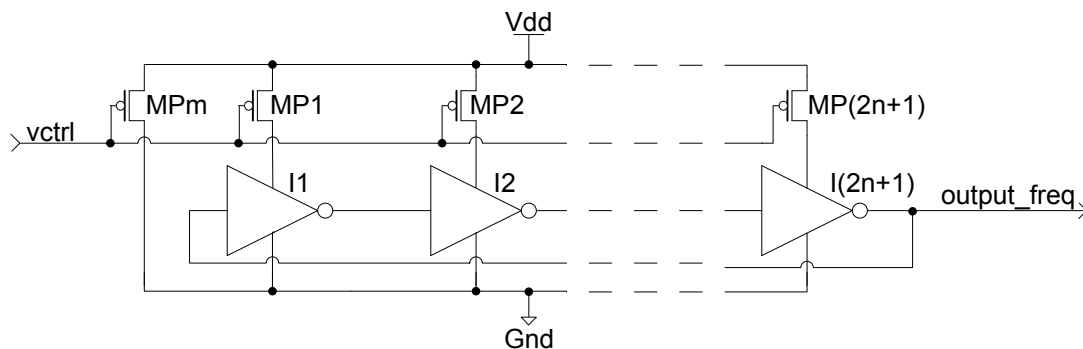


Figura 42. Oscilador controlado por corrente de carga.

Usando como base a Equação (8) da Seção 6.1.1, os limites de frequência definidos na Seção anterior, os parâmetros tecnológicos dos transistores LP-SVt da biblioteca STM 65nm e, como passo inicial, um oscilador de cinco estágios, pode-se estimar a corrente  $I_D$  necessária ao oscilador. Ao aplicar-se a corrente de  $9,1 \mu\text{A}$  ao oscilador, obtém-se uma frequência em torno de 847 MHz. Este erro é esperado, justamente pela aproximação no cálculo da capacitância total do nodo. Por meio de um ajuste fino, chega-se à corrente de  $10,7 \mu\text{A}$  que finalmente produz a frequência de 1 GHz.

A partir deste ponto, analisa-se a possibilidade de alterar o circuito para permitir a paralisação do relógio sem a geração de espúrios e prevendo a redução de sua dissipação de potência.

### 7.2.1. Inibição de sinal de relógio

Inicialmente verificaram-se as características de oscilação presentes no circuito. Ao utilizar um circuito externo que permite isolar o oscilador do circuito que o emprega, nota-se que o oscilador continua ativo, desperdiçando energia. Entretanto, é possível observar que ao abrir-se o anel, a realimentação é desfeita e a oscilação cessa. Por este motivo,

este trabalho busca inibir o relógio abrindo-se o circuito em anel. Entretanto, abrir o circuito poderá provocar metaestabilidade no sinal de relógio podendo também causar comportamento anômalo no circuito dele dependente.

A partir desta observação, verifica-se a necessidade de garantir que, mesmo em laço aberto, o oscilador deve permanecer com valor estável em sua saída. Para tanto, inseriu-se um sistema de memorização que mantém o sinal estável enquanto o anel permanecer aberto. Assim, conforme apresenta a Figura 43, altera-se o esquema original do oscilador em anel com controle de corrente de modo a incluir uma célula de memória, duas chaves S1 e S2 e uma porta inversora para inibir o relógio. Ao acionar-se o sinal de inibição de relógio (*Inibir\_relogio*), a chave S1 é aberta e a chave S2 fecha acionando a memorização do último valor de saída do oscilador e mantendo também um valor estável em todos os nodos internos do oscilador.

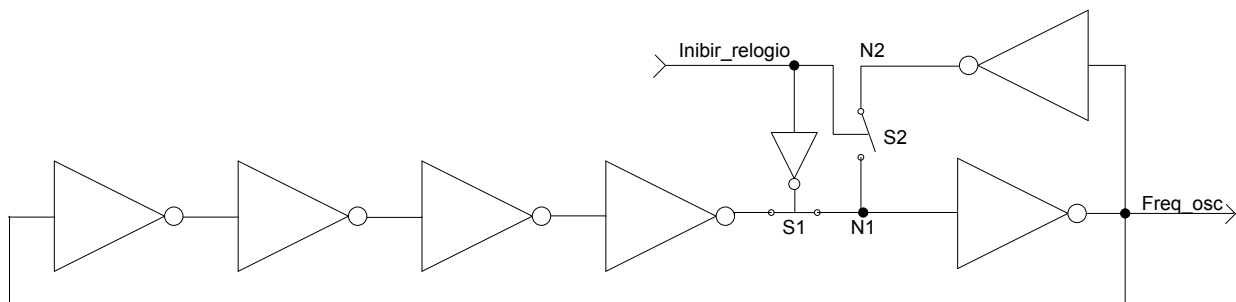


Figura 43. Oscilador em anel mostrando a estrutura simplificada do circuito de inibição de relógio.

Apesar do circuito permitir a paralisação do sinal de relógio, deve-se garantir que o sinal esteja estável antes do chaveamento, para evitar geração de espúrios na saída de relógio. É necessário ter certeza do acionamento do sinal que inibe o relógio seja acionado quando os nodos N1 e N2 tiverem o mesmo valor lógico. Para realizar esta lógica utiliza-se uma porta NAND de três entradas, como se observa na Figura 44.

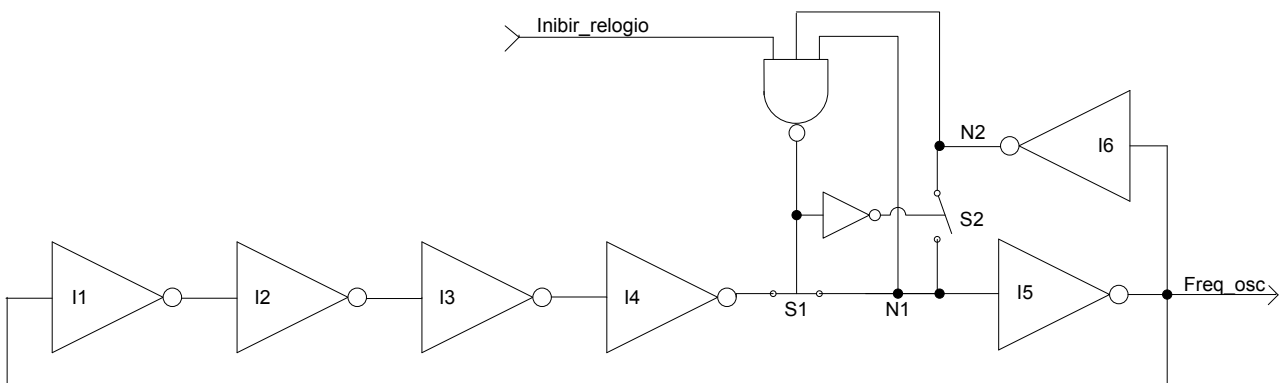


Figura 44. Oscilador em anel com circuito de inibição de relógio.

Analisando-se o comportamento do circuito da Figura 44 aplicando-se a técnica de controle de corrente, se este atuar sobre o estágio I5, ainda existe a possibilidade de se gerar espúrios pelo atraso do sinal que passa por I6. Assim, propôs-se que este inversor não possua controle de corrente, conforme a versão final do oscilador, detalhada na Figura 45.

Ao eliminar o controle de corrente do inversor I5, verificam-se contribuições significativas para o comportamento do circuito. A primeira envolve a certeza de memorização

do valor correto. Como não se reduz a constante de tempo deste transistor, ele atua sempre mais rápido que os demais pertencentes ao anel. Além disto, como a saída de I6 sempre estará em fase com a entrada de I5, pode-se garantir a inexistência de espúrios. A segunda contribuição reside na economia de energia. Ao acionar-se a memória, permite-se que seja completamente desligada a manipulação de corrente dos inversores I1, I2, I3 e I4 podendo ser desconectados da alimentação juntamente com todo o circuito que manipula a corrente destes. Entretanto, esta segunda contribuição não é explorada pelo Autor sendo deixada como trabalho futuro. A terceira contribuição é a forma de onda da saída do oscilador. O controle de corrente ocasiona uma forma de onda triangular na saída do oscilador. Ao evitar o controle de corrente no último estágio, o mesmo regenera a onda quadrada esperada pelos circuitos digitais acionados pelo oscilador.

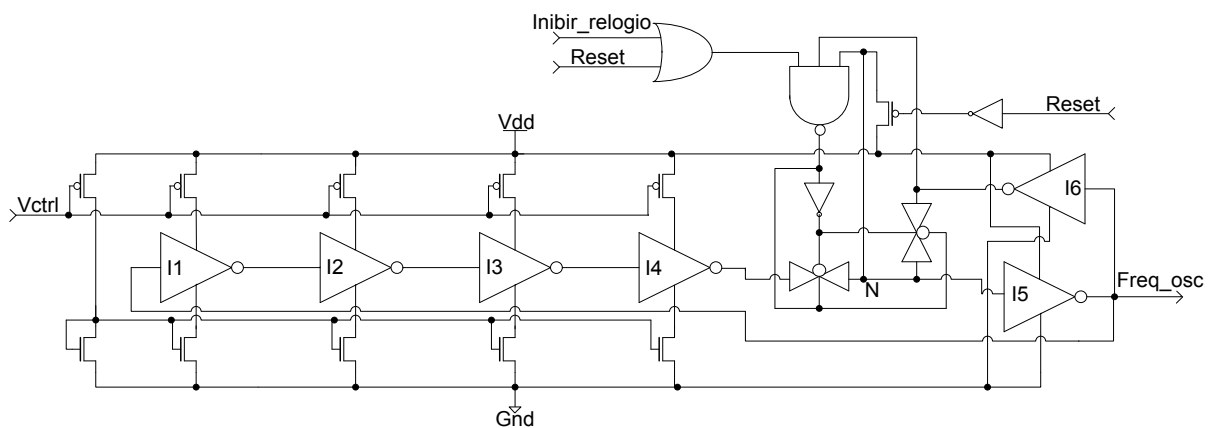


Figura 45. Oscilador em anel com controle de corrente com inibição de relógio (versão final).

Entretanto, a alteração proposta não vem sem desvantagens, pois ela acaba por alterar a constante de tempo do circuito como um todo, tornando necessário um novo ajuste fino executado em simulação para chegar-se a frequência máxima especificada de 1 GHz.

A Figura 45 também identifica o circuito de *reset*, que neste caso é utilizado para inicializar o oscilador. Observa-se que, quando o sinal *reset* subir, o circuito que produz a inibição do relógio é acionado e memoriza-se no nodo marcado como N na Figura 45 o valor lógico “1”. Neste caso, o sinal de *reset* deve permanecer em ‘1’ até que todos os inversores do oscilador possuam valores lógicos estáveis em seus nodos (ou em nível lógico alto ou baixo). Para garantir isto, o *reset* deve permanecer ativo pelo período relativo ao ciclo de relógio de menor frequência que possa ser sintetizada pelo oscilador.

Em relação à carga de saída que o oscilador suporta, deve-se enfatizar que os projetos visaram a aplicação em uma árvore de relógio de um circuito digital. Como tais circuitos possuem características únicas dependendo de sua funcionalidade, sua árvore de distribuição de relógio também varia conforme seu projeto. Por este motivo, o amplificador comumente utilizado na saída do circuito da Figura 45 foi abstraído, devendo ser inserido pela ferramenta de CAD dedicada ao projeto da parte digital do sistema. Entretanto, deve-se informar a esta ferramenta a máxima capacitância de entrada para o sinal de relógio como sendo de 5fF, aquela utilizada nos experimentos aqui realizados.

Aqui também deve-se deixar claro que, caso o circuito digital dependa de um sinal de *reset* síncrono, ou seja, dependente do sinal de relógio, este deve ser fornecido pelo bloco controlador do Gerador Local de Relógio. Sugere-se que ao aplicar o *reset* ao oscilador, a máquina de estados do Bloco Controlador acione instantaneamente um *reset* in-



terno ao módulo digital dele dependente. Depois que o sinal de relógio gerado pelo oscilador estiver estabilizado, conta-se mais um ciclo e posteriormente desativa-se o sinal de *reset* do módulo digital.

Finalmente, a Figura 46 apresenta a interface externa resultante para o oscilador. Pode-se observar que apenas o sinal *Vctrl* é analógico e visa manipular a constante de tempo dos elementos de atraso. Os demais sinais utilizados na interface deste bloco são digitais e binários sendo o sinal *Inibir\_relogio* proveniente do bloco Controlador e o *Reset* gerado externamente ao MPSoC. Para viabilizar a manipulação do sinal analógico *Vctrl* desenvolve-se na próxima Seção a estrutura do bloco Atuador.

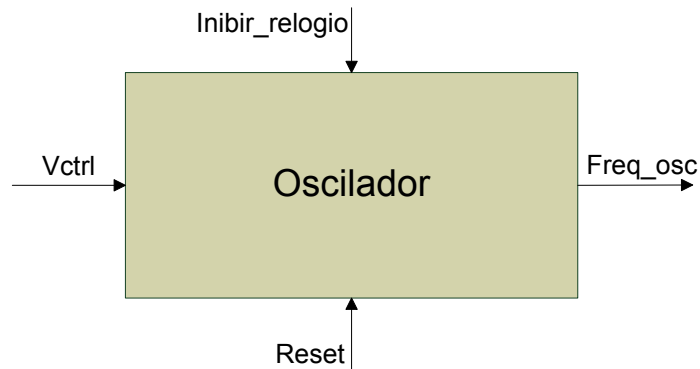


Figura 46. Interface externa do Oscilador com suas respectivas entradas e saídas.

### 7.3. Conversor digital analógico

Dado o bloco oscilador desenvolvido na Seção 7.2, é necessário propor uma estrutura de conversor DA capaz de gerar o sinal *Vctrl*, lembrando que a entrada do atuador aqui será um sinal digital correspondente a uma seleção de frequência, a ser convertida em valor de frequência. Para a escolha do conversor digital analógico a atuar sobre o oscilador, inicialmente analisou-se a diferença entre os passos de frequência a serem gerados pelo oscilador em anel com controle de corrente. Tendo como base os limites da especificação entre 0 Hz (correspondendo à inibição de relógio) e 1 GHz (correspondendo à frequência máxima), determinou-se que passos de aproximadamente 50 MHz seriam suficientes para atender as necessidades de instância do MPSoC alvo. Para simplificar a lógica, adota-se o uso de 4 bits de resolução para definir a frequência de oscilação. Com 4 bits é possível gerar 16 níveis distintos de frequência com intervalos de frequência de 66 MHz.

Ao analisarem-se as duas topologias estudadas e apresentadas anteriormente na Seção 6.2.1.2, verifica-se que a rede W-2W (ver Figura 40) seria uma opção interessante para gerar o circuito planejado. Entretanto, variações de processo podem causar alterações nas dimensões e no comportamento de transistores, resultando em problemas de monotonicidade, ou seja, nem sempre ocorre incremento na saída ao incrementar a entrada binária [BAK10]. Nestes casos, o uso de codificação unária na seleção de frequência garante monotonicidade mesmo com variações de processo, pois ao adicionar-se mais uma fonte de corrente, a resultante sempre será maior que sem esta fonte. Como 4 bits correspondem a 16 valores distintos, o acréscimo em área de mais oito chaves é significativamente pequeno comparado com o ganho proporcionado pela certeza de incremento na frequência.

Aqui vale algumas ressalvas em relação à conversão de valores de seleção de frequência provenientes bloco digital conectado ao gerador local de relógio. Em virtude da alteração para codificação unária diretamente proporcional à entrada, a conversão do valor de seleção de frequência enviado pelo bloco digital deve ser realizado pelo Bloco Controlador. Esse bloco ficará responsável por inverter o valor de seleção de frequência proveniente do bloco digital conectado ao gerador local de relógio e convertê-lo para codificação unária. Normalmente a literatura analógica aborda o uso de chaves conectando e desconectando fontes de corrente à saída, como são vistos na Figura 36 e na Figura 37. Entretanto, durante a conexão da nova fonte de corrente, pode-se conectar momentaneamente o pino de saída ao sinal terra. Como geralmente utilizam-se transistores ligados como diodos para replicar fontes de corrente, a porta deste transistor pode ocasionar tal situação, gerando um pico de corrente neste transistor que replica este estado aos transistores do oscilador em anel com controle de corrente. Durante este breve momento, a frequência de operação poderá passar da frequência máxima tolerada pelo circuito digital, resultando possivelmente em comportamento anômalo ou inesperado.

Para evitar que tais picos afetem o sinal de relógio, o circuito de conversão digital analógico aqui apresentado altera a posição das chaves que conectam as fontes de corrente. Habitualmente ligadas entre o dreno do transistor e a saída do conversor, estas chaves são transferidas então para a porta do transistor utilizado como fonte de corrente, conforme ilustra a topologia da Figura 47. Como agora a chave manipula a tensão de porta, este valor ficará sempre entre o potencial de terra e a tensão enviada pelo espelho da corrente de referência. Assim, garante-se que a corrente de saída desta fonte estará entre a menor corrente possível e a corrente desejada, evitando que o circuito gere picos de frequência. Para reduzir o efeito da queda de tensão provocada pela tensão de limiar das chaves digitais, utilizam-se transistores de tensão de limiar baixa (em inglês, low  $v_T$  transistors ou  $Lv_T$ ) de tamanho mínimo. Pode-se também observar na Figura 47 o transistor M2, utilizado como fonte de corrente mínima, que está constantemente ligado ao espelho de corrente. Como está planejado um circuito de inibição de relógio no bloco oscilador, este transistor visa manter a oscilação mínima. Isto colabora com o incremento de níveis úteis de geração de frequência. O transistor PMOS M18 foi dimensionado para a corrente máxima a ser fornecida ao oscilador em anel. Sendo assim, este transistor possui dimensões idênticas às dos transistores utilizados para controlar a corrente dos inversores do oscilador em anel.

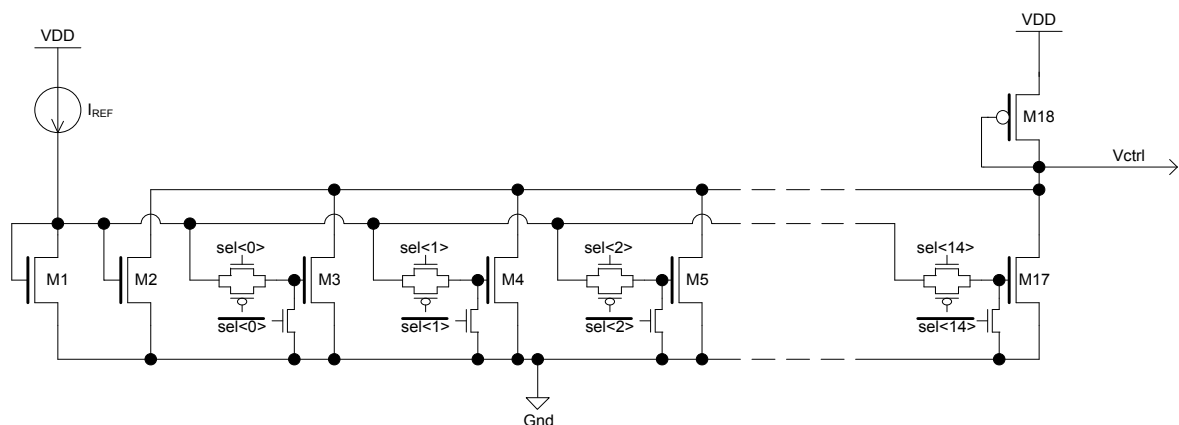


Figura 47. Conversor digital analógico com saída em modo corrente.

Assim, o bloco do Conversor DA, apresentado na Figura 48, apresenta as conexões existentes neste bloco. Vale salientar que os 15 bits de seleção de frequência devem receber valor unário a ser provido pelo Bloco Controlador em razão da seleção de frequência a ele solicitada.

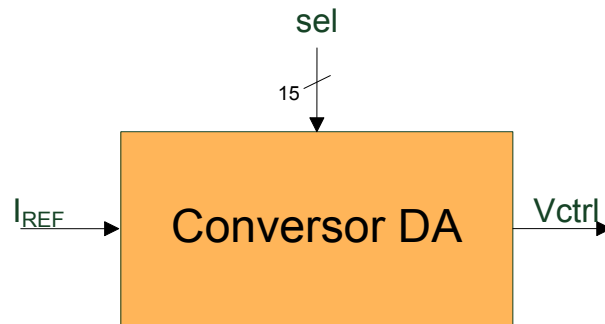


Figura 48. Interface externa do Conversor DA com suas respectivas entradas e saídas.

#### 7.4. Oscilador controlado digitalmente

Ao integrar o conversor digital analógico ao módulo oscilador, cria-se o oscilador controlado digitalmente (DCO). Este módulo permite manipular a frequência de saída a partir de uma entrada digital. Esta Seção discute a integração do conversor apresentado na Seção 7.3 e do oscilador selecionado na Seção 7.2.

Conforme dito anteriormente, a troca de posição das chaves digitais no acionamento das fontes de corrente do conversor selecionado é intencional e visa evitar a geração de picos de frequência acima da selecionada. Desde que a corrente máxima de saída com todas as chaves acionadas não ultrapasse a corrente que gera no oscilador a frequência máxima especificada na Seção 7.1, pode-se produzir qualquer valor de entrada digital, que a saída não passará da frequência máxima projetada. Assim, pode-se dizer que este circuito permite a seleção dinâmica de frequências, por meio de uma entrada digital, sem a necessidade de pausar o relógio durante sua transição entre quaisquer níveis disponibilizados.

Tendo como base a corrente necessária para gerar a frequência de 1 GHz determinada na Seção 7.2, pode-se dimensionar a fonte de corrente de referência e o tamanho do transistor M1. Como o projeto visa área mínima, utilizam-se transistores de tamanho mínimo nas fontes de corrente a serem conectadas à saída (transistores M2 à M17 conforme a Figura 47). Neste ponto, existem três possibilidades de atuar no projeto do conversor: (i) dimensionar a fonte de corrente de referência e manter o tamanho mínimo em M1; (ii) dimensionar o transistor M1 mantendo a corrente de referência; ou (iii) dimensionar a fonte de corrente juntamente com o transistor M1. Para reduzir a dissipação de potência ocasionada por uma fonte de corrente elevada e não incrementar em demasia a área do conversor escolhe-se a terceira opção, dimensiona-se tanto a fonte quanto o transistor M1.

Por fim, apresenta-se na Figura 49 o diagrama de blocos do DCO preliminar. Vale ressaltar que a corrente de referência  $I_{REF}$  é atualmente provida por uma fonte ideal. Está previsto como trabalho futuro, desenvolver uma fonte de corrente de referência para ser integrada ao circuito.

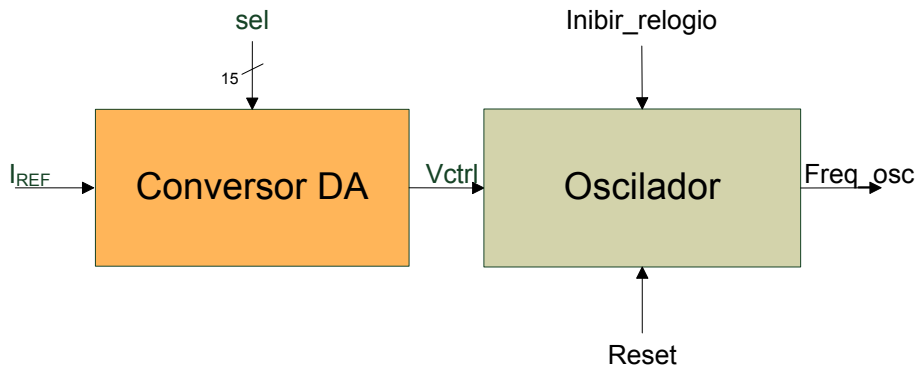


Figura 49. Diagrama de blocos do DCO com Conversor DA e Oscilador, versão preliminar.

## 7.5. Simulações e análises preliminares

Com o auxílio de ferramentas comerciais para simulação de circuitos analógicos, neste caso Spectre da empresa Cadence, simulou-se o DCO proposto na Seção 7.4. Inicialmente, apresentam-se as simulações que visam explorar os diferentes níveis de frequência possíveis de serem gerados. Após, simula-se novamente o DCO alterando-se parâmetros de contorno (em inglês *corner conditions*) para verificar a sensibilidade do circuito com relação à variação de determinados parâmetros de fabricação. Por fim, propõe-se uma alteração para compensar os possíveis desvios observados. O ambiente de simulação é configurado para trabalhar com a corrente de referência nominal, tensão de alimentação do circuito em 1.2V, temperatura de 25°C. Varia-se, no entanto, as condições de contorno para os transistores entre típico-típico (em inglês *typical-typical* ou TT), rápido-rápido (em inglês *fast-fast* ou FF) e lento-lento (em inglês *slow-slow* ou SS) para avaliar a sensibilidade do projeto a variações de dopagem definidas pela tecnologia.

Como é possível observar a partir do exame da Tabela 7 e da Figura 50, os graus de incremento obtidos pelo conversor DA variam de 151 MHz a 38 MHz com incrementos dos valores de entrada no caso Típico-Típico. Contudo, o resultado mais relevante produzido pelas simulações é a variação na frequência máxima provocada pela simples alteração nas condições de contorno dos transistores. Pode-se verificar o quão sensível o gerador de frequências com controle de corrente é a alterações de *corners*. Estas alterações afetam tanto o comportamento dos transistores utilizados como fonte de corrente no conversor DA e no oscilador em anel com controle de corrente quanto nas capacitâncias dos nodos dos elementos de atraso do mesmo oscilador.

Analisou-se a Equação (8) do oscilador em anel com controle de corrente fornecida na Seção 6.1.1, em busca de uma solução para as variações de *corners* de fabricação, de tensão de alimentação e de temperatura. Verifica-se que a única variável disponível para ser controlada é a corrente elétrica incidente no elemento de atraso. Tendo isto em vista, projetou-se um circuito compensador que permite atuar na fonte de corrente de referência de maneira simples para compensar as variações. Discute-se este na Seção 7.6.

Tabela 7 - Relação entre entrada fornecida ao DCO e a frequência de saída.

Entrada unária do DCO ( <i>sel</i> )		Frequência gerada		
Unário	Decimal	Típico-Típico (MHz)	Rápido-Rápido (MHz)	Lento-Lento (MHz)
0000000000000000	0	151,5	234,3	111,7
0000000000000001	1	251,9	372,5	190,5
0000000000000011	2	336,7	486,5	257,8
0000000000000111	3	411,9	586,2	317,6
0000000000001111	4	480,0	675,9	372,1
0000000000111111	5	542,6	757,9	422,2
0000000001111111	6	600,8	833,8	469,0
0000000011111111	7	655,4	904,6	512,9
0000000111111111	8	706,8	971,2	554,6
0000001111111111	9	755,4	1034	593,4
0000011111111111	10	801,6	1093	630,7
0000111111111111	11	845,6	1150	666,3
0001111111111111	12	887,7	1204	700,4
0011111111111111	13	927,9	1255	733,0
0111111111111111	14	966,6	1305	764,5
1111111111111111	15	1004,0	1352	794,7

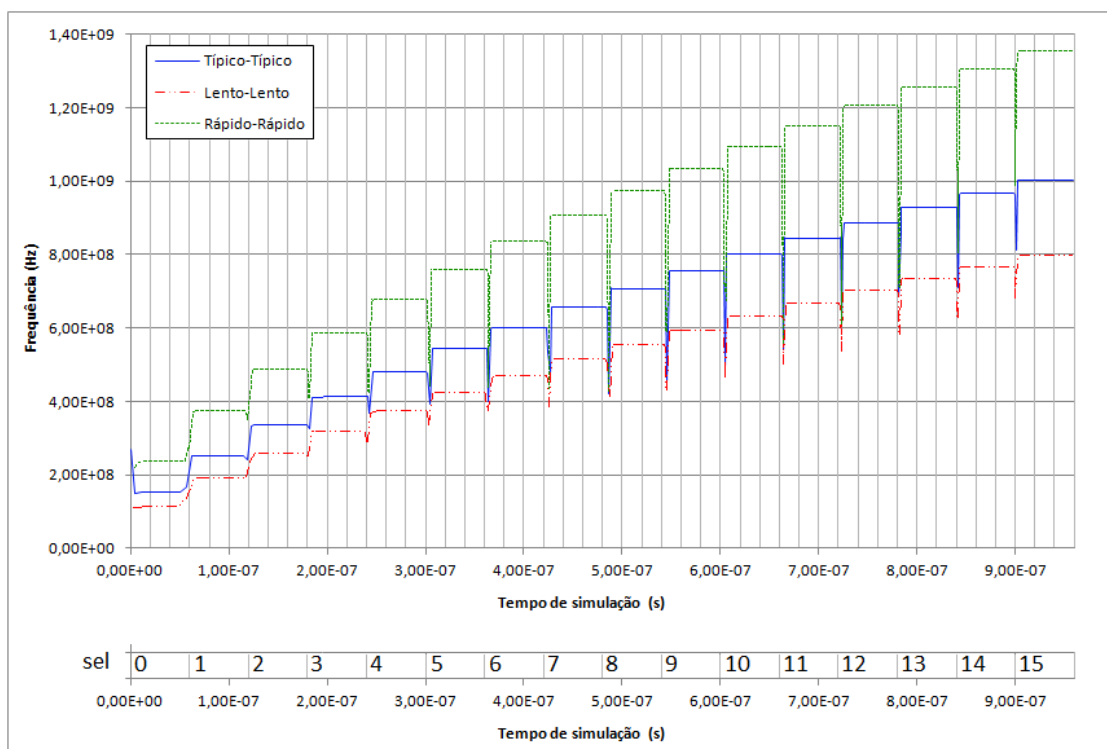


Figura 50. Variação de frequência conforme valores de entrada digital do DCO, considerando caso típico e condições de contorno de variação de parâmetros dos transistores que compõem o DCO.

## 7.6. Compensador de variações PVT

Primeiro é necessário determinar os limites entre os quais o compensador deve atuar. Para realizar este passo, identificaram-se os casos extremos em que se deseja operar o circuito. Segundo a biblioteca STM 65nm, a tensão de alimentação pode variar em torno de +/- 10% do seu valor nominal [STM11]. Adotando a mesma política, usa-se porcentagem idêntica de variação à fonte de corrente de referência. Já com relação à temperatura, determinou-se o uso de classe militar de emprego para este DCO, o que é definido como variações de temperatura entre -55°C e 125°C [ALT12B]. Esta escolha visa gerar um componente robusto. Deve-se ainda levar em conta uma margem extra de segurança, visando atender a variações de descasamento e de dimensionamento dos transistores, identificáveis pela análise de Monte Carlo [JEN91]. Aqui vale uma ressalva interessante. A

biblioteca STM 65nm de baixa potência apresenta um comportamento peculiar quanto aos parâmetros para os *corners* do processo, ou seja, pior caso (em inglês *worst case*) e melhor caso (em inglês *best case*). O pior caso é definido como sendo o que apresenta a menor velocidade possível, ou seja, pior resposta em frequência. O melhor caso é definido como o que apresenta melhor resposta em frequência. Aqui, neste trabalho, com transistores de baixa potência (LP) da ST-65nm, a menor frequência gerada é observada quando a tensão de alimentação é a mais baixa (-10%), quando a fonte de corrente de referência é a menor (-10%), quando o *corner* dos transistores é SS e quando a temperatura é a mais baixa (-55°C). Já a maior frequência é obtida quando a tensão de alimentação é a mais alta (+10%), quando a fonte de corrente é a mais elevada (+10%), quando o *corner* dos transistores está em FF e quando a temperatura é a mais alta (+125°C). No entanto, é possível observar comportamento anômalo ao comumente observado na literatura com relação à temperatura. Segundo [BHA09], este caso já se observa em tecnologias nanométricas e é mais aparente em transistores com tensão de limiar alta ( $Hv_T$ ) ou tensão de limiar padrão ( $Sv_T$ ), este último sendo o caso aqui. Assim, os testes serão realizados seguindo os casos críticos da Tabela 8.

Tabela 8 - Configuração do ambiente de simulação para pior, melhor e caso típico de simulação.

Variável	Pior Caso	Caso Típico	Melhor Caso
Corner	Slow-Slow	Typical-Typical	Fast-Fast
Tensão de alimentação ( $V_{dd}$ )	1,08V	1,2V	1,32V
Fonte de corrente de referência ( $I_{REF}$ )	4,5 $\mu$ A	5 $\mu$ A	5,5 $\mu$ A
Temperatura	-55°C	25°C	125°C

Com estes dois casos, obtêm-se os limites de operação do DCO. O compensador age manipulando a corrente de referência de modo a assegurar que mesmo nestes casos possa-se gerar os 16 níveis distintos de frequência além de garantir a frequência máxima em torno de 1 GHz. Analisando os casos, observa-se que, no pior caso, o problema concentra-se em atingir frequências mais elevadas. Assim, o compensador deverá amplificar a corrente da fonte de corrente para níveis que possibilitem ao oscilador gerar a frequência desejada. Já para o melhor caso, existe dificuldade de atingir as frequências mais baixas. Logo, o compensador deve reduzir a fonte de corrente para valores que permitam gerar-se a frequência de 1 GHz para a seleção de frequência máxima. É necessário também considerar as variações de processo de fabricação e de casamento entre os transistores identificáveis pela simulação de Monte Carlo. Assim, é necessário acrescentar margem de erro ao compensador para torna-lo insensível a tais variações.

Partindo desta base, simulou-se novamente o circuito para determinar o quanto a fonte de corrente necessitaria ser manipulada para cobrir os extremos de operação. Neste ponto, constatou-se uma variação de até uma ordem de grandeza na fonte de corrente necessária. Em simulações das situações de pior e melhor caso com seleção de frequência máxima produziu limites de frequências entre 368 MHz e 2016 MHz. Supondo-se um bom nível de precisão em torno de +/-15 MHz a partir dos valores nominais de frequência esperados (caso de contorno Típico-Típico da Tabela 7), e os dois casos extremos de frequência anteriormente apresentados, obtêm-se a razão em torno de 100 passos. Para compensar os efeitos de variação de processo que incidem no dimensionamento dos transistores (estes podem chegar a quase 50% dependendo da dimensão analisada – neste caso L – [NAS00]), dobra-se esta razão para 200 passos. Assim, determina-se a

necessidade de uma entrada binária de 8 bits para o compensador, o que permite manipular 256 níveis distintos de corrente para o mesmo.

Inicialmente considerou-se usar o conversor W-2W exposto na Seção 6.2.1.2, mas devido a baixa tensão de alimentação, que pode chegar a 1,08V, e ao número de bits selecionados, seria inviável associar 8 transistores em série. Por este motivo, optou-se pelo DAC binário ponderado em modo corrente mostrado na Figura 51.

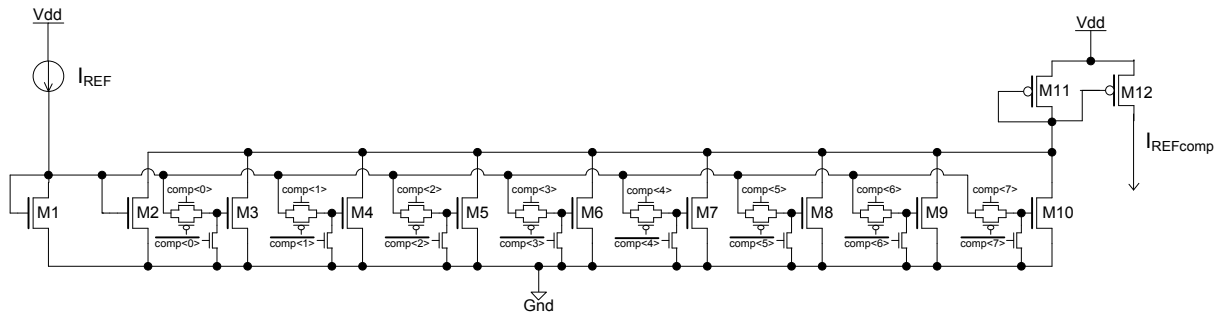


Figura 51. Circuito do compensador de variações de PVT.

Após diversas tentativas de reduzir-se área alterando-se o comprimento ( $L$ ) dos transistores e a associação série/paralelo dos mesmos, decidiu-se apenas alterar a associação paralela de transistores, dobrando a quantidade de transistores ligados em paralelo a cada entrada binária ponderada. Este circuito, por fim, apresentou um aumento de área considerável em relação a versões anteriores, causado pelos seus 256 transistores associados, o que representa praticamente o dobro de espaço ocupado pelo restante do DCO.

Para ser energeticamente econômico, o compensador deve trabalhar com correntes muito baixas e neste ponto observou-se um problema considerável. Ao trabalhar com tais correntes, os transistores entram na região de inversão fraca, fazendo com que a corrente cresça exponencialmente, comandada pela tensão de porta. Ao simular o compensador na situação de melhor caso, onde a tensão de alimentação e a temperatura são maiores, o passo de incremento inicial resulta no aumento exponencial da corrente. Com isso, os primeiros passos de compensação criam passos de até 130 MHz na saída do oscilador.

Para diminuir a influência da inversão fraca, decidiu-se analisar duas possíveis soluções: (i) aumentar o comprimento do canal do transistor ou (ii) utilizar a técnica de empilhamento de transistores [KAO02]. Em transistores de canal curto, os efeitos parasitas da região de depleção entre fonte e corpo e entre dreno e corpo influenciam na carga necessária para inverter o canal do transistor. Para reduzir este efeito, ou aumenta-se a distância entre as regiões de depleção, aumentando  $L$ , ou empilham-se transistores para reduzir a tensão de dreno e, por consequência sua região de depleção [KAO02]. Para o DCO aqui proposto utilizam-se as duas soluções.

No caso de transistores de canal curto, ao aumentar a dimensão  $L$  do transistor, aumenta-se a tensão de limiar do mesmo. Aumentando a tensão de limiar, reduz-se o efeito de corrente de sub-limiar, resolvendo o problema de fuga observado anteriormente. Assim, manipulou-se a dimensão  $L$  por associação de transistores, o que contribui para a linearização da resposta do circuito como um todo. Entretanto, para compensar a redução de corrente provocada por esta manipulação, nos transistores do oscilador em anel são

proporcionalmente aumentadas ambas as dimensões,  $L$  e  $W$ , mantendo a corrente necessária para atingir as especificações.

Por fim, adicionou-se um circuito de pausamento de relógio, que visa permitir que o Bloco Controlador manipule o compensador sem causar comportamento inesperado no módulo digital (roteador ou IP do MPSoC) que dele depende. Isto se deve à possibilidade do compensador momentaneamente selecionar frequências acima de 1 GHz, o que poderia acarretar violações no tempo de configuração e de retenção de dados em seus circuitos de memória. Assim, adiciona-se à saída do bloco oscilador um circuito idêntico ao projetado para inibição de relógio do oscilador em anel. Como explicado anteriormente (ver Seção 7.2.1), este circuito permite que a pausa de relógio seja acionada a qualquer instante de operação. Entretanto, deve-se atentar que, diferentemente do oscilador, o relógio ainda varia em sua entrada. Portanto, deve-se tomar cuidado ao liberar-se este sinal. Sugere-se ao programador do Bloco Controlador executar a seguinte sequência de passos para utilizar o sinal de pausa: (i) acionar o circuito de pausa; (ii) atuar no compensador; (iii) inibir relógio do oscilador; (iv) desativar circuito de pausa; (v) desinibir relógio do oscilador.

Como resultado do desenvolvimento descrito nesta e nas Seções anteriores deste Capítulo, a Figura 52 apresenta o diagrama de blocos do oscilador digitalmente controlado com compensador de variações de processo, tensão e temperatura e circuito de inibição de relógio. Disponibilizam-se dois barramentos de ajuste e seleção de frequência sendo o ajuste fino realizado pela entrada *comp* e o ajuste grosso pela entrada *sel*. Além disto, existem dois circuitos que possibilitam parar o relógio através dos sinais *Inibir\_relogio* e *Pause*. O sinal *Inibir\_relogio* é utilizado para abrir o oscilador e parar completamente o relógio enquanto o *Pause* apenas impede sua propagação ao módulo digital ao qual está conectado. Este último sinal visa permitir a realimentação do Bloco Controlador com o sinal de frequência do oscilador *Freq\_osc* para ajustes nos sinais *comp* e *sel*.

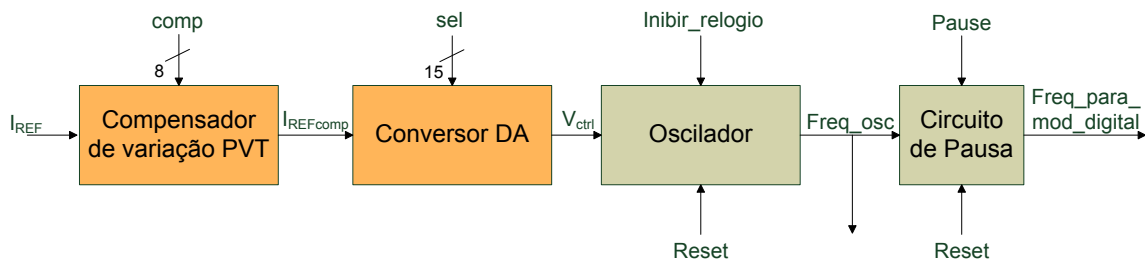


Figura 52. Diagrama de blocos final do DCO.

Ao final do projeto, o DCO apresentado na Figura 52 ocupa uma área total de 828 transistores de tamanho mínimo. Destes, 629 são utilizados no bloco Compensador de variações de PVT, 93 do Conversor DA, 74 do Oscilador e 32 do bloco de Pausa. Vale ressaltar que a fonte de corrente não está contemplada nos dados apresentados.

## 7.7. Operação e características do DCO

Para realizar as simulações do DCO projetado, primeiramente deve-se ajustar a corrente de referência através do bloco compensador de variação de PVT. O Bloco Controlador prevê o uso de um sinal de relógio de referência externo, utilizado como base de



comparação com o relógio gerado pelo oscilador. Na falta deste bloco, pode-se aplicar a seguinte lógica de ajuste:

- O sinal de *Reset* é ativado. Enquanto este sinal estiver ativo, deve-se acionar o circuito de pausa, elevando o sinal de *Pause* para evitar que o relógio, assim que desativado o *Reset*, seja propagado ao módulo digital dele dependente. Inicializa-se tanto o barramento *comp* quanto *sel* para produzir a máxima frequência, ou seja, ambos com todos os bits em '1'. Isso reduz o tempo mínimo de espera para garantir que todos os nodos do inversor estejam com seu valor lógico estável. Ao ter *comp* e *sel* inicializados, após 5ns<sup>1</sup> pode-se desativar o sinal de *Reset*.
- Após desativar o sinal de *Reset*, retorna-se o valor de *comp* para zero, ou seja, "0x00" e incrementa-se sucessivamente seu valor até que a frequência máxima selecionada em *sel*, ou seja, próximo de 1 GHz, seja atingida;
- Depois de determinar o valor de compensação, aciona-se o circuito de inibição de relógio pelo sinal *Inibir\_relogio*;
- Por fim, desativa-se a pausa do circuito baixando o sinal *Pause* e, logo após, desativa-se o sinal *Inibir\_relogio* liberando-o ao módulo digital.

A Figura 53 apresenta as frequências de saída geradas ao variar-se a entrada *comp* do Compensador de variação de PVT.

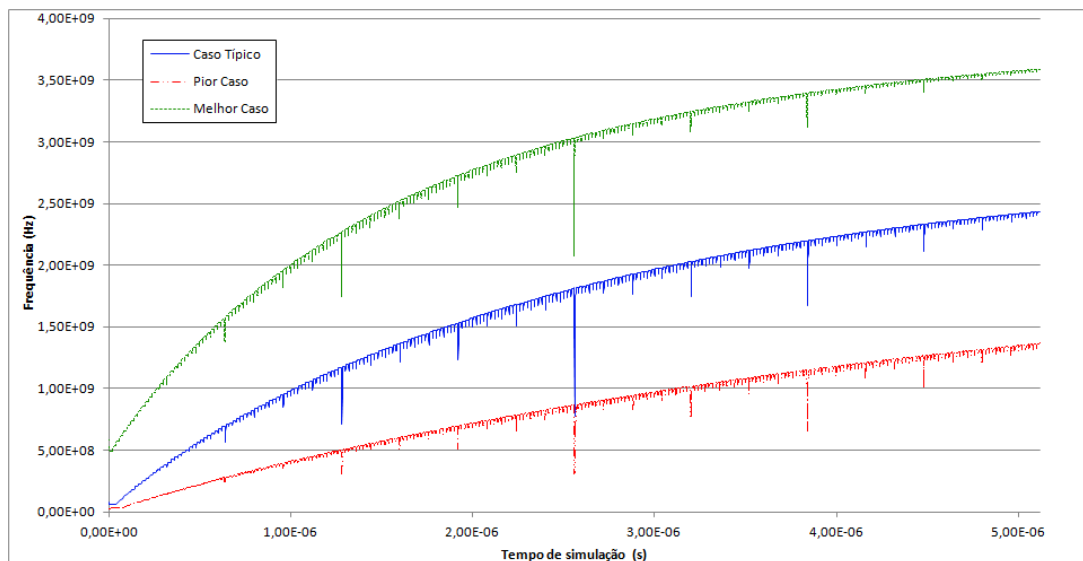


Figura 53. Frequência de saída do DCO conforme incremento no compensador de PVT (*comp*).

Para tanto, esta simulação inicializa o barramento *comp* com "0x00" e incrementa-se este valor digitalmente até "0xFF", mantendo a entrada *sel* com todos os bits em nível lógico '1'. O maior passo de incremento em frequência disponibilizado pelo compensador é de 40 MHz para o melhor caso, ou seja, quando a velocidade dos transistores é maior. Deste modo, pode-se garantir uma precisão de +/- 20 MHz para atingir a frequência de 1 GHz. A Figura 54 apresenta o detalhe da forma de onda da frequência de saída do osci-

<sup>1</sup> Referente ao período da máxima frequência considerando que todos os casos possam-se atingir a frequência de 1 GHz multiplicado pelo número de estágios.

lador variando-se a entrada do compensador *comp* para atingir as situações de: (a) melhor caso, (b) pior caso e (c) caso típico.

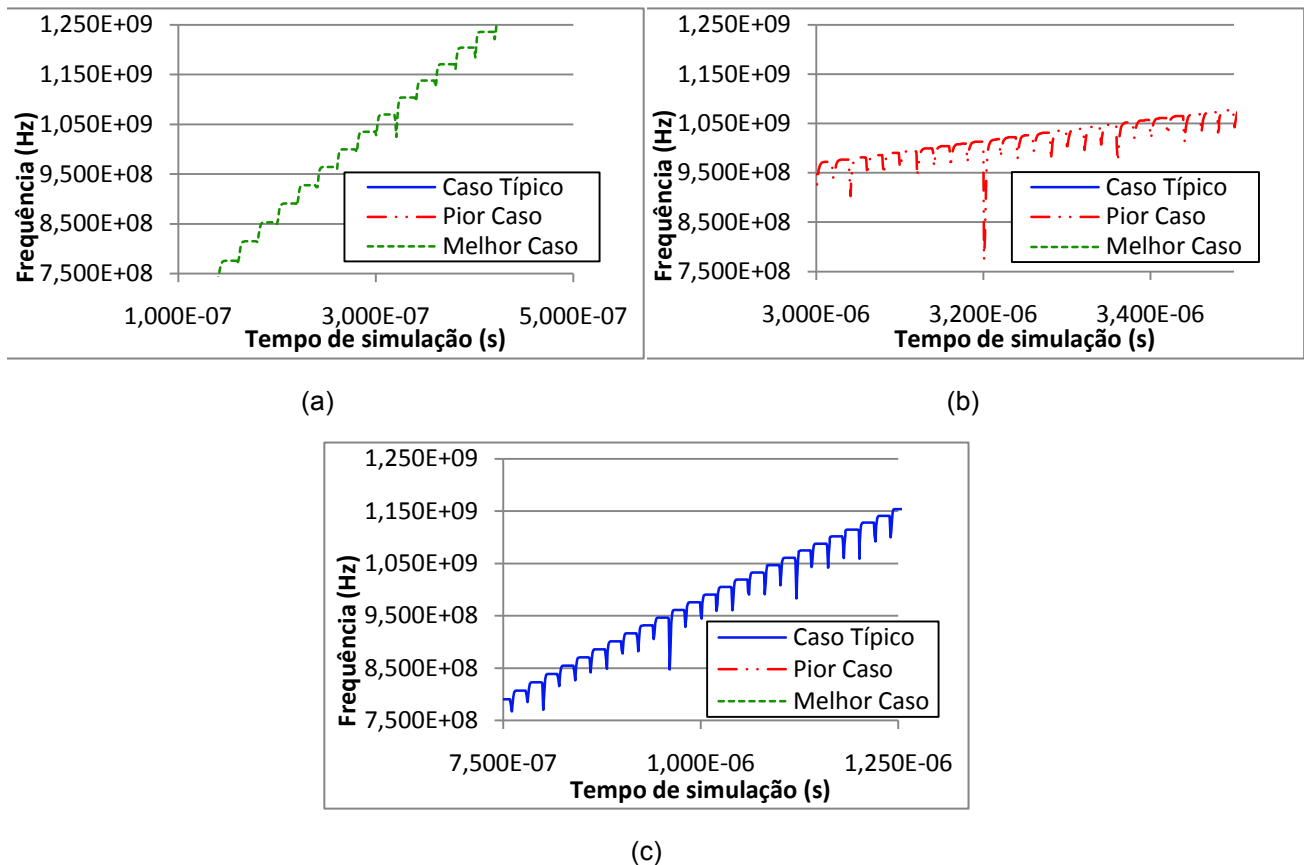


Figura 54. Detalhe das frequências geradas conforme variação do compensador *comp*.

Ao variar-se o passo do compensador, observa-se a geração de espúrios que podem ser maiores dependendo do valor de *comp*. Estes valores transitórios mais acentuados são causados pelo chaveamento das entradas binárias, que acabam por apresentar elevada capacitância nos bits mais significativos. Ao conectar-se estes transistores, necessita-se primeiramente carregar suas capacitâncias de porta, provocando uma queda na corrente gerada durante este período. Isto resulta em uma redução momentânea da frequência gerada.

Ao executar-se o procedimento de ajuste do compensador de PVT, obteve-se para o melhor caso o valor 0Dh, para o pior caso o valor 9Dh e para o caso típico, 33h. Vale ressaltar que como o valor de compensação ultrapassa 7Fh no pior caso, necessita-se de 8 bits para o valor de *comp*.

Após o ajuste no compensador, varia-se a entrada de seleção de frequência *sel*, começando com todo o barramento desligado e ligando-se um a um os pinos do mesmo até todos estarem ligados. A Figura 55, por sua vez, mostra a escala de frequências atingida em cada caso para toda a faixa de valores da entrada *sel*.

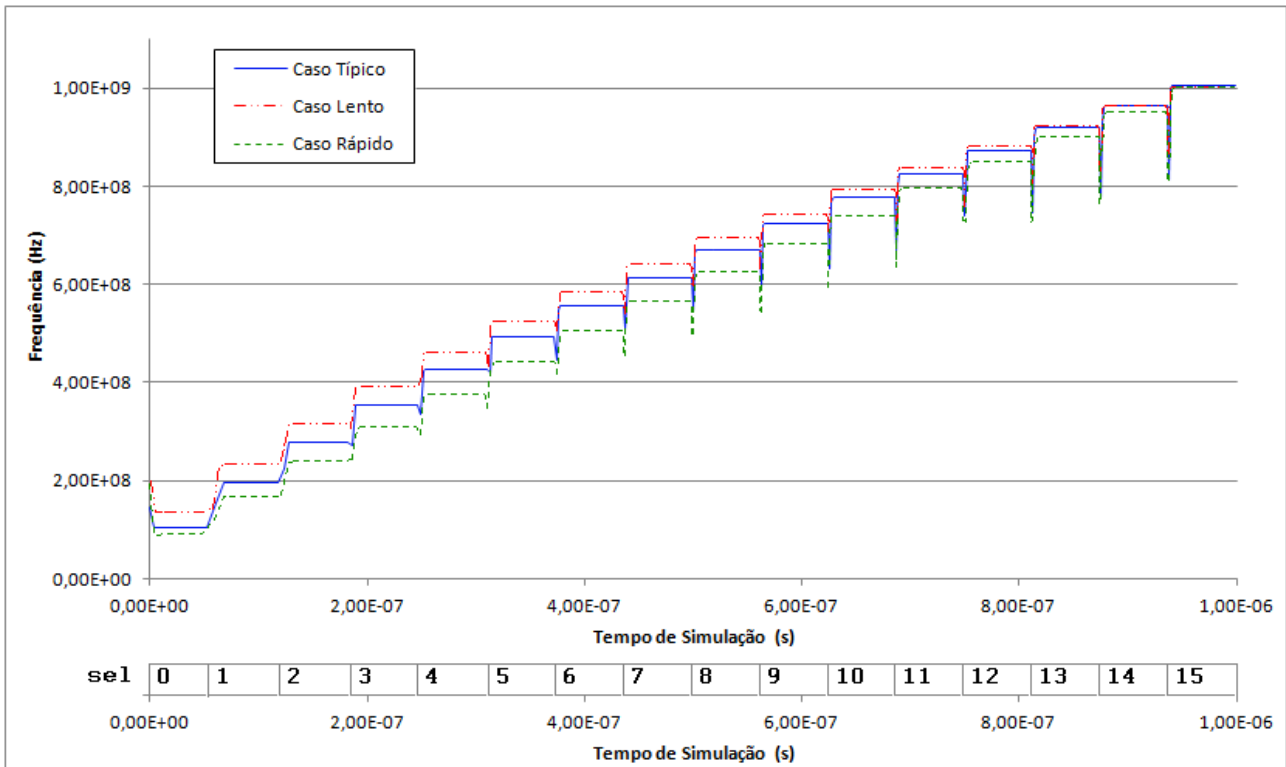


Figura 55. Escala de frequências, conforme entrada fornecida ao DCO (*sel*) com compensação de PVT.

Como resultado, observam-se os 16 níveis de frequência gerados pelo DCO no pior, melhor e no caso típico do circuito, conforme conteúdos da Tabela 9. Também se observa na Figura 55, que a transição do sinal dá-se sem a presença de espúrios, e também as frequências selecionadas.

Conforme exposto na Seção 7.3, o conversor digital analógico projetado permite a troca de frequências a qualquer momento sem a geração de espúrios para a frequência selecionada. Após ajustar a frequência máxima para 1 GHz pelo compensador, é possível aplicar qualquer entrada a *sel* com garantia de não ultrapassar a frequência máxima. Deste modo, é possível utilizar lógicas de conversão binárias para unárias (mesmo se estas produzirem espúrios), pois a frequência máxima não será ultrapassada.

Entretanto, mesmo atingindo a frequência de 1 GHz nos casos extremos, ainda seria interessante saber o quão sensível o circuito é em relação a variações de descasamento e de processo de fabricação. Simulações Monte Carlo visam justamente avaliar a porcentagem de DCOs que não atingiriam a frequência máxima especificada. Para avaliar este dado, determinam-se os extremos para garantir a cobertura de Monte Carlo. Foram analisados dois casos: o melhor e o pior.

No melhor caso, o problema não consiste em atingir frequências altas. Estas facilmente podem ser obtidas por meio do compensador, como se pode observar no gráfico da Figura 53. Assim, para se garantir a cobertura do melhor caso, a maior parte das amostras simuladas deve atingir frequências abaixo de 1 GHz. Portanto, simula-se este caso com o compensador com sua entrada mínima e o sinal de seleção de frequência com sua entrada máxima. Conforme se observa na Figura 56, os resultados da simulação Monte Carlo neste caso mostram que somente uma amostra das 5000 executadas não conseguiu atingir frequência abaixo de 1 GHz, o que aponta uma cobertura de 99,98% neste caso.

Tabela 9 - Relação entre entrada fornecida ao DCO com compensador de PVT e a saída de frequência do oscilador.

Entrada unária ao DCO (sel)		Frequência gerada		
Unário	Decimal	Caso Típico (MHz)	Melhor Caso (MHz)	Pior Caso (MHz)
0000000000000000	0	105,5	90,4	134,6
0000000000000001	1	196,0	166,5	232,1
0000000000000011	2	278,4	238,4	315,8
0000000000000111	3	354,6	307,6	390,8
0000000000001111	4	425,5	374,7	459,5
0000000000011111	5	492,1	439,8	523,3
0000000000111111	6	554,9	503,0	583,0
0000000001111111	7	614,3	564,5	639,2
0000000011111111	8	670,8	624,3	692,4
0000000111111111	9	724,7	682,4	742,8
0000001111111111	10	776,2	739,0	790,8
0000111111111111	11	825,6	794,0	836,5
0001111111111111	12	873,1	847,5	880,1
0011111111111111	13	918,8	899,6	921,8
0111111111111111	14	962,8	950,4	961,7
1111111111111111	15	1005,0	999,8	999,8

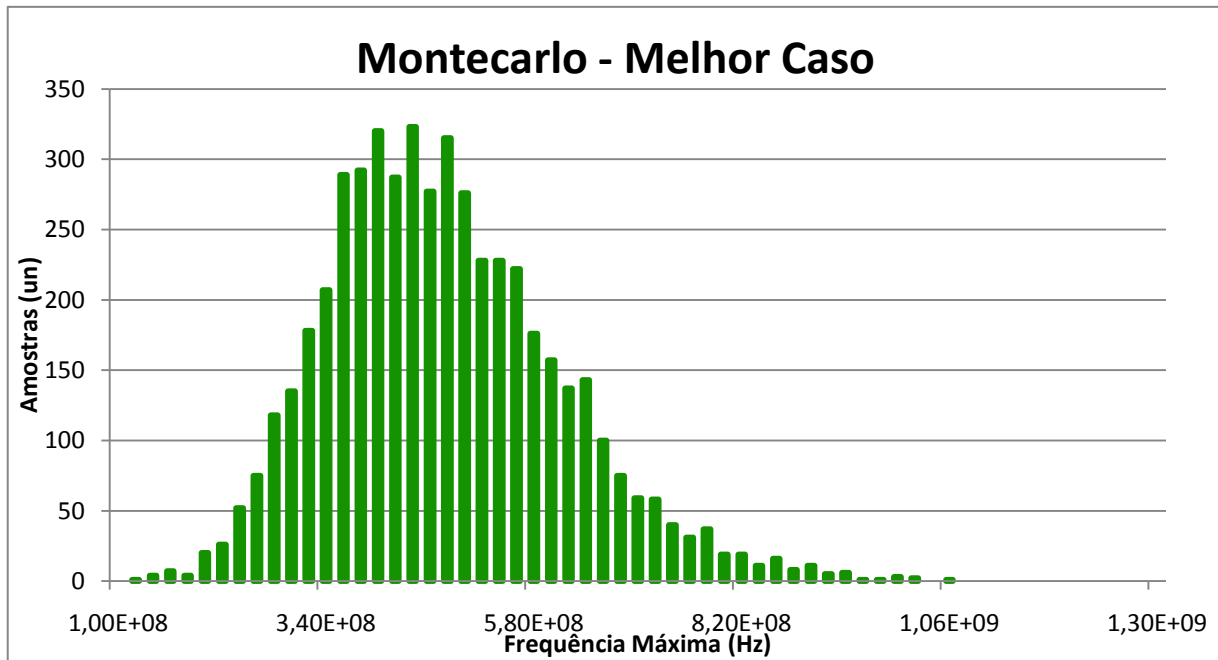


Figura 56. Resultado da simulação Monte Carlo para o melhor caso (5000 amostras).

No pior caso, a questão é atingir altas frequências. Pelo gráfico da Figura 53 pode-se facilmente identificar que atingir frequências baixas não é difícil neste caso, mas garantir que seja possível atingir 1 GHz ou ir além deste ponto é problemático. Com isso, a simulação utiliza o maior fator de compensação possível em sua entrada *comp* e seleciona-se a máxima frequência possível com todos os bits de *sel* em nível lógico '1'. Segundo os resultados da simulação Monte Carlo, apresentados na Figura 57, garante-se que 95% das amostras atingiram frequências acima de 1 GHz.

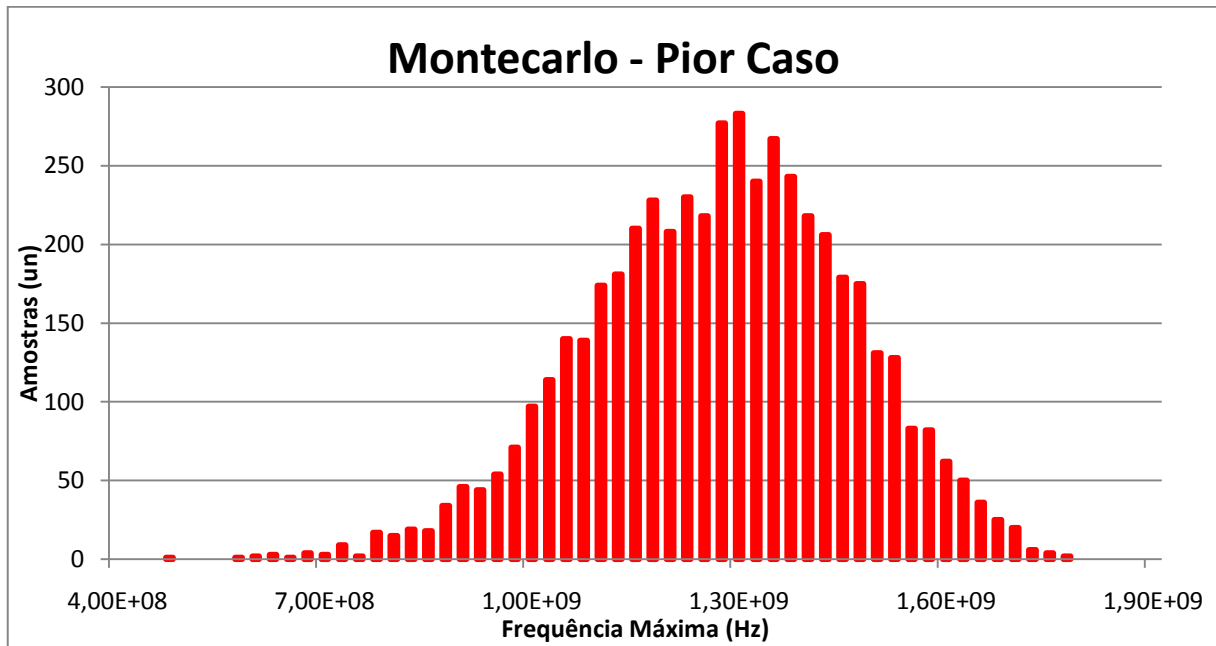


Figura 57. Resultado da simulação Monte Carlo para o pior caso (5000 amostras)

Como a probabilidade destes casos extremos realmente ocorrerem na prática é pequena, pode-se afirmar que é possível produzir satisfatoriamente este DCO para aplicações de grau militar. Garante-se que pelo menos 95% dos circuitos produzidos atenderão às especificações nas condições extremas, gerando a frequência máxima de 1 GHz e mais outros 15 níveis inferiores distintos.

Como a seleção de frequência pela entrada *sel* pode ser realizada a qualquer instante, o oscilador conta com o circuito de inibição de relógio, que pode ser acionado a qualquer instante. A Figura 58 apresenta um acionamento de inibição de relógio no exato instante de conflito de realimentação no nodo N da Figura 45.

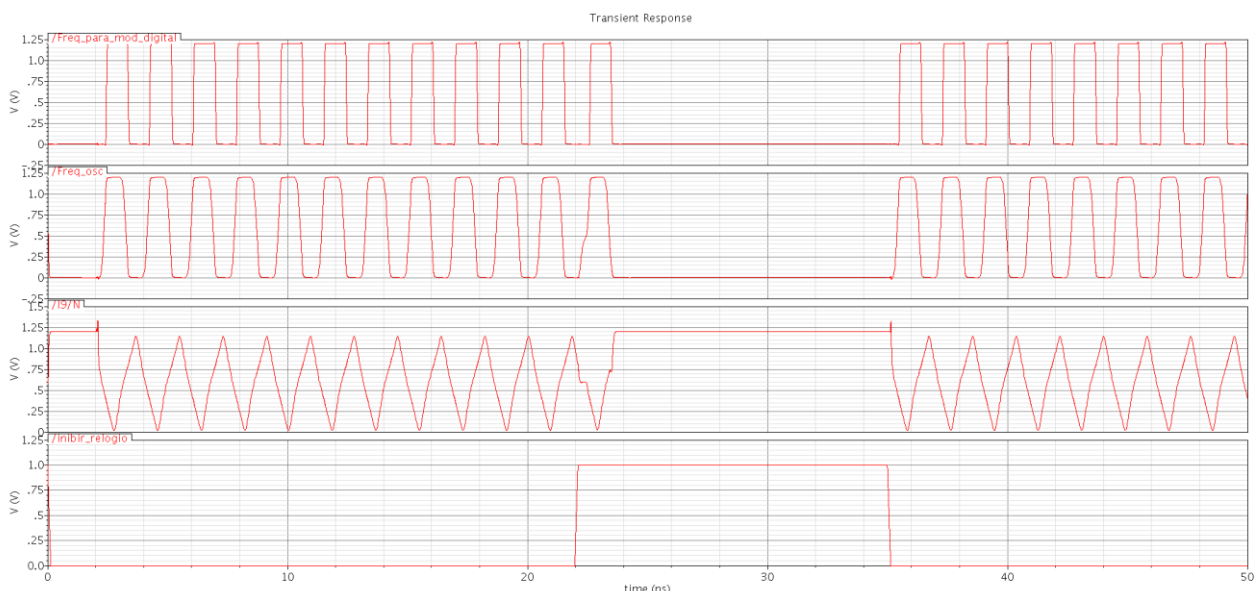


Figura 58. Ação do sinal *Inibir\_relogio* em ponto de conflito no nodo N do DCO.

No entanto, pode-se observar que o sinal de relógio ainda permanece operando até após ele ir para o valor lógico '0'. Neste exato momento, a lógica de controle do circuito inibidor de relógio realimenta a última célula do oscilador e abre o anel, garantindo a integridade de seu nível. A Figura 59 apresenta o instante anterior ao da Figura 58 onde

se pode observar que o relógio é inibido um ciclo antes. O pequeno pico de tensão observado em *Freq\_osc* não será propagado ao circuito digital conectado a este gerador devido à utilização dos inversores presentes no circuito de pausa e de amplificadores instanciados na síntese da árvore de relógio do circuito digital. Assim que se baixa a entrada *Inibir\_relogio*, a memória é desconectada e o anel novamente é ligado, retornando normalmente a seu estado de oscilação.

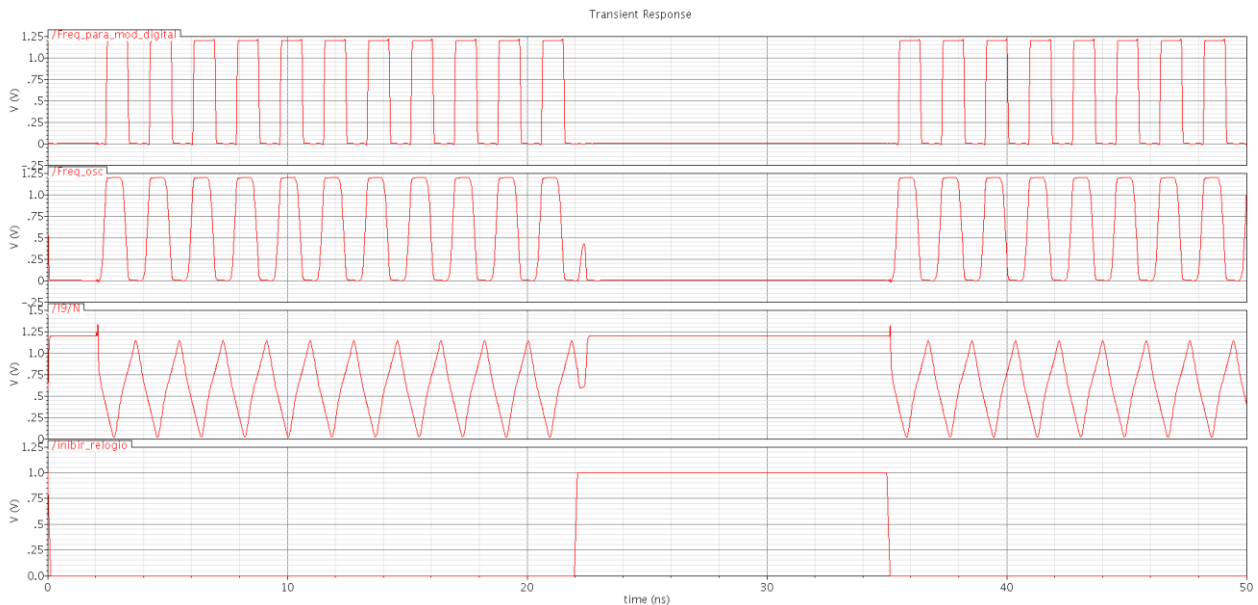


Figura 59. Ação do sinal *Inibir\_relogio* em ponto de conflito no nodo N do DCO 0.1ps antes do visto na Figura 58.

Com relação à dissipação de potência deste DCO, executou-se a simulação de todas as frequências constantes na Tabela 9. A Tabela 10 apresenta a relação entre a entrada *sel* e a potência dissipada em cada frequência. Pode-se relacionar diretamente esta Tabela com a Tabela 9 em relação a frequência e potência.

Tabela 10 - Dissipação de potência do DCO conforme entrada *sel* e caso testado.

Entrada unária ao DCO ( <i>sel</i> )		Potência dissipada		
Unário	Decimal	Caso Típico ( $\mu\text{W}$ )	Melhor Caso ( $\mu\text{W}$ )	Pior Caso ( $\mu\text{W}$ )
0000000000000000	0	27,76	36,83	35,60
0000000000000001	1	31,78	41,71	39,47
0000000000000011	2	35,13	45,81	42,99
0000000000000111	3	38,43	49,50	46,40
0000000000001111	4	41,63	53,73	49,50
0000000000111111	5	44,66	57,69	52,62
0000000011111111	6	47,36	61,37	55,54
0000000111111111	7	50,32	65,01	58,27
0000001111111111	8	52,94	68,68	60,94
0000011111111111	9	55,67	72,38	63,64
0000111111111111	10	58,09	75,85	66,24
0001111111111111	11	60,67	79,42	68,61
0011111111111111	12	63,11	82,74	71,14
0111111111111111	13	65,56	86,02	73,50
1111111111111111	14	67,88	89,34	75,89
1111111111111111	15	70,12	92,62	78,13

Como conclusão, desenvolveu-se aqui um oscilador digitalmente controlado com compensação de variações de PVT. Este oscilador possui capacidade de seleção de 16 diferentes níveis de frequência com inibidor de relógio e circuito de pausa para realimentação do Bloco Controlador. Os resultados apontam uma área estimada de apenas 828

transistores de dimensões mínimas. Verificou-se a operação correta do compensador e as diferentes frequências, dependendo do caso de teste abordado. Também se confirmou a viabilidade de produção deste DCO pelas simulações de Monte Carlo para casos extremos. Apresentou-se ainda o caso de teste para o circuito de inibição de relógio e sua operação correta. Por fim, contabilizou-se a dissipação de potência média em função da frequência selecionada e do caso de teste.

Ao inibir-se o relógio, a dissipação de potência também é reduzida. Assim, ao elevar-se o sinal de *Inibir\_relogio* e inserir-se no barramento *sel* o valor “0”, ou seja, todos os bits em ‘0’, obtém-se o menor consumo do DCO. Para o caso típico, a dissipação de potência média está em torno de 22,3  $\mu\text{W}$ , para o melhor caso, este valor fica próximo de 22,8  $\mu\text{W}$  e 35,4  $\mu\text{W}$  para o pior caso. Este consumo se deve principalmente ao circuito de compensação, que corrige a variação de PVT.





## 8. HEMPS-GLP COM GLR

O MPSoC HeMPS-GLP original descrito no Capítulo 5 pressupõe a existência de diferentes sinais de relógios disponíveis aos roteadores por diferentes árvores de relógios e que são selecionados conforme a necessidade. Entretanto, o projeto deste MPSoC aqui descrito prevê o uso de geradores locais de frequência baseados no DCO projetado e abordado no Capítulo 7. O que se propõe é desenvolver um módulo que aja como gerador local de relógio para possibilitar o controle e a redução de potência da HeMPS-GLP.

Para tanto, inicialmente abordam-se diferentes técnicas de modelamento do gerador local de relógio. Posteriormente, integra-se este modelo aos demais módulos do MPSoC. Após, apresenta-se o ambiente de geração deste MPSoC com geração local de relógio, uma expansão da ferramenta Gerador HeMPS-GLP. Por fim, discute-se uma nova ferramenta de geração do MPSoC baseada no Gerador HeMPS-GLP, que possibilita sua síntese física em ASIC para estimar área e dissipação de potência.

### 8.1. Modelamento do GLR

O diagrama de blocos da Figura 60 apresenta o MPSoC HeMPS-GLP com geradores locais de relógio (GLRs).

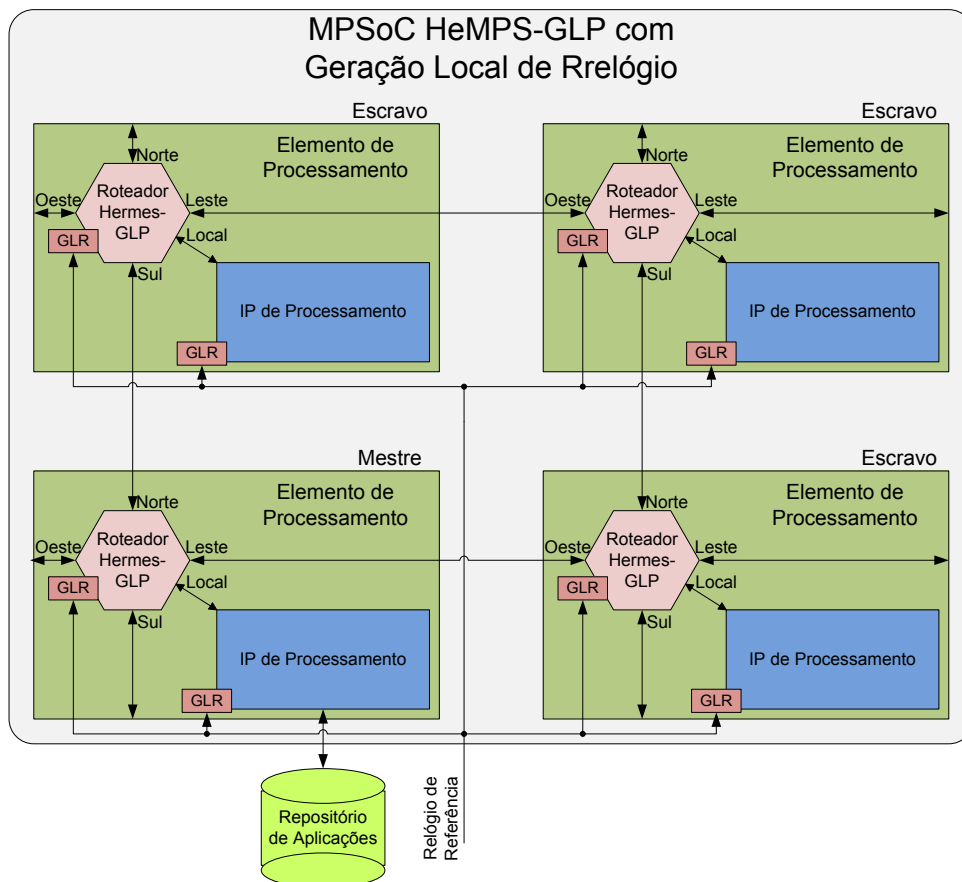


Figura 60. O MPSoC HeMPS-GLP com módulos de geração local de relógio (GLR).

Para garantir a precisão no processo de geração local de frequências, cada GLR necessita do bloco controlador incluído na Figura 33. Para controlar a frequência de operação do GLR, planeja-se utilizar um relógio de referência global e externo ao MPSoC.

Contudo, este relógio apenas será utilizado para comparar com o relógio gerado pelo DCO a fim de ajustá-lo. Logo, este relógio externo não é diretamente utilizado no módulo IP ao qual o GLR estará conectado podendo inclusive ser de baixa frequência e com es-corregamento. Sua utilidade é estabelecer uma referência única e precisa de frequência (a fase é irrelevante) para todos os blocos GLR. O resultado é uma linha global, sim, mas de impacto reduzido sobre o consumo total de energia do sistema, ao contrário de uma rede convencional de distribuição de relógio. A partir deste relógio de referência, o bloco controlador ajusta sua frequência de saída a fim de aproximá-la o máximo possível da curva de frequências gerada no caso típico memorizada internamente. Assim, o gerador local de relógio deve apresentar o mesmo comportamento do DCO projetado no Capítulo 7 já compensado. Por este motivo, parte-se do princípio que se deve modelar o comportamento do DCO com seu processo de compensação, visando emular o GLR.

Para modelar o comportamento e integrar o GLR é necessário atentar para a quantidade de valores que este módulo pode assumir em suas entradas e saídas (tanto valores contínuos como discretos) e com a disposição temporal das alterações de sua entradas e saídas (contínuas ou discretas). Estas propriedades ditarão qual melhor modelo comportamental adotar. Neste caso, tanto as entradas quanto as saídas são digitais. Por serem digitais, podem-se adotar modelos discretos com relação aos valores de entrada e saída. Já com relação ao tempo, deve-se observar a falta de sincronia na geração de relógios. Neste caso em particular, deve-se procurar linguagens de modelamento que possam trabalhar com entradas e saídas binárias e com comportamento contínuo no tempo.

O modelamento comportamental utiliza como base o comportamento da onda física do circuito, aproximando-a por modelos matemáticos simplificados, visando reduzir sua complexidade [JAN04]. O modelo comportamental pode descrever um circuito analógico ou digital, dependendo da abstração que se dá aos valores físicos de tensão ou corrente.

Os sinais que necessitam de resposta contínua nestas grandezas físicas são chamados de sinais analógicos [ACC03]. As linguagens mais usuais para descrição comportamental de sinais contínuos são *Verilog-A*, *Verilog-AMS* e *VHDL-AMS*. Para reproduzir o comportamento físico em seus nodos, estas linguagens utilizam modelos matemáticos que aproximam o comportamento real com o simulado, visando redução no tempo de simulação e algum grau de precisão em seu comportamento. Como vantagem, são necessários passos de simulação reduzidos para a geração da onda contínua no tempo.

Já os sinais digitais possuem seus níveis físicos abstraídos pela discretização em níveis lógicos [CAL98]. As linguagens mais comuns para descrever este tipo de sinal são *Verilog*, *VHDL*, *SystemVerilog* e *SystemC*, que reproduzem o comportamento discreto de sinais. Neste caso, como ambas as entradas e saídas do GLR são digitais, pode-se assumir que este modelamento traria melhores resultados em termos de comportamento e tempo de simulação. Entretanto, estas linguagens não permitem analisar a dissipação de potência do DCO presente no GLR. Por este motivo, o modelo deve gerar um arquivo de saída que registrará as transições em seus sinais para posteriormente estimar sua potência dissipada. Como a linguagem SystemC possui facilidades para manipulação de arquivos, adotou-se esta última para prover a modelagem do módulo GLR.

Em paralelo ao trabalho aqui apresentado, está em andamento o projeto do bloco controlador do GLR utilizando um módulo SystemC para emular o DCO desenvolvido aqui

[HEC11]. Como aqui o que se busca é emular o comportamento do gerador local de relógio já compensado e controlado, pode-se adaptar este módulo para atuar como o gerador local de frequência.

Antes de explicar o funcionamento do modelo GLR em SystemC, devem-se assumir algumas características deste módulo:

- Assume-se que o bloco controlador atua no compensador e garante a estabilidade da frequência gerada, segundo os 16 níveis distintos de relógio do caso típico, que constam na Tabela 9;
- O bloco conversor irá converter a entrada binária em unária, invertendo seu valor em virtude da lógica de seleção de frequência da HeMPS-GLP apresentada na Seção 5.3.

O módulo de geração local de relógio em linguagem SystemC é basicamente constituído por três arquivos: “*ClkTable.cpp*” (detém as frequências de operação do gerador), “*ClkAttributes.cpp*” (calcula os períodos de relógio), e “*LocClkGenerator.cpp*” (executa a manipulação de frequência conforme entradas). Para produzir a frequência solicitada, o gerador utiliza uma lógica de inversão sucessiva de sinal. Consultando a tabela de frequências constantes no arquivo “*ClkTable.cpp*”, calculam-se os períodos relativos às frequências possíveis de serem geradas pelo arquivo “*ClkAttributes.cpp*”. Estes períodos são armazenados para determinar o intervalo em que o sinal permanecerá ativo ou desativo. Antes de começar o laço de geração de relógio, o modelo inicializa o pino de frequência de saída com o valor lógico ‘0’. A partir deste ponto o programa entra em laço repetindo indefinidamente a seguinte lógica:

1. Aguarda até o sinal de inibir relógio possuir nível lógico ‘0’;
2. Lê o valor de seleção de frequência do módulo ao qual está conectado;
3. Atribui ao pino de frequência de saída valor lógico ‘1’;
4. Aguarda metade do período da frequência solicitada;
5. Lê o valor de seleção de frequência do módulo ao qual está conectado;
6. Atribui ao pino de frequência de saída o valor lógico ‘0’;
7. Aguarda metade do período da frequência solicitada.

Seguindo esta lógica, o modelo do GLR descrito em SystemC reproduz em sua saída a frequência solicitada. Como dito anteriormente, ao ser executado, este módulo de geração também produz um arquivo por gerador, indicando as transições nos valores de entrada e seus respectivos instantes na simulação. Com estes valores pode-se estimar a potência dissipada em cada GLR utilizando os dados da Tabela 10. Entretanto, esta estimativa de potência dependeria do caso sendo imposto o circuito (típico, melhor ou pior). Deixa-se esta análise e a respectiva resolução destas questões como indicação de trabalho futuro a partir deste projeto.

## 8.2. Integração do modelo do GLR ao MPSoC HeMPS-GLP

Após o desenvolvimento do modelo SystemC do GLR, algumas alterações na estrutura de arquivos e nas arquiteturas projetadas para o MPSoC HeMPS-GLP necessitam ser modificadas. Boa parte destas alterações diz respeito ao bloco controlador de relógio presente na NoC Hermes-GLP.

Inicialmente, o bloco controlador de relógio controlava e gerava o relógio para o módulo ao qual estava ligado. Entretanto, com o advento do gerador local de relógio, a geração de frequência é fornecida externamente ao roteador e não mais internamente a este. Assim, o novo controlador de relógio é responsável por duas operações nos roteadores: definir qual frequência de operação será utilizada e comandar a inibição de relógio.

Como o módulo GLR é externo, é necessário alterar a interface do roteador para permitir a propagação do sinal de seleção e inibição de relógio. Ao selecionar a frequência desejada, o roteador se comunica com o gerador local de relógio, solicitando uma frequência e recebendo em troca o sinal de relógio selecionado. Deste modo, o comportamento do roteador e da própria codificação de hardware acaba se assemelhando muito ao circuito final, o que permite a integração do gerador local de relógio final de maneira simplificada.

Para os IPs de Processamento a adaptação é simplificada. Assim como no roteador, o GLR é instanciado externamente ao módulo. Entretanto, este módulo não possui por enquanto interface com o GLR e, por este motivo, não é possível trocar sua frequência dinamicamente. Para permitir seu uso nestes módulos, o sinal de seleção de frequência é fornecido pelo arquivo de estímulos *testbench* que fixa seu valor em uma determinada frequência. Internamente, ainda é desativada a função de inibição de relógio deixando estes permanentemente ligados, por enquanto.

## 8.3. Gerador HeMPS-GLP com geração local de relógio

Para comprovar o funcionamento e permitir que outros usuários usufruam do módulo de Geração Local de Relógio, desenvolveu-se um novo Gerador HeMPS-GLP. Este novo ambiente fornece a opção de utilização do módulo de GLR para a NoC e também para os IPs de Processamento. Entretanto, como mencionado na Seção anterior, o sistema de seleção dinâmica de frequência para estes módulos ainda não foi implementado, possibilitando apenas selecionar estaticamente uma das 16 diferentes frequências de operação para cada um.

No campo de Seleção de NoC do novo Gerador HeMPS-GLP, em destaque na Figura 61, existe uma nova opção, denominada “Hermes-GLP LCG” referente ao gerador local de relógio (em inglês *local clock generator* ou LCG). Ao selecioná-la, a NoC Hermes-GLP é ativada com 16 níveis distintos de relógio e com os períodos relativos às frequências que constam no caso típico do DCO da Tabela 9.

Para os IPs de Processamento se disponibiliza a seleção de frequência pelo número de relógio constante na tabela de índice referente à frequência do gerador de relógio local. Para auxiliar na escolha, o usuário pode consultar a lista de períodos de relógio da NoC, pois ambos módulos compartilham o mesmo conjunto de períodos para os GLRs.

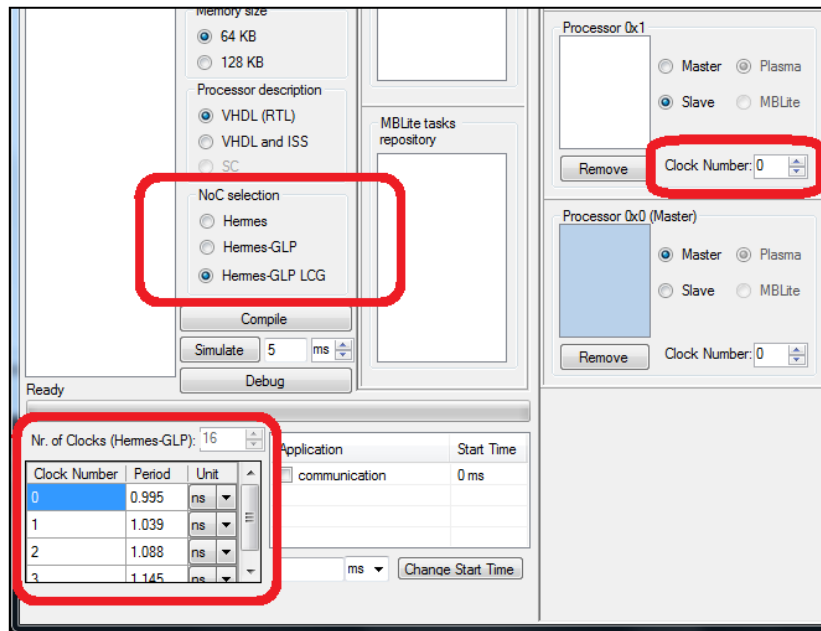


Figura 61. Gerador HeMPS-GLP com gerador local de relógio.

O restante da operação de simulação é idêntico àquela comentada na Seção 5.7, sendo a única diferença a geração de arquivos de registro das variações nas entradas dos geradores locais de relógio na pasta do cenário. O arquivo “*lcg\_router\_XY.txt*” apresenta as variações impostas ao gerador local de relógio do roteador das coordenadas XY. Já o arquivo “*lcg\_ip\_XY.txt*” apresenta os dados de frequência de operação do IP de Processamento da coordenada XY.

#### 8.4. Gerador HeMPS-GLP sintetizável

Com o objetivo de coletar dados referentes à dissipação de potência proporcionada pelo MPSoC HeMPS-GLP com geração local de relógio, desenvolveu-se a ferramenta Gerador HeMPS-GLP Prototipável. Este ambiente, além de permitir a simulação comportamental do MPSoC com o uso de módulos de memória e bancos de registradores STM 65nm também possibilita sua prototipação em ASIC, constituindo-se mais uma contribuição original deste trabalho.

Para a simulação do MPSoC prototipável algumas alterações são necessárias em função dos blocos de registradores e memórias originais da HeMPS. Estes módulos foram originalmente emprestados dos FPGAs da Xilinx através da biblioteca *Unisim*, mas são inadequados para prototipação em silício. Assim, novos módulos que os substituem foram solicitados e posteriormente fornecidos pela STMicroelectronics. Com estes módulos foi enviado o código VHDL funcional e VHDL com atrasos. Neste caso, para aproximar-se mais da realidade, utilizaram-se os modelos VHDL com atraso.

Por meio de simulações, identificou-se que a frequência máxima de operação dos modelos das memória e dos bancos de registradores disponibilizados é em torno de 400 MHz, limitando a velocidade dos IPs de Processamento. Para evitar que se utilizem frequências acima deste valor, foram instanciados redutores de frequência que dividem por dois a onda gerada pelo gerador local de relógio. Entretanto, como o gerador pode gerar até 1 GHz, a interface do Gerador HeMPS-GLP limita a seleção de frequência aos números de relógio com frequências abaixo de 800MHz. Assim o usuário pode selecionar qual-

quer frequência entre o número de relógio 15 e 5 dos geradores locais de relógio. Vale salientar que esta redução é aplicada nos IPs aqui utilizados. Caso o IP permita a operação a 1 GHz, o gerador pode ser diretamente conectado à sua entrada de relógio.

A Figura 62 apresenta a forma de onda comportamental de operação do MPSoC HeMPS-GLP com GLR. É possível observar o chaveamento correto do relógio e a operação correta do circuito de inibição deste obedecendo à mesma lógica utilizada no DCO desenvolvido neste trabalho.

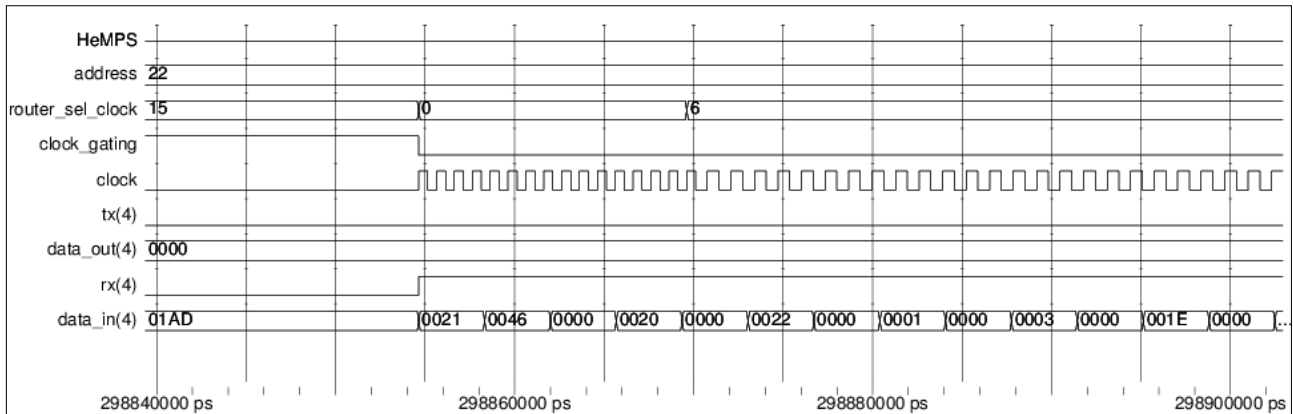


Figura 62. Forma de onda do roteador 22 da HeMPS-GLP com GLR.

Por fim, para extrair dados referentes à área e à dissipação de potência, deve-se obter o circuito descrito em nível de portas lógicas para poder estimar tais valores com maior precisão. Para tanto, o uso de ferramentas comerciais é primordial e, neste caso optou-se por utilizar o *RTL Compiler* da Cadence. Entretanto, como existem muitas peculiaridades neste projeto, necessita-se isolar os domínios de relógio e informar do fornecimento externo de relógio, além de estimar as capacitâncias máximas das árvores de relógio. Para finalizar a definição para geração do circuito do MPSoC, solicita-se ao *RTL Compiler* a utilização de bibliotecas de transistores do caso típico, para manter a compatibilidade com o modelo de GLR desenvolvido.

Após estas definições, executa-se a geração do circuito em nível de portas lógicas. Neste ponto observou-se um fato interessante. Foi identificada uma incompatibilidade na integração do banco de registradores fornecidos pelo fabricante e o processador MIPS-Lite. Originalmente o banco de registradores do processador possuía acesso assíncrono para leitura e síncrono para escrita. Por este motivo sua frequência máxima de operação não ultrapassou 250 MHz após a síntese. Entretanto, pela falta de tempo hábil, decidiu-se prosseguir com a geração do circuito mesmo com frequência de operação tão baixa.

Apresenta-se na Tabela 11 os resultados extraídos do *RTL Compiler* para um IP de processamento e para um roteador do MPSoC. Considerando o roteador Hermes-GLP o menor módulo com frequência produzida por um gerador local de relógio no MPSoC, é possível observar que o número de células ocupadas é de 5488. Como o roteador não possui macro-células da STM 65nm, como memórias ou registradores, as células anteriormente ditas podem ser desde portas inversoras que usam dois transistores, até portas complexas e flip-flops, que podem consumir oito ou mais transistores. Estima-se que cada célula ocupa em média quatro transistores. Assim, estima-se a área do roteador a em torno de 22000 transistores, podendo ser até maior em função de balanços de compensação de mobilidade de cargas entre transistores P e N.

Tabela 11 - Relatório simplificado de módulos do MPSoC HeMPS-GLP em nível de porta.

Dado avaliado	Roteador Hermes-GLP	IP de processamento
Células	5488	7366
Potência ao chaveamento	20,04 mW (3,7 mW por fila)	192,3 mW
Potência de fuga	2,65 $\mu$ W	87,3 $\mu$ W

Com relação à dissipação de potência, comparou-se a dissipação máxima do DCO já compensado apresentado na Tabela 10 com a do roteador da Tabela 11. Pode-se observar que a dissipação média de potência seja em torno de 20 mW por chaveamento sendo 3,7 mW o dissipado por fila do roteador. Comparados com o gasto máximo de aproximadamente 93  $\mu$ W do DCO, pode-se ver que a dissipação de potência deste gerador deva ser bem inferior a do roteador. Entretanto, este é o pior caso de dissipação de potência e como o circuito ainda possui um sistema de inibição dinâmica de relógio, este dado tende a cair em aplicações reais.

Com estes dados pode-se reverificar que o DCO deve ocupar em torno de 3,8% da área e dissipar cerca de 2,5% da potência do roteador. Entretanto, vale lembrar que a fonte de corrente ainda não é contabilizada nestes cálculos e nem o bloco controlador, ambos fora do escopo deste trabalho. Para se contabilizar estes últimos dados, necessita-se projetar a fonte de corrente e o chegar ao leiaute completo tanto do DCO quanto do roteador. Aqui, deixa-se a referência para trabalho futuro nesses pontos visando obter dados mais precisos sobre área e dissipação de potência do DCO.





## 9. CONCLUSÕES E TRABALHOS FUTUROS

### 9.1. Conclusões

Este trabalho apresentou o desenvolvimento de um MPSoC GALS com sistema de DFS aplicável individualmente a cada módulo principal do MPSoC. Também mostrou o projeto de um oscilador controlado digitalmente com compensação de PVT, que será utilizado como base para o GLR do MPSoC em questão.

As principais contribuições deste trabalho são os aprimoramentos da rede Hermes-GLP com suporte à parametrização, o desenvolvimento de um MPSoC HeMPS-GLP objetivando permitir a aplicação de DFS a cada um de seus módulos, o projeto de um oscilador digitalmente controlado com compensação de PVT integrável à HeMPS-GLP, e um ambiente de prototipação voltado ao projeto ASIC do MPSoC aqui desenvolvido.

A rede Hermes-GLP foi refinada e encontra-se completamente funcional além de possuir suporte à parametrização de variáveis como tamanho da fila, quantidade de relógios disponibilizados e definição de padrão de seleção de frequência. Ainda, modificou-se a estrutura de votação e de propagação de seleção de frequência para garantir sua correta operação. Esta rede, além de validada por programas de simulação comercial foi também incorporada ao MPSoC HeMPS, sintetizada em FPGA na plataforma HardNoC e analisada em ASIC em uma tecnologia 65nm.

O novo MPSoC HeMPS-GLP oferece suporte a DFS em uma ampla gama de frequências de operação. Permite-se ao usuário configurar via software a seleção de frequência dos roteadores do caminho de uma mensagem. Disponibilizou-se um ambiente de teste que permite ao usuário configurar as frequências disponibilizadas aos roteadores e a frequência de operação dos IPs de Processamento individualmente. A compatibilidade com versões anteriores foi mantida neste novo ambiente de geração, o que permitindo manipular também projetos síncronos. A ferramenta Gerador HeMPS-GLP foi validada por meio da execução de aplicações de teste em ferramentas comerciais sendo seus resultados obtidos conforme esperado.

O oscilador controlado digitalmente como passo fundamental do desenvolvimento do GLR apresenta 16 níveis distintos de frequência, conforme especificado. Com o uso do compensador de PVT permite-se ajustar a uma precisão de +/- 20% a frequência máxima de 1 GHz em casos extremos de condições de contorno. As simulações de variação de processo e descasamento apresentam resultados animadores, pois garantem pelo menos 95% de circuitos produzidos operacionais. A área ocupada pelo DCO quando comparada à estimativa de área do roteador é animadora, sendo na casa dos 4%. Entretanto, por falta de tempo hábil, deixou-se o leiaute para a próxima etapa do trabalho assim como o desenvolvimento da fonte de corrente de referência. Espera-se que ao término da geração de leiaute, o DCO ocupe menos que os 4% estimados em vista da aproximação pelo número de células no roteador.

Por fim, apresentou-se um ambiente de geração de código sintetizável em ASIC para identificação de potência dissipada e área estimada do MPSoC HeMPS-GLP com geração local de relógio. Para emular o GLR, adaptou-se um módulo SystemC já desen-

volvido para modelar o comportamento do DCO compensado e aplicá-lo ao MPSoC HeMPS-GLP sintetizável na tecnologia STM 65nm. Apesar de resultados preliminares terem sido apresentados, por falta de tempo hábil deixa-se como trabalho futuro a simulação de aplicações reais em nível de portas para extrair com maior precisão a dissipação de potência do MPSoC.

## 9.2. Trabalhos futuros

A continuação desse trabalho prevê primeiramente o desenvolvimento da fonte de corrente de referência, a ser utilizada como base para o oscilador controlado digitalmente. Posteriormente, deve-se produzir o leiaute do DCO, visando extrair dados mais precisos de área ocupada, potência dissipada e de tempos de atraso dos conversores DA. Também coloca-se aqui a possibilidade do uso de espelhos de corrente do tipo *cascode*, que poderiam melhorar consideravelmente a linearidade dos conversores DA. Com isso, a quantidade de bits de *comp* seria reduzida, proporcionando um DCO menor em área. Além disso, é indispensável a inserção de um sistema que comande, via software, o GLR do IP de Processamento para permitir o uso de seleção dinâmica de frequências nestes módulos. Necessita-se também realizar um levantamento quantitativo mais preciso da potência dissipada com simulações em nível de transistores de aplicações reais, tanto no MPSoC HeMPS quanto no MPSoC HeMPS-GLP, para identificar a redução da potência dissipada. Deve-se também extrair as potências médias das transições no DCO e desenvolver-se uma calculadora que converta os arquivos de registro dos geradores locais de relógio modelados em SystemC em potência média dissipada pelo DCO. Posteriormente, deve-se incorporá-la ao ambiente do Gerador HeMPS-GLP, visando estimar a dissipação de potência dos GLRs conforme a aplicação executada.

Também se deixa como trabalho futuro o desenvolvimento do bloco controlador do GLR. Verifica-se, contudo, que o controlador a ser utilizado nos roteadores pode ser simplificado, visando reduzir sua área. Para estes módulos, como não há comprometimento na latência de pacotes (congestionamentos imprevisíveis), a frequência também não necessita de exatidão. Assim, seria suficiente apenas ajustar o compensador para aproximar os níveis como apresentado na Figura 55. Para tanto, passos incrementais consecutivos no barramento *comp* do DCO podem reduzir a complexidade do controlador reduzindo sua área. Para os demais módulos IPs, o ajuste deve ser preciso envolvendo os dois ajustes disponibilizados para atingir o nível mais próximo possível do solicitado pelo IP, objetivando menor dissipação de potência. Com este aumento de complexidade, este controlador possivelmente terá área maior. Entretanto, em relação a área do IP, este aumento não deverá apresentar muita relevância.

Outro ponto importante que deve ser foco de atividade futura é o desenvolvimento de políticas em software para gestão de potência dos IPs de Processamento que podem estar subutilizados por apresentar baixa carga de processamento. A ideia é por software, permitir que o *microkernel* ou outro programa identifique a carga do IP e reduza sua frequência ou até iniba-a se necessário.

O estudo de viabilidade de aplicar técnicas de DVFS também poderia contribuir com a redução de dissipação de potência da HeMPS-GLP. Outra técnica também poderia

ser empregada, analogamente a inibição de frequência, inibindo a tensão de alimentação do circuito digital, quando este não estiver em uso.



## REFERÊNCIAS BIBLIOGRÁFICAS

- [ACC03] Accelera, Inc. “*Verilog-AMS Language Reference Manual*”. Captured in: <http://www.designers-guide.org/verilogams/VlogAMS-2.1-pub.pdf>, Jul 2012.
- [ALM11] Almeida, G.; Busseuil, R.; Carara, E. A.; Hébert, N.; Varyani, S.; Sassatelli, G.; Benoit, P.; Torres, L.; Moraes, F. G. “*Predictive Dynamic Frequency Scaling for Multi-Processor Systems-on-Chip*”. In: IEEE International Symposium on Circuits and Systems (ISCAS’11), 2011, pp. 1500-1503.
- [ALT12A] Altera, Inc. “*8 bits Gray counter*”. Captured in: <http://www.altera.com/support/examples/vhdl/vhd-gray-counter.html>, Jul 2012.
- [ALT12B] Altera, Inc. “*Military Temperature Range Qualified Devices*”. Captured in: <http://www.altera.com/devices/common/military/mil-temp.html>, Jul 2012.
- [BAK10] Baker, R. “*CMOS: Circuit Design, Layout, and Simulation*”. John Wiley & Sons and IEEE Press, IEEE Series on Microelectronic Systems, 3<sup>rd</sup> Edition, New Jersey, 2010, 1174p.
- [BHA09] Bhasker, J.; Chadha, R. “*Static Timing Analysis for Nanometer Designs: A Practical Approach*”. Springer, New York, 2009, 572p.
- [BEI08] Beigné, E.; Clermidy, F.; Miermont, S.; Vivet, P. “*Dynamic Voltage and Frequency Scaling Architecture for Units Integration within a GALS NoC*”, In: International Symposium on Networks-on-Chip (NoCs’08), 2008, pp.129-138.
- [CAL98] Calazans, N. “*Projeto Lógico Automatizado de Sistemas Digitais Sequenciais*”. Imprinta, Rio de Janeiro, 1998, 318p.
- [CAR09] Carara, E.; Oliveira, R.; Calazans, N.; Moraes, F. “*HeMPS - A Framework for Noc-Based MPSoC Generation*”. In: IEEE International Symposium on Circuits and Systems (ISCAS’09), 2009, pp. 1345-1348.
- [HEC12] Heck, G.; Guazzelli, R.; Moraes, F.; Calazans, N.; Soares, R. “*HardNoC: A platform to validate networks on chip through FPGA prototyping*”. In: Southern Programmable Logic Conference (SPL’12), 2012, pp.15-20.
- [HEC11] Heck, L. “*Relógios Locais em MPSoCs GALS e NoCs*”. Plano de Estudo e Pesquisa para Mestrado, Programa de Pós-Graduação em Ciências de Computação, PUCRS, 2011, 28p.
- [HO01] Ho, R.; Mai, K.; Horowitz, M. “*The Future of Wires*”. Proceedings of the IEEE, 89(4), Jan 2001, pp. 490-504.
- [HOP12] Höppner, S.; Eisenreich, H.; Henker, S.; Walter, D.; Ellguth, G.; Schüffny, R. “*A Compact Clock Generator for Heterogeneous GALS MPSoCs in 65-nm CMOS Technology*”. IEEE Transaction on VLSI Systems, Early access article, Captured in <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6166353>, Jul 2012, 5 pages.

- [JAN04] Jantsch, A. *“Modeling Embedded Systems and SoC’s: Concurrency and Time in Models of Computation”*. Morgan Kaufmann Publishers, San Francisco, 2004, 356p.
- [JEN91] Jensen, G.; Lund, B.; Fjeldly, T.; Shur, M. *“Monte Carlo Simulation of semiconductor devices”*. Computer Physics Communication, 67(1), Aug 1991, pp. 1-61.
- [JER05] Jerraya, A.; Wolf, W. *“The what, why and how of MPSoCs”*. In: Multiprocessor Systems-on-Chip, Morgan Kaufmann-Publishers, 2005, pp. 1-18.
- [JOH97] Johns, D.; Martin, K. *“Analog Integrated Circuit Design”*. John Wiley & Sons, New Jersey, 1997, 706 p.
- [KAO02] Kao, J.; Narendra, S.; Chandrakasan, A. *“Subthreshold leakage modeling and reduction techniques”*. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD’02), 2002, pp.141-148.
- [KIM03] Kim, N.; Austin, T.; Baauw, D.; Mudge, T.; Flautner, K.; Hu, J.; Irwin, M.; Kandemir, M.; Narayanan, V. *“Leakage current: Moore’s law meets static power”*. IEEE Computer, 36(12), Dec 2003, pp. 68-75.
- [KLA08] Klapf, C.; Missoni, A.; Pribyl, W.; Holweg, G.; Hofer, G. *“Analyses and design of low power clock generators for RFID TAGs”*. In: 2008 PhD Research in Microelectronics and Electronics (PRIME’08), Jun 2008, pp.181-184.
- [LUD11] Ludovici, D.; Strano, A.; Gaydadjiev, G. N.; Bertozzi, D. *“Mesochronous NoC Technology for Power-Efficient GALS MPSoCs”*. In: Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (OCMC’11), 2011, pp. 27-30.
- [MOR04] Moraes, F.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. *“HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip”*. Integration, the VLSI Journal, 38(1), Oct 2004, pp. 69-93.
- [NAS00] Nassif, S. *“Design for variability in DSM technologies”*. In: International Symposium on Quality Electronic Design (ISQED’00), 2000, pp. 451-454.
- [OGR09] Ogras, U.; Marculescu, R.; Marculescu, D.; Jung, E. *“Design and Management of Voltage-Frequency Island Partitioned Networks-on-Chip”*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 17(3), Mar 2009, pp. 330-341.
- [PON08A] Pontes, J. *“Projeto e Prototipação de Interfaces e Redes Intrachip Não-síncronas em FPGAs”*. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2008, 116p.
- [PON08B] Pontes, J.; Moreira, M.; Soares, R.; Calazans, N. *“Hermes-GLP: A GALS Network on Chip Router with Power Control Techniques”*. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI’08), 2008, pp. 347-352.
- [RAZ98] Razavi, B. *“RF Microelectronics”*. Prentice Hall, Los Angeles, 1998, 335p.

- [RAZ02] Razavi, B. *“Design of Analog CMOS Integrated Circuits”*. Tata McGraw-Hill, 2002, 684 p.
- [ROS12] Rosa, T.; Larréa, V.; Calazans, N.; Moraes, F. *“Power Consumption Reduction in MPSoCs through DFS”*. In: Symposium on Integrated Circuits and Systems (SBCCI’12), 2012. 6 p.
- [SOB08] Sobczyk, A.; Luczyk, A.; Pleskacz, W. *“Controllable Local Clock Signal Generator for Deep Submicron GALS Architectures”*; In: IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS’08), 2008, pp. 14-17.
- [SPA02] Sparsø, J.; Furber, S. (Eds). *“Principles of asynchronous circuit design - A systems perspective”*. Springer, London, 2002, 360p.
- [STM11] STMicroelectronics. *“CMOS065 Design Rule Manual 65 nm Bulk CMOS Process”*. Revision 1.2.3, 2011, 164 p.
- [TOF10] Toffolo, V. *“Um Gerador de Relógio com Escalonamento Dinâmico de Frequência para Sistemas Globalmente Assíncronos Localmente Síncronos”*. Trabalho de Conclusão de Curso, Engenharia de Computação, PUCRS, 2010, 72p.
- [VAN08] Vangal, S.; Howard, J.; Ruhl, G.; Dighe, S.; Wilson, H.; Tschanz, J.; Finan, D.; Singh, A.; Jacob, T.; Jain, S.; Erraguntla, V.; Roberts, C.; Hoskote, Y.; Borkar, N.; Borkar, S. *“An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS”*. IEEE Journal of Solid-State Circuits, 43(1), Jan 2008, pp. 29-41.
- [XIL10] Xilinx, Inc. *“Virtex-5 user guide”*. Hardware Reference Manual, UG190, V5.3, May 2010, 385 p.
- [YAD12] Yadav, M.; Casu, M.; Zamboni, M. *“DVFS Based on Voltage Dithering and Clock Scheduling for GALS Systems”*. In: 14th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC’12), May 2012, pp.118-125.





## APÊNDICE A

### A.1. Codificação de ponteiros Johnson

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity johnson_pointer is
port(
    clock:          in std_logic;
    reset:          in std_logic;
    pointer:        out std_logic_vector (TAM_JOHNSON_POINTER-1 downto 0);
    buff_addr:      out std_logic_vector (TAM_POINTER-1 downto 0)
    );
end johnson_pointer;

architecture johnson_pointer of johnson_pointer is
    signal john      : std_logic_vector (TAM_JOHNSON_POINTER-1 downto 0);
    signal address   : std_logic_vector (TAM_POINTER-1 downto 0);
begin

    process(clock, reset)
    begin
        if reset = '1' then
            john <= (others => '0');
            address <= (others => '0');
        elsif clock'event and clock = '1' then
            john <= john(TAM_JOHNSON_POINTER-2 downto 0) & not john(TAM_JOHNSON_POINTER-1);
            address <= address + 1;
        end if;
    end process;

    pointer <= john;
    buff_addr <= address;
end johnson_pointer;

```

## A.2. Codificação de ponteiros Gray

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity gray_pointer is
port(
    clock:          in  std_logic;
    reset:          in  std_logic;
    pointer:        out std_logic_vector (TAM_POINTER-1 downto 0)
    buff_addr:      out std_logic_vector (TAM_POINTER-1 downto 0)
    );
end gray_pointer;

architecture gray_pointer of gray_pointer is
    signal gray, no_ones_below_gray      : std_logic_vector (TAM_POINTER downto 0);
begin

    process(clock, reset)
    begin
        if reset = '1' then
            gray(0) <= '1'; -- Parity bit
            gray(TAM_POINTER downto 1) <= (others => '0');
        elsif clock'event and clock = '1' then
            gray(0) <= not(gray(0));
            for i in 1 to (TAM_POINTER) loop
                if ((i = (TAM_POINTER) or gray(i-1) = '1') and no_ones_below_gray(i-1) = '0')
                then
                    gray(i) <= not(gray(i));
                end if;
            end loop;
        end if;
    end process;

    no_ones_below_gray(0) <= '0';
    no_ones_below_gray(TAM_POINTER downto 1) <= gray(TAM_POINTER-1 downto 0) or \
    no_ones_below_gray(TAM_POINTER-1 downto 0);

    pointer <= gray(TAM_POINTER downto 1);
    buff_addr <= gray(TAM_POINTER downto 1);
end gray_pointer;

```

### A.3. Codificação de ponteiros *Johnson* com conversor binário

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity johnson_pointer_with_converter is
port(
    clock:          in  std_logic;
    reset:          in  std_logic;
    pointer:        out std_logic_vector (TAM_JOHNSON_POINTER-1 downto 0);
    buff_addr:      out std_logic_vector (TAM_POINTER-1 downto 0)
);
end johnson_pointer_with_converter;

architecture johnson_pointer_with_converter of johnson_pointer_with_converter is
    signal john      : std_logic_vector (TAM_JOHNSON_POINTER-1 downto 0);
    signal address   : std_logic_vector (TAM_POINTER-1 downto 0);
    signal xor_vect  : std_logic_vector (TAM_JOHNSON_POINTER-2 downto 0);

    type or_vect_type is array (0 to TAM_POINTER-2) of \
std_logic_vector((TAM_JOHNSON_POINTER/2)-1 downto 0);
    type ored_vect_type is array (0 to TAM_POINTER-2) of \
std_logic_vector((TAM_JOHNSON_POINTER/2)-2 downto 0);

    signal or_vect      : or_vect_type;
    signal ored_vect    : ored_vect_type;
begin

    process(clock, reset)
    begin
        if reset = '1' then
            john <= (others => '0');
        elsif clock'event and clock = '1' then
            john <= john(TAM_JOHNSON_POINTER-2 downto 0) & not john(TAM_JOHNSON_POINTER-1);
        end if;
    end process;

    XORs_rd : for i in 0 to TAM_JOHNSON_POINTER-2 generate
        xor_vect(i) <= john(i) xor john(i+1);
    end generate XORs_rd;

    ORs : for j in 0 to TAM_POINTER-2 generate
        ORs2 : for k in 0 to (TAM_JOHNSON_POINTER/2)-1 generate
            or_vect(j)(k) <= xor_vect_rd(k + ((k/(2**j))*(2**j)));
        end generate ORs2;

        ORed : for l in 1 to (TAM_JOHNSON_POINTER/2)-2 generate
            ored_vect(j)(l) <= ored_vect_rd(j)(l- \ 1) or or_vect_rd(j)(l+1);
        end generate ORed;

        ored_vect(j)(0) <= or_vect(j)(0) or or_vect(j)(1);
    end generate ORs;

    encoderOut : for m in 0 to TAM_POINTER-2 generate
        address(m) <= ored_vect(m)((TAM_JOHNSON_POINTER/2)-2);
    end generate encoderOut;
    address(TAM_POINTER-1) <= rjohn(0);

    pointer <= john;
    buff_addr <= address;
end johnson_pointer_with_converter;

```