Uma FIFO Básica para Transferência de Dados no Domínio de Clocks não Relacionados

Duarte L. Oliveira[1], Kledermon Garcia[1,2], Lester A. Faria[1], Roberto d'Amore[1]
[1]Divisão de Engenharia Eletrônica do Instituto Tecnológico de Aeronáutica – ITA – IEEA – SJC – SP – Brasil
[2]Divisão de Sistemas Aeronáuticos do Instituto de Aeronáutica e Espaço – IAE-DCTA – SJC – SP – Brasil

Resumo — **Sistemas complexos demandam, cada vez mais, alta desempenho, baixa dissipação de potência e altas taxas de integração entre diferentes módulos. Plataformas militares sejam elas aeronaves, embarcações ou veículos de superfície, mostram-se como um agregado de equipamentos que necessitam se comunicar, nem sempre possuindo protocolos de comunicação triviais entre diferentes módulos de processamento. Neste sentido, First-in-first-out (FIFO) são estruturas de memória amplamente usadas para realizar a transferência de dados entre os módulos de processamento, permitindo uma maior modularidade e aumento de desempenho. Neste artigo propomos três novas FIFOs básicas de modo misto. As FIFOs podem operar com clock misto, ou com a entrada (ou a saída) sendo assíncrona. As novas FIFOs são voltadas para VLSI (*Very Large Scale Integration*), FPGAs (*Field Programmable Gate Array*) e podem ser usadas eficientemente em qualquer topologia dos sistemas digitais heterogêneos do tipo SOC (*Systems-on-Chip*) ou GALS (*Globally-Asynchronous Locally-Synchronous*). Como principais características, tem-se que a sua memória é constituída por um conjunto de registradores que operam como um *pipeline* assíncrono, enquanto as interfaces, que sincronizam o sinal de clock, o fazem de forma eficiente. Os resultados obtidos pelas FIFOs são bastante promissores, seja em área, potência dissipada e/ou taxa de transferência (*throughput*).**

***Palavras-Chave*— sistemas heterogêneos; múltiplos clocks; SOC; GALS**

I. INTRODUÇÃO

Sistemas complexos demandam, cada vez mais, alta performance, baixa dissipação de potência e altas taxas de integração entre diferentes módulos. Neste sentido, plataformas militares, sejam elas aeronaves, embarcações ou veículos de superfície, mostram-se como um agregado de equipamentos que necessitam se comunicar, nem sempre possuindo protocolos de comunicação triviais entre tais módulos de processamento, geralmente se constituindo de circuitos e sistemas digitais.

Sistemas digitais contemporâneos devem necessariamente basear-se no conceito "*System-on-Chip – SoC*". A principal razão para isso é a de satisfazer a crescente demanda por maior desempenho, reutilização e baixa potência [1]. Circuitos SoC são compostos por módulos funcionais, que podem ser IP-*cores* (nucleos de propriedade intelectual), desenvolvido por diferentes fornecedores. Estes IP-*cores* são pré-concebidos, verificados, testados e otimizados para alto desempenho, permitindo também a redução de custos e menor tempo de desenvolvimento.

Duarte L. Oliveira, duarte@ita.br , Tel +55-12-3947-6813, Fax +55-12-3947-6930. Kledermon Garcia, kledermonkg@iae.cta.br; Lester A. Faria, lester@ita.br; Roberto d'Amore, damore@ita.br

Circuitos SoC, quando implementados usando um sinal de relógio (*clock*) global, estão sujeitos às penalidades de velocidade e potência (defasagem da fase do clock - *clock skew*, redes de distribuição, etc.), tornando-se a sua análise de temporização muito difícil [2]. Circuitos SoC, quando implementados num ambiente heterogêneo (ver Fig. 1) (o qual permite módulos síncronos de operarem com diferentes clocks e interagirem com módulos assíncronos) tendem a reduzir sensivelmente os problemas relacionados ao sinal de *clock*, aumentando a modularidade e desempenho, bem como reduzindo a potência. Estes circuitos também podem ser implementados em dispositivos FPGAs (*Field Programmable Gate Array*), porém levando o problema de *clock skew* a ficar ainda drástico, devido ao atraso significativo entre as macro-células. No que diz respeito a FPGAs, estas têm se tornado uma solução bastante popular para a implementação de circuitos digitais, devido ao seu baixo custo e curto tempo de desenvolvimento [3].

First-in-first-out (FIFO) são estruturas de memória amplamente usadas para realizar a transferência de dados entre os módulos de processamento. Neste tipo de estrutura, o primeiro dado que entra é o primeiro a sair. Em sistemas digitais de alto desempenho e de alta complexidade, como SOCs, é cada vez mais necessário transferir dados entre módulos de freqüências de *clock* diferentes, ou até mesmo não relacionadas. Assim, FIFOs são componentes cada vez mais importantes para que o projeto torne-se mais modular (ver Fig. 1), com maior desempenho.

Neste artigo propomos uma nova FIFO básica de modo misto, as quais cobrem a necessidade de um sistema heterogêneo, como mostra a Fig. 1. Para cobrir esta necessidade, três variantes da FIFO básica são propostas. Elas são básicas porque não fornecem as informações de FIFO vazia e FIFO cheia, enquanto são classificadas como de modo misto porque interagem com o ambiente em diferentes formas de entrada e saída (ver Fig. 2). A FIFO 1 opera com o *clock* misto, como mostram as Fig. 1, Fig. 3 e Fig. 4. A FIFO 2 opera na entrada na forma assíncrona e na saída na forma síncrona (ver Fig. 1). A FIFO 3 inverte a operação da FIFO 2 (ver Fig. 1). Somente quatro componentes são usados na construção das novas FIFOs, quais sejam: os registradores; controle assíncrono responsável pela transferência dos dados e que opera no protocolo *handshaking* de duas fases [2]; e as interfaces de entrada e de saída, que são responsáveis em interagir sincronizadamente com o ambiente. As três FIFOS propostas usam uma memória composta de N registradores, operam como um *pipeline* assíncrono e têm um alto *throughput* e baixa área.

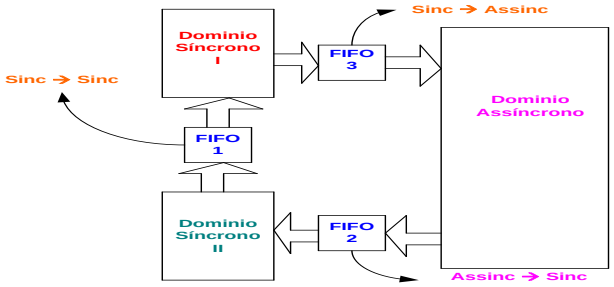

Fig. 1. Sistema digital com domínios de tempos misturados.

Fig. 2. Estrutura de entrada/saída da FIFO proposta.

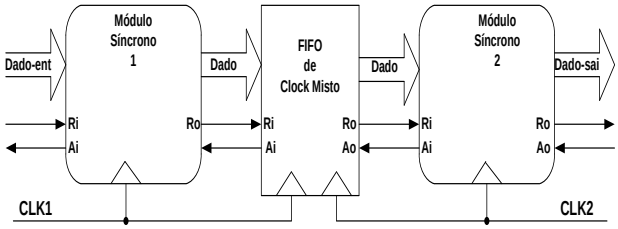

Fig. 3. Sistema digital com *clock* misto e FIFO básica.

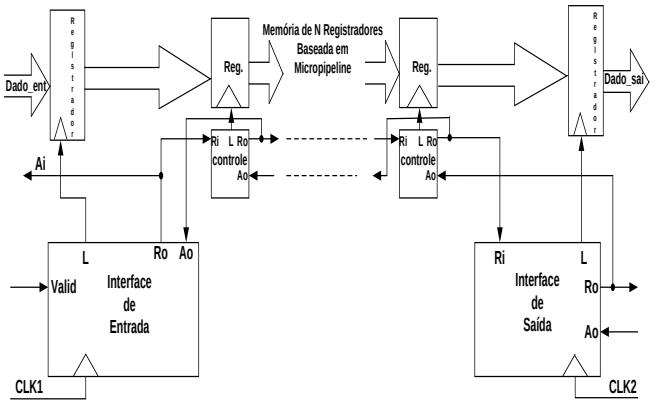

Fig. 4. FIFO básica de *clock* misto: arquitetura proposta.

II. Projeto de Fifos: Visão Global

Uma adequada escolha da estrutura de memória a empregar na FIFO pode ter um impacto significativo sobre a potência, desempenho e custo de um projeto. Diferentes projetos e aplicações de FIFOs podem ser encontrados na literatura. Em [4-6] mostram-se aplicações envolvendo FPGAs e em [7-9] mostra-se uma aplicação em VLSI (very large scale integration). O projeto de uma FIFO envolve diferentes arquiteturas pelas quais o projetista deve optar. Primeiramente, devemos classificar a estrutura de uma FIFO entre duas opções: *pipeline* ou armazenamento direto (AD) que usa uma memoria do tipo RAM (*Random Access Memory*). Se é AD, possui a vantagem de que o dado não se movimenta na FIFO, havendo uma redução do tempo de latência e também na potência dissipada, mas com a desvantagem de o controle ser mais complexo e de haver uma redução no *throughput*. Uma outra opção que se deve tomar é se a FIFO será síncrona ou assíncrona. Se a FIFO for síncrona, ela pode ter *clock* simples ou dual-*clock*, isto é, há um *clock* para entrada dados e um outro *clock* para a saida dos dados. Neste caso, os dois *clocks* podem ser de frequências diferentes ou *clocks* com a mesma frequência nominal, mas de fases diferentes. Há autores que denominam FIFO de dual-*clock* também como FIFO assíncrona,o que nem sempre é utilizado. Além disso, podemos ainda ter a entrada ou a saída de uma FIFO no estilo assíncrono. A Figura 5 mostra as variáveis de entrada e de saída de uma FIFO de clock misto ou dual.

Fig. 5. Estrutura de entrada/saída de uma FIFO.

Diferentes estilos de projeto de sistemas digitais procuram usar FIFOs para reduzir o custo de comunicação entre os módulos, como por exemplo: SOC e GALS (globalmente assíncrono localmente síncrono) [10,11]. Muitas aplicações são implementadas no estilo GALS na topologia ponto-a-ponto (ver Fig. 6), porém esta arquitetura tem como limitação um alto custo de comunicação, uma vez que os dados são transferidos por unidade e não por um pacote de dados.

Uma forma eficiente para transferir pacotes de dados é utilizar uma FIFO assíncrona. Diferentes projetos de FIFOs foram propostos na literatura, focando nas aplicações e usando diferentes técnicas e tecnologias [4-9]. A Figura 7 mostra um sistema GALS com uma FIFO assíncrona básica enquanto a Fig. 8 mostra uma FIFO assíncrona que permite maior flexibilidade para os módulos produtor e consumidor de dados, o que é conseguido através dos sinais *Full* (fila cheia) e *Empty* (fila vazia).

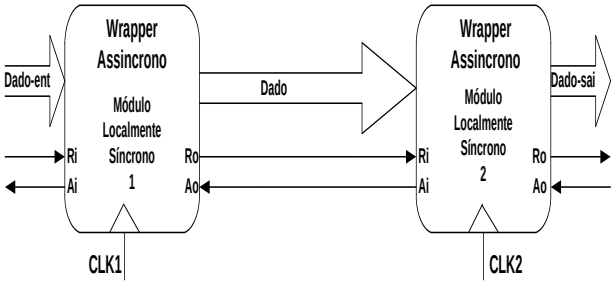

Fig. 6. Sistema GALS ponto-a-ponto convencional.

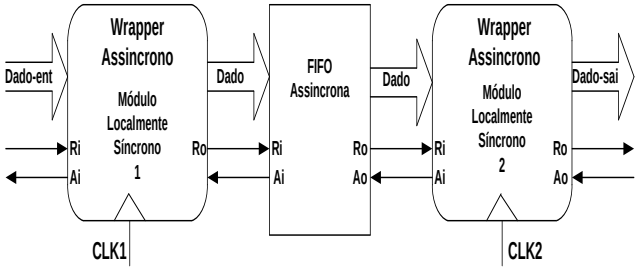

Fig.7. Sistema GALS com FIFO assíncrona básica.

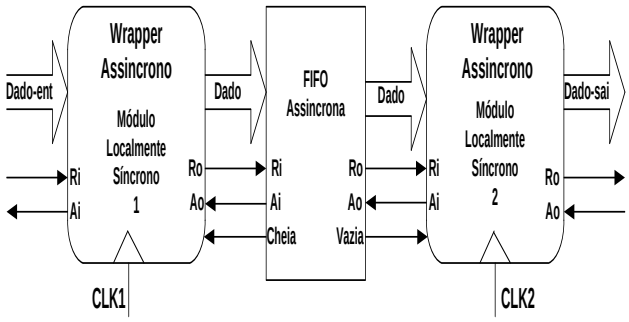

Fig. 8. Sistema GALS com FIFO assíncrona.

III. Projeto de Fifo: FPGAS

Dispositivos programáveis, como FPGAs, são desenvolvidos para projetos síncronos [3]. Os esforços para prototipagem de projetos assíncronos em FPGAs comerciais [12] e FPGAs acadêmicas [13] foram relatados somente recentemente. Os problemas na implementação de sistemas assíncronos em FPGAs comerciais estão relacionados com o seu controle [14], quais sejam:

a) Processo de mapeamento livre de *hazard* lógico das funções Booleanas do controle em blocos lógicos (*macrocells)*. Neste caso, as ferramentas comerciais utilizadas para decomposição e para mapear funções Booleanas em LUTs (*Look-Up Table*) não estão preparadas para atender aos requisitos de *hazard* lógico, o que pode causar mau funcionamento do circuito se a intervenção manual para corrigir o problema não for realizada [14];

b) O processo de roteamento interno entre as macrocélulas pode introduzir atrasos significativos. Tais atrasos podem resultar em *hazard essencial* e levar a um mau funcionamento do circuito [2]. O modelo de atraso do circuito define a forma de como resolver o problema do hazard essencial, por meio da inserção de elementos de atraso nas linhas de feedback ou escolhendo macrocélulas que satisfaçam a condição de *isochronic fork* [2]

O projeto de uma FIFO assíncrona no estilo *pipelin*e simplifica os requisitos do controle, nos quais a parte mais crítica são os contadores assíncronos que têm as funções de contar a entrada e saída de dados, gerando as variáveis de FIFO cheia e vazia. As funções Booleanas desses contadores no mapeamento podem ter problemas de *hazard*.

IV. Fifo Básica de Clock Misto

A Figura 2 mostra os sinais de entrada e de saída da FIFO básica de *clock* misto proposta. Ela é composta por quatro diferentes blocos, quais sejam (ver Fig. 4): a) registradores, que são baseados em flip-flops D; b) controle assíncrono para cada registrador, que é responsável pela transferência dos dados; c) interface de entrada síncrona, que é responsável por armazenar inicialmente os dados na FIFO; e d) interface de saída síncrona, que é responsável pela leitura dos dados da FIFO.

Oliveira et al. [15] propôs uma arquitetura micropipeline linear para implementar sistemas digitais assíncronos. Ela opera no protocolo *handshake* de duas fases e é voltada para FPGAs. Devido à grande quantidade disponível de flip-flops D, os seus registradores são baseados nestes. Esta arquitetura, sem os estágios de processamento, forma uma FIFO assíncrona básica, isto é, sem os sinais de *Full* (cheia) e *Empty* (vazia). O protocolo de duas fases permite aumentar a taxa de *throughput* da FIFO bem como reduzir o tempo de processamento dos *ports* de entrada e de saída do *wrapper* assíncrono [2]. Os *ports* devem trabalhar no protocolo de 2-fases. O controle é composto pelos sinais: *Ri* (*request input*), *Ao* (*acknowledge output*), *Ai* (*acknowledge input*), *L* (*load*) and *Ro* (*output request*). A memória da FIFO básica proposta é baseada no micropipeline de [15] (ver Fig. 9). A Fig. 10 mostra a descrição do controle, que foi feita usando a especificação STG (*signal transition graph*) proposta em [16], enquanto a Fig. 11 mostra o circuito lógico do controle de [15].

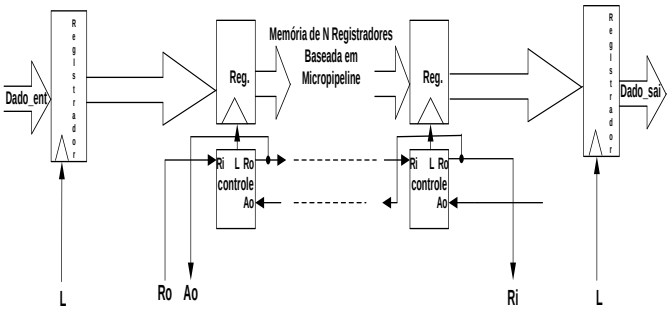

Fig. 9. FIFO assíncrona: estrutura interna.

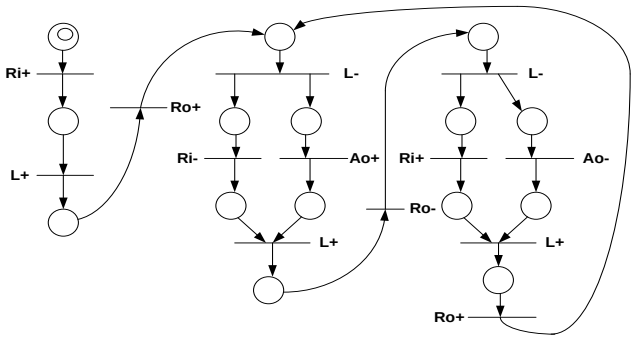

Fig. 10. Grafo de transição de sinal [16]: controle interno da FIFO baseada no pipeline assíncrono de [15].

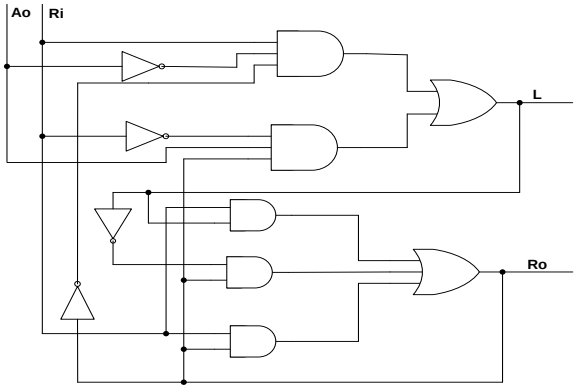

Fig. 11. Circuito lógico: controle interno da FIFO baseada no pipeline assíncrono de [15].

V. Síntese da Interface de Entrada

A interface de entrada da FIFO básica de clock misto foi proposta em [17]. Ela é composta por um controlador assíncrono e por um circuito de sincronização. Esta interface faz a interação de um ambiente heterogéneo, tipo síncrono, com o micropipeline. O controlador assínrono aceita

diretamente o sinal de relógio, estando sincronizado com o módulo síncrono que está fornecendo os dados. O controlador de entrada interage com o micropipeline (memória da FIFO) no protocolo de duas fases. Por conseguinte, o pedido (*Request*) de armazenamento dos dados ocorre em ambas as bordas deste sinal, enquanto o sinal de aceite (*Acknowledge*) necessita ser sincronizado, (sinal A_{CLK}) [4]. Este último é responsável por armazenar os dados no último registrador. A Figura 12 mostra o esquema geral da interface de entrada, a qual consiste em dois flip-flops de sincronização e de um controlador de entrada. O controlador de entrada foi descrito na especificação XBM (*extended burst-mode*) [18] (ver Fig. 13). Ele é composto pelos sinais: *Valid* (dados validos), *Ao* (saída aceita), *CLK* (sinal do clock), *L* (load), *Ro* (pedido de saída) e A_{CLK} (saída aceita sincronizada).

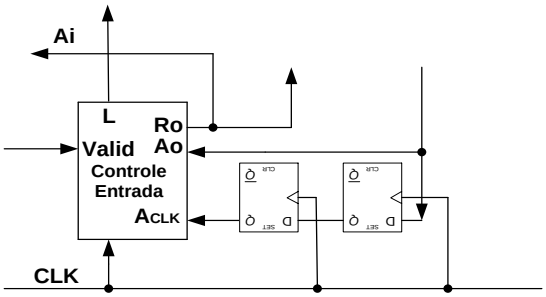

Fig. 12. Interface de entrada: esquema.

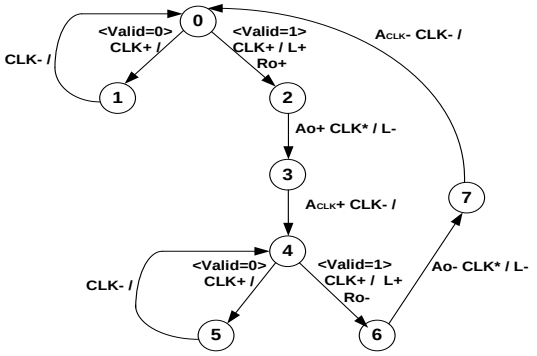

Fig. 13. Controle de entrada: especificação XBM.

VI. Síntese de Interface de Saída

A interface de saída da FIFO básica de clock misto foi proposta em [17]. Ela é composta por um contrador assíncrono e por um circuito de sincronização. Esta interface faz a interação de um ambiente heterogêneo, tipo micropiepline, para síncrono. O controlador assíncrono aceita diretamente o sinal de relógio, por conseguinte, está sincronizado com o módulo síncrono que está recebendo os dados. O controlador interage com o micropipeline (memória da FIFO) no protocolo de duas fases. Ele é responsável por armazenar os dados no primeiro registrador, enquanto o sinal de pedido (Request) necessita de ser sincronizado (Ri_{CLK}) [4]. A Figura 14 mostra o esquema geral da interface de saída, que consiste de um MUX 2x1, e de dois flip-flops para sincronização, bem como o controlador de entrada. O controlador de entrada foi descrito na especificação XBM (*extended burst-mode*) [18] (ver Fig. 15). Ele é composto pelos sinais: *Ri* (pedido de entrada), *Ao* (saída aceita), *CLK* (sinal do clock), *L* (load), *Ro* (pedido de saída) e Ri_{CLK}(pedido de entrada sincronizada).

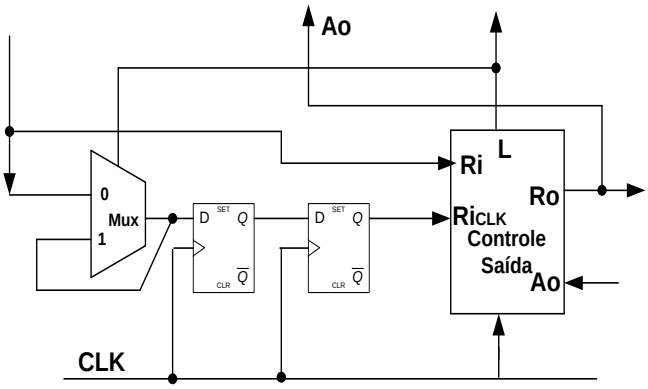

Fig. 14. Interface de saída: esquema.

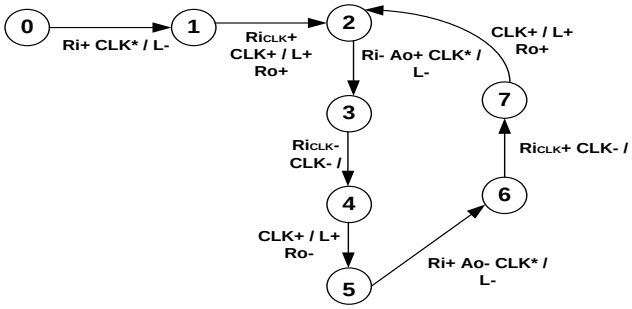

Fig. 15. Controle de saída: especificação XBM.

VII. FIFO Básica de Modo Misto

A FIFO básica de modo misto proposta tem três variantes que permitem cobrir qualquer topologia de um sistema heterogêneo (ver Fig. 1). A Figura 4 mostra a FIFO de clock misto. Ela permite a transferência de dados entre dois domínios síncronos, sendo formada por quatro componentes já discutidos anteriormente (registradores, controle, interface de entrada e interface de saída). A Figura 16 mostra a FIFO básica proposta, a qual permite a transferência de dados entre um domínio assíncrono de entrada e um domínio síncrono de saída. Ela é formada por três componentes (registradores, controle e interface de entrada). A Figura 17 mostra a FIFO básica proposta, a qual permite a transferência de dados entre o domínio síncrono de entrada e um domínio assíncrono de saída, sendo formada por três componentes (registradores, controle e interface de saída). A FIFO básica que permite a transferência entre domínios assíncronos é a de [15] (ver Fig. 9). Ela é formada por dois componentes (registradores e controle).

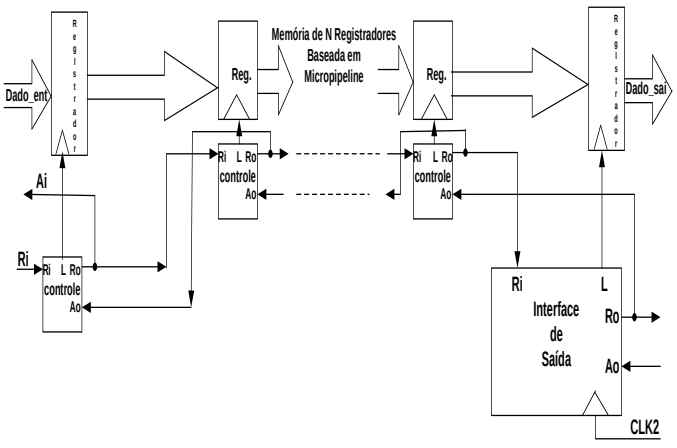

Fig. 16. FIFO de modo misto: entrada assíncrona – saída síncrona.

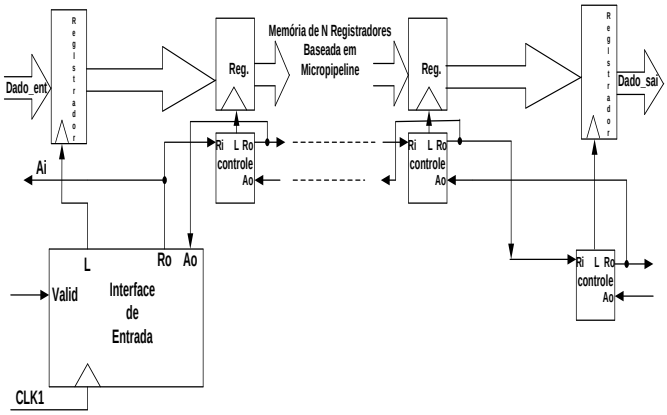

Fig. 17. FIFO de modo misto: entrada síncrona – saída assíncrona.

VIII. SIMULAÇÃO E RESULTADOS

Os controles assíncronos de entrada (ver Fig. 13) e saída (ver Fig. 15) respectivamente das interfaces de entrada e de saida da FIFO foram sintetizados pela ferramenta 3D [18]. As FIFOs de modo misto propostas foram sintetizadas e simuladas com uma memória de 8x8 bits, no software Quartus II, versão 9.1 [19], considerando o dispositivo alvo Altera Stratix II (EP2S15F484C3). As FIFOS assíncronas de [15,20] foram também simuladas. A FIFO assíncrona finaliza todos os modos necessários e suficientes para comunicação entre módulos de um sistema heterogêneo. As Figuras 18, 19, 20, 21 e 22 mostram as simulações das FIFOs propostas e das FIFOs assíncronas de [15] e [20]. As simulações mostram as formas de onda, conforme o esperado, e que satisfazem às especificações dos controles, mostrando ser livres de *hazard*. As interfaces de entrada e de saída permitem interagir com os módulos síncronos neste dispositivo da ALTERA em frequências de até F_{MAX}=500MHZ.

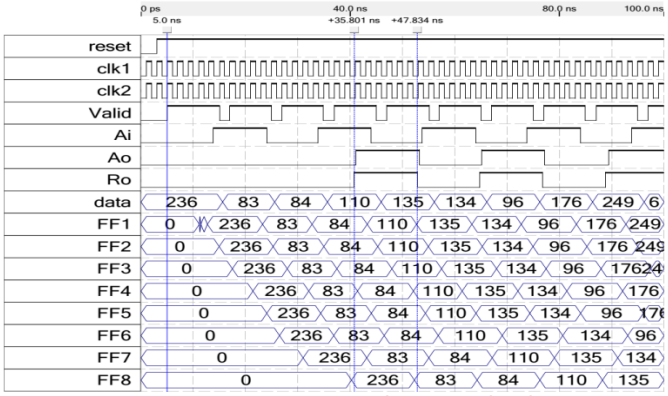

Fig. 18. Simulação: FIFO de entrada síncrona – saída síncrona.

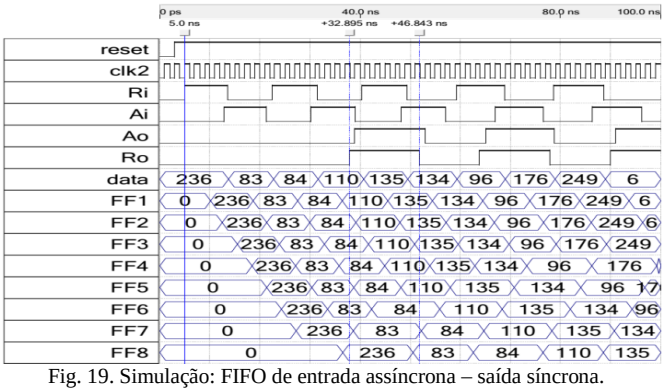

Fig. 19. Simulação: FIFO de entrada assíncrona – saída síncrona.

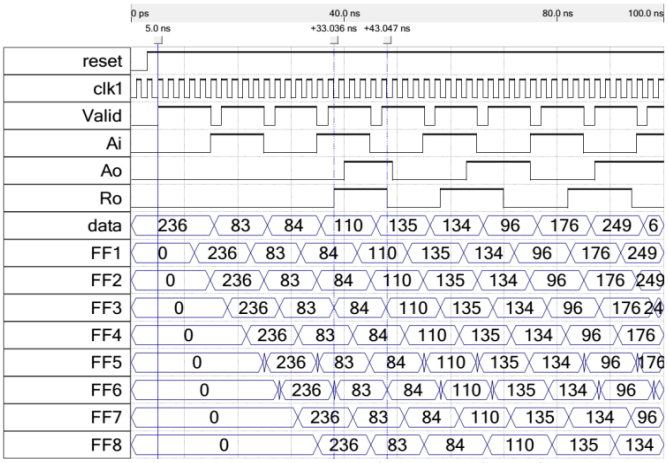

Fig. 20. Simulação: FIFO de entrada síncrona – saída assíncrona.

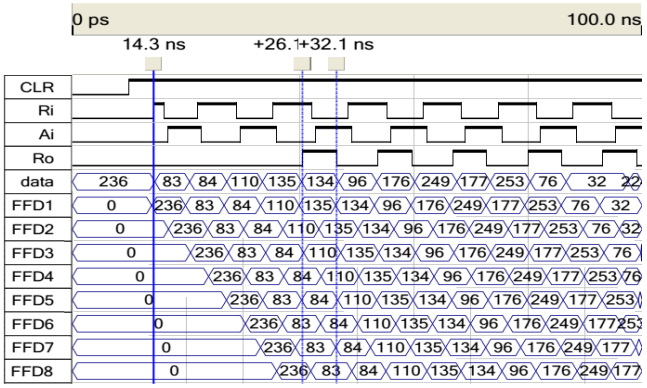

Fig. 21. Simulação: FIFO assíncrona de [15].

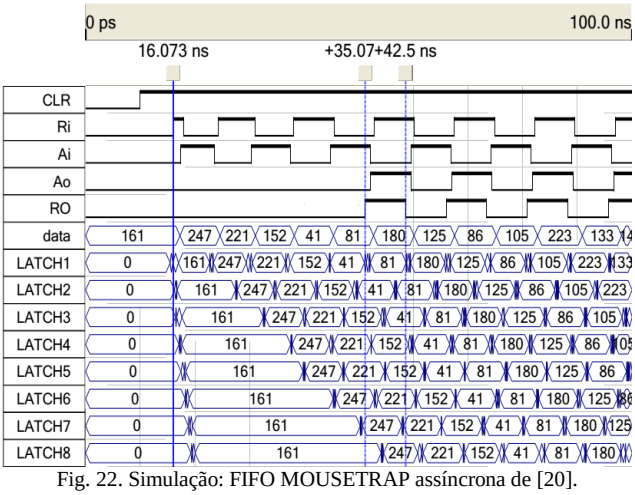

Fig. 22. Simulação: FIFO MOUSETRAP assíncrona de [20].

A Tabela I apresenta os resultados das FIFOs básicas de modo misto propostas, bem como as FIFOS assíncronas MOUSETRAP [20] e a proposta em [15]. Os trabalhos de [15] e [20] propõe arquiteturas para implementar sistemas pipeline assincrono de alto desempenho, que podem ser usadas como FIFO básica assíncrona. As informações apresentadas na tabela I são: latência, taxa de transferência em MOPS (10^6 operações por segundo), área (número de LUTs e número de Flip-Flops) e dissipação de energia. Para avaliar a arquitetura proposta, usamos as FIFOs assincronas MOUSETRAP [20] e de [15]. Fazendo uma comparação das FIFOs assíncronas com as FIFOs de modo misto propostas, houve penalidades como era esperado, porque elas não precisam das interfaces de entrada e/ou de saida. No tempo de latência foi de até 37%, enquanto na taxa de transferência

(*throughput)* foi de até 115%, na potência dissipada de até 30% e na área (LUTs + FFs) de até 9%. Apesar de tais penalidades, considerando a complexidade dos projetos nos quais as FIFOs de modo misto tiveram as interfaces entrada e/ou saída introduzidas para permitir a sincronização entre os módulos síncronos, pode-se dizer que a penalidade foi considerada baixa.

TABELA I RESULTADOS: FIFOS BÁSICAS

FIFO Simplificada Pipeline	Tempo de Latência	Throughput MOPS	Potência Dissipada	Macrocélulas	
				Número Flip-Flops	Número LUTs
MOUSETRAP	35,07ns	134	**347,55mw**	**0**	171
Baseado em [15]	**26,1ns**	**166**	508,72mw	64	**58**
Figura 4	35,801ns	83	494,41mw	68	66
Figura 16	32,895ns	77	488,50mw	66	59
Figura 17	33,036	99	483,96mw	68	63

IX. CONCLUSÃO

Neste trabalho, propomos três novas arquiteturas para FIFOs básicas de modo misto. As três FIFOs usam memória no estilo pipeline assíncrono e os seus registradores são baseados em flip-flops do tipo D. Estas arquiteturas estão voltadas para implementações em VLSI e FPGAs e podem ser usadas eficientemente em sistemas GALS com topologia ponto-a-ponto, ou SOC. Elas cobrem eficientemente as necessidades de um sistema heterogêneo, como ocorrem em sistemas complexos tais quais os existentes em vetores militares mais modernos. As FIFOs propostas tem como características a interação entre ambientes diferentes e não têm as informações de FIFO cheia e vazia. Os três controladores assíncronos usados no projeto das FIFOs são livre de hazard. Para aumentar a robustez e permitir uma fácil implementação em dispositivos como FPGAs, os controladores assíncronos das interfaces de entrada e de saída podem ser implementados na arquitetura de clock local [21,22]. Apesar das penalidades, como era de se esperar, as FIFOs propostas, quando comparadas às FIFOs assíncronas, se mostraram eficientes em área e potência dissipada, com bons desempenhos no caso latência e *throughput*. Tais resultados obtidos mostram-se potencialmente interessantes para aplicações práticas, facilitando e otimizando a integração de sistemas complexos. Para trabalhos futuros, pretendem-se desenvolver arquiteturas para as três FIFOs propostas que contenham as variáveis de FIFO cheia e vazia e comparar com outras FIFOs de modo misto. Estas informações permitem mais flexibilidade na interação da FIFO com o ambiente externo.

REFERÊNCIAS

[1] G. DE Micheli, "An Outlook on Design Technologies for Future Integrated Systems," IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 28, no.6, pp. 777-789, June 2009.

[2] C. J., Myers, "*Asynchronous Circuit Design*", Wiley & Sons, Inc., 2004, 2ª edition.

[3] J. J. Rodriguez, et. Al., "Features, Design Tools, and Applications Domains of FPGAs", *IEEE Trans. on Industrial Electronics*, vol. 54, No. 4, pp.1810-1823, August 2007.

[4] V. Vijaya, et al., "FPGA Implementation of RS232 to Universal serial bus converter," IEEE Symposium on Computers & Informatics, pp.237-242, 2011.

[5] S. Yu, et al., "Implementation of a Multi-channel UART Controller Based on FIFO Technique and FPGA," Second IEEE Conference on Industrial Electronics and Applications, pp.2633-2638, 2007..

[6] F. Crowe, et al., "Single-Chip FPGA Implementation of a Cryptographic Co-Processor," Proc. ICFPT, pp.279-285, 2004.

[7] R. W. Apperson, "*A Dual-clock FIFO for the Reliable Transfer of High-Throughput Data Between Unrelated clock Domains*," Master of Science, University of California – Davis, 2002.

[8] P. Huang, W. Hwang, "2-Level FIFO Architecture Design for Switch Fabrics in Network-on-Chip," IEEE ISCAS, pp.4863-4866, 2006.

[9] T. Chelcea, S. M. Nowick, "Robust Interfaces for Mixed-Timing Systems," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 8, pp.857-873, August 2004.

[10] D. M. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems*, PhD thesis, Stanford University, October 1984.

[11] M. Rubini, and C. Rajasekaran, "Design of high performance system-on-chips using Field Programmable Gate Arrays (FPGA)," 2014 International Conference on Communications and Signal Processing (ICCSP), pp.358-362, 2014.

[12] M. Tranchero and L. M. Ryneri, "Implementation of Self-Timed Circuits onto FPGAs Using Commercial Tools", 11th Euromicro Conf. on Digital System Design Architectures, Methods and Tools, pp.373-380, 2008.

[13] N. Huot, et al., "FPGA architecture for multi-style asynchronous logic," Proc. Of the Design, Automation and Test in Europe Conference and Exhibition, 2005.

[14] P. S. K. Siegel, "*Automatic Technology Mapping Asynchronous Designs*," PhD thesis, Stanford University, February, 1995.

[15] D. L. Oliveira, et al., "Using FPGAs to Implement Asynchronous Pipeline," 5th IEEE Latin American Symposium on Circuits and Systems, Santiago, Chile, 2014.

[16] T. –A. Chu, "*Synthesis of Self-Timed VLSI Circuits from Graph-Theory Specifications*," PhD. Thesis, June, 1987, Dep. Of EECS, MIT.

[17] D. L. Oliveira, et al., "Low-Power Interface for Interacting Asynchronous Pipeline with Synchronous on FPGAs, " submetido ao 25th International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS, 2015.

[18] K. Y. Yun e D. L. Dill, "Automatic Synthesis of Extended Burst-Mode Circuits: Part I (Specification and Hazard-.Free Implementation) and Part II (Automatic Synthesis)," *IEEE Trans. on CAD of Integrated Circuit and Systems*, Vol. 18:2, pp. 101-132, Feb. 1999.

[19] Altera Corporation, 2015, www.altera.com.

[20] M. Singh and S. M. Nowick, "MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines", *IEEE Trans. on VLSI Systems*, vol.15, no. 6, pp.684-698, June, 2007.

[21] D. L. Oliveira, et al., "Design of Locally-Clocked Asynchronous Finite State Machines using Synchronous CAD tools," 4th IEEE Latin American Symposium on Circuits and Systems, 2013.

[22] T. Curtinhas, et al., "SICARELO: A Tool for Synthesis of Locally-Clocked Extended Burst-Mode Asynchronous Controllers," 5th IEEE Latin American Symposium on Circuits and Systems, 2014.