Uma Interface Assíncrona Robusta para Sistemas Localmente Síncronos–Globalmente Assíncronos

[1]Duarte L. Oliveira, [1]Eduardo Lussari, [2]Sandro S. Sato e [1]Lester A. Faria
[1]Divisão de Engenharia Eletrônica – Instituto Tecnológico de Aeronáutica – ITA – IEEA – SJC – SP – Brazil
[2]Departamento de Informática – ETE Ferraz de Vasconcelos – SP – Brazil

***Resumo* – Sistemas digitais contemporâneos devem necessariamente basear-se no conceito "System-on-Chip - SoC". Um estilo interessante para projeto em SoC é o paradigma GALS (*Globally Asynchronous, Locally Synchronous*), que pode ser usado para VLSI_DSM (*Very Large System Integration - Deep-Sub-Micron*). A principal desvantagem na concepção de um sistema de GALS é a interface assíncrona (*wrapper* assíncrono - AW) quando implementada em VLSI_DSM. Há um estilo de projeto típico AW, baseado em controladores assíncronos que fornece comunicação entre os módulos (chamados de ports), mas os Controladores de Ports são geralmente sujeitos a risco (*hazard*) essencial. No que diz respeito a esta desvantagem, este trabalho propõe um invólucro (*wrapper*) robusto assíncrono, que é livre de hazard essencial e permite autonomia total para os módulos localmente síncronos. Os nossos controladores de ports foram implementados em uma arquitetura hibrida (standard RS e máquina de Huffman), obedecem ao modelo de atraso de portas e fios com atraso limitado (*Bounded gate and wire delay – BGWD*) e interagem com o ambiente no modo I_b/O_b o que permite um melhor desempenho. As características da arquitetura da nossa interface mostram ter um grande potencial de aplicação em sistemas de VLSI_DSM.**

Palavras-chave: especificação modo rajada; risco; máquinas de estado finito; gated-clock; lógica assíncrona; interface

I. INTRODUÇÃO

Sistemas digitais contemporâneos geralmente são implementados em Sistema de Integração Muito Grande (VLSI, *Very Large Scale Integration*) e deve necessariamente basear-se no conceito "System-on-Chip - SoC". A razão para isso é satisfazer a demanda crescente por maior desempenho, possibilidade o reuso, e requisitos de baixo consumo de potência [1,2]. Circuitos SoC são compostos por módulos funcionais, os quais podem ser IP-cores (núcleos de propriedade intelectual) oferecidos por muitos fornecedores diferentes. Esses núcleos são pré-concebidos, verificados, testados e otimizados para alto desempenho, fornecendo redução de custos e tempo de desenvolvimento. Uma vez que os circuitos SoC são implementados em tecnologia DSM (*deep-sub-micron*), é sabido que em nanotecnologia CMOS DSM (por exemplo, 70nm, 500M transistores e f = 2,5 GHz) os atrasos nos fios são significativos e a diferença entre atrasos mínimos e máximos nas portas é grande [3].

Duarte L. Oliveira, duarte@ita.br , Tel +55-12-3947-6813, Fax +55-12-3947-6930. Eduardo Lussari, lussari@gmail.com; Sandro S. Sato, shoiti@uol.com.br; Lester A. Faria, lester@ita.br.

Portanto, quando os circuitos SoC são implementados usando um sinal de relógio global, eles estão sujeitos a penalidades de velocidade e potência (*clock skew*, redes de distribuição, etc), fazendo da análise de temporização algo muito complexo [4].

Metodologias de projeto assíncrono [5,6] podem, naturalmente, eliminar tais desafios, removendo o sinal de relógio global do projeto. Classes diferentes de circuitos assíncronos podem ser usadas para implementar SoCs, que podem ser construídas a partir de módulos completamente assíncronos, entretanto este tipo de circuito não é uma solução amplamente aceita. As principais razões para isso são: a) falta de ferramentas confiáveis para o projeto assíncrono; b) dificuldades para projetar e testar circuitos livres de todo tipo de risco (*hazard)*; c) cultura limitada em projeto assíncrono; d) ausência de *IPs* assíncronos [7].

No que diz respeito a esta situação, soluções intermediárias foram desenvolvidas entre "totalmente síncrono" e "totalmente assíncrono", tais como a metodologia GALS (*Globally Asynchronous Locally Synchronous*). O termo GALS foi usado pela primeira vez por Chapiro em sua tese de doutorado [8]. Um sistema de GALS consiste em diversos módulos funcionais síncronos que comunicam entre si sob a forma assíncrona. Neste artigo, referimo-nos aos sistemas GALS como um sistema digital dividido em módulos funcionais (que podem ser IPs), que carregam os seus próprios sinais de relógio individuais, que não estão relacionados entre si. Um esquema de comunicação assíncrono é usado para a comunicação entre estes diferentes módulos com diferentes domínios de relógio. A fim de lidar com a comunicação assíncrona entre os módulos, um circuito de interface tem de ser adicionado à volta de cada um dos módulos síncronos, proporcionando um invólucro (ou *wrapper*) assíncrono. O termo invólucro assíncrono foi usado pela primeira vez em [9]. Esta interface local pode ser construída com o uso de relógios locais, FIFO, controlador de comunicação assíncrona (*ports* de entrada, *ports* de saída), etc. Techan et al. em [10] mostram estilos diferentes para interfaces assíncronas dedicadas a sistemas GALS. A Fig. 1 mostra uma interface genérica com um módulo síncrono. Sistemas GALS foram usados com sucesso em muitas implementações, incluindo circuitos de aplicação específica – ASIC (*Application Specific Integrated Circuit)* [11,12] e FPGA [13].

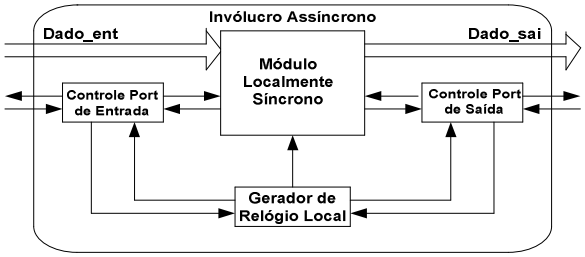

Fig. 1. Invólucro assíncrono

As interfaces assíncronas que fazem uso de *ports* de comunicação são interessantes porque permitem remover o esquema de *handshake* assíncrono do módulo síncrono, permitindo que o módulo síncrono seja projetado usando as técnicas padrões de projeto síncrono. Embora a metodologia GALS tenha resolvido os problemas relacionados com o sinal de relógio global, a comunicação entre os módulos é realizada no paradigma assíncrono, sendo, portanto, sujeitas a seus problemas inerentes.

A. *Implementação de ports: diferentes metodologias*

Diferentes tipos de *ports* têm sido sintetizados no estilo de *síntese lógica* [5]. Como um exemplo, os *ports* propostos em [14] foram especificadas em STG (Gráfico de Transição de Sinal), que é uma especificação do tipo Petri-net, sendo sintetizada na ferramenta Petrify [5]. Esses *ports* devem cumprir a exigência bifurcação isocrônica (*isochronic fork*) [5,6], mas a realização deste requisito em VLSI_DSM apresenta um alto nível de dificuldade. Esta exigência diz que para fios com bifurcação (fan-out >1) os atrasos nesses fios devem ser iguais [5].

Por outro lado, os *ports* propostos em [15,16] foram especificados em Modo Rajada Estendido XBM (*Extended burst-mode*) e Modo Rajada BM (*Burst Mode*). Esses *ports* foram sintetizados, respectivamente, nas ferramentas 3D [17] e Minimalist [18]. Eles interagem com o meio ambiente no modo fundamental generalizado (GFM) (*Generalized Fundamental Mode*), exigindo uma análise de temporização e estando sujeitos a *hazard* essencial, especialmente no domínio da tecnologia DSM. No modo GFM uma nova entrada tipo rajada pode ser aceita (ativada) se o circuito estiver estabilizado, isto é sem nenhuma atividade elétrica. No que diz respeito a esta última desvantagem, no caso o modo GFM a inserção de elementos de retardo pode ser necessária, mas esta solução (VLSI_DSM) degrada a capacidade de teste e de tempo de ciclo.

Este artigo propõe um novo invólucro assíncrono para GALS. Uma vez que uma grande desvantagem na concepção de invólucros assíncronos é a síntese dos *ports*, o invólucro assíncrono proposto mostrou ser robusto. Seus *ports* são livres de *hazard* essencial, sendo facilmente implementados em VLSI_DSM. Eles foram sintetizados pelo método proposto em [19], mas sintetizados em uma arquitetura hibrida (standard RS e máquina de Huffman), cuja finalidade é obter uma redução de área [5]. Para aumentar o desempenho os nossos *ports* operam no modo I_b/O_b de [20].

Outras vantagens deste invólucro são: 1) autonomia total para os módulos localmente síncronos ao interagir com o invólucro assíncrono proposto e 2) como seus *ports* interagem com o ambiente no modo I_b/O_b, não necessita de análise de temporização, portanto são mais robustos do que o modo GFM. No modo I_b/O_b, uma nova entrada do tipo rajada é imediatamente aceita quando todos os sinais de rajada de saída alterarem seus valores, isto é forem ativados.

Este trabalho está estruturado da seguinte forma: seções II e III apresentam as arquiteturas propostas de invólucro assíncrono e gerador de *gated-clock*, respectivamente, enquanto a seção IV apresenta a síntese de *ports* robustos. Seção V discute o invólucro assíncrono proposto e, finalmente, a seção VI apresenta algumas conclusões e sugere trabalhos futuros.

II. INVÓLUCRO ASSÍNCRONO: ARQUITETURA

O principal objetivo da arquitetura proposta consiste em proporcionar uma interface de interação fraca entre o módulo localmente síncrono (LSM) e interface assíncrona. A Figura 2 mostra as variáveis "dados disponíveis" e "aceitar dados", como as únicas usadas para comunicação entre o LSM e a interface. Quando *dados disponíveis = '1'*, os dados estão prontos para serem transmitidos, enquanto que quando *dados aceitar = '1'* os dados foram recebidos.

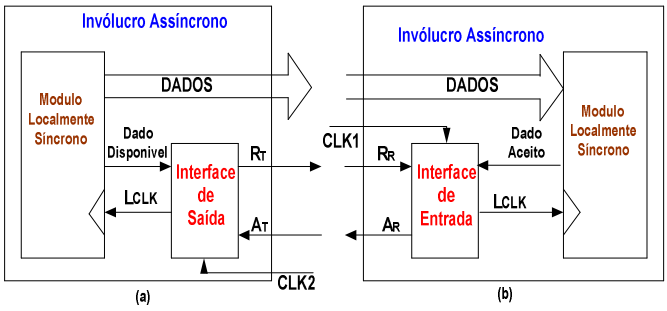

Fig. 2. Módulo localmente síncrono com interface fraca: a) saída; b) entrada

A Figura 3 mostra a arquitetura do controle de comunicação de saída proposto, que implementa a interação fraca entre a interface e o LSM, enquanto que as Fig. 4 e 5 mostram, respectivamente, os invólucros assíncronos de entrada e saída, com a inserção de um gerador de *gated-clock*. Finalmente, a Fig. 6 mostra o invólucro assíncrono proposto completo que recebe e transmite os dados.

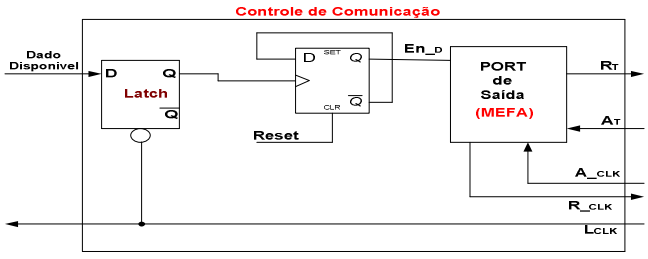

Fig. 3. Controle de comunicação de saída

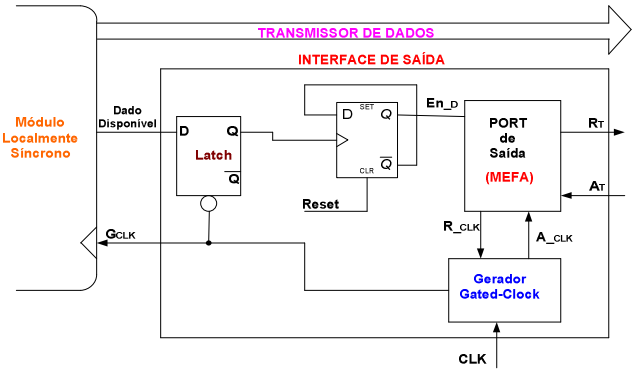

Fig. 4. Invólucro assíncrono proposto (saída).

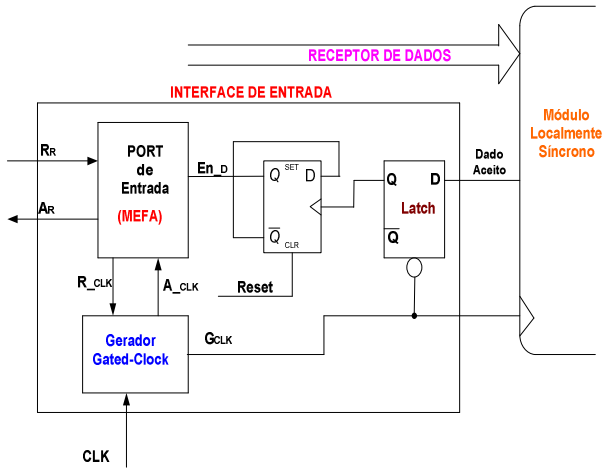

Fig. 5. Invólucro assíncrono proposto (entrada)

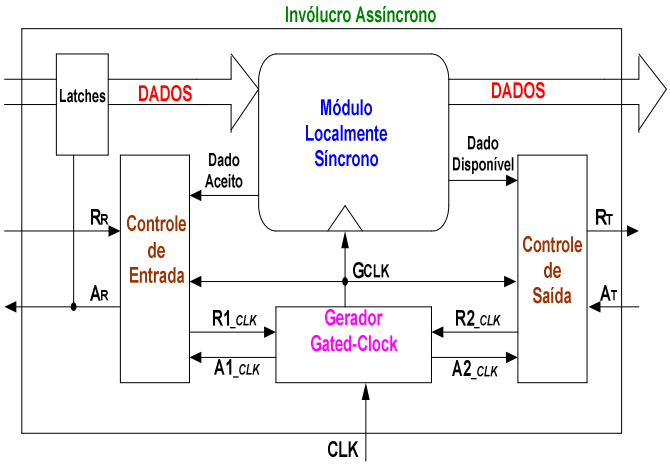

Fig. 6. Invólucro assíncrono proposto (com I/O)

III. GERADOR DE GATED-CLOCK

Neste trabalho, propomos também um gerador de *gated-clock* (GCG), composto basicamente de dois sincronizadores e um *gated-clock*. A Figura 7 mostra o diagrama de temporização da GCG proposto com a ativação e desativação do sinal GCLK. Enquanto a Fig. 8 mostra a arquitetura do GCG, a Fig. 9 mostra a topologia do *gated-clock*, e a Fig. 10 mostra a topologia de seu sincronizador. A parada do sinal GCLK ocorre quando R_CLK vai de 0 → 1 e após dois ciclos de relógio do sinal "Stop" vai de 0 → 1, que determina a parada do sinal GCLK.

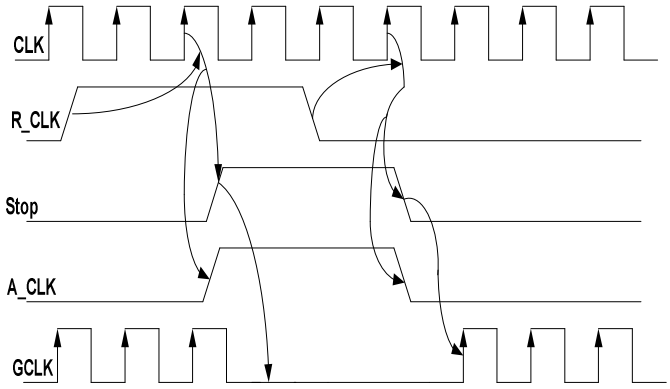

Fig. 7. Diagrama de tempo: gerador de gated-clock

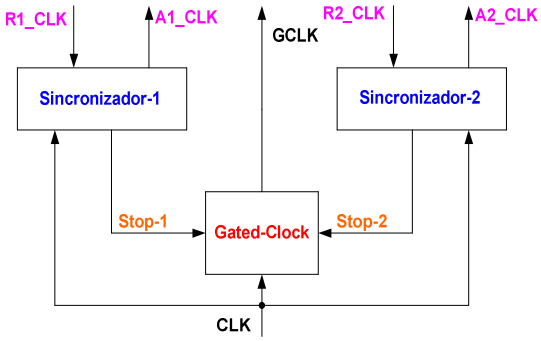

Fig. 8. Arquitetura para o gerador de gated-clock proposto

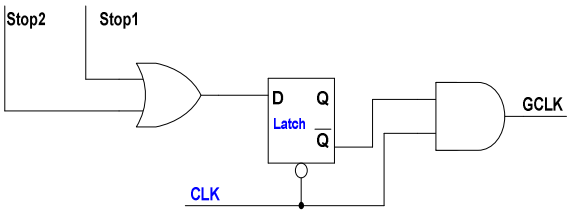

Fig. 9. Topologia do gated-clock

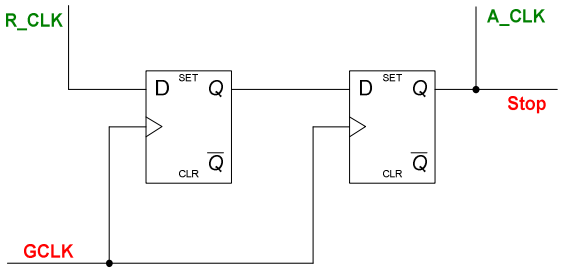

Fig. 10. Topologia do sincronizador

IV. PROJETO: PORTS (AFSM)

Os *ports* de entrada e saída utilizados no nosso invólucro assíncrono já haviam sido propostos anteriormente em [15]. Eles são descritos na especificação XBM (como mostrado na Fig. 11 e 12). Eles tinham sido sintetizados no modelo de atraso onde portas e fios possuem atrasos limitados (BGWD), portanto delimitados em mínimo e máximo, sendo que estão sujeitos a risco essencial, e interagem com o ambiente no GFM [17,18]. A especificação XBM dos *ports* de entrada e saída de [15] atendem o conceito de sinal essencial, isto é, em cada transição de estado deve haver, pelo menos, um sinal essencial. Para ser caracterizado como um sinal essencial, um novo sinal de entrada do XBM não pode ter sido mencionado na transição do estado anterior [19]. Como um exemplo, a Fig. 12 apresenta a especificação do *port* de saída: na transição de estado 2 → 3, o sinal A_CLK é essencial, porque na transição de estado 1 → 2 não foi mencionado.

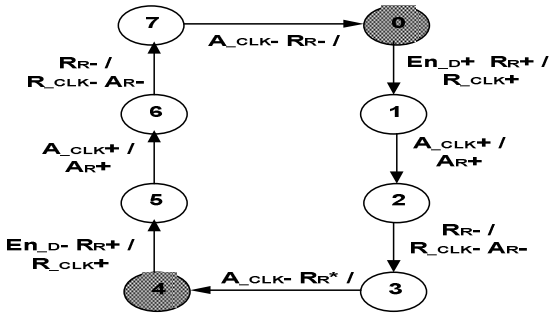

Fig. 11. Especificação XBM: *port* de entrada.

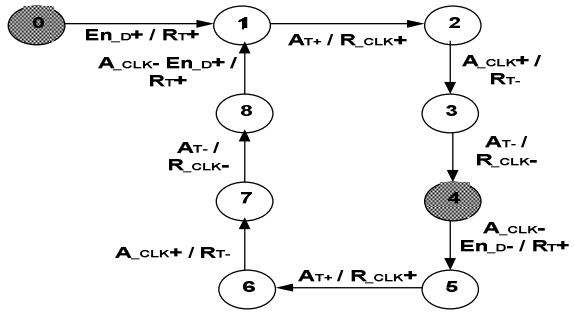

Fig. 12. Especificação BM: *port* de saída.

Oliveira et al. [19] mostram que, se a especificação XBM satisfaz o sinal essencial, então é possível gerar uma cobertura lógica (implicantes primos), satisfazendo a regra proposta por Ungle [21], que especifica as condições para uma especificação estar livre de *hazard* essencial. A Figura 13 mostra a tabela de fluxo de estados do *port* de saída, com a introdução de um sinal de estado 'Z' para resolver os conflitos [17]. O sinal de estado Z é introduzido na forma que satisfaça o modo de operação que é I_b/O_b [20]. Usando o procedimento de [19] a Fig. 14 mostra todas as células (mintermos) utilizadas na cobertura lógica que assegurem que o *port* de saída está livre de *hazard* essencial (células com valor em azul são criadas para garantir cobertura livre de *hazard* lógico). O *port* de saída foi implementado na arquitetura de Huffman com realimentação de saída (HM_OF). Fazendo uma síntese análoga o *port* de entrada foi implementado utilizando a arquitetura híbrida (padrão RS e HM_OF). Finalmente, a Fig. 15 e 16 mostram, respectivamente, os circuitos lógicos dos *ports* de entrada e de saída.

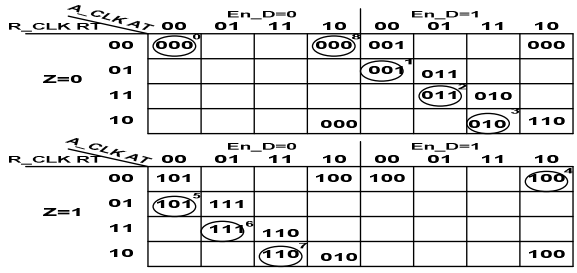

Fig. 13. Tabela de fluxo de estados: *port* de saída.

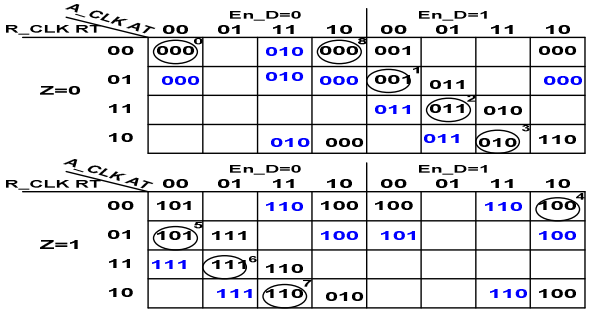

Fig. 14. Tabela de fluxo de estados: *port* de saída (cobertura).

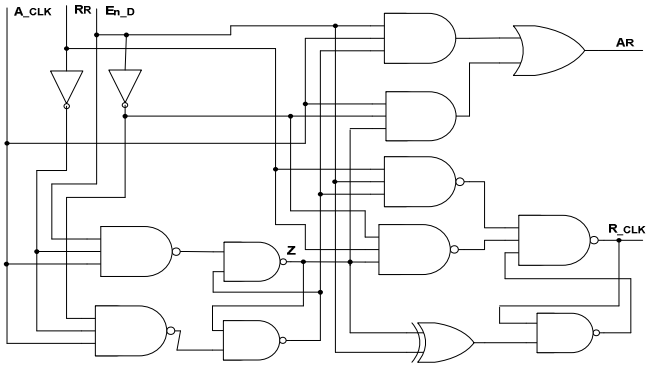

Fig. 15. Circuito lógico: *port* de entrada.

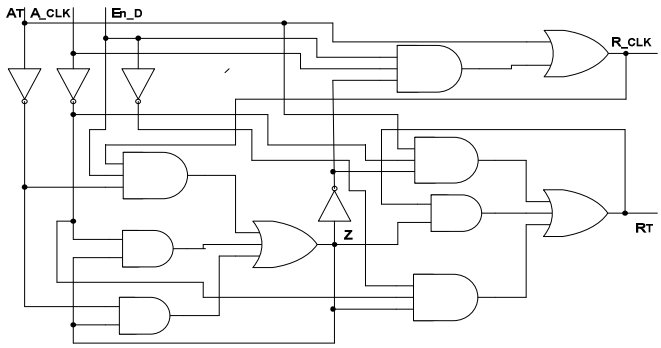

Fig. 16. Circuito lógico: *port* de saída.

V. DISCUSSÃO & SIMULAÇÃO

Oliveira et al., em [20], apresentam uma lista de vantagens dos sistemas GALS, o que leva a uma conclusão que o projeto GALS pode desempenhar um papel relevante no futuro do projeto digital, mas uma grande desvantagem para este uso é a interface assíncrona.

Focando nestes tipos de aplicações, a interface assíncrona (invólucro) proposta livre de *hazard* tem um grande potencial para implementações em VLSI_DSM. Apesar dos nossos ports serem originalmente propostos em [15], há três diferenças significativas nos circuitos resultantes: 1) os nossos *ports* são livres de *hazard* essencial, enquanto os de [15] não estão garantidos estarem livres de *hazard* essencial; 2) os nossos *ports* têm um melhor desempenho, porque eles operam no modo I_b/O_b, enquanto que os de [15] operam no modo GFM; 3) os nossos *ports* foram implementados na arquitetura hibrida (standard RS e HM_OF) que permite uma redução de área, enquanto os de [15] foram implementados somente na arquitetura HM_OF. Figuras 17 e 18 mostram as simulações livres de hazard dos ports de I/O do invólucro assíncrono proposto, demostrando que a arquitetura proposta satisfaz a especificação XBM.

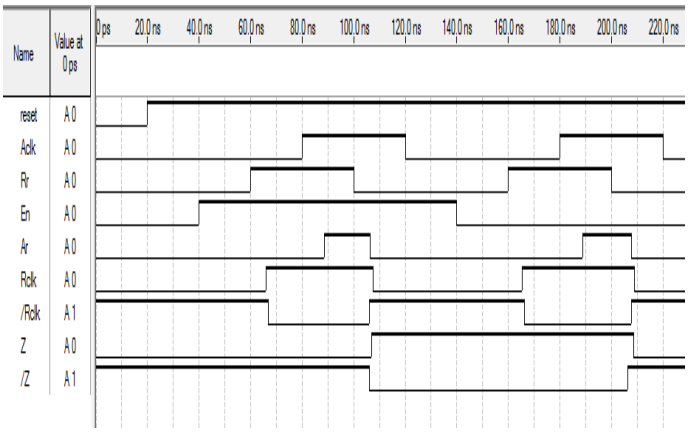

Fig. 17. Simulação do *port* de entrada.

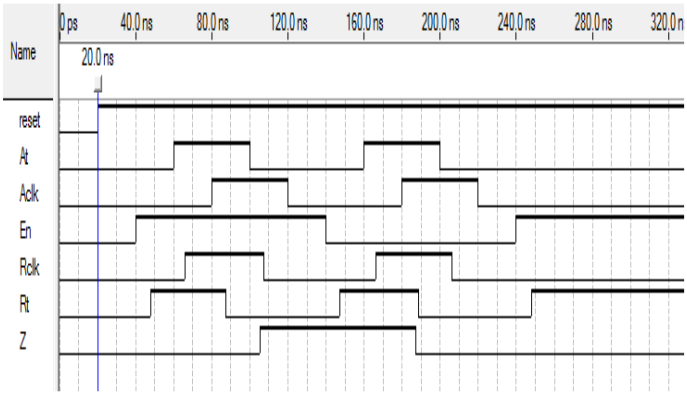

Fig. 18. Simulação do *port* de saída.

VI. CONCLUSÃO

Sistemas GALS implementados em VLSI_DSM mostraram ser um estilo de projeto interessante para SoCs, mas os problemas típicos relativos à interface assíncrona, especialmente para o projeto do invólucro assíncrono, acabam sendo uma desvantagem. No que diz respeito a esta situação, foi proposta uma nova arquitetura para o invólucro assíncrono que é capaz de superar os problemas anteriormente discutidos, mostrando ser uma boa opção para os projetistas que precisam implementar GALS em VLSI_DSM. Os resultados obtidos mostraram que a arquitetura proposta está livre de *hazard* essencial e permite autonomia total para os módulos localmente síncronos. Ela atende ao modelo de atraso BGWD e interage com o ambiente no modo I_b/O_b. Comparando com os *ports* da literatura, os nossos tem propriedades interessantes, que leva a ter um grande potencial de aplicação em sistemas de VLSI_DSM. Pretende-se para trabalhos futuros o projeto de uma interface assíncrona robusta para implementação GALS, envolvendo FIFO.

REFERENCES

[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] K. D. Muller-Glaser, et. al. "Multiparadigm Modeling in Embedded Systems Design", *IEEE Trans. on Control Systems Technology*, vol. 12, no. 2, March 2004.

[3] D. Goldhaber-Gordon, et al., "Overview of Nanoelectronic Devices," *Proc. of the IEEE*, vol. 85, No. 4, April 1997, pp.521-540.

[4] A. J. Martin and M. Nystrom, "Asynchronous Techniques for System-on-Chip Design," *Proc. of the IEEE*, vol.94, no. 6, pp.1089-1120, June 2006.

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

[6] J. Sparzo and S. Furber, "Principles of Asynchronous Circuits Design", Kluwer Academic Publishers, 2001.

[7] W. Hardt, et. al., "Architecture Level Optimization for Asynchronous IPs", Proc. 13th Annual IEEE Int. Conf. ASIC/SOC, pp.158-162, 2000.

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

[9] D. S. Bormann and P. Y. K., "Asynchronous Wrappers for Heterogeneous Systems," Proc. Int. Conf. Computer Design (ICCD), pp.307-314, October 1997.

[10] P. Techan, M. Greenstreet, and G. Lemieux, "A Survey and Taxonomy of GALS Design Styles," *IEEE Design & Test of Computers*, vol. 24, pp.418-428, September-October 2007.

[11] F. K. Gurkaynak, et al., "GALS at ETH Zurich: Success or Failure ?," , Proc. 12th IEEE Int. Symposium on Asynchronous Circuits and Systems, pp. 150-159, 2006.

[12] M. Krstic´, et al., "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook," *IEEE Design & Test of Computers*, vol. 24, pp. 430-441, September-October 2007.

[13] A. Kumala, et al., “Reliable GALS Implementation of MPEG-4 Encoder with Mixed Clock FIFO on Standard FPGA,” Int. Conf. on Field Programmable Logic and Application, pp. 1-6, 2006.

[14] E. Amini, M. Najibi and H. Pedram, ”Globally Asynchronous Locally Synchronous Wrapper Circuit based on Clock Gating,” Proc. Emerging VLSI Technologies and Architectures, pp.193-199, 2006.

[15] J. Muttersbach, T. Villiger, and W. Fichtner, “Practical Design of Globally-asynchronous Locally-synchronous System,” Proc. IEEE 6th Int. Symposium Advanced Research in Asynchronous Circuits and Systems, pp.52-59, 2000.

[16] J. Pontes, et al., “SCAFFI: an Intrachip FPGA asynchronous interface based on hard macros,” 25th Int. Conf. on Computer Design, pp.541-546, 2007.

[17] K. Y. Yun and 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, February, pp. 101-132, 1999.

[18] R. M. Fuhrer, et al., “Minimalist: An environment for the Synthesis, verification and testability of burst-mode machines,” Technical Report, Columbia University, TR-CUCS-020-99, 1999.

[19] D. L. Oliveira, et al., “Burst-Mode Asynchronous Controllers on FPGA,” *Int. Journal of Reconfigurable Computing*, vol. 2008, pp.1-10, 2008.

[20] D. L. Oliveira, et al., “Synthesis of Robust Conrollers for GALS_FPGA from Multi-Burst Graph Specification,” Proc. IEEE VII Southern Conference on Programmable Logic (SPL), pp. 123-129, Cordoba, Argentina, 2011.

[21] S. H. Unger, *Asynchronous Sequential Switching Circuits*, John Wiley & Sons Inc, 1969.