Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

THIAGO FERREIRA DE PAIVA LEITE

Projeto de Circuitos Integrados Assíncronos QDI em Tecnologia FD-SOI 28nm

Campina Grande, Paraíba
Agosto de 2015

THIAGO FERREIRA DE PAIVA LEITE

Projeto de Circuitos Integrados Assíncronos QDI em Tecnologia FD-SOI 28nm

Relatório de Estágio Integrado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica

Orientador:

Professor Raimundo Carlos Silvério Freire, D. Sc.

Campina Grande, Paraíba
Agosto de 2015

AGRADECIMENTOS

Agradeço primeiramente à minha mãe, Selma, ao meu pai, Gilson, minha irmã Maíra e minha tia Ana pelo apoio e companheirismo em todos os momentos, fator essencial para superação das adversidades encontradas ao longo de minha carreira acadêmica.

Agradeço a toda minha família e amigos, que com todo carinho e apoio, não mediram esforços para eu chegar a esta etapa da minha vida.

Agradeço também aos colegas e orientadores do laboratório TIMA pelo apoio e acompanhamento durante o estágio.

“Algo só é impossível até que alguém duvide e resolva provar o contrário. ”

Albert Einstein.

RESUMO

A redução contínua das dimensões dos transistores nos circuitos integrados tem provocado um aumento da vulnerabilidade de circuitos digitais à variações de tensão, temperatura e fabricação. Para reduzir esses impactos, tecnologias de fabricação como a *Fully Depleted Silicon on Insulator* e o paradigma assíncrono de concepção de circuitos digitais tem sido apontado como soluções robustas à essas variações. Neste estágio propõe-se o desenvolvimento de um fluxo de projeto de circuitos integrados assíncronos QDI em tecnologia FD-SOI. A concepção do leiaute de um microprocessador ARM7 simplificado síncrono foi proposta como tarefa inicial, objetivando-se uma melhor compreensão do problema e das ferramentas disponíveis para sua solução. Por fim, foram feitas simulações e síntese de circuitos *half buffer* duplo e somador completo assíncronos.

Palavras-chave: Circuitos Assíncronos, FD-SOI, QDI, Circuitos Integrados.

ABSTRACT

The downscale of semiconductors' feature size has led to increases in vulnerability of digital circuitry to process, voltage and temperature variations. To reduce these impacts, manufacturing technologies such as FD-SOI and the asynchronous design approach have been pointed as robust solutions to this problem. This internship proposes an asynchronous QDI IC design flow using the FD-SOI technology. The design of the layout of a synchronous simplified ARM7 microprocessor has been proposed as an initial task aiming to better understand the problem and the tools available to solve it. At the end, simulations and logical synthesis of asynchronous QDI double half buffer circuit and full adder were performed.

Keywords: Asynchronous circuits, FD-SOI technology, QDI systems, Integrated Circuits.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

TIMA – *Laboratoire des Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés* (Laboratório de Técnicas de Informática e de Microeletônica para Arquiteturas de Sistemas Integrados)

CNRS – *Centre National de la Recherche Scientifique* (Centro Nacional de Pesquisa Cinetífica)

UJF – *Université Joseph Fourier*

INPG – *Institut National Polytechnique de Grenoble* (Instituto Nacional Politécnico de Grenoble)

AMS – *Analog Mixed Signals* (Sinais Analógicos e Mistos)

EDA – *Electronic Design Automation*

CAD – *Computer Aided Design* (Projeto Auxiliado por Computador)

FET – *Field Effect Transistor* (Transistor de Efeito de Campo)

FD-SOI – *Fully Depleted Silicon on Insulator*

BOX – *Buried Oxide* (Óxido Enterrado)

UTBB – *Ultra Thin Body and BOX* (Corpo e Óxido Enterrado Ultrafinos)

RDF – *Random Dopant Fluctuation* (Flutuação Aleatória de Dopantes)

BB – *Body Biasing* (Polarização de Corpo)

PVT – *Process, Voltage, Temperature* (Fabricação, Tensão, Temperatura)

QDI – *Quasi Delay Insensitive* (Quase Insensível ao Atraso)

DI – *Delay Insensitive* (Insensível ao Atraso)

HDL – *Hardware Description Language* (Linguagem de Descrição de Hardware)

P&R – *Place and Route* (Alocação e Roteamento)

RTL – *Register Transfer Level* (Nível de Transferência entre Registradores)

TCL – *Tool Command Language* (Linguagem de Comando de Ferramenta)

VHDL – *Very High Speed Integrated Circuit Hardware Description Language*

RISC – Re*duced Instruction Set Computer* (Computador com Jogo de Instruções Reduzido)

LEF – *Library Exchange Format* (Formato de Troca de Biblioteca)

SoC – *System on Chip* (Sistema em Chip)

DIMS – *Delay Insensitive Minterms Synthesis*

SUMÁRIO

1 INTRODUÇÃO

A evolução da eletrônica tem possibilitado criações antes inimagináveis. Dispositivos eletrônicos cada vez mais compactos e computacionalmente poderosos têm sido desenvolvidos e possibilitado feitos inovadores nos mais variados ramos do conhecimento. Essa constante redução nas dimensões dos circuitos eletrônicos deu origem aos termos microeletrônica e nanoeletrônica, hoje comumente aplicados.

Historicamente, a evolução dos circuitos integrados tem seguido a Lei de Moore, que prevê um aumento de 100% do número de transistores nos *chips* a cada 18 meses a um custo fixo. A confirmação dessa "profecia", proferida pelo então presidente da Intel, Gordon E. Moore, tem sido possível graças às constantes pesquisas na área de microeletrônica e consequente desenvolvimento de tecnologias de fabricação cada vez mais eficiente e que possibilitam integração de circuitos em escalas cada vez menores.

Contudo, a continuação da lei de Moore tem sido cada vez mais difícil de ser seguida à medida que as novas tecnologias de fabricação apresentam dimensões próximas às dos átomos. Efeitos físicos antes negligenciados, como o aumento significativo das perdas elétricas por corrente de fuga, a aceleração do processo de envelhecimento dos circuitos e a maior variabilidade dos circuitos devido ao processo de fabricação, tem impactado o desempenho dos transistores.

Uma das possíveis soluções para a redução dos impactos negativos da miniaturização dos transistores é o uso da tecnologia de fabricação *Ultra Thin Body & BOX Fully-Depleted Silicon On Insulator* (UTBB FD-SOI), que será abordada no capítulo seguinte. A fabricante ST Microelectronics (2014a, 2014b) utiliza a vantagem do preço competitivo para justificar a tecnologia FD-SOI como uma possibilidade para dar-se continuidade à tendência de evolução prevista pela lei de Moore, já que tecnologias concorrentes para os mais recentes nodos possuem custo de fabricação maiores (JONES, 2012).

Alternativamente à lei de Moore, uma outra forma de analisar a tendência de desenvolvimento dos dispositivos microeletrônicos foi proposta por Koomey et al. (2011). Nesse estudo foi feita a análise da tendência de desenvolvimento dos dispositivos eletrônicos sob o ponto de vista do consumo. Tal estudo reflete o atual interesse por

sistemas com menor consumo de energia, condição essencial para implementação de portabilidade em sistemas microeletrônicos.

Uma proposta interessante para a implementação de dispositivos com menor consumo de energia e também a redução de problemas oriundos da redução do tamanho dos transistores é o paradigma assíncrono de projeto de circuitos (SPARSØ, 2006). Diferentemente dos circuitos síncronos, que seguem o ritmo do sinal de relógio para sincronização, nos circuitos assíncronos o próprio fluxo de dados sincroniza a informação entre os blocos do sistema (BASTOS, 2010; BEREEL, 2010; RENAUDIN, 2000; SPARSØ, 2006). A ausência do sinal de relógio reduz o consumo e a interferências, induzida pelas linhas que carregam esse sinal. O paradigma Assíncrono será tratado com maiores detalhes nos capítulos subsequentes deste trabalho.

Uma fusão do uso de dispositivos feitos em tecnologia UTBB FD-SOI juntamente com a aplicação de uma lógica de concepção assíncrona, tal como proposto por Hamon et al. (2013), possibilita a redução de consumo de energia mantendo-se um bom nível de desempenho do circuito. Tal capacidade torna-se possível graças a algumas formas de polarização passíveis de serem implementadas com a tecnologia FD-SOI e as peculiaridades dos protocolos de sincronização implementados pelos circuitos assíncronos.

Neste estágio, realizado no laboratório TIMA, após estudo da tecnologia FD-SOI e dos circuitos assíncronos, um fluxo de projeto para circuitos integrados assíncronos QDI foi proposto e passou por alguns testes iniciais. Esse projeto servirá de base para vários outros trabalhos do grupo de pesquisa no qual foi desenvolvido, a saber, o grupo CDSI, bem como de outros grupos de pesquisa do laboratório TIMA.

1.1 OBJETIVO DO PROJETO

O objetivo deste estágio foi estudar a tecnologia de fabricação FD-SOI e o paradigma assíncrono de projeto de circuitos integrados para propor um fluxo de desenvolvimento de CIs assíncronos para aplicações com baixo consumo de energia.

As novas formas de polarização propostas pela tecnologia FD-SOI aliadas aos circuitos assíncronos oferecem uma interessante solução para aplicações que exigem baixo consumo de energia, sem que haja impactos significativos ao desempenho.

Este trabalho foi desenvolvido no contexto do projeto europeu THINGS2DO, que visa criar um ambiente favorável ao desenvolvimento de aplicações em tecnologia FD-SOI, escolhida estrategicamente para tirar proveito de alguns pontos fortes da Indústria de Semicondutores da Europa (Things2Do, 2013).

1.2 APRESENTAÇÃO TIMA – LABORATÓRIO DE TÉCNICAS DE INFORMÁTICA E DE MICROELETRÔNICA PARA ARQUITETURAS DE SISTEMAS INTEGRADOS

1.2.1 APRESENTAÇÃO GERAL

O TIMA é um laboratório de pesquisa compartilhado entre profissionais de três instituições francesas: o CNRS (*Centre National de la Recherche Scientifique*) e as universidades Grenoble-INP (*Institut National Polytechnique de Grenoble*) e UJF (*Université Joseph Fourier*). Grande parte das pesquisas do TIMA são realizadas no contexto de projetos de cooperação com parceiros industriais e acadêmicos, apoiados por subsídios regionais, nacionais ou internacionais.

Os principais temas de pesquisa do laboratório TIMA são: especificação, projeto, verificação e teste de sistemas integrados, tanto analógicos quanto digitais, para processadores com vários núcleos e SoCs (*Systems-on-Chips*).

Mais especificamente, o laboratório é dividido em quatro grupos de pesquisa (AMfoRS, CDSI, RIS, RMS e SLS), cada qual com uma linha de pesquisa bem definida.

- AMfoRS – Grupo de arquiteturas e métodos para sistemas resilientes. Trabalha principalmente com especificação multi-nível e verificação de hardware/software *on-chip*; modelagem em nível de sistema; análise e testes de confiabilidade de sistemas; e segurança de sistemas integrados.
- CDSI – Grupo de projeto de dispositivos, circuitos e sistemas integrados. Trabalha com sistemas assíncronos, com ênfase em aplicações para segurança; amostragem e processamento não uniformes; e desenvolvimento de tecnologias para sistemas integrados em escala micro e nanométricas.

- RIS – Grupo de sistemas integrados robustos. Trabalha com arquiteturas auto adaptativas e tolerantes a falha; gerenciamento de energia; arquiteturas com teste e reparação autônomos; arquiteturas paralelas robustas em *chip* único; avaliação da robustez; e arquiteturas para aplicações espaciais.
- RMS – Grupo de sistemas de sinal misto de confiança. Trabalha com projeto e teste de circuitos analógico, de sinal misto e de rádio frequência; técnicas de avaliação de sistemas AMS e RF; calibração de dispositivos de RF; e modelagem em alto nível de sistemas heterogêneos e multi-físicos.
- SLS – Grupo de síntese em nível de sistemas. Tem como foco de pesquisa arquiteturas paralelas e reconfiguráveis; *softwares* para sistemas integrados; e síntese, geração e simulação de sistemas integrados digitais.

O estágio relatado neste documento foi realizado no grupo CDSI do laboratório TIMA.

1.2.2 O PROJETO THINGS2DO

O projeto europeu THINGS2DO é focado na construção de um "ecossistema" para a tecnologia FD-SOI, que foi estrategicamente escolhida para tirar proveito de alguns pontos fortes da indústria europeia de semicondutores.

O ecossistema proposto baseia-se em três pilares:

- Ferramentas EDA – Automação é a base para execução de projetos complexos e a chave para a portabilidade. A indústria de EDA na Europa conta com implantações de grandes empresas norte-americanas, bem como um "ecossistema" de pequenas e médias empresas europeias ativas neste domínio.
- Produção de IPs – Disponibilidade de blocos de construção pré-concebidos é uma necessidade absoluta para qualquer tecnologia emergente. Concomitantemente, o estabelecimento de um ambiente de implementação é uma parte vital ao desenvolvimento de SoCs complexos.

- Serviços de integração entre IPs e ferramentas EDA. Há uma variedade de empresas de pequeno e médio porte na Europa focadas neste tema, fornecendo ofertas de serviços para trazer o potencial inovador da tecnologia FD-SOI para os sistemas e aplicações finais.

Os objetivos do projeto THINGS2DO estão voltadas para a criação e valorização dos três pilares supracitados com metas específicas para aplicações finais europeias. As áreas identificadas para a aplicação deste projeto são: biomédica, aeronáutica e espacial, entre outros.

Como esse projeto impactará significativamente aplicações finais, uma área em que a Europa tem um notável destaque, a industrial européia como um todo beneficiar-se-á dos resultados do projeto THINGS2DO.

1.3 ORGANIZAÇÃO DO RELATÓRIO

Este documento está organizado da seguinte forma: na seção 2 o embasamento teórico necessário à compreensão das atividades desenvolvidas no estágio será abordado em detalhes. A seção 3 descreverá cada atividade realizada, a seção subsequente mostrará os resultados obtidos em cada uma das atividades. Por fim, a seção 5 tratará das conclusões deste trabalho.

2 CONTEXTO DO PROJETO

2.1 A TECNOLOGIA FD-SOI

A tecnologia SOI (*Silicon On Insulator*) de fabricação de semicondutores, industrializada desde o final do século XX, usa um substrato silício-isolante-silício (uma camada de óxido de silício isolante é inserida entre duas redes monocristalinas de átomos de silício) ao invés do tradicional substrato de silício dos circuitos integrados convencionais, comumente chamados tipo *bulk* (CELLER; CRISTOLOVENEAUNU, 2003).

A tecnologia FDSOI (*Fully-Depleted Silicon On Insulator*), por sua vez, faz uso de *wafers* SOI aperfeiçoados a fim de criar transistores de efeito de campo (FETs) que possuam canais de condução completamente depletados. Ser completamente depletado significa ter a camada de silício sobre isolante com espessura igual à profundidade da região de depleção (SINGH; SAXENA; RASTOGI, 2011).

A Figura 1 e a Figura 2 comparam a tecnologia SOI com FD-SOI e FD-SOI com a tradicional tecnologia tipo bulk, respectivamente.

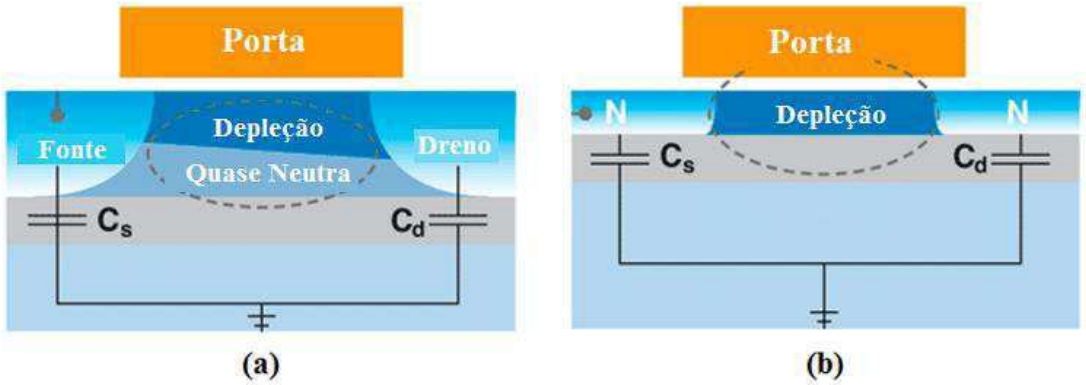

Figura 1. Corte transversal de transistores em tecnologia SOI (a) e FD-SOI (b). (MENTOR GRAPHICS, 2008. Modificada)

Na Figura 1 pode-se identificar o canal de condução formado nos dois transistores (SOI a esquerda e FD-SOI a direita) na camada superior de silício – região em azul escuro localizada entre a fonte (*source*) e o dreno (*drain*).

Na Figura 1a) observa-se um transistor com canal parcialmente depletado, ou seja, neste há condução apenas em parte do silício localizado entre a fonte e o dreno. Em outras

palavras, o campo elétrico formado na região do canal não é suficientemente forte para que haja uma inversão completa do tipo de portadores majoritário nesta área. Tal fato pode ser explicado pela forma como se dá a fabricação de semicondutores SOI. Nos primórdios do processo de fabricação dessa tecnologia, não era possível obter-se espessuras de camadas de cristais de silício e de óxido enterrado suficientemente finas para a formação de uma região de depleção completa.

Com o desenvolvimento de um novo processo de fabricação, chamado de *SmartCut*, finas espessuras de silício (entre 5 e 50 nm) são factíveis (HARS, 2012). Isso permite a formação de um campo elétrico mais forte em toda a região inter-óxidos, o que, por sua vez, permite a obtenção de canais de transistor completamente depletados, o que deu origem a tecnologia FD-SOI.

A Figura 1b) exemplifica a tecnologia supracitada. Nela pode-se observar que o canal de condução ocupa toda a região inter óxidos (região entre a fonte e o dreno), ou seja, obtém-se uma depleção completa dessa região, daí o nome *Fully-Depleted*.

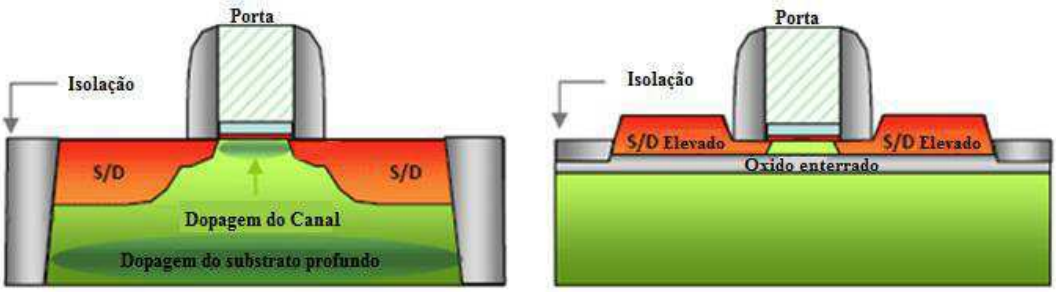

Figura 2. Seção transversal de transistor em Tecnologia tipo bulk, à esquerda, e em tecnologia FD-SOI, à direita. (THINGS2DO, 2013)

Na Figura 2 observam-se as diferenças entre a convencional tecnologia tipo bulk e a tecnologia FD-SOI. As diferenças são basicamente a inserção de uma camada fina de óxido enterrado (BOX) no transistor FD-SOI; no transistor tipo bulk a camada mais profunda do substrato passa por um processo de dopagem (*Deep Well Implante*); e as camadas de silício que formar a fonte/dreno são elevadas nos transistores produzidos em tecnologia FD-SOI.

2.1.1 ÓXIDO ENTERRADO

Na tecnologia FD-SOI 28nm, um transistor possui um canal de espessura máxima de aproximadamente 7nm. O canal do transistor e o substrato são separados por uma pequena camada de óxido enterrado (do inglês, *Buried Oxide*), cuja espessura é de aproximadamente 25nm. Essa camada recebe o nome, em alguns trabalhos da literatura científica, de *Ultra Thin Body & Buried Oxide* (UTBB).

Em alguns trabalhos da literatura especializada, o substrato na tecnologia FD-SOI, região abaixo da camada de óxido enterrado, é também chamado de Back-Plane (BP), ou, em português, plano de fundo (HAMON, BEIGNE, 2003). Desta forma, é possível diferenciá-lo do substrato de transistores fabricados em tecnologia bulk, nos quais o substrato é denominado de *Body*. A presença de uma camada isolante entre o canal e o plano de fundo permite fazer 3 diferentes constatações:

- Há uma certa dificuldade na circulação de corrente elétrica entre o plano de fundo e a Fonte/Dreno, devido à presença do óxido enterrado (CRISTOLOVENEAUNU, LI, 1995; CELLER, CRISTOLOVENEAUNU, 2003.);
- Não há necessidade de dopagem da camada de silício onde se forma o canal do transistor, o que reduz o fenômeno de *Random Dopant Fluctuation* (RDF), variação da concentração de impurezas dopantes no silício (HAMON, BEIGNE, 2013; ST MICROELECTRONICS 2013; NAZAROV et al., 2011). O ajuste da tensão de limiar, em contrapartida, é controlado pelo metal da porta (NAZAROV et al., 2011);
- O tipo de impureza dopante no plano de fundo não precisa ser oposto ao utilizado na dopagem do silício das regiões que formam a fonte e o dreno. Pode-se obter, portanto, uma inversão do estilo de dopagem usado em FETs do tipo bulk. (HAMON, BEIGNE, 2003; JACQUET 2013).

Como consequência direta da primeira observação feita, há uma redução na corrente de fuga entre fonte/dreno e o plano de fundo. Desta maneira, é possível dizer que o consumo do dispositivo é menor, devido à menor quantidade de perdas por correntes de fuga entre substrato, canal, Fonte e Dreno.

Random Dopant Fluctuations são as variações na concentração de impurezas implantadas, intrínsecas ao processo de fabricação dos transistores. Esse fenômeno é apontado como um dos principais fatores que provocam variações da tensão de limiar de

um transistor para outro. Como a quantidade total de átomos que constituem o canal de um transistor vem tornando-se cada vez menor, à medida que novas tecnologias vem sendo desenvolvidas, uma pequena variação na concentração de impurezas, entre um transistor e outro, tem um impacto significativo na performance do dispositivo.

Como o silício da região que forma o canal dos transistores FD-SOI não é dopado, efeitos de RDF são eliminados (HAMON, BEIGNE, 2003).

A respeito da inversão de dopagem comentada na terceira observação, pode-se afirmar que tal capacidade possibilita mudanças na amplitude da tensão de alimentação dos transistores FD-SOI. Esta tensão influencia a Tensão de Limiar, a qual será tratada com maiores detalhes na subseção seguinte.

2.1.2 TENSÃO DE LIMIAR (V_{TH})

A Tensão de Limiar, do inglês, *Threshold Voltage*, pode ser definida como a menor diferença de potencial elétrico entre fonte e dreno necessária para a formação de um canal de condução entre eles. O fator de substrato (*body-effect coeficient*), por sua vez, é uma medida do grau de influência da polarização do substrato do transistor (*Body Biasing*, BB) na sua Tensão de Limiar (RABAEY, 2003).

Para compreender melhor tal influência e sua relação com outras variáveis, pode-se observar a equação que define a Tensão de Limiar, obtida a partir do modelo de Shichman-Hodges:

$$V_T = V_{T0} + \gamma(\sqrt{|-2\emptyset_F + V_{SB}|} - \sqrt{|-2\emptyset_F|}) \quad (1)$$

Na qual:

V_T é a Tensão de Limiar;

V_{SB} é a Diferença de Potencial entre o Contato de Substrato (*Body Contact*) e o Contato da Fonte (*Source Contact*);

V_{T0} é Tensão de Limiar para $V_{SB} = 0$. Normalmente dependente do processo de fabricação;

$\emptyset_F$ é o Potencial de Fermi; e

γ é o chamado coeficiente de corpo, fator de substrato ou *body-effect coefficient*.

Observar-se que, para transistores em tecnologia convencional, a tensão de limiar é positiva para transistores MOS do tipo N (NMOS) e negativas para transistores do tipo

P (PMOS) (RABAEY, 2003). Observa-se também que quanto maior for o fator de substrato (γ), maior será a influência da tensão de substrato na Tensão de Limiar.

Um estudo de Cattaneo (2009) sobre o comportamento de transistores em tecnologia SOI mostrou que o efeito de corpo das tecnologias SOI é menor do que em outras tecnologias devido principalmente à espessura do óxido enterrado. Porém, transistores em tecnologia SOI com óxido enterrado ultrafino possuem efeito de corpo mais expressivo do que outros tipos de transistores de tecnologias SOI precedentes (HAMON; BEIGNE, 2003; OHTOU; SARAYA; HIRAMOTO, 2008; NOEL et al., 2011).

O impacto da espessura do óxido enterrado fica evidente no estudo comparativo, feito por Liu et al. (2011), entre o efeito de corpo para BOXs com 10 e 25nm, Figura 3 (a) e Figura 3 (b), respectivamente.

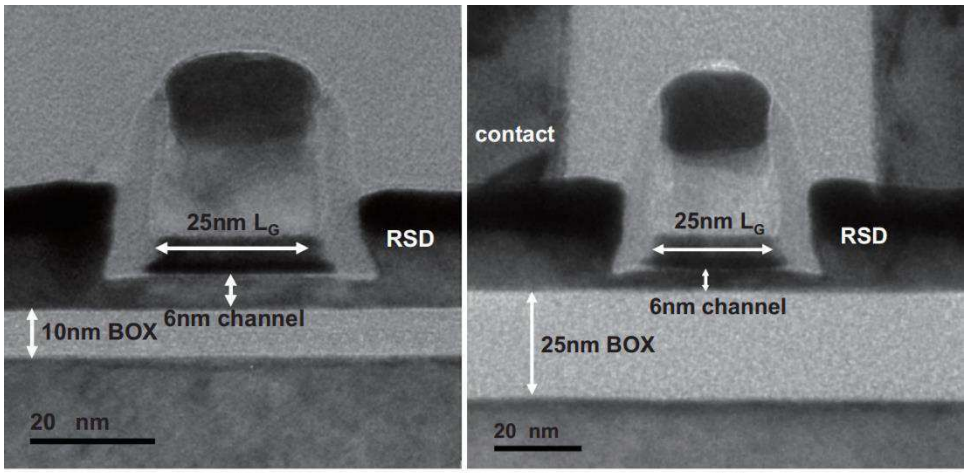

Figura 3. Transistores FD-SOI com óxido enterrado de 10nm (a) e 25nm (b). (LIU et al., 2011)

As medições de tensão de limiar obtidas no referido estudo foram:

- 120 mV/V, para um BOX de 10 nm;
- 60 mV/V, para um BOX de 25 nm.

Analisando-se o comportamento de um transistor com relação à tensão de limiar, pode-se dizer que quanto menor for a magnitude desta, mais fácil será levar o transistor a um estado de condução, visto que tensões de porta menores provocarão a formação de um canal de condução. Com isso, pode-se afirmar que nesse caso os transistores tornam-

se mais rápidos. Porém, uma tensão de limiar baixa resulta em mais perdas por corrente de fuga.

Por outro lado, o aumento da tensão de limiar faz com que a barreira de potencial a ser vencida para formação de um canal de condução no transistor seja maior que no caso anterior. Sendo assim, torna-se mais difícil que o transistor entre em estado de condução, o que acaba por reduzir as perdas por corrente de fuga. Todavia, neste caso percebe-se que os transistores se tornam mais lentos.

Vale salientar que a inversão do BP, comentada na subseção anterior, permite uma configuração antes impossível em tecnologias MOS anteriores. Um canal com mesmo tipo de dopante que a fonte e o dreno seria um curto circuito e não um transistor. Essa inversão do BP é chamada de *Flip-Well* (FW), ou Poço Invertido. Um comparativo entre esse tipo de BP e o convencional, tal como na tecnologia bulk, é apresentado na Figura 4. Nessa figura também pode-se observar as margens de polarização para a tecnologia FD-SOI 28nm.

Existe ainda um terceiro tipo de poço, o *Single-Well*. Nos transistores com esse tipo de BP a mesma impureza dopante é usada para formar os substratos dos transistores PMOS e NMOS. Com um poço único, aumentar a tensão de limiar para um tipo de transistor, reduziria a tensão de limiar para o seu complementar. Esta configuração de BP não será detalhada neste trabalho.

Observa-se do lado direito da Figura 4 as amplitudes das tensões de polarização dos substratos, passíveis de serem aplicadas a cada tipo de BP. Essas amplitudes são limitadas pela corrente de polarização direta e reversa da junção PN, formada entre os substratos dos transistores PMOS e NMOS. Esses limites de tensão de polarização sã válidos para a tecnologia FD-SOI 28nm.

Vale salientar que o esquema de polarização do BP pode ser manipulado para tornar o circuito mais lento ou mais rápido, de acordo com a necessidade e com a limitação da configuração de poço escolhida (convencional ou invertida).

Com um esquema de polarização FBB (*Foward Body Biasing*), há uma redução na tensão de limiar, o que torna os transistores mais rápidos, a custo de um aumento na corrente de fuga, conforme comentado anteriormente. Adotar esse tipo de polarização em transistores com substrato convencional é permitido até cerca de 300 mV, conforme estudos feitos por Flatresse e Jacquet (2013). Contudo, conforme visto na Figura 4, esse esquema de polarização é mais adequado para transistores com substrato invertido, dado

que nesse caso há uma faixa de tensão maior que polariza inversamente a junção PN formada entre os poços dos transistores PMOS e NMOS.

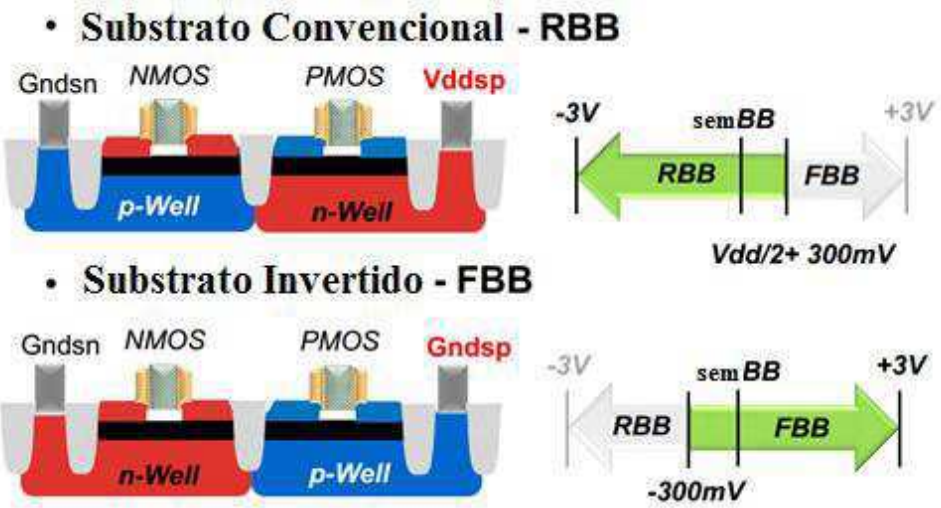

Figura 4. FD-SOI com poço convencional, acima, poço invertido, abaixo, e suas respectivas margens de tensão de polarização do plano de fundo. (FLATRESSE, 2013)

Analogamente com esquema de polarização RBB (*Reverse Body Biasing*), há um aumento da tensão de limiar, o que torna os transistores mais lentos, porém com menor corrente de fuga. Esse esquema de polarização é mais adequado para transistores com substrato convencional, pois nesse caso há uma faixa de tensão maior que polariza inversamente a junção PN formada entre os poços dos transistores PMOS e NMOS, conforme visto na Figura 4.

É possível, também, implementar esquemas de polarização que hora deixem o circuito mais rápido, para obter-se um melhor desempenho enquanto o circuito estiver ativo, hora deixem o circuito mais lento, com a finalidade de reduzir as perdas de energia, enquanto o circuito não estiver sendo utilizado. Tal estratégia é sugerida pela ST Microelectronics e apontada como uma das vantagens da tecnologia FD-SOI.

Sinais de ativação/desativação são abundantes em alguns tipos de lógica de concepção de circuitos assíncronos (HAMON, BEIGNE, 2003), que serão apresentadas com mais detalhes na próxima seção.

2.1.3 POLYBIASING

Uma outra característica peculiar da tecnologia FD-SOI é o *polybiasing*. Essa técnica consiste em alterar, em tempo de projeto, o comprimento do canal do transistor,

ou seja, o poli silício de porta. A Figura 5 mostra como o polybiasing altera a constituição física dos transistores.

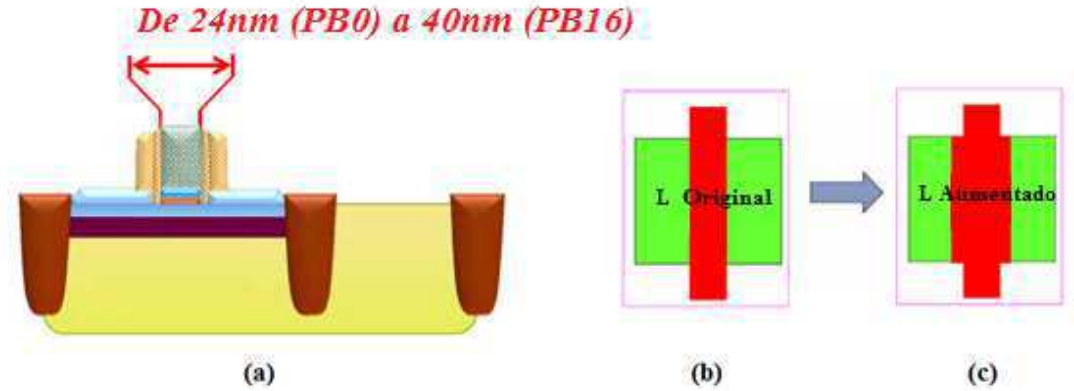

Figura 5. Corte transversal de um transistor FD-SOI (a). Variação do comprimento do canal em função do *polybiasing* (b) e (c). (FLATRESSE, 2013)

Na Figura 5b) observa-se um exemplo de vista do transistor da Figura 5a), obtida com auxílio de uma ferramenta CAD, na qual não há *polybiasing*, ou seja, o transistor retratado tem comprimento mínimo de canal. As regiões em verde são a fonte e o dreno e a parte em vermelho representa o poli silício de porta. Na Figura 5c), é mostrado como é feita a variação do comprimento do canal do transistor, em relação a seu tamanho original retratado na Figura 5b), para que se obtenha o efeito de *polybiasing* desejado. Observa-se que o desenho do comprimento de porta maior é feito apenas na região do canal do transistor.

A variação no comprimento do canal resulta na alteração da Tensão de Limiar. A seguinte analise pode ser feita: quanto maior for o comprimento do canal, maior é o valor desta tensão, e menores as perdas por corrente de fuga *sub-threshold*. Alterar o comprimento do canal de um transistor faz com que este se comporte como um transistor fabricado em tecnologias com comprimentos de canal maiores. Sendo assim, uma das vantagens do *polybiasing* que pode ser apontada é que tecnologias com nodos menores geralmente apresentam melhor qualidade de fabricação que as tecnologias de nodos maiores.

Outra vantagem que pode ser apontada é que a variação dos valores de V_{th} com o uso de *polybiasing* permite a fabricação de circuitos compatíveis com arquiteturas multi-V_{th}. Além disso, conforme escolhas de projeto, pode-se usar diferentes valores de polybiasing para obter-se desempenhos e consumos diferentes.

O design kit da tecnologia FD-SOI 28nm provido pela ST Microelectronics apresenta uma biblioteca *stardard cells* com 4 opções de comprimento de canal, conforme apresentado na Tabela 1. Cada um representando, portanto, uma opção de polybiasing que varia de 0 a 16nm (P0 à P16). Seus comprimentos efetivos, no entanto, variam de 24 a 40nm.

Tabela 1. *Polybiasing*, comprimentos de projeto e comprimentos efetivos disponíveis para biblioteca standard cells FD-SOI 28nm.

Polybiasing	**Comprimento de Projeto (nm)**	**Comprimento Efetivo (nm)**
P0	30	24
P4	34	28
P10	40	34
P16	46	40

Nota-se que, para a tecnologia de 28nm, o canal do transistor possui um comprimento mínimo de 30nm. Para transistores com esse comprimento de canal, o comprimento efetivo – que é a distância real entre os terminais de Fonte e Dreno é de 24nm. A diferença entre as duas distâncias deve-se à difusão dos dopantes por baixo do óxido de porta.

A existência de diferentes bases de medição para o comprimento de projeto (30nm mínimo para a tecnologia FD-SOI 28nm), o comprimento de referência da tecnologia (28nm) e o comprimento efetivo (24nm para a tecnologia FD-SOI 28nm) são resultado de decisões comerciais.

2.2 CIRCUITOS ASSÍNCRONOS

Em sistemas síncronos, que têm sido amplamente estudados e utilizados desde os primórdios da eletrônica digital, um sinal de relógio (*clock*) rege a propagação de dados pelo circuito, controlando o momento de memorização de informações em registradores. Uma ilustração do funcionamento geral de circuitos síncronos pode ser observada na Figura 6.

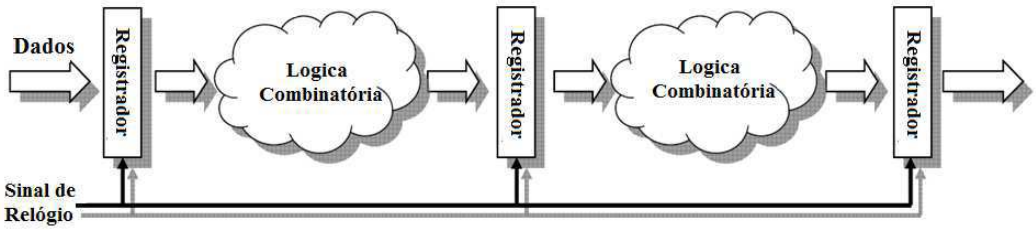

Figura 6. Estrutura básica dos sistemas síncronos (FRAGOSO, 2005. Modificada).

Sistemas assíncronos, por outro lado, não usam sinal de relógio para sincronização de dados, daí a razão para tais sistemas também serem conhecidos, na literatura científica, como circuitos de tempo próprio ou auto temporizados, no inglês *self-timed circuits*. Ao invés disso protocolos de transferência de dados ativam ou não a alteração de células de memória, controlando, assim, a propagação de informações ao longo dos circuitos assíncronos. (RENAUDIN; RIGAUD, 2000; SPARSØ, 2006; BEEREL, OZDAG, FERRETTI, 2010). Um exemplo de um módulo componente de um sistema assíncrono pode ser visto na Figura 7.

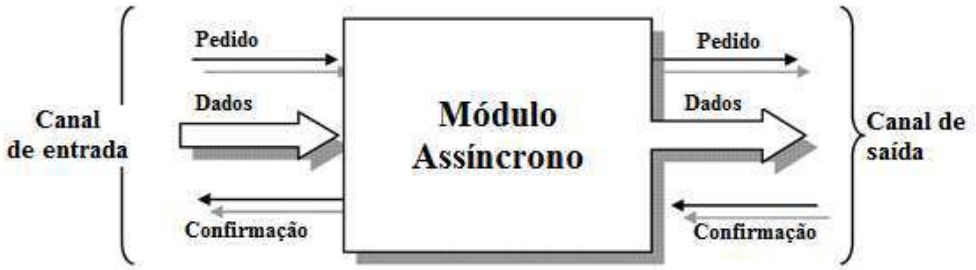

Figura 7. Componente de um Circuito Assíncrono (FRAGOSO, 2005. Modificada).

Percebe-se claramente a ausência de um sinal de relógio no módulo da Figura 7. Outra observação a respeito da referida figura é que os sinais de pedido e confirmação representam um tipo de protocolo de transferência de dados, tal como comentado no parágrafo anterior, que será tratado com mais detalhes posteriormente. Ainda a respeito dessa figura, percebe-se que o protocolo de comunicação é implementado tanto no canal de entrada como no canal de saída, para garantir que não haverá perda de dados ao longo do circuito.

Como os sinais de relógio não são necessários em circuitos assíncronos, a dissipação de energia nas linhas que formam a árvore de relógio (*clock tree*) é eliminada, ou seja, há uma redução no consumo de energia do circuito. Experimentos feitos por Berkel et al. (1994), demonstraram uma redução no consumo de energia de 80% em um determinado circuito assíncrono quando comparado ao seu equivalente síncrono.

Concomitantemente, a não existência de sinal de relógio resulta em menor Interferência Eletromagnética (EMI, do inglês: *Electromagnetic Interference*) entre linhas de relógio e demais linhas (BEEREL; OZDAG; FERRETTI, 2010; SPARSØ, 2006).

Sabendo-se que algumas funções lógicas necessitam de tempos de execução menores que um ciclo de relógio para serem concluídas, um determinado circuito assíncrono pode ser mais rápido que seu equivalente síncrono. A velocidade de operação em circuitos assíncronos é limitada, portanto, apenas pelo atraso de propagação da informação (BEEREL; OZDAG; FERRETTI, 2010).

Uma outra vantagem observada em circuitos assíncronos é que estes são mais robustos a variações PVT (do inglês, *Process, Voltage, Temperature*), ou seja, seu comportamento tem menor dependência de variabilidade nas características elétricas (Tensão de Limiar, velocidade etc) decorrentes de variações no processo de fabricação ou nas condições de temperatura e tensão de alimentação. Variações PVT são cada vez mais vez mais significativas em tecnologias mais atuais (BEEREL; OZDAG; FERRETTI, 2010).

Contudo, circuitos assíncronos podem necessitar mais área para a concepção de sua lógica. Isso se deve principalmente devido à necessidade de circuitos de controle (BEEREL; OZDAG; FERRETTI, 2010). Sendo maior a área, maiores serão as superfícies de contato por onde circulam as correntes de fuga. Deste modo, uma maior área pode resultar em um aumento do consumo por perdas, principalmente estáticas (SPARSØ, 2006).

Um outro empecilho à uma maior difusão da lógica de concepção assíncrona é a escassez de ferramentas CAD voltadas para esse tipo de circuitos, a maioria das ferramentas comerciais existentes visam a concepção de circuitos síncronos. A ausência de ferramentas de teste e vetores de testes também é apontada por SPARSØ (2006) como uma problemática clássica da concepção assíncrona.

As etapas básicas para a concepção de circuitos assíncronos são: definição do modelo de atrasos do circuito, definição do protocolo de comunicação e forma de

codificação dos dados e, finalmente, a escolha dos componentes básicos (MOREIRA; GUAZZELLI; CALAZANS, 2012).

Definir o modelo de atraso significa determinar a forma como as informações fluirão no circuito em função do tempo. O modelo mais simples, e também o mais otimista, assume que o atraso das informações no circuito será fixo, nesse caso o tempo de propagação é conhecido. Alternativamente, pode-se assumir uma escala de atraso, ou seja, nesse caso assume-se que o atraso de propagação não é conhecido, mas estará delimitado por um valor mínimo e um valor máximo (em inglês, *bounded delay*). Um outro modelo passível de ser aplicado assume que o tempo de propagação da informação no circuito é desconhecido, ou seja, $0 < t_{atraso} < \infty$ (no inglês, *unbounded delay*) (SPARSØ, 2006).

O modelo de atraso está intimamente ligado à classe de circuito assíncrono a qual o circuito pertence. Diversas estudos na literatura científica exploram com grande riqueza de detalhes essa classificação dos circuitos assíncronos quanto ao modelo de atraso, como Renaudin e Rigaud (2000), SparsØ (2006), Beerel, Osdag e Ferretti (2010) entre tantos outros. Esses autores classificam os circuitos assíncronos, quanto ao modelo de atraso, em 5 classes distintas, a saber:

Circuitos de Huffmam: também conhecidos como máquinas sequenciais assíncronas, esta classe é a origem dos circuitos assíncronos. Estes circuitos são concebidos a partir de portas lógicas convencionais, que geram as saídas do circuito e computam seu próximo estado. O estado seguinte é armazenado em um conjunto de linhas de realimentação. Exemplos clássicos de circuito pertencem à essa classe são os flip-flops. Os circuitos dessa classe possuem um modelo de atraso limitado (*bounded delay*).

Circuitos Micropipeline: são circuitos similares aos síncronos com pipeline, porém, com a substituição do sinal de relógio por um protocolo *handshake* de comunicação, implementado localmente para fins de sincronização. As partes dos circuitos dessa classe responsáveis pela implementação do protocolo de comunicação têm um modelo de atraso ilimitado. Contudo, a parte do circuito responsável pelo processamento da informação assume um modelo de atraso limitado.

Circuitos Independente da Velocidade: são circuitos que utilizam um modelo de atraso ilimitado em todas as suas portas. Em seu modelo, os atrasos nos fios são considerados insignificantes ou menores que os das portas. Porém, à medida que os fios se tornam mais extensos e mais complexos a afirmação de que os atrasos sejam insignificantes ou menores do que as portas não pode ser garantida.

Circuitos Insensíveis ao Atraso (*Delay Insensitive* em inglês): são similares aos circuitos independentes da velocidade, mas, o fator de atraso ilimitado é estendido também aos fios. Conforme descrito por Martin (1990), a implementação deste tipo de lógica é limitada e não prática.

Circuitos Quase Insensíveis ao Atraso (*Quasi Delay Insensitive* em inglês): Essa classe de circuitos assíncronos representa a solução assíncrona mais robusta e factível, e trata-se do tipo de implementação estudado neste trabalho. Mas detalhes sobre essa classe são apresentados na próxima subseção.

A Figura 8 compara as classes de circuitos assíncronos previamente descritas em termos de sua robustez, complexidade e da quantidade de premissas a respeito do atraso. Quanto mais premissas em relação ao atraso, menos robusto e menos complexo é a classe de circuitos assíncronos, tal como pode ser visto na Figura 8.

Uma vez caracterizado o modelo de atraso adotado por um circuito assíncrono, os próximos passos para que se possa projetar um circuito assíncrono são a definição do protocolo de comunicação e forma de codificação dos dados a serem adotados, e finalmente a definição dos elementos básicos que comporão o circuito, escolhas essas que estão intimamente ligadas à classe de circuitos assíncronos que será utilizada. Essas duas últimas etapas de projeto serão explanadas nas seções que se seguem, para o caso dos circuitos QDI, que são a classe alvo deste trabalho.

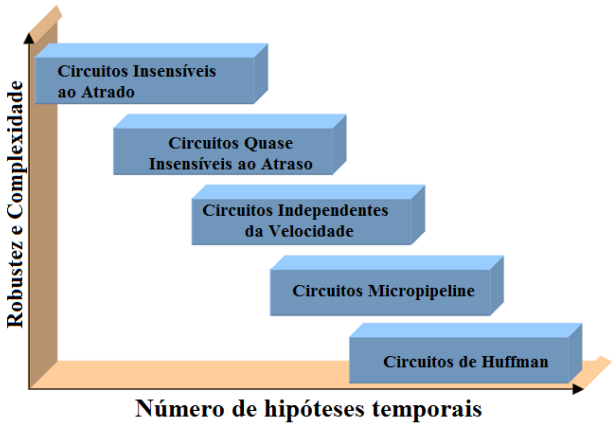

Figura 8. Robustez e Complexidade versus quantidade de premissas Temporais para as classes de circuitos assíncronos (FRAGOSO, 2005. Modificada)

2.2.1 Circuitos Quase Insensíveis ao Atraso (QDI)

Nesta classe de circuitos assíncronos, assim como em circuitos DI, o modelo ilimitado é utilizado para caracterizar o atraso em todas as portas que os constituem.

Contudo, uma ressalva é feita com relação aos atrasos nos fios dos circuitos QDI. Diferentemente de circuitos DI, que assumem que todos os fios têm modelo de atraso ilimitado, circuitos QDI assumem que alguns fios são isocrônicos, ou seja, tem atrasos similares. Essa condição tem que ser verdadeira em alguns fios que se ramificam (MARTIN, 1990). Um exemplo de aplicação dessa premissa de atraso em um circuito pode ser visto na Figura 9.

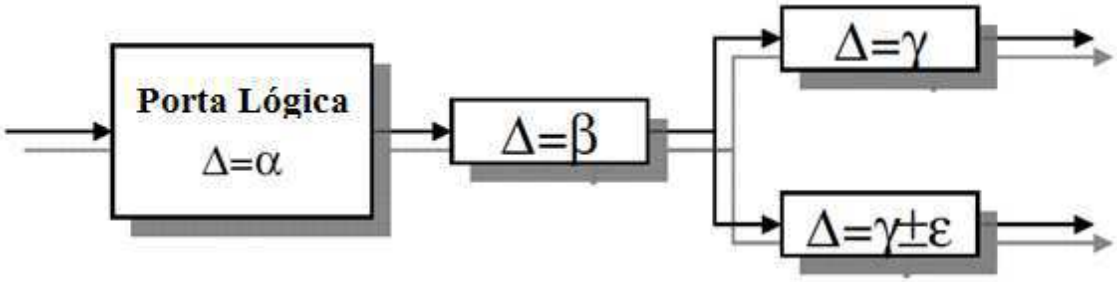

Figura 9. Ramos de fio isocrônicos em circuitos assíncronos QDI (FRAGOSO, 2005. Modificada).

Na Figura 9, a letra grega Δ representa o atraso; α, β e γ representam valores quaisquer; e ε representa uma pequena variação. O bloco "*Porte*" representa uma porta lógica qualquer. Percebe-se, então, que os atrasos na porta e no fio de saída do circuito são arbitrários (α e β, respectivamente), ou seja, nesse caso adotou-se um modelo de atraso ilimitado para a porta e para o fio de saída.

Pode-se perceber também que o fio de saída da porta se ramifica em dois, um com um atraso arbitrário γ e o outro com uma pequena variação nesse mesmo atraso (γ± ε). Esses ramos representam caminhos críticos neste exemplo e, por isso, conforme dito anteriormente, assumiu-se a premissa de que as ramificações do fio de saída da porta são fios isocrônicos.

Uma forma de implementação de um sistema assíncrona, analogamente a sistemas síncronos pode ser observado no Figura 10.

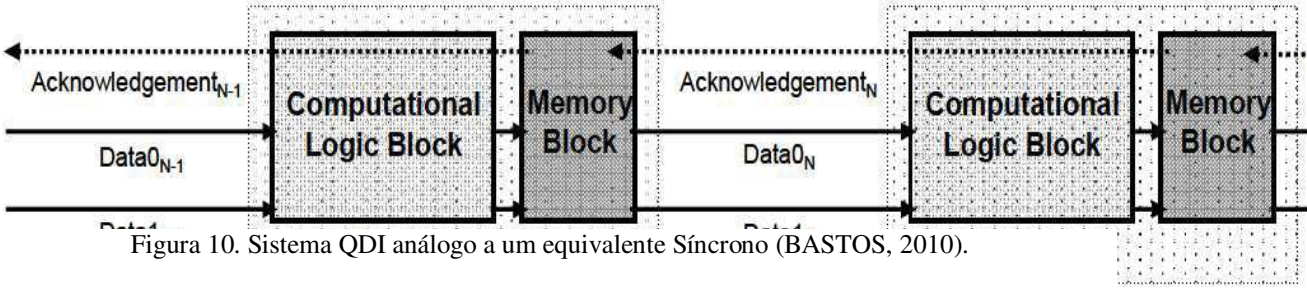

Figura 10. Sistema QDI análogo a um equivalente Síncrono (BASTOS, 2010).

Os blocos de memória (*Memory Block*) dos circuitos QDI são equivalentes aos registradores dos circuitos síncronos e os blocos de computação lógica (*Computational Logic Block*) são equivalentes aos blocos combinatórios dos sistemas síncronos, com uma ressalva. Para que o modelo de atraso adotado pelos circuitos QDI seja respeitado em um sistema assíncrono como um todo, é necessário projetar os blocos de computação lógica utilizando como elemento básico portas de Muller, também denominadas C-Elements (BASTOS, 2010), que serão detalhados nas seções seguintes.

Para que haja sincronização e não haja perda de dados entre os estágios de um sistema QDI, é necessário implementar um protocolo de comunicação entre um estágio e outro. Os protocolos de comunicação geralmente utilizados nos circuitos QDI são agrupados na literatura em dois grandes grupos, o protocolo duas fases, que não será abordado nesse trabalho, e o protocolo quatro fases, que será tratado com detalhes na próxima subseção.

Para que se possa implementar o protocolo de comunicação quatro fases sem nenhuma pressuposição acerca de atrasos, é necessário implementar um esquema de codificação de dados. Tal codificação, denominada codificação insensível ao atraso, permite a detecção da chegada de novas informação a cada estágio do sistema QDI sem necessidade de sinas adicionais (sinal de pedido da Figura 7) (FRAGOSO, 2005).

Pode-se perceber que, no sistema apresentado na Figura 10 a codificação insensível ao atraso foi implementada, uma vez que não há sinal de pedido.

2.2.2 PROTOCOLO 4 FASES

Conforme mencionado anteriormente, existem duas formas básicas de protocolos para circuitos assíncronos: protocolo 2 ou 4 fases. Um protocolo tradicional de comunicação utiliza 2 fases para realizar um ciclo computacional completo. As transições que definem em qual das duas fases o circuito se encontra.

Um protocolo de 4 fases detecta pelo menos três eventos em ciclo completo de transmissão de informações: demanda/pedido, confirmação e um ou mais sinais de dados. A implementação desse protocolo de 4 fases é mais simples do que uma implementação em protocolo de 2 fases, pois implementar um esquema de detecção de fase é mais simples do que implementar um esquema de detecção de transição (FRAGOSO, 2005).

Contudo, a necessidade de detecção de mais eventos resulta em um circuito mais lento se comparado à um circuito que implemente o protocolo em 2 fases (MOREIRA;

GUAZZELLI; CALAZANS, 2012). Ainda assim, para várias aplicações as vantagens do protocolo de 4 fases se sobressaem, fazendo com que o mesmo seja mais utilizado (MOREIRA; GUAZZELLI; CALAZANS, 2012; GREAVES, 2013).

Na Figura 11 são apresentadas as 4 fases deste protocolo. Os sinais de entrada, saída e confirmação de um estágio de circuito assíncrono são apresentados nas linhas um, dois e três da forma de onda apresentada, respectivamente.

Na primeira fase, os dados válidos são detectados e processados. A passagem do sinal de confirmação (*acknowledgment*) a zero, que certifica a validade dos dados no canal de saída, marca o fim da primeira fase e começo da segunda. O estágio precedente, então, detecta o sinal de confirmação e faz com que a entrada passe para um estado invalido, nesse caso a entrada retorna a zero.

A terceira fase acaba quando o estágio atual detecta o estado inválido da entrada e passa o sinal de confirmação à 1 novamente. O começo da quarta fase marca o fim do processamento dos dados, e termina com a chegada de uma nova entrada, quando a entrada passa novamente a um estado válido. Observa-se na Figura 11 que uma codificação insensível ao atraso foi implementada nesse no circuito assíncrono em questão.

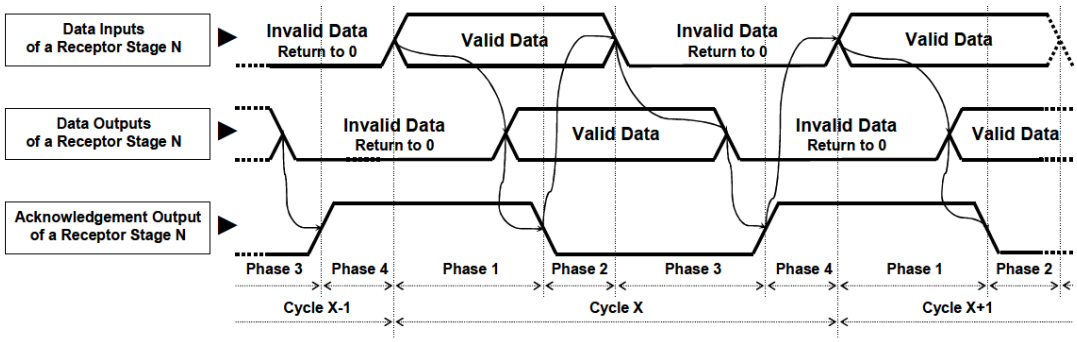

Figura 11. Representação das fases do protocolo de comunicação 4 fases (BASTOS, 2010)

2.2.3 C-ELEMENTS

Células *C-Element*, também denominadas portas de Muller são elementos de memória normalmente usados em circuitos assíncronos. O princípio de funcionamento de uma C-Element de duas entradas é caracterizado pela tabela verdade apresentada na

Tabela 2, na qual A(n) e B(n) representam entradas e Y(n) representa a saída da porta C-Element no instante n.

Tabela 2. Tabela Verdade de C-Element de 2 entradas

A(n)	B(n)	Y(n)
0	0	0
0	1	Y(n-1)
1	0	Y(n-1)
1	1	1

Percebe-se pela tabela verdade que as portas de Muller possuem basicamente duas funções lógicas:

- Buffer, quando as entradas A e B são idênticas; e
- Memória, preservando o valor anterior no caso de suas entradas A e B serem distintas;

Nota-se que o funcionamento de uma porta C-Element é equivalente ao de um *Latch* SR. Porém, portas SR não seguem o modelo de atraso ilimitado, já que não se pode assegurar que suas duas entradas serão iguais à 1 simultaneamente (WUU; VRUDHULA, 1993). O mesmo não ocorre com circuitos C-Elements, já que a alteração de estado só ocorre depois que as duas entradas passam a ter o mesmo valor. Devido ao seu funcionamento, portas Muller podem ser também usadas na sincronização de diversos sinais.

Existem diversas arquiteturas para circuitos C-Elements, dentre estas, as quatro mais básicas são C-Element Convencional, C-Element Dinâmica (*Dynamic*), C-Element Simétrica (*Symmetric*) e C-Element de Realimentação Fraca (*Week Feedback*). Um estudo mais aprofundado dos tipos de portas Muller foge do escopo deste trabalho.

3 ATIVIDADES DESENVOLVIDAS

Para que os objetivos desse projeto fossem cumpridos com sucesso, uma série de atividades foram executadas. Começou-se pela caracterização de circuitos síncronos em linguagem de descrição de hardware VHDL, criação de ambientes de teste para a verificação comportamental dos mesmos e utilização de ferramentas EDA para síntese lógica e concepção do leiaute, no inglês *Place and Route* (P&R).

Uma vez que um fluxo de projeto de leiaute, a partir de um circuito descrito em HDL, foi estabelecido para circuitos síncronos em tecnologia FD-SOI 28nm, algumas adaptações foram feitas no fluxo de projeto proposto para viabilizar a concepção de leiautes de circuitos assíncronos QDI. Cada etapa e cada atividade desenvolvida será abordada nas subseções desse capítulo.

3.1 PROPOSIÇÃO DE UM FLUXO DE PROJETO PARA PROJETO DE CIRCUITOS SÍNCRONOS EM TECNOLOGIA FD-SOI 28NM

O estabelecimento de um fluxo de projeto de circuitos integrados, desde a descrição do circuito, em nível RTL, até a obtenção do leiaute final, é fundamental para que o processo de produção de *chips* seja eficiente e seja o mais reutilizável possível. A forma como as etapas de tal fluxo é organizada deve levar em consideração as nuances da tecnologia de fabricação à qual o circuito integrado será mapeado.

Neste contexto, a fim de aplicar os conhecimentos acerca da tecnologia FD-SOI adquiridos na fase de revisão bibliográfica, e também conhecer as ferramentas utilizadas no laboratório onde realizou-se o estágio, a primeira tarefa realizada foi a adaptação de um fluxo de projeto de CIs padrão à tecnologia FD-SOI 28nm.

O fluxo de projeto proposto parte da descrição do circuito que se deseja implementar, em linguagem HDL e nível RTL, e tem como produto final a implementação física do circuito em questão em termos de formas geométricas correspondentes às camadas de metal, óxido e semicondutores que constituirão o circuito integrado, tal como os fluxos de projeto de CIs clássicos. Um fluxograma que representa o referido fluxo de projeto proposto pode ser visto na Figura 12.

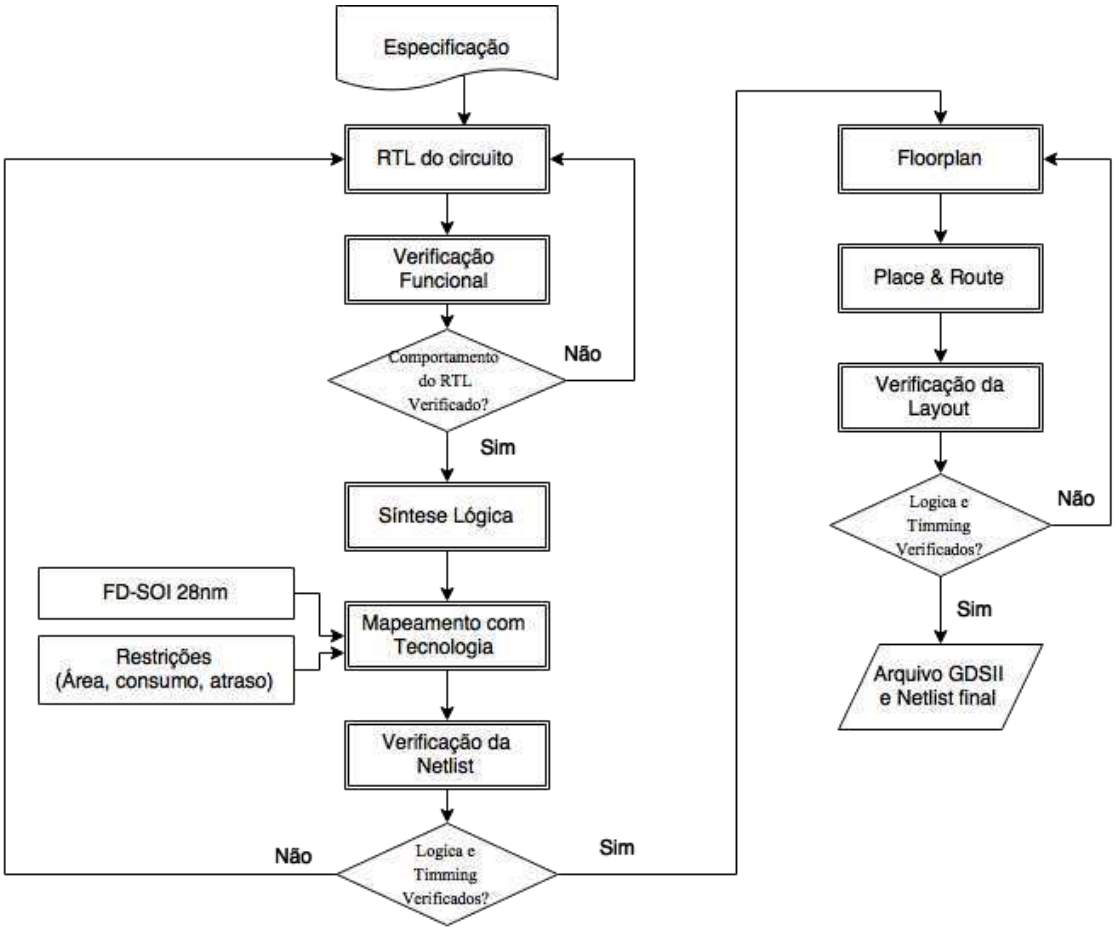

Figura 12. Fluxo de projeto de CIs síncronos.

Primeiramente uma arquitetura é concebida, a partir das especificações do sistema que se deseja projetar, e uma descrição do hardware é feita em nível RTL, utilizando-se a linguagem VHDL. Esta é a primeira etapa do fluxo de projeto proposto, conforme fluxograma da Figura 12.

Em seguida, a fim de verificar se o comportamento do circuito descrito está de acordo com o que foi especificado, o código RTL passa por um processo de verificação funcional. Um ambiente de verificação (*testbench*) é criado, também em VHDL, e simulado no software ModelSim, da Mentor. Uma vez que a descrição do hardware em questão atende às expectativas comportamentais, segue-se para a etapa de síntese lógica.

Nesta, o código HDL é traduzido em portas lógicas básicas e flip-flops independentes de qualquer tecnologia. Após algumas otimizações, a ferramenta de síntese mapeia o conjunto de portas lógicas à células padrão da tecnologia FD-SOI, levando em consideração algumas restrições de tempo, consumo e área, conforme observado na Figura 12. Nesta etapa o software Design Compiler, da Synopsys, foi utilizado como ferramenta de síntese. O conjunto de portas lógicas obtido, conhecida também como

netlist, passa por uma análise, para verificar se a funcionalidade e os atrasos nas portas estão de acordo com a especificação e as restrições definidas.

Caso a netlist obtida esteja de acordo com o esperado, em termos comportamentais e de restrições de tempo, área e consumo, segue-se a elaboração do leiaute. Pode-se definir concepção de leiaute como o processo de criação de uma representação física precisa de uma *netlist* que esteja em conformidade com as restrições impostas pelo processo de fabricação e pelos requisitos de desempenho (CLEIN, 2000). Para a criação do leiaute, é necessário seguir uma séria de passos, os quais serão melhor detalhados na parte da seção de resultados referente a esta tarefa. O software Soc Encounter, da Cadence, foi utilizado para executar as etapas de *floorplan* e *place & route*.

A concepção do leiaute foi dividida em apenas duas etapas no fluxograma da Figura 12 (*Floorplan* e *Place & Route*), porém várias sub etapas são necessárias em cada uma dessas etapas, as quais serão descritas com mais detalhes na seção de resultados.

Finalmente faz-se uma última análise na netlist pós leiaute obtida, para verificar se o leiaute obtido está de acordo com as especificações e com as restrições de área, consumo e tempo.

A execução das etapas do fluxo de projeto da Figura 12 é regida por uma série de scripts escritos em *shell script* e TCL, responsáveis pela configuração e controle adequado das ferramentas EDA que são utilizadas ao longo. O laboratório onde realizou-se o estágio dispunha de uma série de *scripts shell* e em formato TCL que implementavam um fluxo de projeto padrão para uma outra tecnologia. O objetivo principal dessa atividade consistiu em adaptar os scripts existente às nuances da tecnologia FD-SOI 28nm, algumas das quais foram descritas no capítulo 2. Os resultados obtidos vão ser mostrados na próxima seção, passo a passo, com um exemplo de leiaute de um microprocessador ARM7 que foi feito.

3.2 PROJETO DE UM PROCESSADOR ARM7 SIMPLIFICADO, DO RTL AO LEIAUTE

Após o estabelecimento do fluxo de projeto específico para a tecnologia FD-SOI 28nm, a descrição em nível RTL de um microprocessador passou por todas as etapas

propostas, a fim de testar a funcionalidade do fluxo proposto, inicialmente para circuitos síncronos.

O microprocessador utilizado foi um ARM7 simplificado, cuja descrição de hardware já estava em andamento, no momento do início do estágio. Tal hardware foi escolhido pela equipe do laboratório onde realizou-se o estágio por tratar-se de um processador relativamente simples indicado para aplicações de baixo custo e que visem baixo consumo de energia.

Trata-se de um microprocessador RISC cujas instruções têm tamanho fixo igual a 32-bits. As instruções capazes de ser executadas nesse microprocessador são compatíveis com o conjunto de instruções de microprocessadores ARM, família ARM7.

Assim como os microprocessadores dessa família, a estrutura do microprocessador desenvolvido conta com um pipeline de três estágios, para assegurar que os sistemas de processamento e memória sejam utilizados continuamente. Por motivos de simplicidade e limitação de tempo, não foram implementados coprocessadores, nem cache de instruções ou de dados.

O principal objetivo desta tarefa foi verificar os módulos da arquitetura que já estava sendo desenvolvida, efetuando as devidas correções arquiteturais, produção da netlist em nível de portas lógicas e em seguida concepção do leiaute. Maiores detalhes sobre o microprocessador projetado serão dados nas subseções seguintes. Os resultados da verificação, síntese lógica e o leiaute obtido serão apresentados no próximo capítulo, na seção correspondente a esta tarefa.

3.2.1 CONJUNTO DE INSTRUÇÕES IMPLEMENTADAS

As seguintes instruções foram implementadas na arquitetura proposta:

1. Instruções de processamento de dados;
2. Instruções *branch*;
3. Instruções de acesso a memória (*load word* e *store word*).

A estrutura de instruções de processamento de dados (em inglês *Data Processing*) pode ser vista na Figura 13.

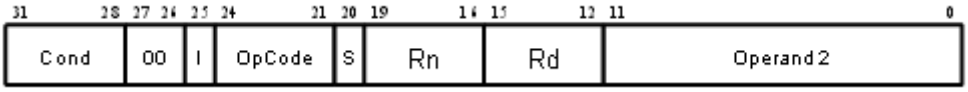

Figura 13. Codificação de instruções de processamento de dados.

Nesse tipo de instrução, cada grupo de bits, aqui referenciados como campos, ilustrado na Figura 13, tem a seguinte codificação:

- I - Indica se o segundo operador da instrução é um valor imediato ou um registrador;
- OpCode – Indica qual operação de processamento de dados deve ser executada. Uma lista completa das operações codificadas por esse campo da instrução pode ser vista na documentação de referência dos processadores ARM7 (1994);
- S – Indica se o registrador CPSR (r16 do banco de registradores) será atualizado ou não;
- Rn – Endereço do registrador que será o primeiro operando da operação que será executada;
- Rd – Endereço do registrador de destino;
- Operand2 – Representa o segundo operando da instrução a ser executada. Pode ser um registrador ou um valor imediato.

O campo “Cond” está presente em qualquer tipo de instrução da família ARM7. Todas as instruções dessa família são condicionais, ou seja, elas somente serão efetivamente executadas caso a condição teste codificada pelo campo “Cond” seja verdadeira. Uma lista completa dos tipos de teste codificados por esse campo das instruções pode ser vista na documentação de referência da família de processadores ARM7 (1994).

A estrutura das instruções tipo branch pode ser vista na Figura 14.

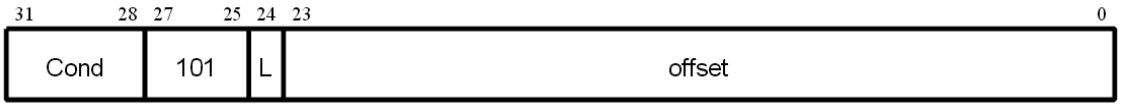

Figura 14. Codificação de instruções branch.

Nesse tipo de instrução, cada campo ilustrado na Figura 14 tem a seguinte codificação:

- L – Indica se a corrente instrução é um branch (L = 1) ou branch com link (L = 0). O processador projetado não implementa funções de branch com link, portanto esse bit é sempre 0;

- offset – Valor que será adicionado ao apontador de programa (PC) a fim de conseguir-se o endereço da próxima instrução a ser executada;

A estrutura das instruções de acesso a memória pode ser vista na figura 15.

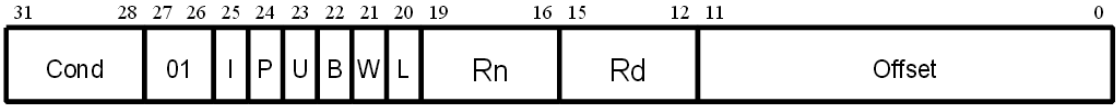

Figure 15. Codificação de instruções de acesso a memória.

Nesse tipo de instrução, cada campo ilustrado na Figura 14 tem a seguinte codificação:

- I – Indica se o segundo operador da instrução é um valor imediato ou um registrador;
- P – Indica se o valor do offset deve ser modificado antes (P = 1) ou depois (P = 0) que o registrador base for usado como endereço. No processador projetado a modificação se dá sempre depois da utilização do registrador base como endereço, ou seja, P é sempre 0;
- U – Indica se o valor do offset dever ser subtraído (U = 0) ou adicionado (U = 1) do valor presente no registrador base;
- B – Indica se será transferido apenas um byte da memória (B = 1) ou uma palavra inteira (B = 0). No processador projetado esse bit é sempre 0 pois apenas transferências de palavras completas são permitidas.
- W – Indica se o valor modificado da base com o offset deve ser reescrito no registrador base (W = 1) ou não (W = 0). No processador projetado o registrador de base sempre permanece com o valor que tinha antes de ser modificado pelo offset, ou seja, esse bit é sempre 0.

3.2.2 ARQUITETURA DO PROCESSADOR

O microprocessador projetado implementa apenas um modo de operação, *User Mode*. Nesse modo de operação um conjunto de 17 registradores são utilizados, dos quais, 15 são de uso geral, 1 guarda o endereço da instrução que está sendo executada, conhecido como registrador PC e 1 registrador de estado, denominado CPSR, que guarda

informações a respeito do modo de operação no qual o processador está operando e também informações a respeito do resultado da operação que está sendo executada (resultado negativo, presença de *carry out*, *overflow* ou resultado igual a zero).

Cada vez que uma instrução é executada, o registrador de estado é consultado. Caso a condição imposta pela instrução seja verdadeira, o banco de registradores será atualizado em resposta à execução da instrução. Caso contrário, a instrução também é executada, mas seu resultado não altera os registradores do processador.

A arquitetura do processador projetado pode ser vista na Figura 16.

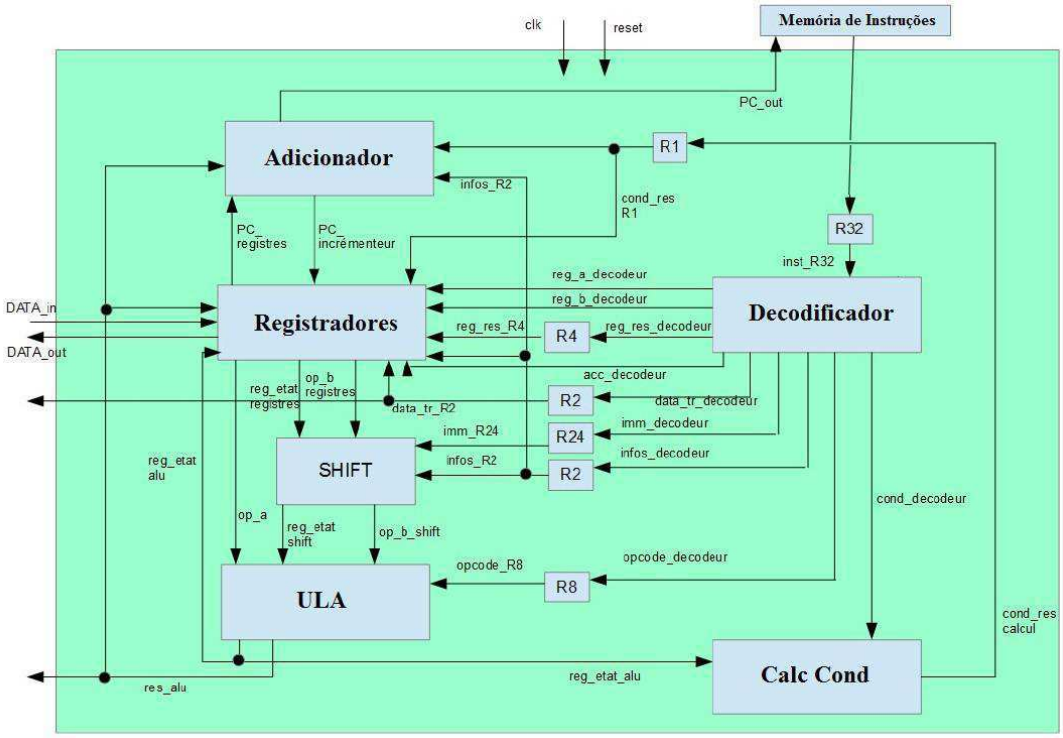

Figura 16. Arquitetura do microprocessador ARM7 projetado.

As funções de cada bloco dessa arquitetura são definidas a seguir:

- Decodificador – Lê a instrução a ser executada e produz os sinais de controle que serão necessários à correta execução da instrução, seleciona os endereços dos registradores a serem utilizados durante o estágio de execução da instrução e também do registrador de destino;
- Registradores – Conjunto de registradores que são guardam os valores a serem usados na execução das instruções no processador;
- ULA (Unidade Lógica e Aritmética) – Executa operações lógicas e aritméticas, entre elas soma, subtração, “e” e “ou” lógico;

- Shift – Executa o tipo correto de shift no segundo operador, que pode ser o valor contido em um registrador ou um valor imediato, conforme detalhado na subseção anterior;
- Calc Cond – Determina se os resultados da execução das instruções serão armazenados ou não no banco de registradores. As várias possibilidades de testes que essa unidade pode executar dependo do campo "Cond" da instrução, conforme comentado anteriormente;
- Adicionador – Atualiza o valor do PC, somando-o a 4. Também é responsável por selecionar se a próxima instrução deve ser lido do endereço atualizado do PC ou do resultado de uma instrução tipo *branch*.
- Registradores (R1, R2, R4, R8, R24 e R32) – Esses registradores foram incluídos na arquitetura para garantir a sincronização, ou seja, para garantir que os dados corretos cheguem em cada bloco no momento certo.

Conforme dito anteriormente, o microprocessador projeto implementa três estágios de pipeline, que são leitura da instrução, decodificação e execução. A cada ciclo do sinal de relógio (clk na Figura 16) os vários blocos da arquitetura do microprocessador executam um dos referidos estágios.

Analisando o "ciclo de vida" de uma instrução arbitrária no microprocessador da Figura 16, em uma primeira subida do sinal de relógio, uma palavra de 32 bits (que representa uma instrução) é lida da memória de instruções e armazena no registrador R32 da Figura 16. O endereço a ser lido é informado pelo sinal PC_out.

Quando da ocorrência de uma segunda subida do sinal de relógio, a instrução armazenada em R32 é então decodificada, segundo premissas apresentadas da subseção anterior. É durante esse estágio do pipeline que o conjunto de registradores recebe os endereços corretos dos registradores a serem utilizados no próximo estágio do pipeline. É também nesse estágio que o bloco shift calcula o segundo operador corretamente, em caso de cálculos com um valor imediato.

Após a terceira borda de subida do sinal de relógio, a instrução é executada. A execução das instruções e atualização dos registradores em função do resultado das instruções acontecem em um mesmo estágio do pipeline, para obedecer aos três estágios de pipeline implementados na família de processadores ARM7.

3.3 ADAPTAÇÃO DO FLUXO DE PROJETO PARA CIRCUITOS ASSÍNCRONOS QDI

Uma vez que o fluxo de projeto de CIs síncronos apresentado na seção 3.1 foi testado com o microprocessador apresentado na seção 3.2, uma série de modificações foram testadas a fim de conseguir-se desenvolver um fluxo similar ao proposto em 3.1, que seja capaz de gerar leiaute de circuitos assíncronos QDI.

O objetivo desta tarefa foi propor um fluxo de projeto de circuitos assíncronos QDI que utilizasse as consolidadas ferramentas EDA que são tipicamente utilizadas para concepção de circuitos síncronos. Apesar de existirem algumas ferramentas EDA específicas para desenvolvimento de circuitos assíncronos, alguns contrapontos podem ser identificados.

O sistema Balsa (2010), por exemplo, requer a utilização de linguagem CSP (HOARE, 1985), que descreve os circuitos de forma diferente das consolidadas linguagens de descrição de hardware dos fluxos síncronos. Segundo estudos feitos por Zhou et al. (2014), outras ferramentas também foram desenvolvidas com o intuito de disseminar o desenvolvimento de circuitos assíncronas, mas exigem conhecimento de linguagens pouco conhecidas ou apresentam algumas desvantagens em relação à otimização, se comparadas às ferramentas para circuitos síncronos disponíveis.

Neste contexto, o objetivo desta tarefa é propor um fluxo de projeto de CIs assíncronos que faça o mínimo de alterações possíveis no fluxo de projeto da Figura 12. Sendo assim, pensou-se em algumas alterações para o referido fluxo, as quais podem ser vistas da Figura 17.

As partes em vermelho representam as alterações propostas no fluxograma da Figura 12 a fim de conseguir-se o leiaute de circuitos assíncronos QDI. A primeira modificação que se deve fazer diz respeito ao nível descrição do circuito. Inicialmente, por simplicidade, optou-se por usar apenas códigos de circuitos descritos em nível de portas lógicas, não em nível comportamental como era feito no fluxo síncrono. O motivo de tal restrição é possibilitar o uso de ferramentas de síntese síncronas no procedimento de síntese lógica. Futuramente, uma etapa pode ser adicionada a fim de "traduzir" circuitos descritos em nível comportamental em circuitos assíncronos em nível de portas lógicas, tal como proposto por Zhou et al. (2014). Contudo essa funcionalidade não faz parte do fluxo proposto por este trabalho.

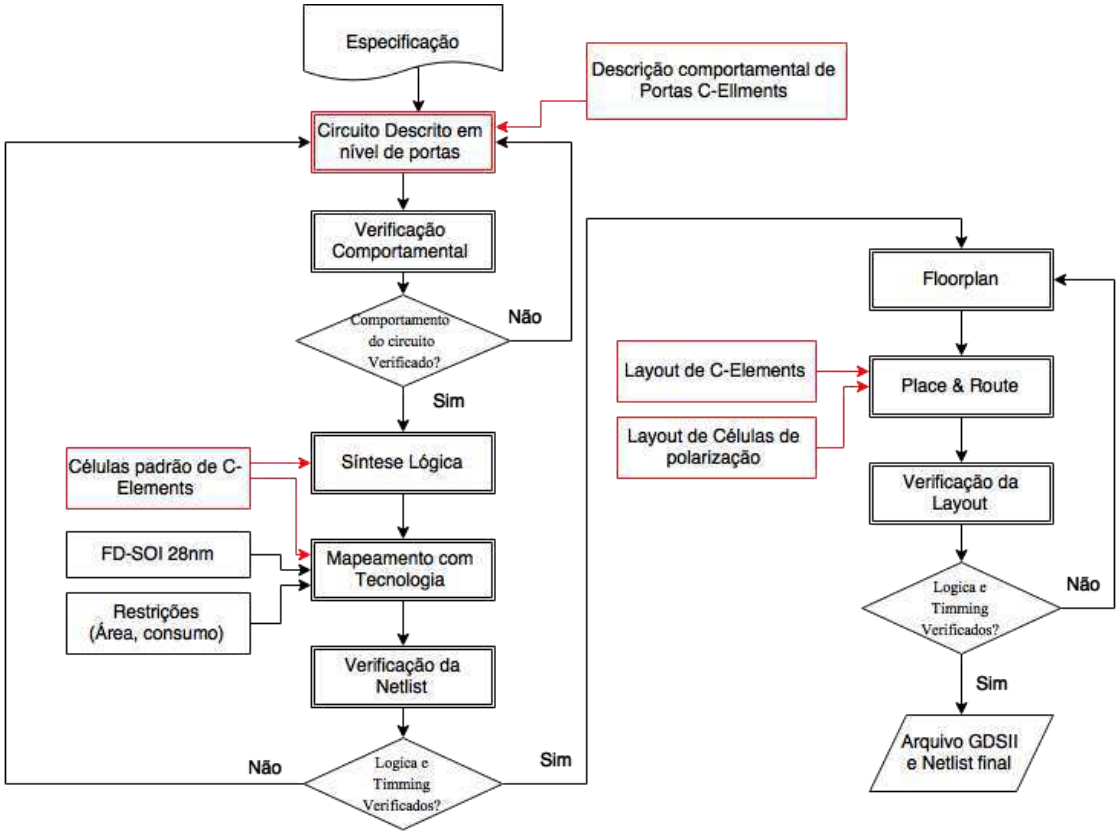

Figura 17. Fluxo de projeto de CIs assíncronos QDI.

Nessa etapa de descrição de hardware é importante descrever os circuitos usando portas de Muller, para garantir que o modelo de atraso adotado pelos circuitos QDI seja respeitado, conforme descrito na fundamentação teórica deste trabalho. Após simulação comportamental inicial, caso o circuito funcione conforme especificado, segue-se o fluxo com a etapa de síntese lógica.

Uma alteração importante que tem que ser feita nesta etapa é impedir que a ferramenta de síntese lógica substitua as portas de Muller por um conjunto de outras portas lógicas. No fluxo de projeto de CIs assíncronos proposto, esse problema é contornado inserindo-se células padrão das portas de Muller necessárias à cada circuito, conforme pode ser visto no fluxograma da Figura 17. O procedimento de caracterização das células padrão das portas C-Elements segue o fluxograma da Figura 18.

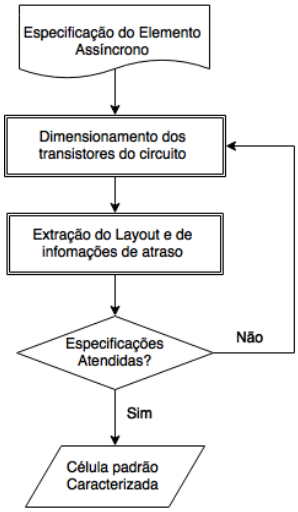

Figura 18. Processo de caracterização de células padrão assíncronas.

Esse fluxograma descreve o que deve ser feito para que se consiga a descrição de portas de Muller em termos de seu consumo, atrasos e tamanho. Tal descrição é codificada em arquivos nos formatos db e lib, os quais são usados pela ferramenta de síntese para fazer o mapeamento da netlist com a tecnologia FD-SOI 28nm. Além dos arquivos necessários à etapa de síntese lógica e mapeamento com a tecnologia FD-SOI, o procedimento descrito pelo fluxograma da Figura 18 tem como saída um arquivo em formato lef, o qual é utilizado nas etapas de concepção do leiaute.

O procedimento descrito na Figura 18 também deve ser seguido para caracterização de células de polarização. Estas, por sua vez, podem ser utilizadas nas etapas de P&R para criar esquemas de polarização do plano de fundo dos transistores FD-SOI a fim de aumentar o desempenho do circuito ou diminuir seu consumo, assim como descrito na fundamentação teórica deste trabalho.

As etapas do fluxograma da Figura 18 não serão discutidas em maiores detalhes neste trabalho pois não estão no escopo deste trabalho.

Os resultados obtidos nesta tarefa serão apresentados na seção seguinte.

4 RESULTADOS

4.1 PROCESSADOR ARM7 SIMPLIFICADO

4.1.1 VERIFICAÇÃO FUNCIONAL DA ARQUITETURA

Primeiramente cada módulo do processador ARM7 projetado foi testado separadamente para garantir que seu comportamento era o esperado. Os ambientes de verificação utilizados para esse fim, descritos em linguagem VHDL, tinham a estrutura tal qual apresentado na Figura 19.

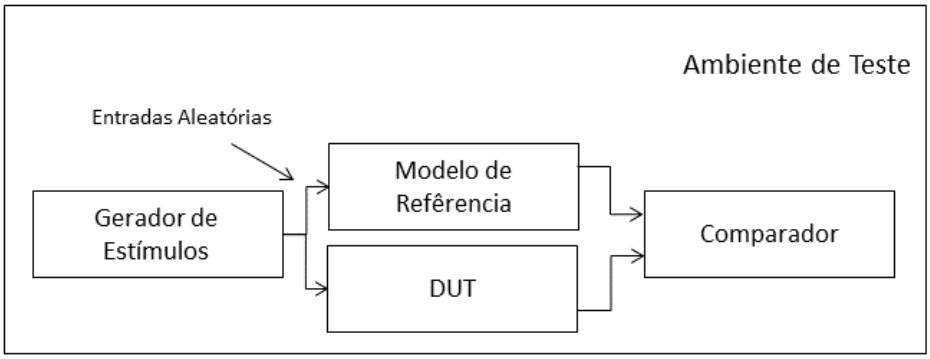

Figura 19. Estrutura do Ambiente de testes.

O gerador de estímulos, como o próprio nome sugere, é o componente do testbench responsável por gerar as entradas que serão usadas como teste. O DUT, do inglês *Design Under Test*, é o bloco que se deseja verificar. O modelo de referência é um componente presente em várias metodologias de verificação funcional, como OVM e UVM. Geralmente são modelos previamente validados do funcionamento de cada bloco. Nesta tarefa usou-se como modelo de referência os blocos similares do microprocessador open source, STORMcore.

O comparador, por sua vez, compara as saídas do modelo de referência com as saídas do DUT, contando o número de saídas que são iguais e o número de saídas que são diferentes entre esses dois blocos.

Essa estrutura de testbench foi usada para verificar os blocos ALU, Decodeur e Shif da arquitetura mostrada na Figura 16. O simulador ModelSim, da Mentor foi usado para simular os ambientes de teste. Exemplos de formas de onda obtidas durante a verificação do modulo ALU pode ser visto na figura 20 e na figura 21.

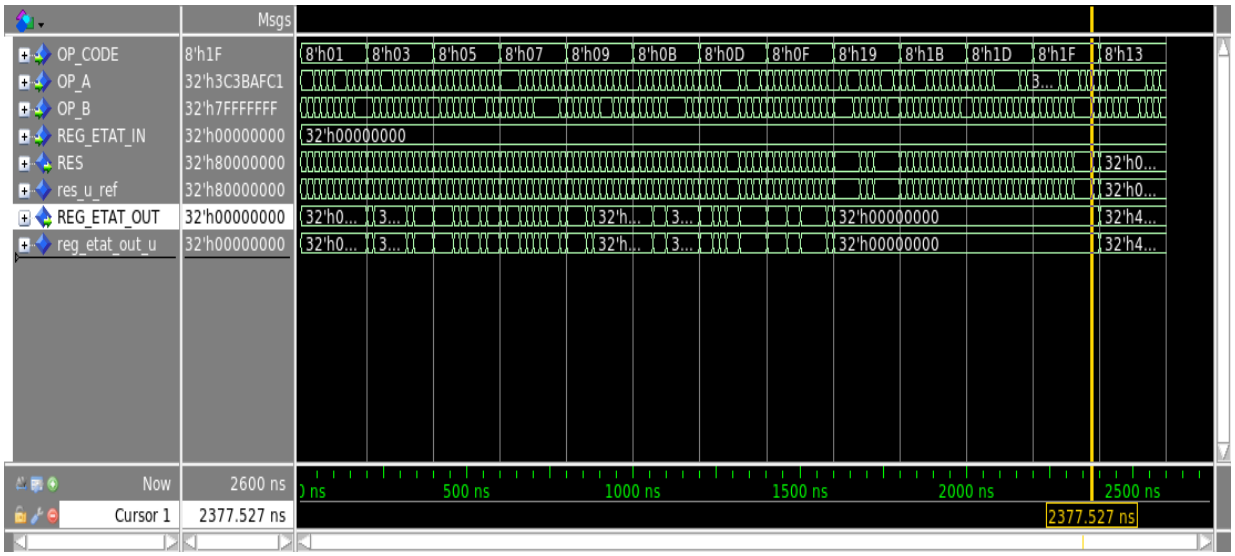

Figura 20. Forma de onda gerada pelo testbench do módulo ALU.

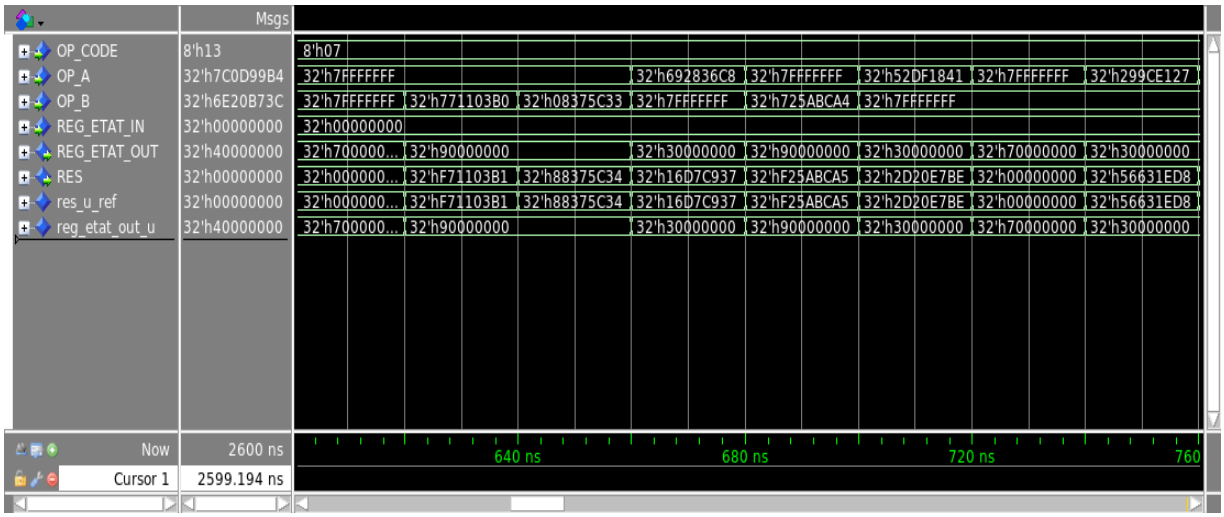

Figura 21. Forma de onda gerada pelo *testbench* do módulo ALU, com zoom nos instantes iniciais.

Na Figura 20 a variável OP_CODE sinaliza ao módulo ALU qual operação deve ser efetuada, as variáveis OP_A e OP_B representam respectivamente o primeiro e segundo operandos, REG_ETAT_IN representa o valor do registrador de estado CPSR antes que o módulo o altere e REG_ETAT_OUT representa o valor atualizado deste registrador. A variável RES representa o resultado da operação efetuada pela ULA. As

variáveis res_u_ref e reg_etat_out_u representas as saídas do modelo de referência, as quais serão comparadas à RES e REG_ETAT_OUT, respectivamente.

A Figura 21 é apenas uma ampliação da figura anterior. Nela pode-se observar que, conforme esperado, as saídas do DUT estão de acordo com as do modelo de referência, portanto, para a estratégia de verificação adotada, conclui-se que o comportamento do código RTL no módulo ALU condiz com a especificação.

Para facilitar a visualização dos resultados, o comparador imprime no terminal de saída do ModelSim a quantidade de valores comparados iguais (*matches*) e a quantidade de valores comparados diferentes (*mismatches*) para cada saída. A figura 22 mostra uma parte desse arquivo gerado.

No testbench do módulo ALU, todos as opções de opcode foram testadas. O módulo gerador de estímulos gerou aleatoriamente as entradas OP_A e OP_B. Para cada operação a ser testada, 10 combinações de operadores foram geradas e comparadas com as saídas do modelo de referência.

Os ambientes de verificação dos blocos “Calcul Condition”, Registres, “PC Incrementeur” e registradores não seguiram a estrutura da Figura 19, tendo sido verificados pela introdução de algumas entradas, não aleatórias, e verificação da coerência de suas saídas a partir das formas de ondas geradas.

```
sim:/test_bench_alu/ALU1/ALU1/RES
add wave -noupdate -format logic
sim:/test_bench_alu/ALU1/../res_u_ref
add wave -noupdate -format logic
sim:/test_bench_alu/ALU1/../reg_etat_out_u
run 600000ns
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of RES
matches is 10 And mismatches: 0 Of 10 Tests for OP_CODE: 1
#    Time: 160 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of
REG_ETAT_OUT matches is 10 And mismatches: 0 Of 10 Tests for
OP_CODE: 1
#    Time: 160 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of RES
matches is 10 And mismatches: 0 Of 10 Tests for OP_CODE: 3
#    Time: 360 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of
REG_ETAT_OUT matches is 10 And mismatches: 0 Of 10 Tests for
OP_CODE: 3
#    Time: 360 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of RES
matches is 10 And mismatches: 0 Of 10 Tests for OP_CODE: 5
#    Time: 560 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of
REG_ETAT_OUT matches is 10 And mismatches: 0 Of 10 Tests for
OP_CODE: 5
#    Time: 560 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of RES
matches is 10 And mismatches: 0 Of 10 Tests for OP_CODE: 7
#    Time: 760 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of
REG_ETAT_OUT matches is 10 And mismatches: 0 Of 10 Tests for
OP_CODE: 7
#    Time: 760 ns  Iteration: 1  Instance: /test_bench_alu
# ** Note: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Number of RES
matches is 10 And mismatches: 0 Of 10 Tests for OP_CODE: 9
```

Figure 22. Parte do Arquivo gerado pelo comparador do ambiente de verificação do módulo ALU no ModelSim.

A verificação do funcionamento do processador ARM7 como um todo se deu também de forma diferente. A estrutura do ambiente de verificação usado para testar seu comportamento é mostrada na figura 23.

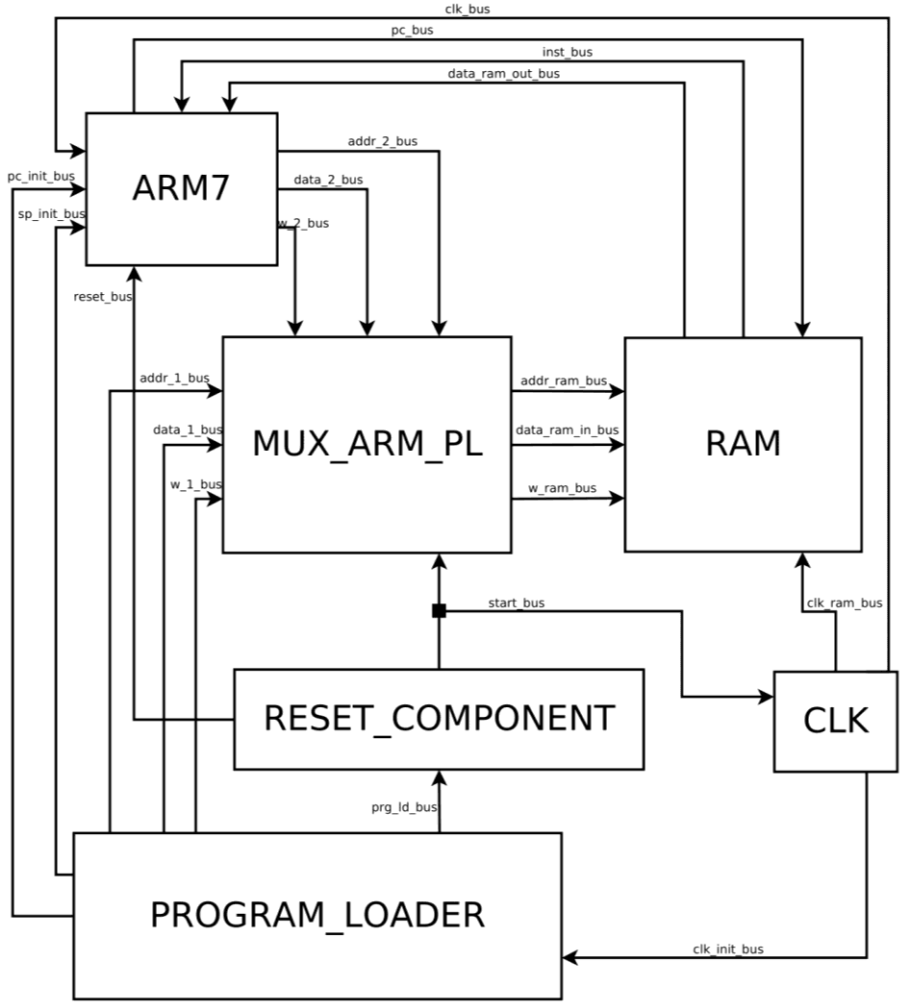

Figura 23. Estrutura do ambiente de verificação do microprocessador ARM7 simplificado

O ambiente é composto pelos seguintes blocos:

- O processador ARM7 que se deseja verificar;
- RAM – Funciona como memória de dados e memória de instruções;
- Program_Loader – Responsável por carregar as instruções presentes em um arquivo binário, gerado por um assembler, na memória de instruções;
- Reset_Component – Indica ao processador o momento de começar a funcionar, ou seja, o momento em que as instruções já foram carregadas na memória;

- Mux_Arm_PL – Coordena o acesso à memória. Como o bloco RAM foi implementado para ter as funcionalidades de memória de instruções e memória de dados, esse módulo faz-se necessário para que a memória correta seja acessada.

Para que o teste funcione corretamente, é necessário existir, na mesma pasta onde se encontram os arquivos dos blocos que compõem o ambiente de teste, um arquivo binário contendo as instruções do programa a ser executado no microprocessador. As etapas para conseguir-se tal arquivo eram objeto do trabalho de um outro grupo de estagiários e fogem do escopo deste trabalho.

Tendo-se um arquivo binário contendo as instruções que serão usadas para testar o ARM7, o módulo Program_Loader lê cada instrução desse arquivo e as envia ao bloco Mux_Arm_PL, que as envia para o bloco RAM juntamente com os endereços corretos nos quais cada instrução deve ser armazenada. Durante esse processo de carregamento de instruções, o módulo Reset_Component certifica-se de que o módulo ARM7 não tentará acessar a memória mantendo a entrada reset do microprocessador em nível lógico alto.

Uma vez que as instruções já estão devidamente armazenadas na memória, o módulo Reset_Component coloca a saída reset_bus em nível lógico baixo, o que faz com que o módulo ARM7 comece a processar as instruções. O endereço da primeira instrução a ser executada é passado pela entrada pc_init_bus, durante o estado de reset.

A cada ciclo do sinal de relógio, o microprocessador sob teste lê a instrução armazenada no endereço indicado no barramento pc_bus. As instruções são carregadas no barramento inst_bus pelo módulo RAM. Como o microprocessador ARM projetado tem três estágios de pipeline, o resultado da execução de uma determinada instrução só é observado três ciclos do sinal de relógio após ter sido lida da memória.

A verificação do comportamento do microprocessador foi feita, então, a partir das formas de onda geradas pelo ambiente de testes no simulador ModelSim. A Figura 24 mostra um exemplo de programa utilizado para verificação do ARM7.

No exemplo de programa executado, mostrado na figura 24, observa-se a execução de duas instruções, de processamento, mais especificamente uma subtração (sub), e uma instrução de acesso a memória, mais especificamente uma operação de escrita (str). Os estágios de processamento de cada instrução e a delimitação do tempo que cada uma delas leva para ser completamente processada estão explicitamente indicados na Figura 24. O momento exato em que o resultado da instrução fica pronto

está marcado na figura por setas (“Instruction 1 result” e “Instruction 2 result”). O local onde ocorre a mudança está marcado em amarelo.

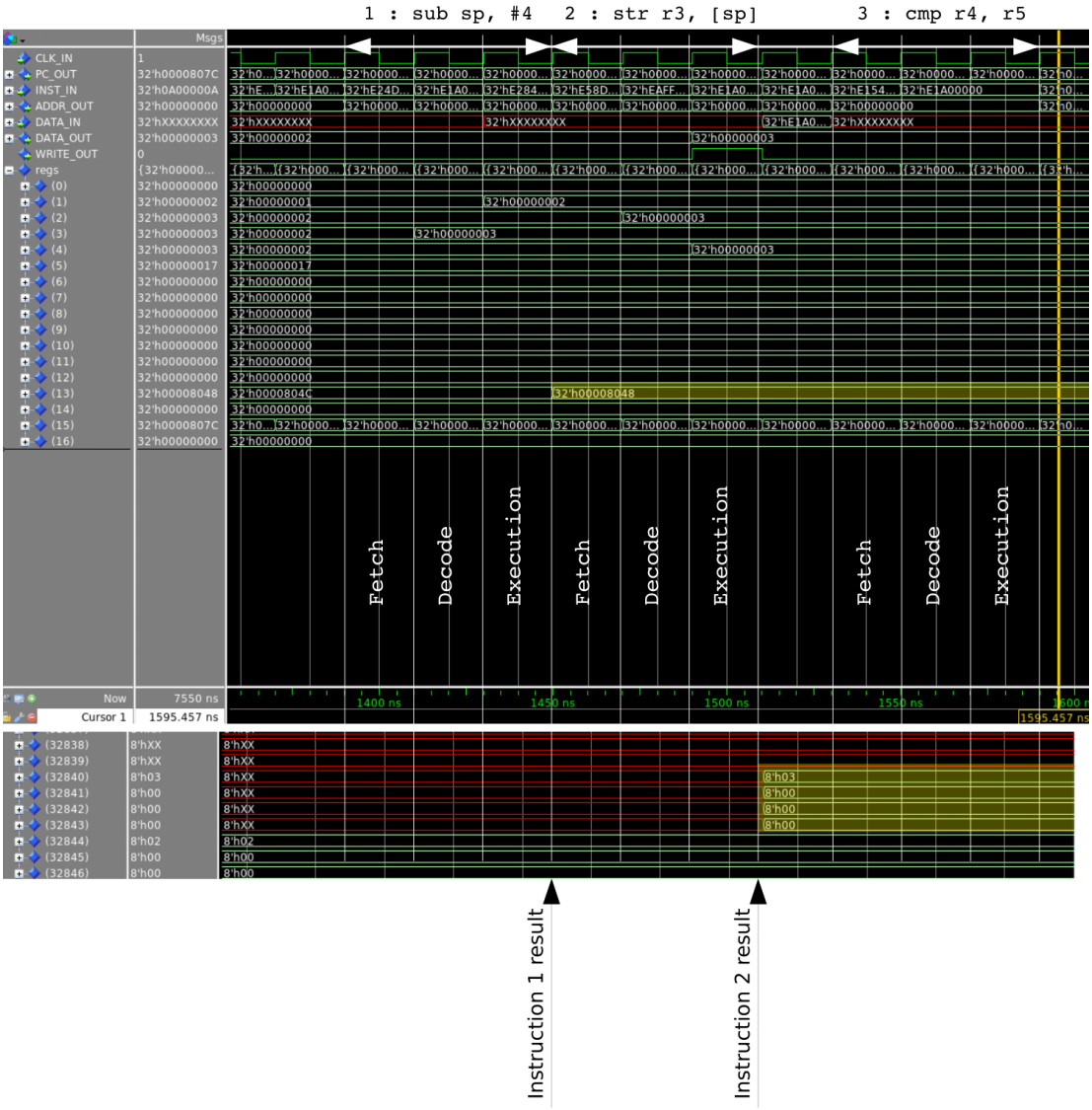

Figura 24. Forma de onda gerada pelo ambiente de testes do microprocessador ARM7

Após o procedimento de verificação funcional, alguns erros foram corrigidos, os quais encontram-se na Tabela 3.

Tabela 3. Erros encontrados na arquitetura do microprocessador ARM7 após verificação funcional.

Problema	**Causa do problema**	**Nível de verificação em que foi detectado**
As instruções permanecem no barramento de entrada por dois ciclos do sinal de relógio	Os registradores do módulo "Registres" estavam sendo atualizados de forma errada	Verificação da arquitetura completa
Instruções que seguem um branch eram executadas	Não havia nenhum módulo de previsão de branch.	Verificação da arquitetura completa
Algumas Instruções de processamento de dados davam resultados errados (CMP, CMN, TEQ e TST)	Não tinham sido implementas (CMP, CMN, TEQ e TST)	Verificação do módulo ALU
Instruções de acesso a memória não funcionavam	Não havia nenhum sinal que indicasse à memória se a operação de acesso a memória era uma leitura ou escrita.	Verificação da arquitetura completa
A primeira instrução era sempre executada duas vezes	Estado inicial do registrador r15 (PC) estava errado	Verificação da arquitetura completa
Alguns valores de endereços da próxima instrução eram incorretos	O cálculo de shift com rotação estavam sendo implementado de forma errada	Verificação da arquitetura completa

4.1.2 SÍNTESE LÓGICA

Uma vez terminada a fase de verificação funcional, prosseguiu-se o fluxo de projeto de CIs proposto com a síntese lógica do código VHDL que descreve o comportamento do circuito. Para tanto, utilizou-se a ferramenta de síntese lógica da Synopsys, Design Compiler.

Nessa etapa, conforme dito previamente, o código em nível RTL será convertido em uma netlist e em seguida mapeado à tecnologia FD-SOI 28nm. Células padrão da

tecnologia são utilizadas na fase de mapeamento com a tecnologia. Nesse ponto, deve-se escolher que características de transistores FD-SOI deseja-se utilizar. Conforme descrito na fundamentação teórica deste relatório, existem opções de células padrão com polybiasing ou sem, com plano de fundo convencional ou invertido.

Na síntese lógica do microprocessador ARM7 simplificado usou-se células padrão sem polybiasing e com substrato convenciona. Os resultados de síntese obtidos podem ser vistos nas figuras a seguir.

A Figura 25 representa uma visão geral do processador sintetizado, em termos de entradas e saídas. A Figura 26 mostra um circuito esquemático, gerado pela ferramenta de síntese, contendo todos os módulos da arquitetura interligados. A Figura 27 é uma ampliação da Figura 26, na parte do circuito que representa o módulo decodificador. Percebe-se que uma vez feita a síntese lógica, o circuito passa a ser representado realmente por portas lógicas.

Em termos de desempenho, a síntese lógica gerou um circuito com área de 5518.3 μm^2 e um consumo total de 0.17mW para uma frequência de 100 MHz. Os relatórios de desempenho gerados pela ferramenta de síntese encontram-se em anexo.

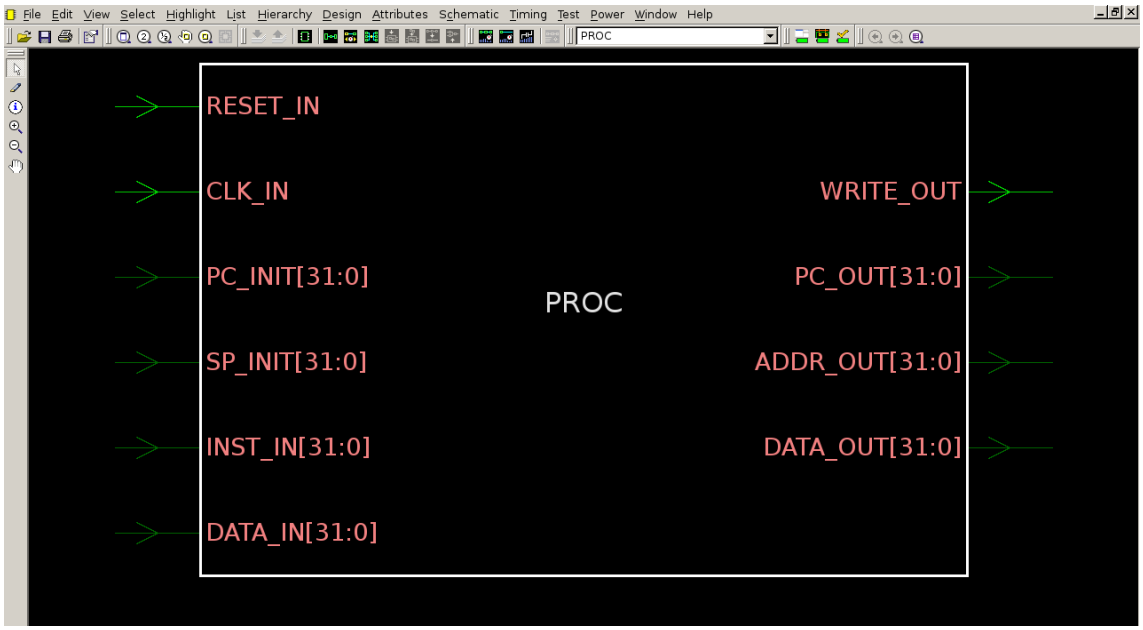

Figura 25. Vista externa do processador ARM7 sintetizado.

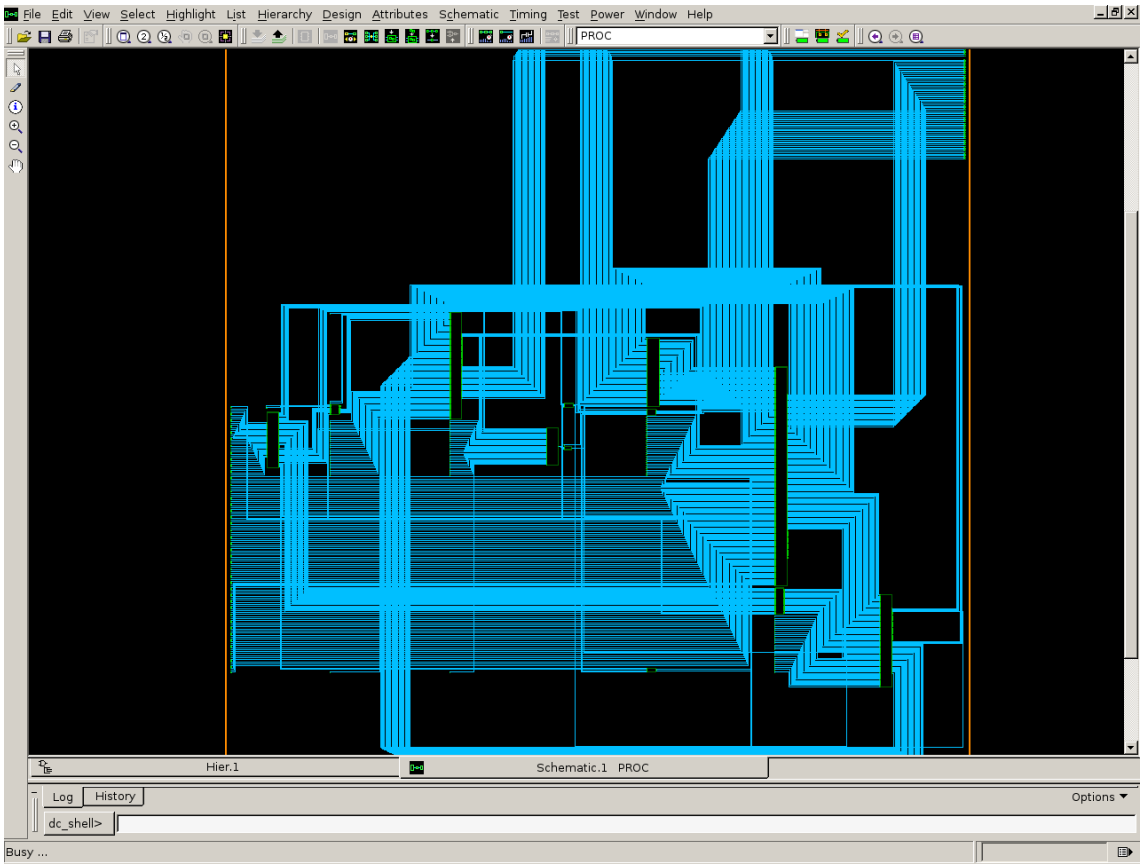

Figura 26. Circuito esquemático gerado pela ferramenta de síntese.

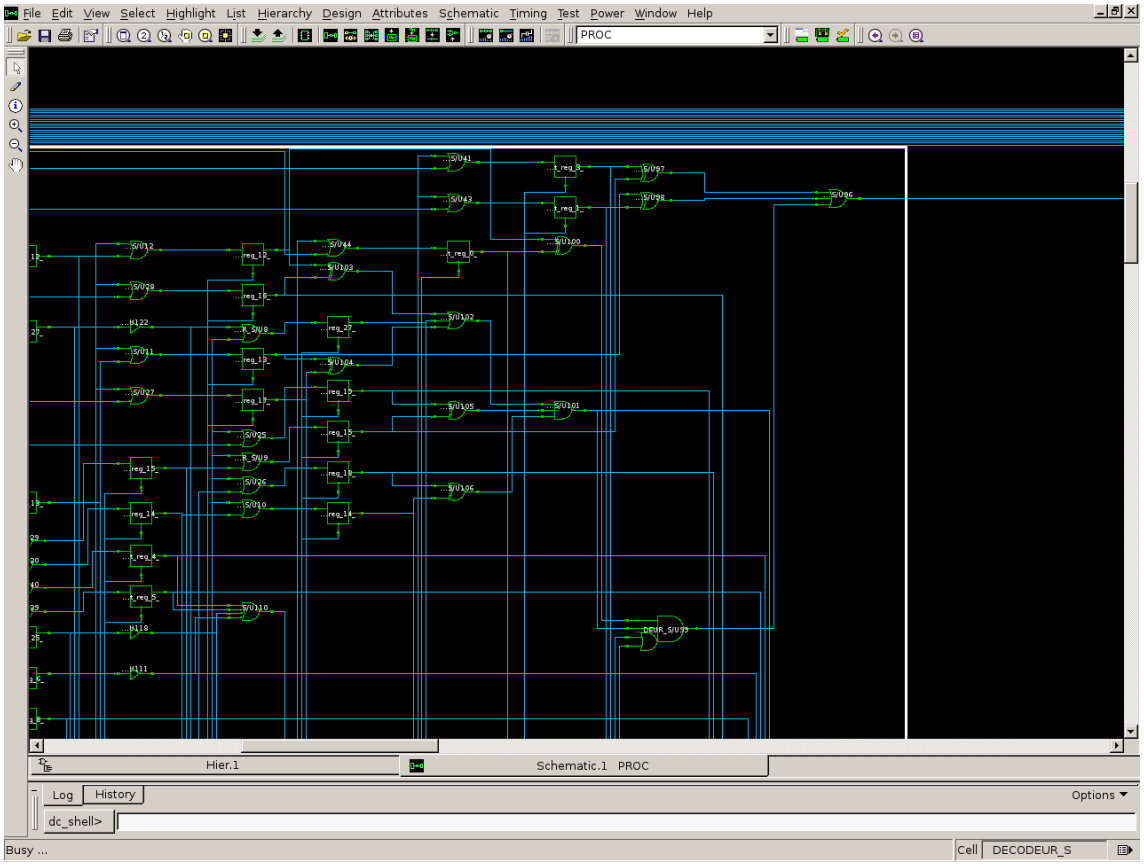

Figura 27. Parte do circuito do modulo Decodificador sintetizado.

4.1.3 CONCEPÇÃO DO LEIAUTE

Uma vez que a netlist foi gerada, sem nenhuma violação de timing e respeitando as restrições imposta, esta servira como ponto de partida para que o leiaute possa ser criado. Nessa tarefa utilizou-se o software Soc Encouter, da Cadence.

Algumas etapas são necessárias para que o leiaute seja produzido corretamente. A Figura 12 mostra o processo de concepção do leiaute em duas etapas, mas, na verdade vários, estas subdividem-se em várias outras etapas.

Primeiramente, especifica-se uma tentativa inicial de posicionamento das portas lógicas padrão que compõe a netlist obtida. Esse posicionamento inicial é feito em linhas e colunas que são definidas pela ferramenta de leiaute. Esse planejamento inicial é denominado Floorplan e é geralmente seguido da especificação das dimensões das linhas de metal que alimentarão o circuito (VDD e GND). A Figura 28 mostra o leiaute do processador ARM7 após essas etapas.

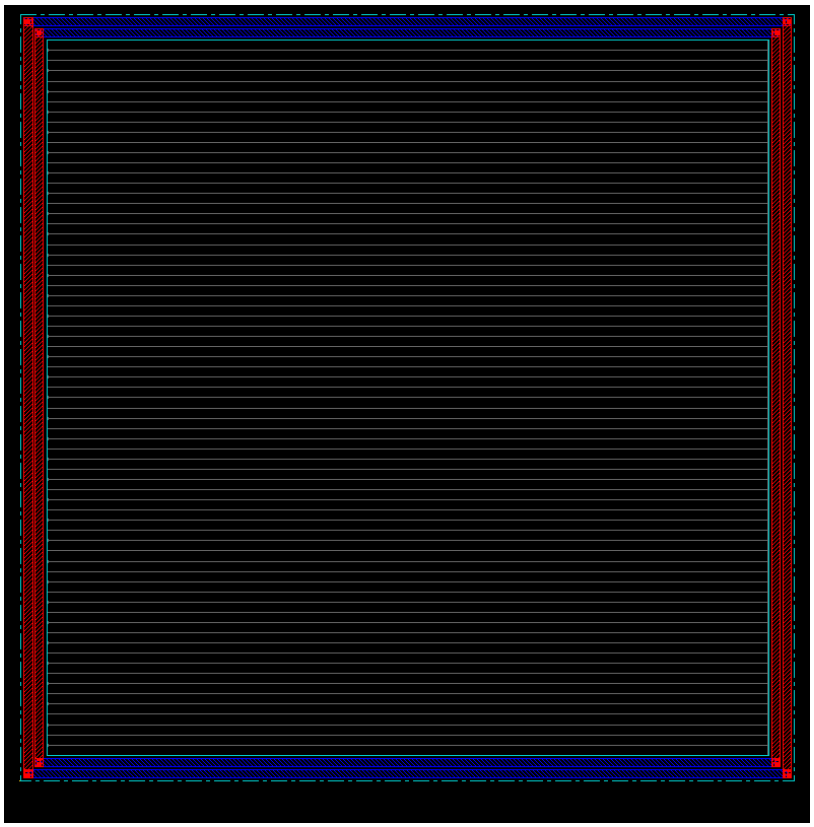

Figura 28. Leiaute do microprocessador ARM7 com linhas de alimentação (anéis em azul e vermelho) e *floorplan* inicial.

Em seguida, as células padrão são alocadas nas linhas predefinidas, conforme pode-se observar na Figura 29. Cada retângulo localizado entre as linhas de alimentação representa uma das portas lógicas do netlist. Essa etapa é denominada placement.

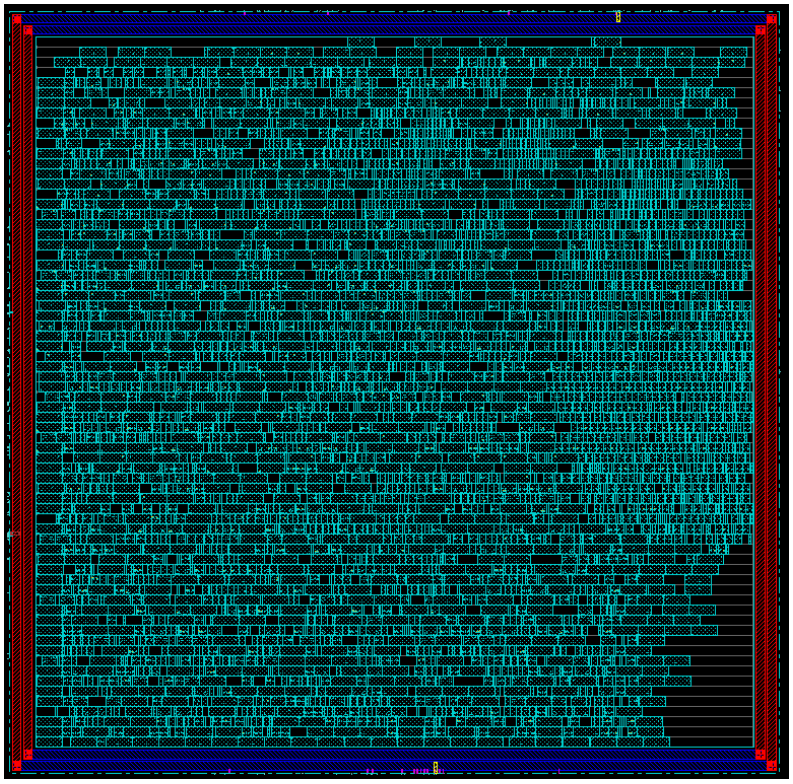

Figura 29. Leiaute do microprocessador ARM7 com células padrão alocadas.

Uma vez que as portas são alocadas, a árvore de relógio (do inglês *clock tree*) é sintetizada para obedecer às restrições de máximo *skew*, atraso, *jitter*, entre outros parâmetros, definidos na fase de síntese lógica. Para tanto a ferramenta de P&R pode rearranjar alguns módulos, ou até mesmo inserir buffers e inversores.

Em seguida, os espaços que não estão sendo ocupados são preenchidos com células de preenchimento (*filler cells*) para evitar problemas de planificação das camadas de metal e tornar o leiaute manufaturável. Uma vez terminado o preenchimento, inicia-se o processo de roteamento (*route*), no qual as células padrão são conectadas umas às outras, conforme especificado na netlist. Ao fim destas etapas, a ferramenta de concepção do leiaute faz algumas otimizações e correção de quaisquer violações que venham a ocorrer, e obtém-se o leiaute final, que pode ser visto na Figura 30.

O leiaute foi realizado em uma área total de 7117.5 µm2, apresenta um consumo total estimado em 0.23mW, para um sinal de relógio de 100MHz. O máximo *clock skew* reportado pela ferramenta de concepção de leiaute foi 29.1ps.

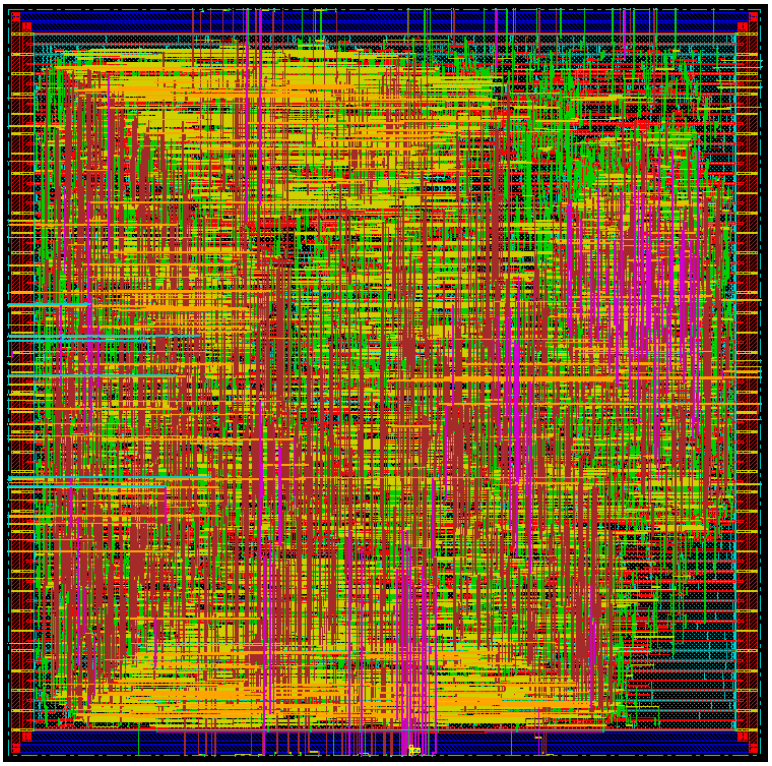

Figura 30. Leiaute final do microprocessador ARM7 projetado.

4.2 PROPOSTA DE FLUXO DE PROJETO DE CIS ASSÍNCRONOS QDI

O objetivo final dessa tarefa foi adaptar o fluxo de projeto de CIs utilizado na tarefa anterior para o paradigma de circuitos assíncronos QDI. Em função da limitação de tempo, apenas a fase de descrição do circuito em nível de portas lógicas, síntese lógica e simulação da descrição do circuito e da netlist obtida foram feitas.

Primeiramente, descreveu-se o circuito da Figura 31 em linguagem VHDL em nível de portas lógicas. Esse circuito representa uma célula de memória (*double half buffer*) amplamente utilizada em circuitos assíncronos QDI. Nesse circuito, observa-se dois estágios de um circuito assíncrono QDI, o primeiro deles composto pelos C-Elements C1 e C2 e pela porta N1; e o segundo estágio composto por C3, C4 e N2. O ambiente de testes utilizado para verificação do comportamento desse módulo simula uma comunicação em protocolo quatro fases com codificação de dados *dual-rail*.

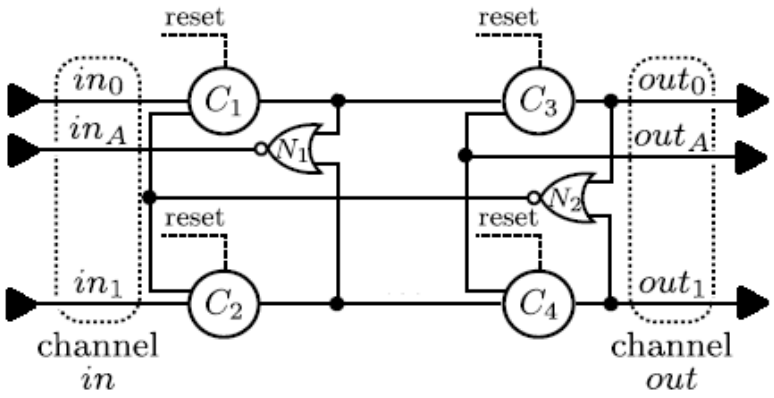

Figura 31. Circuito esquemático de um *Double Half Buffer*. (OUCHET et al., 2010. Modificado)

A simulação feita com a descrição do circuito em HDL pode ser vista na Figura 32. Nessa simulação, como não há informação sobre os atrasos nas portas que compões os circuitos as fases 1 e 3 começam e acabam no mesmo instante. O comportamento do circuito será melhor verificado na simulação pós-síntese.

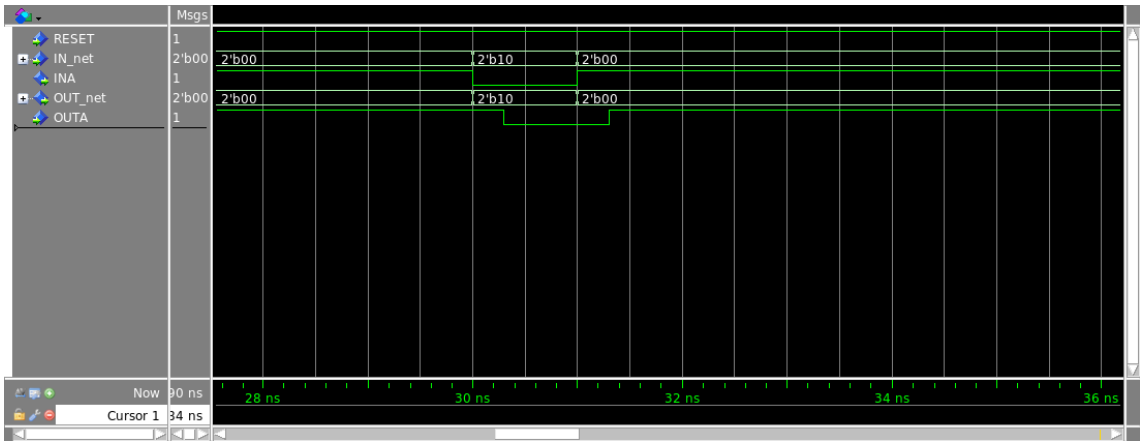

Figura 32. Simulação comportamental de um circuito *double half buffer*.

Para que se conseguisse fazer a síntese lógica, sem que a caracterização das portas C-Elements descrita na Figura 18 estar pronta, o seguinte procedimento foi adotado:

1. Criou-se uma biblioteca comportamental contento alguns tipos de portas de Muller, as quais continham informações de atrasos de subida e decida (*rise delay* e *fall delay*). As informações de atraso foram conseguidas a partir de simulações em nível Spice, as quais não serão discutidas pois não fazem parte do escopo deste trabalho;

2. O circuito descrito em VHDL foi sintetizado, com portas lógicas quaisquer postas no lugar das portas de Muller (denominadas portas *dummy*);
3. As portas dummy são substituídas pelas portas de Muller pertencentes à biblioteca criada no passo 1. Essas alterações são feitas no arquivo que representa a netlist e no arquivo SDF gerados pela ferramenta de síntese;
4. A simulação pós-síntese é feita com a netlist e o arquivo SDF modificados.

Como resultado, obteve-se a forma de onda da Figura 33. Nessa figura, as fases do protocolo quatro fases, referentes ao primeiro estágio do circuito descrito foram delimitadas por barras nas cores amarelo, vermelho e azul. Como a caracterização das células de Muller não ficaram prontas a tempo, esse circuito não passou da fase de simulação pós-síntese.

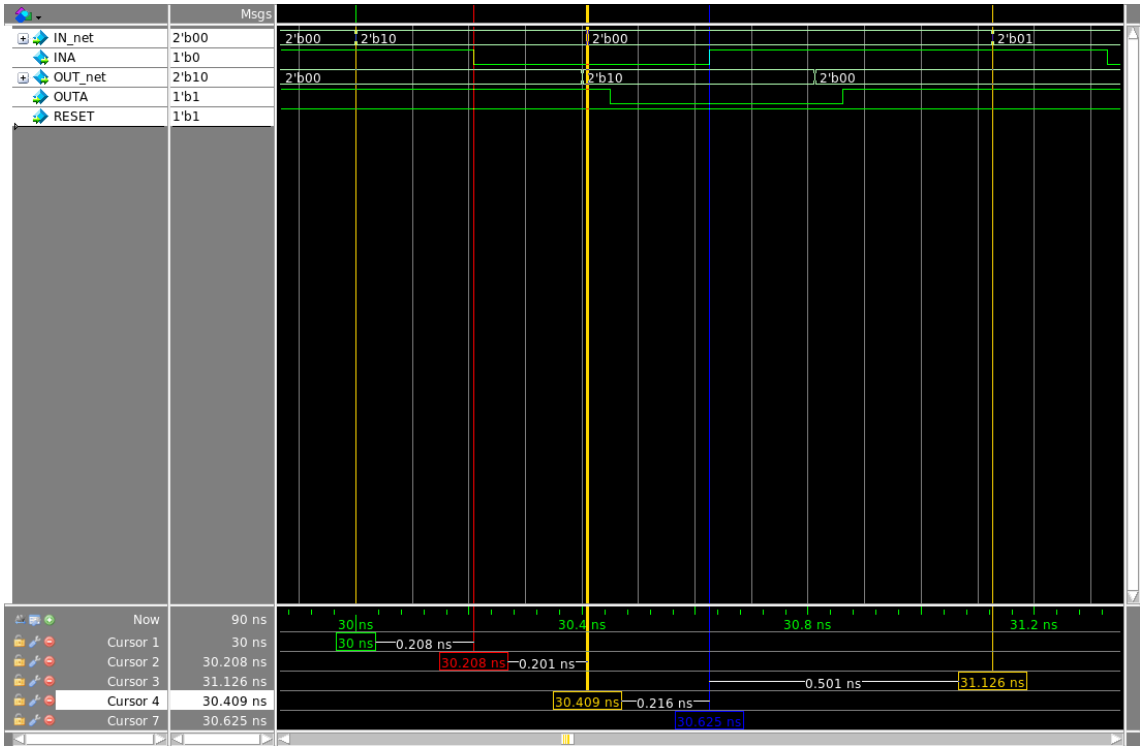

Figura 33. Simulação pós-síntese do circuito do *double half buffer*.

Em seguida, simulações foram feitas com um circuito somador completo assíncrono. O método DIMS, do inglês *Delay Insensitive Minterms Synthesis*, foi utilizado para determinar o circuito lógico que mapeia o comportamento desejado (SPARSO, 2006), chegando-se ao circuito da Figura 34. As portas que contêm um "C" no seu interior representam portas C-Element.

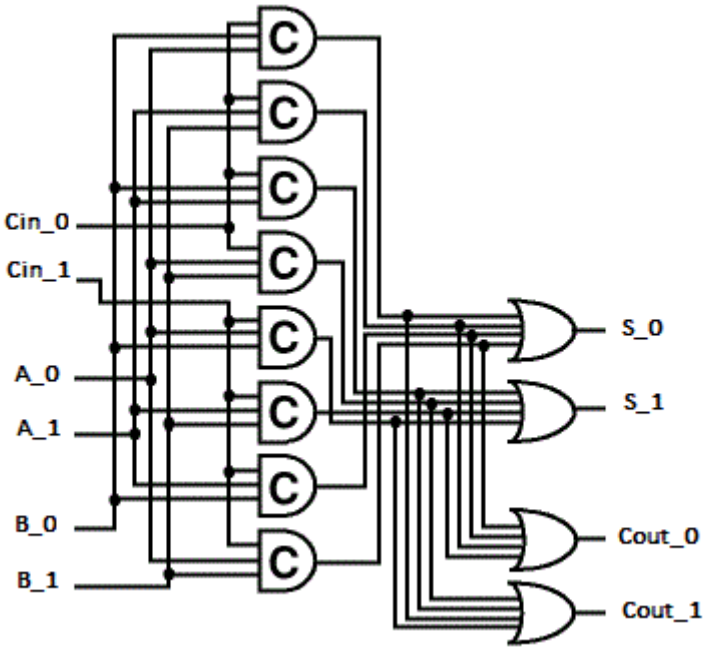

Figura 34. Circuito somador completo assíncrono.

Para este circuito, apenas simulação comportamental foi feita. O resultado dessa simulação pode ser visto na Figura 35. O ambiente de teste usado para testar o comportamento do circuito descrito simula um protocolo de comunicação em quatro fases, com retorno a zero. Entradas de saídas “01” codificam um bit “0” e caso sejam “10” codificam um bit “1”.

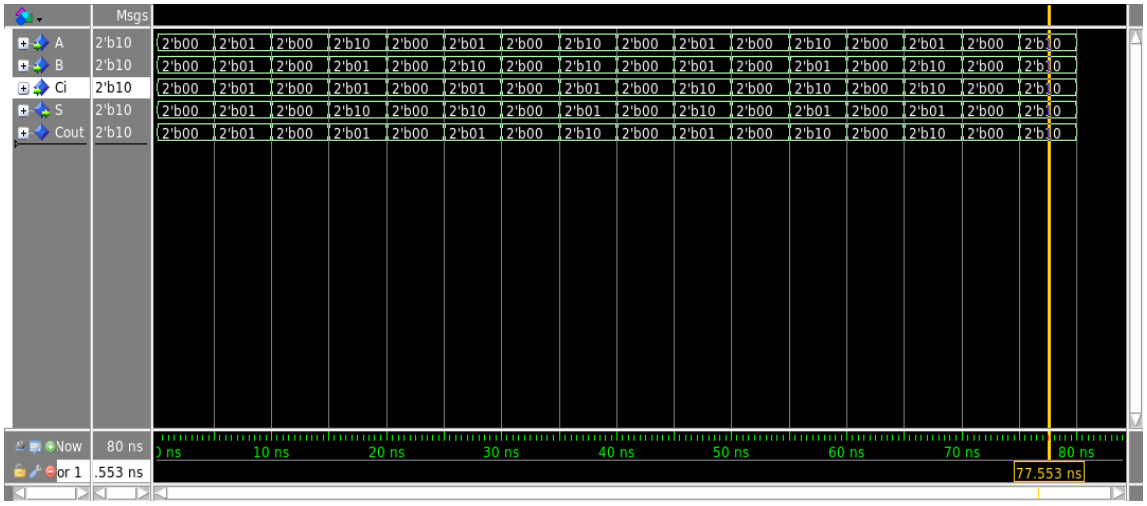

Figura 35. Simulação comportamental do circuito somador complete assíncrono.

5 CONCLUSÕES

Várias atividades distintas foram realizadas durante este estágio. A revisão bibliográfica feita, que compõe a fundamentação teórica deste relatório, foi essencial para o entendimento e desenvolvimento das mesmas.

A criação do leiaute no microprocessador ARM7, apesar de não representar, inicialmente, a tarefa principal do estágio, mostrou-se extremamente importante para o entendimento dos fluxos clássicos de projeto de CIs, das ferramentas normalmente utilizadas para verificação, síntese lógica e *place* e *route*, e para o melhor conhecimento da tecnologia FD-SOI.

Com a finalização do fluxo completo com um projeto real, percebeu-se a importância de cada uma das etapas do fluxo de projeto de CIs, o que foi fator preponderante para a compreensão das modificações necessárias no fluxo de projeto de circuitos integrados assíncronos.

Durante a fase de verificação, vários erros foram encontrados e corrigidos, conforme apresentado na seção de resultados, mas acredita-se que a aplicação de metodologias de verificação funcional padrão, como por exemplo UVM (*Universal Verification Methodology*) devem ser implementadas para que se consiga construir ambientes de verificação mais robustos e mais completos.

As etapas de síntese lógica e concepção do leiaute foram realizadas com apenas um tipo de células padrão, células com tensão de limiar reduzida e sem *polybiasing*. Contudo, analisar o desempenho de um mesmo circuito, sintetizado com células padrão que implementem as várias possibilidades de aprimoramento disponíveis na tecnologia FD-SOI 28nm, implementando, inclusive, esquemas de polarização do plano de fundo dos transistores, pode ajudar a compreender melhor as potencialidades dessa tecnologia.

No que concerne a proposição de um fluxo de projeto de Circuitos Integrados assíncronos QDI, o algoritmo testado para síntese lógica e simulação pós-síntese mostrou-se válido, uma vez que conseguiu-se simular *netlists* de alguns circuitos assíncronos simples. A concepção de leiautes, no entanto, depende de um procedimento de caracterização de portas de Muller e células de polarização conforme descrito no fluxograma da Figura 18, que serão importantes para a finalização do fluxo proposto pela Figura 17.

Referências Bibliográficas

Bastos, R. P. Transient-Fault Robust Systems Exploting Quasi-Delay Insensitive Asynchronous Circuits. Programa de Pós-Graduação em Microeletrônicas, Universidade Federal do Rio Grande do Sul, Porto Alegre,2010.

BEEREL, P. A. OZDAG, R. O.; FERRETTI, M. A Designer's Guide to Asynchronous VLSI. Cambridge University Press, 2010.

BERKEL, K. V; BURGESS, R.; KESSELS, J.; RONCKEN, M.; SCHAW, F. Asynchronous Circuits for Low Power: A DCC Error Corrector. IEEE Design & Test of Computers, 1994.

Cattaneo, M. Efeito de Corpo em transistores SOI de porta dupla vertical. Dissertação de Mestrado. Centro Universitário da FEI, São Bernardo do Campo, 2009.

CELLER, G. K.; CRISTOLOVENEAUNU, S. Frontiers of silicon-on-insulator. Journal of Applied Physics Volume 93, p. 4955 à 4978. American Institute of Physics, [S. l.], 2003.

Cristoloveanu, S.; Balestra, F. Technologie silicium sur isolant (SOI). Les Sélections : Dossier Techniques de l'ingénieur e2380. Éditions T. I., 2013.

CRISTOLOVENEAU, S.; LI, S. S.. Electrical characterization of silicon-on-insulator materials and devices. Springer, Norwell, 1995

DEMONE, P. The Stuff Dream Are Made Of [Part 2]. [S. l.], 2002. Disponível em: <http://www.realworldtech.com/cmos-logic/>. Acessado em: 06/2015.

Faynot, O.; Vandooren, A.; Ritzenthaler, R.; Poiroux, T.; Lolivier, J.; Jahan, C.; Barraud, S.; Ernst, T.; Adrieu, F.; Casse, M.; Giffard, B; Deleonibus, S.. Advanced SOI MOSFETs: structures and devices physics. Silicon-on-Insulator Technology and Devices XII, p.: 1 a 10. The Electrochemical Society, Inc, [S. l.], 2005.

FLATRESSE, F. UTBB-FDSOI Design & Migration Methodology. ST Microelectronics. [S. l.], 2013. Disponível em: <http://cmp.imag.fr/documents/doc/UTBB-FDSOI%20Design%20and%20Migration%20Methodology_.pdf>. Acessado em: 07/2015.

Greaves, D. J. Four-Phase Handshake in Synchronous, Asynchronous and Behavioural Forms – Revision Notes. University of Cambridge, Cambridge, 2013. Disponível em: <https://c;.cam.ac.uk/~djg11/wwwhpr/fourphase/fourphase.html>. Acessado em: 06/2015.

Hamon; J., Beigne; E.. Automatic Leakage Control for Wide Range Performance QDI Assynchronous Circuits in FD-SOI technology. IEEE Computer Society: International Symposium on Asynchronous Circuits and Systems, [S. l.], 2013.

Jacquet, D. Architectural choices & design-implementation methodologies for exploiting extended FD-SOI DVFS & body-bias capabilities. SOI technology Summit, VLSI Symposium, Shanghai, 2013

JONES, H. Economic impact of the technology choices at 28nm/20nm. International Business Strategies. Los Gatos, 2012. Disponível em: http://www.soiconsortium.org/pdf/Economic_Impact_of_the_Technology_Choices_at_28nm_20nm.pdf . Acessado em 05/2015.

KOOMEY, J.G. ; Berard, S. ; Sanchez, M.; Wong, H. Implications of Historical Trends in the Electrical Efficiency of Computing. Annals of the History of Computing, IEEE, [S. l.], 2011.

Liu, Q. et al. Impact of Back Bias on Ultra-Thin Body and BOX (UTBB) Devices. Proc. Symp. VLSIT, [S. l.], 2011, pp. 160–161.

MARTIN, A. J. The Limitations to Delay-Insensitive in Asynchronous Circuits. The 6th MIT Conference on Advantage Research in VLSI. Proceedings MIT Press, 1990.

Mentor Graphics. Fully Depleted (FD) vs Partially Depleted (PD) SOI. Advanced Substrate News. SOITEC, [S. l.], 2008. Disponível em: <http://www.advancedsubstratenews.com/2008/05/fully-depleted-fd-vs-partially-depleted-pd-soi>. Acessado em: 05/2015.

MOORE, G. E. Cramming More Components onto Integrated Circuits. Proceedings of the IEEE, vol. 86, n 1. IEEE, [S. l.], 1998.

Rabaey, J. M.; Chandrakasan, A.; Nikolic, B. Digital Integrated Circuits, A design Perspective. 2nd Ed. Prentice Hall, 2003.

SINGH, R. K.;SAXENA, A.; RASTOGI, M.. Silicon on Insulator Technology Review. International Journal of Engineering Sciences & Emerging Technologies, 2011. Volume 1, Issue 1, pp: 1-16

SKOTNICKI, T. Competitive SOC with UTBB SOI. SOI Conf. , Phoenix A., 2011

SPARSØ, J. Asynchronous Circuit Design, A Tutorial. Technical University of Denmark, 2006.

ST MICROELECTRONICS. Learn more about FD-SOI. [S. l.], 2014 Disponível em: <http://www.st.com/web/en/about_st/learn_fd-soi.html>. Acesso em: 05/2015. [2014a]

ST MICROELECTRONICS. An introduction to FD-SOI. St Online Media, [S. l.], 2013 Disponível em: <http://www.youtube.com/watch?v=uvV7jcpQ7UY> Acesso em: 04/2015.

ST MICROELECTRONICS. Continuing Moore's law. [S. l.], 2014. Disponível em: <http://www.st.com/web/en/about_st/moore_law.html>. Acesso em: 05/2015. [2014b]

HOARE, C. A. R. Communicating Sequential Processes. Englewood Cliffs, NJ: Prentice-Hall, 1985.

Martin, A. J. Synthesis of Asynchronous VLSI Circuits. Technical Report. California Institute of Technology, 1991.

ZHOU, R.; CHONG, K. S.; GWEE, B. H.; CHANG, J. S.; HO W. G. Synthesis of Asynchronous QDI Circuits using Synchronous Coding Specifications. Circuits and Systems (ISCAS), IEEE International Symposium, 2014.