Implementação em PLD de Máquinas de Estados Finitos Assíncronas de Relógio Local

[1]Duarte L. Oliveira, [1]Diego Bompean, [1]Tiago Curtinhas, [1]Lester A. Faria, [1]Noé Alles e [2]Leonardo Romano
[1]Divisão de Engenharia Eletrônica do Instituto Tecnológico de Aeronáutica – ITA – IEEA – SJC – SP – Brazil
[2]Departamento de Engenharia Elétrica do Centro Universitário da FEI – SBC – SP – Brazil

Resumo — **Controladores baseados em máquinas de estado finito (MEF) síncronas são grandemente usados para projeto da unidade de controle em um sistema digital embarcado (SDE). Estes sistemas possuem requisitos críticos, tais como consumo de potência, robustez, velocidade, etc. O paradigma assíncrono possui características interessantes que pode ser uma alternativa para este projeto. Neste artigo propomos um método para projeto de MEF assíncrona no estilo de relógio local. Este estilo reduz os requisitos da lógica assíncrona e viabiliza a síntese em dispositivos programáveis (*Programmable Logic Device* – PLD). As PLDs são uma alternativa de projeto rápido e de custo reduzido e são bastante difundidas no projeto SDE. O nosso método parte de uma especificação popular denominada modo rajada estendida (*extended burst-mode – XBM)* e usa as técnicas do paradigma síncrono para realizar a síntese.**

***Palavras–chave* — relógio local, síntese lógica, maquinas de estado finito assíncronas, especificação modo rajada, PLD**

I. INTRODUÇÃO

Sistemas digitais embarcado podem requerer alta capacidade de integração, alta velocidade e baixo consumo de energia [1]. Eles podem necessitar de baixa interferência eletromagnética, serem robustos aos efeitos radiativos, como também a variações de temperatura e de tensão de alimentação. Um caminho que pode satisfazer os requisitos do ambiente embarcado é a implementação destes sistemas na tecnologia PLD (*Programmable Logic Device*), nos dispositivos: CPLD (*complex* PLD) e FPGA (*Field Programmable Gate Array*) [2,3]. Os dispositivos FPGAs tornaram-se um meio popular de implementar circuitos digitais. Tecnologia FPGA tem crescido consideravelmente nos últimos anos, gerando FPGAs de até 50 milhões de portas, permitindo assim que sistemas digitais complexos possam ser programados em tais dispositivos [3,4]. FPGAs de alto desempenho são implementados na tecnologia MOS *Deep-Sub-Micron* (MOS-DSM). Essa tecnologia necessita operar com baixo ruído e a diferença entre o atraso máximo e mínimo nas linhas, portas é maior quando comparado com outras tecnologias MOS, e o atraso em uma linha pode ser maior que o atraso em uma porta [5]. Sistemas digitais síncronos usam um sinal de relógio global para sincronizar as suas operações e são bastantes populares devido à simplicidade de projeto. Também há uma oferta abundante de ferramentas CAD comercial para síntese automática.

Duarte L. Oliveira, duarte@ita.br, Tel. +55-12-3947-6813, Fax +55-12-3947-6930. Diego Bompean, diegobompean@yahoo.com.br; Tiago Curtinhas, thiagohd@gmail.com; Lester A. Faria, lester@ita.br; Noé Alles, noealles@ita.br; Leonardo Romano, leoroma@uol.com.br.

Um sério problema na tecnologia MOS-DSM é conviver com o sinal de relógio global, porque ele é um grande causador de ruído, de alta emissão eletromagnética, consome uma parte significativa da potência e definir a distribuição do sinal de relógio é uma tarefa com complexidade crescente (por exemplo: relógio defasado – *clock skew*). Análise de temporização de circuitos digitais MOS-DSM síncronos de alta integração é extremamente difícil. Uma característica comum nos sistemas eletrônicos embarcados é o fato de serem alimentados por bateria. Como são alimentados por bateria é desejável que as baterias tenham uma longa vida útil, portanto a potência dissipada é um parâmetro muito importante na concepção de tais sistemas [6]. Em um sistema digital á parte seqüencial é o principal contribuinte para a dissipação de potência dinâmica. Estudos recentes têm mostrado que em tais sistemas o relógio consome uma grande porcentagem (15% a 45%) da potência do sistema. Uma interessante alternativa para projeto digital embarcado, porque ele elimina os problemas causados pelo sinal de relógio e aumenta a robustez é o *paradigma assíncrono*.

A. Sistemas digitais assíncronos

Sistemas assíncronos operam por eventos não possuem um sinal global que sincroniza as operações. A sincronização é realizada por protocolos do tipo *Handshaking*. No projeto de sistemas digitais assíncronos temos que definir o *estilo* de projeto, e em qual *classe* o circuito vai operar corretamente. A classe define o modelo de atraso e em que modo de operação o circuito se comunica com o ambiente [7]. Os circuitos assíncronos podem ser classificados em duas classes: **a)** portas e linhas com atrasos delimitados (*bounded gate and wire delay – BGWD*); **b)** portas e linhas com atrasos quaisquer (indefinidos), mas finitos (*unbounded gate and wire delay – UGWD*). Um importante circuito assíncrono que obedece ao modelo BGWD é o modo-rajada e extensões (*burst-mode circuits*) [8,9]. O circuito insensível ao atraso (*delay insensitive circuits – DI*) obedece ao modelo UGWD [7]. Este modelo é o mais robusto e isento de qualquer análise de temporização. Martin [10] mostra que a aplicação deste modelo é muito restrita. Existe variantes menos restrita deste modelo de atraso, como os circuitos independentes da velocidade (*speed independent circuits – SI*) e os circuitos quase insensíveis ao atraso (*quasi delay insensitive circuits - QDI*) [7]. A comunicação do circuito com o ambiente ou é realizada no modo fundamental generalizado (MFG) (por exemplo: modo-rajada) [8] ou no modo Entrada/Saída (M_E/S) (por exemplo: DI, SI, QDI) [7,11,12]. No MFG a mudança de um novo conjunto de entradas o circuito deve

estar estabilizado. No M_E/S a mudança de um sinal de saída pode imediatamente habilitar a mudança de um sinal na entrada.

No aspecto do projeto assíncrono há três diferentes estilos: **a)** decomposição do tipo controlador + *data-path* [13]; **b)** *micropipeline* [14]; **c)** composição com macromódulos [15]. Os três estilos podem ser projetados nas diferentes classes de circuitos assíncronos. O estilo de projeto assíncrono por decomposição é familiar aos projetistas e é voltado para aplicações de controle intensivo. Ele tem duas variantes: *a)* projetos que envolvem *data-path* assíncrono, portanto sem qualquer inserção de elementos de atraso. A especificação denominada STG (*signal transition graph* – Petri-net) é a mais apropriada para descrever o controlador, que vai interagir com o *data-path* assíncrono [11]. O *data-path* assíncrono envolve componentes *dual-rail*, que tem um custo muito alto; *b)* projetos que envolvem *data-path síncrono*, um elemento de atraso é inserido e ele define o tempo de ciclo da transição de estado (ver Fig. 1). A especificação denominada modo rajada estendida (*extended burst-mode* – XBM) é a mais apropriada para descrever o controlador (máquina de estado finito assíncrona – MEFA), e que vai interagir com o *data-path* síncrono [16]. A variante (b) implica em circuitos com menor consumo de potência, menor área e simplifica o projeto quando comparado com a variante (a). Em anos recentes, alguns circuitos assíncronos da vida real foram projetados com sucesso e eficientemente baseados nas MEF assíncronas modo rajada estendido (MEFA_XBM) [17-19]. Neste trabalho nós focamos o projeto das MEFA_XBM.

Dois são os pontos fracos de MEF assíncronas: dificuldade no projeto e poucas ferramentas para síntese automática. A dificuldade do projeto é que o circuito deve ser livre de risco (*hazard*) e de corrida crítica [7]. O projeto deve satisfazer os diferentes tipos de risco, como: funcional, lógico e seqüencial. O projeto de MEFA_XBM pode ser sintetizado em diferentes arquiteturas. Uma interessante arquitetura é a de relógio local, porque reduz os requisitos da lógica assíncrona [8,20-28]. Os métodos propostos em [8,26,27] para relógio local estão voltados para a especificação BM que é limitada para descrever interação com data-path síncrono. Outras restrições estão na minimização de estados e na minimização lógica, onde as equações de saída devem ser livres de risco lógico [8].

Neste artigo introduzimos um novo método para síntese de MEFA_XBM com relógio local. O nosso método parte da especificação XBM e usa o paradigma síncrono de síntese de MEF síncrona de saída direta. O nosso método projeta a MEFA_XBM como uma MEF síncrona de saída direta, portanto usa ferramentas do paradigma síncrono. O sinal de relógio é substituído por uma função Booleana combinatória que gera localmente o sinal relógio. No nosso relógio somente há transição, quando um conjunto de sinais de entrada (rajada) rotulados em uma transição de estado for ativado, portanto reduz a atividade do relógio. A única restrição é que á função geradora de relógio tem que ser livre de risco lógico. As MEFA_XBM de relógio local sintetizadas pelo nosso método são implementadas na arquitetura da Fig. 2. Elas podem ser sintetizadas em qualquer PLD, como CPLDs e FPGAs sem a necessidade de satisfazer qualquer tipo de mapeamento de macro-células. Como restrições, devemos inserir um elemento de atraso para garantir o tempo de *setup* dos FFs e a função combinatória geradora de relógio necessita de uma cobertura livre de risco lógico.

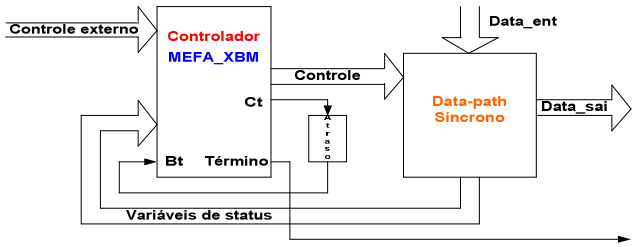

Fig. 1. Arquitetura: MEFA_XBM + Data-path síncrono.

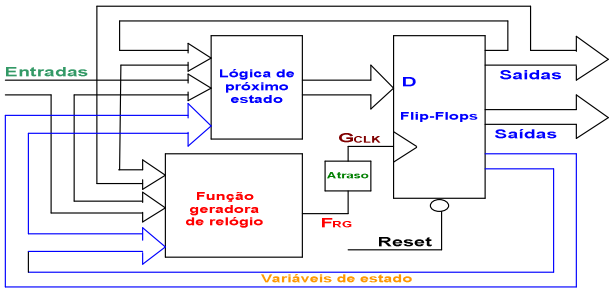

Fig. 2. Arquitetura proposta: MEFA com RL para PLD.

II. VISÃO GERAL: MÁQUINAS DE ESTADO FINITO

Máquinas de estado finito são grandemente usadas no projeto de sistemas digitais. Elas se apresentam em diferentes modelos, são usadas nos diferentes paradigmas de projeto, são sintetizadas em diferentes arquiteturas e implementadas em diferentes tecnologias.

A. Máquinas de estado finito assíncronas em PLDs

O paradigma de projeto envolvendo PLDs é bastante popular para prototipagem e produção de circuitos digitais, isto é devido ao seu baixo custo e tempo curto de projeto. O seu foco tem sido circuitos digitais síncronos. Houve alguns recentes esforços para prototipagem assíncrona. Ela foi realizada em FPGAs comerciais [29] e FPGAs assíncronas acadêmicas [30]. Existem duas razões porque FPGAs comerciais têm dificuldades em implementar MEF assíncronas:

a. *Processo de mapeamento* de funções *Booleanas* livre de risco em blocos lógicos (macro-células) pode introduzir risco lógico. As ferramentas comerciais de decomposição e mapeamento em FPGAs baseadas em *look-up tables* (LUTs) não estão preparadas para decomposição das funções Booleanas livre de risco lógico. A decomposição lógica deve satisfazer os requisitos propostos em [31].
b. *Processo de roteamento* interno entre blocos lógicos pode introduzir atrasos significativos que pode resultar em risco seqüencial do tipo essencial. Este tipo de risco pode ser solucionado por inserção de elementos de atraso nas linhas de realimentação ou se a especificação satisfaz o requisito de sinal essencial [32].

B. MEF assíncronas de relógio local

Muitas propostas de MEF assíncronas de relógio local (MEF_RL) foram introduzidas nas últimas quatro décadas [20-28]. As MEF_RL também conhecidas como máquinas auto-sincronizadas, são máquinas que geram um simples relógio local. O objetivo destas propostas era permitir múltiplas mudanças de entrada (*multiple input chance* – MIC) e reduzir os requisitos da lógica assíncrona. A maior parte das propostas gera circuitos lentos e grandes inviabilizando as aplicações reais. Estas propostas são baseadas ou em flip-flops não convencionais e/ou no uso excessivo de elementos de atraso. Nowick em seu PhD [8] propõe uma síntese automática que parte da especificação modo rajada (*burst-mode (BM) specification*) e implementa na arquitetura de relógio local baseada em latches (ver Fig. 2). Esta especificação suporta uma classe de MIC. As vantagens são: o relógio somente é ativado quando há mudança de estado e o elemento de memória é baseado em latches. Elementos de atraso podem ser necessários para solucionar problemas de temporização. Nowick em [33,34] aplica em casos reais e mostra a eficiência do seu método.

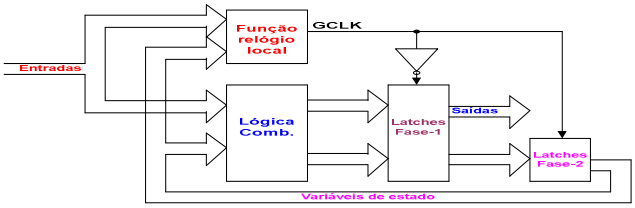

Fig. 3. Arquitetura: MEFA de relógio local de [8].

C. Arquitetura com saída direta

Um interessante tipo de MEF síncrona foi descrito em [35,36]. Ela é conhecida como MEF síncrona com saída direta (MEFS_SD), onde os sinais de saída são usados como sinais de estado (ver Fig. 4). Quando comparemos com os modelos tradicionais Moore e Mealy, temos: a) Pomeranz et al. [35] mostra o potencial das MEFS_SD, como redução do tempo de latência, área e eliminação dos glitches nos sinais de saída; b) Valeri [36] mostra que as MEFS_SD ocupam um menor número de macro-células em uma PLD. Há duas vantagens usando sinais de saída como sinais de estado:

- Redução ou eliminação das variáveis de estado, portanto podemos ter redução de área. Também aumentamos a observabilidade e a controlabilidade, que permite facilitar a testabilidade;
- Na execução clássica das máquinas modelo Moore há três blocos (lógica de excitação, flip-flops, lógica de saída). Nas máquinas de saída direta há somente dois blocos (lógica de excitação e flip-flops), portanto há uma redução no tempo de ciclo (aumento de taxa de relógio);

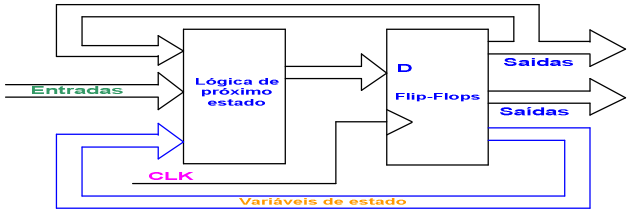

Fig. 4. Arquitetura: MEFS de saída direta.

III. ESPECIFICAÇÃO MODO-RAJADA ESTENDIDA

Nowick [8] propôs uma especificação denominada modo rajada (burst-mode – BM). Transições podem ocorrer quando uma ou múltiplas entradas mudam seu nível lógico, $0 \rightarrow 1$, ou $1 \rightarrow 0$ (sinais sensíveis á transição – ***TSS***). Quando não há nenhuma mudança na entrada à máquina permanece em seu estado estável. As rajadas devem ser monotônicas, isto é, elas podem mudar somente uma única vez durante a cada transição. Um estado inicial deve existir. Na especificação BM, a transição de estado é rotulada com entrada rajada / saída rajada, onde a saída rajada pode ser vazia e somente sinais rotulados podem mudar. A especificação BM deve satisfazer três propriedades para ter condições de implementação [8]: *1.* Polaridade dos sinais, que é a comutação da transição do sinal {+,–}; *2.* Ponto de entrada única; *3.* Conjunto máximo. Yun [9,16] propôs a especificação modo rajada estendida (*extended burst-mode* – XBM) adicionando duas características: sinais irrelevantes direcionados (*directed don't-care*), que permite um sinal mudar concorrentemente com os sinais de saída e sinais condicionais, que depende do nível do sinal (*level sensitive signals*– ***LSS***) com comportamento não monotônico. As restrições mencionadas acima foram generalizadas para permitir a extensão proposta por Yun em seu PhD [9].

Nós ilustramos a especificação com o benchmark SCSI. Fig. 5 mostra uma especificação XBM do SCSI com 4 entradas (*Cntgt1, Fain, Ok, Rin*), 2 saídas (*Aout, Frout*) e estado inicial *0*. A descrição *Rin+ Fain- / Aout+* na transição $5 \rightarrow 3$ significa que a saída (Aout: 0 → 1) vai ser ativada quando a entrada rajada for ativada (Rin**: 0→1** *AND* Fain: **1→0**). O sinal LSS *cntgt1* é usado para descrever a exclusão mútua entre as transições $3 \rightarrow 6$ e $3 \rightarrow 4$. O sinal irrelevante direcionado Rin^* na transição 4 → 5 significa que o sinal *Rin* pode ou alterar o seu valor ou permanecer no seu antigo valor. Todo a transição de estado deve ter pelo menos um sinal de entrada denominado de "*compulsory*". Um sinal de entrada é "compulsory" se na transição de estado anterior, ele não é irrelevante direcionado.

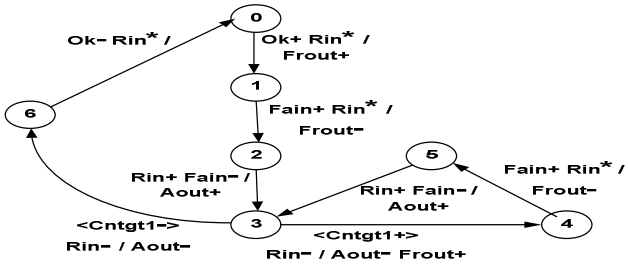

Fig. 5. Especificação MRE: SCSI.

IV. FUNÇÃO GERADORA DE RELÓGIO

A função geradora de relógio f_{GR} tem que ser livre de conflito (GR_LC). Nesta seção introduzimos o conceito de conflito para a função f_{GR}. O conflito está relacionado quando duas ou mais transições de estado que contêm estados em comum, mas com valores de saída da função f_{GR} diferentes. Formalmente definimos conflito como:

Definição 1: *Conflito*. Seja as transições de estado *Ti* e *Tj* quaisquer de uma especificação XBM da função f_{GR} com os respectivos cubos de transição Ti[*A,B*] e Tj[*C,D*]. Diz-se que há um conflito na f_{GR} entre *Ti* e *Tj* se e somente se:

1. A∩C≠∅ ou A∩D≠∅ ou B∩C≠∅ ou B∩D≠∅.
2. Existe um estado K∈ [Ti,Tj] | $f_{GR}(Ti_{-K})\neq 0$ ou $f_{GR}(Tj_{-K})\neq 0$.

A definição de conflito usa o conceito de cubo de transição. Fig. 6a,b mostram respectivamente uma especificação BM e a sua descrição na tabela verdade. Este exemplo possui cinco conflitos na função f_{GR}. O cubo de transição na transição de estado 0 →1 é descrito pelos cubos de entrada A=a,b,x,y=2200, onde 2 significa don't-care e cubo de saída B=a,b,x,y=1122. Para solucionar os conflitos usamos o algoritmo baseado na proposta por Yun et al. [9]. Figura 7 mostra uma tabela verdade para cada transição de estado da especificação BM. Figura 8 mostra a fusão de tabelas que obedece a definição 2. Figura 9 mostra a tabela verdade final, onde necessitou de duas variáveis de estado para eliminar os cinco conflitos. A função combinatória minimizada f_{GR} é obtida usando o algoritmo proposto em [9].

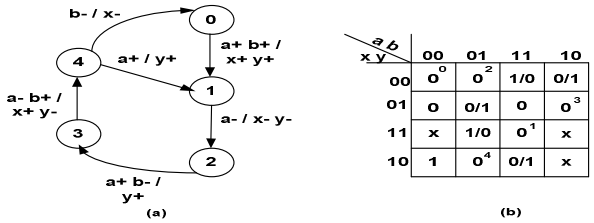

Fig.6. Especificação: a) grafo BM; b) tabela verdade da f_{GR}.

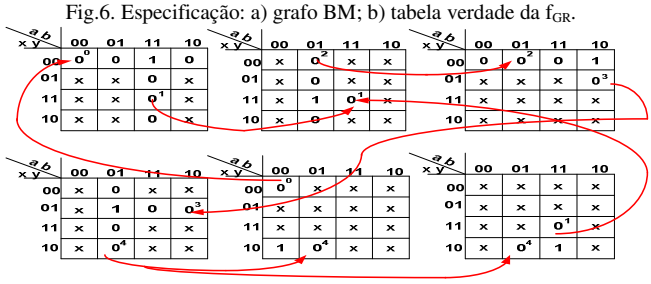

Fig.7. Tabelas verdade por transição de estado da f_{GR}.

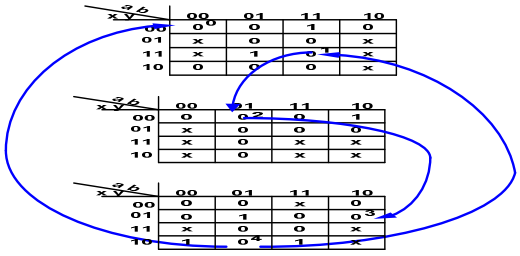

Fig. 8. Fusão de Tabelas verdade da função f_{GR}.

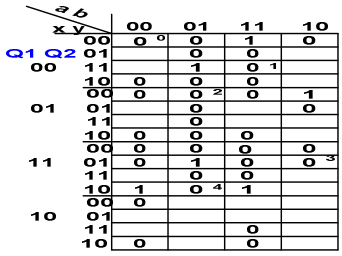

Fig. 9. Tabela verdade da função f_{GR} sem conflitos.

V. METODOLOGIA: SÍNTESE DE MEFA COM RL

A nossa metodologia para síntese de MEFA_RL está voltada para implementação em PLDs, portanto a nossa arquitetura usa flip-flops D como elemento de memória (ver Fig. 1). A metodologia tem quatro passos e parte da especificação XBM:

1. Usando o procedimento da seção IV extrair a função f_{GR} de relógio livre de conflitos e de risco lógico (GR_LCR).
2. Se houve inserção de variáveis na função GR_LCR reformular a especificação XBM, caso contrário ir para o passo 3.
3. Sintetizar a especificação XBM como uma MEF síncrona de saída direta usando o método em [35].
4. Substituir o sinal de relógio pela função GR_LCR e calcular o elemento de atraso para satisfazer o tempo de setup.

Figura 10 mostra a especificação BM reformulada obtida da especificação original e da tabela da Fig. 9. Figura 11 mostra a tabela de transição de estados da especificação BM reformulada que está voltada para o paradigma síncrono, no caso MEFS de saída direta. A máquina é inicializada no estado 0 (*abQ1Q2xy=000000*). Figura 11 mostra o elemento de atraso assimétrico próprio para os compiladores de PLDs.

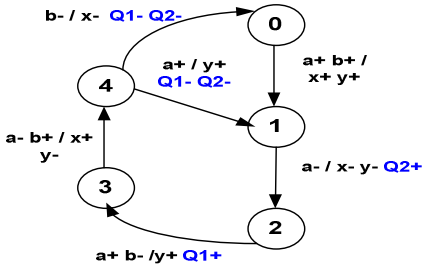

Fig. 10. Especificação BM reformulada.

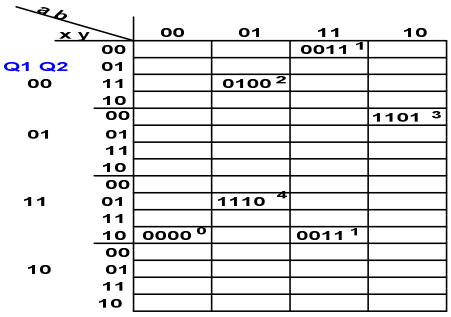

Fig. 11. Tabela de transição de estados da MEFS de saída direta.

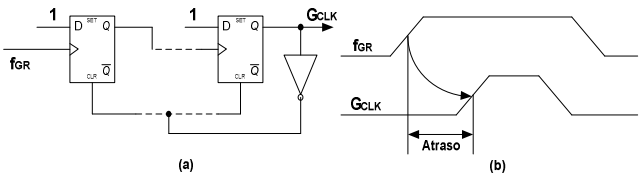

Fig. 12. Elemento de atraso proposto: a) circuito; b) temporização.

VI. CASO DE ESTUDO

Para ilustrar o nosso método usamos a especificação XBM descrita em Fig. 5 do controlador de barramento SCSI (*small computer systems interface*) como definido pela ANSI standard X3.131-1986 que é um protocolo físico e lógico para comunicação entre computadores e dispositivos periféricos [17]. Figuras 13, 14 e 15 descrevem o passo 1 que é gerar a função f_{GR} livre de risco lógico e de conflitos. Como houve quatro conflitos a especificação necessitou de duas variáveis de estado (Q1 e Q2) para eliminá-las. Figura 16 mostra a especificação XBM reformulada obtida no passo 2. Figuras 17 e 18 mostram a síntese no paradigma síncrono da MEF que foi obtida no passo 3.

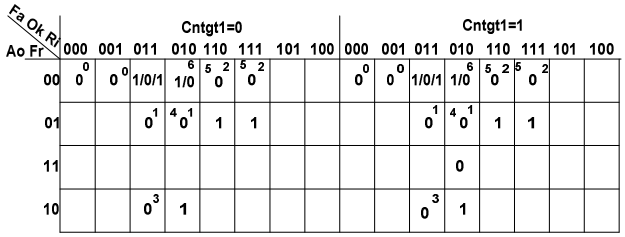

Fig. 13. Tabela verdade com conflitos da função f_{GR} do SCSI.

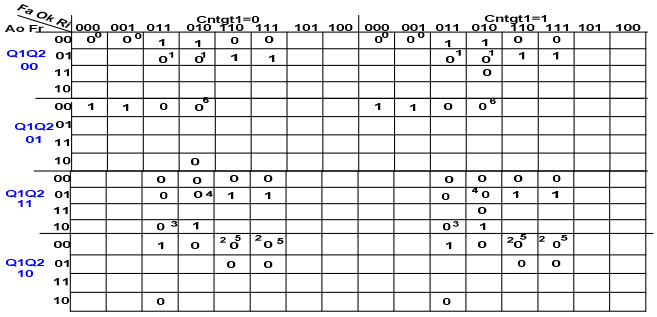

Fig. 14. Tabela verdade sem conflitos da função f_{GR} do SCSI.

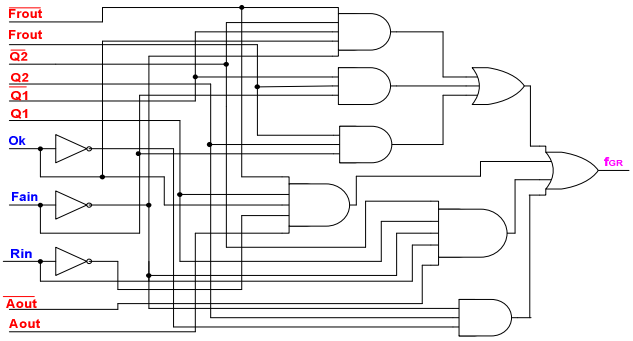

Fig. 15. Circuito lógico: função f_{GR} do SCSI.

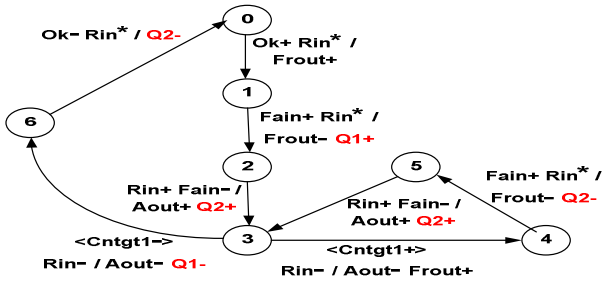

Fig. 16. Especificação XBM reformulada do SCSI.

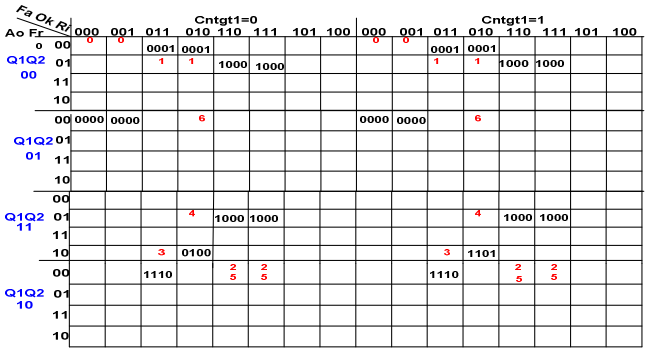

Fig. 17. Tabela de transição de estados do SCSI.

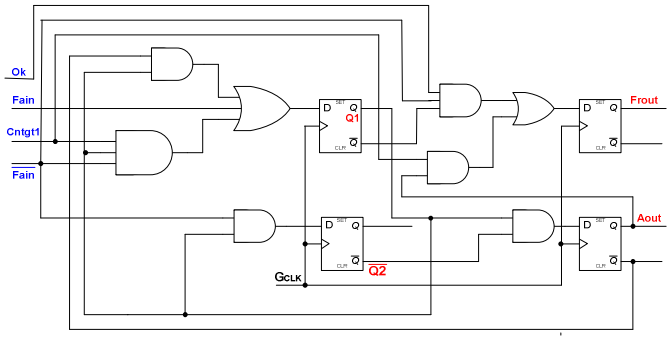

Fig. 18. Circuito lógico: MEFS de saída direta do SCSI.

VII. DISCUSSÃO & SIMULAÇÃO

O projeto digital que estamos tratando está voltado para PLDs. A nossa arquitetura é baseada em flip-flops D e não em latches, porque nestes dispositivos a lógica combinatória e o flip-flop D estão na mesma macro-célula, portanto se obtêm um melhor desempenho e redução de LUTs. Para aplicações que são descritas pela especificação BM a função f_{GR} pode ser obtida por uma cobertura lógica convencional, isto é usando a minimização lógica do paradigma síncrono. As MEFA_XBM_RL interagem com o ambiente no MFG. A nossa MEFA_XBM_RL foi simulada compilada na ferramenta ALTERA, software QUARTUS II, versão 9.1, família CYCLONE III, e dispositivo EP3C16F484C6. Figura 19 mostra as formas de onda livre de risco extraídas da simulação do benchmark SCSI e que foi sintetizado na seção VI. Figura 20 mostra os tempos de latência de cada transição de estado. O nosso circuito ocupou 16 LUTs e 6 flip-flops D, onde 2 flip-flops foram usados no elemento de atraso.

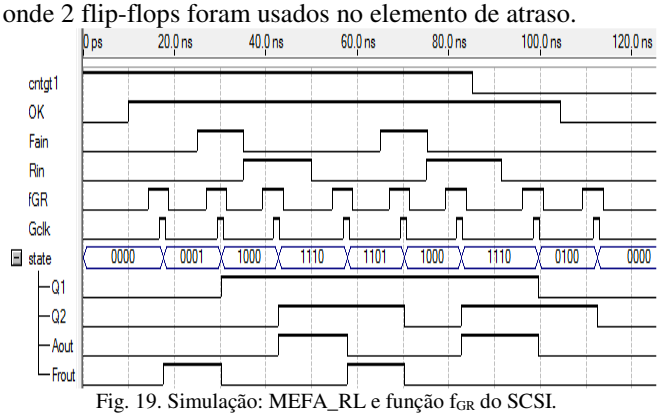

Fig. 19. Simulação: MEFA_RL e função f_{GR} do SCSI.

Transição de estado	Tempo de latência
0 → 1	7.635ns
1 → 2	5.323ns
2 → 3	7.663ns
3 → 4	7.91ns
4 → 2	5.664ns
3 → 6	7.91sn
6 → 0	8.078ns

Fig. 20. Resultados: tempo de latência das transições de estado.

VIII. CONCLUSÃO

Neste artigo introduzimos um novo método para síntese MEFA_XBM. Este método sintetiza no estilo de relógio local, isto permite reduzir os requisitos da lógica assíncrona, que é importante quando se pretende implementar em PLDs. A MEFA_XBM é toda sintetizada no paradigma síncrono, somente a função geradora de relógio requer um procedimento, que sintetiza a função livre de conflitos e de riso lógico. Trabalhos futuros propor um ambiente voltado para PLD para síntese automática de MEFA_XBM_RL e propor uma função geradora de relógio robusta que aceita qualquer cobertura lógica.

REFERÊNCIAS

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

[2] 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.

[3] P. P. Czapski and A. Sluzek, “A Survey on System-Level Techniques for Power Reduction in Field Programmable Gate Array (FPGA)-Based Devices”, The Second Int. Conf. on Sensor Technologies and Applications, pp.319-327, 2008.

[4] Altera Corporation, 2009, www.altera.com.

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

[6] L. Yuan, et al., “Research on the Problems of Satellite Borne FPGA Based Finite State Machine,” 2nd Int. Symposium on Systems and Control in Aerospace and Astronautics (ISSCAA), pp. 1-4, 2008.

[7] C. J., Myers, “Asynchronous Circuit Design”, Wiley & Sons, Inc., 2004, 2a edition.

[8] S. M. Nowick, “Automatic Synthesis of Burst-Mode Asynchronous Controller”, PhD thesis, Stanford University, 1993.

[9] K. Y. Yun, “Synthesis of Asynchronous Controller for Heterogeneous,” PhD thesis, Stanford University, 1994.

[10] J. Martin, “The Limitations to Delay Insensitive in Asynchronous Circuits,” 6th MIT Conference on Advanced Research in VLSI Processes, pp.263-277, 1990.

[11] T. –A. Chu, “Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications”, PhD thesis, Dept. of EECS, MIT, June 1987.

[12] D. J. Barnhart, et. al. “Radiation Hardening by Design of Asynchronous Logic for Hostile Environments”, IEEE Journal of Solid-State Circuits, vol. 44, No. 5, pp.1617-1628, May 2009.

[13] L. Plana e S. M. Nowick, “Architectural Optimization for Low-Power Nonpipelined Asynchronous systems”, IEEE Trans. on VLSI Systems, vol.6 no.1, pp.56-65, March 1998.

[14] E. Sutherland, “Micropipelines”, Communication of the ACM, vol. 32, No.6, pp.720-738, June 1989.

[15] S. F. Nielsen, et. al., “Behavioral Synthesis of Asynchronous Circuits Using Syntax Directed Translation as Backend”, IEEE Trans. on VLSI Systems, vol. 17, no. 2, pp. 248-261, February 2009.

[16] 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.

[17] K. Y. Yun e D. L. Dill, “A High-Performance Asynchronous SCSI Controller,” Proc. Int. Conf. Computer Design (ICCD), pp. 44-49, 1995.

[18] K. Y. Yun, et al., "The design and verification of a high-performance low-control-overhead asynchronous differential equation solver," IEEE Transactions on VLSI Systems, vol. 6, no 4, pp.643-655, Dec.1998.

[19] S. Rotem, et al., "RAPPID: An asynchronous instruction length decoder," in Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 60-70, April, 1999.

[20] F. Aghdasi, “Survey of Self-clocked Controllers,” Microelectronic Journal, vol. 26, pp.659-682, 1995.

[21] J. G. Bredeson, e P. T. Hulina, “Generation of a Clock Pulse for Asynchronous Sequential Machines to Elimination Critical Races,” IEEE Trans. Comput., C20, pp. 225-226, 1971.

[22] H. Y. H. Chuang e S. Das, “Synthesis of Multiple Input Chance Asynchronous Machines using Controlled Excitation and Flip-Flops,” IEEE Trans. Conput., C229120, pp. 1103-1109, 1973.

[23] S. H. Unger, “Self-synchronized Circuits and Nonfundamental Mode Operation,” IEEE Trans. Comput., C26(3), pp. 278-281, 1977.

[24] A. B. Hayes, “Store state Asynchronous Sequential Circuits,” IEEE Trans. Computer, C30(8), pp. 596-600, 1981.

[25] W. S. VanSchek e R. F. Tinder, “High Spped Externally Asynchronous/ Internally Clocked Systems, “IEEE Trans. on Computer, vol. 46, no. 7, pp.824-829, July 1997.

[26] J. Pasanen e B. Oelman, “Locally Clocked AFSMs with Dynamic Latch Implementation,” Proc. IEEE 6th Int. Conf. on Electronics, Circuits and Systems, pp.1643-1646, 1999.

[27] F. Aghdasi e A. Bhasin, “DMA Controller Design using Self-Clocked Methodology,” IEEE AFRICON, pp.443450, 2004.

[28] M. Kovac, “Autosynchronous Circuits Design Methodology,” IEEE Int. Conf. Radioelektronika, pp. 215-218, 2009.

[29] M. Tranchero e 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.

[30] N. Huot, et. al., “FPGA architecture for multi-style asynchronous logic,” Proc. of the Design, Automation and Test in Europe Conference and Exhibition, 2005.

[31] P. Sigel, G. De Micheli e D. Dill, “Decomposition methods for library binding of speed-independent,” Proc. Int. Conf. Computer-Aided Design, pp.558-565, 1994.

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

[33] S. M. Nowick, et al. “Practical Asynchronous Controller design,” Proc. Int. Conf. Computer Design, IEEE ICCD, pp. 341-345, 1992.

[34] S. M. Nowick et al, “The Design of a High Performance Cache Controller: A Case Study in Asynchronous Synthesis” *Integration, the VLSI Journal*, Vol. 15, no 3, pp. 241-262, October 1993.

[35] I. Pomeranz, e Cheng, K.-Ting, “STOIC: State Assignment Based on Output/Input Functions”. IEEE Trans. On CAD of Integrated Circuits and Systems. Vol.12, No.8, august, pp.1123-1131, 1993.

[36] S. Valeri, “Synthesis of Sequential Circuits on Programmable Logic Devices Based on New Models of Finite State Machines”, Proc. Symp. Digital Systems Design – Euromicro, pp.170-173, 2001.