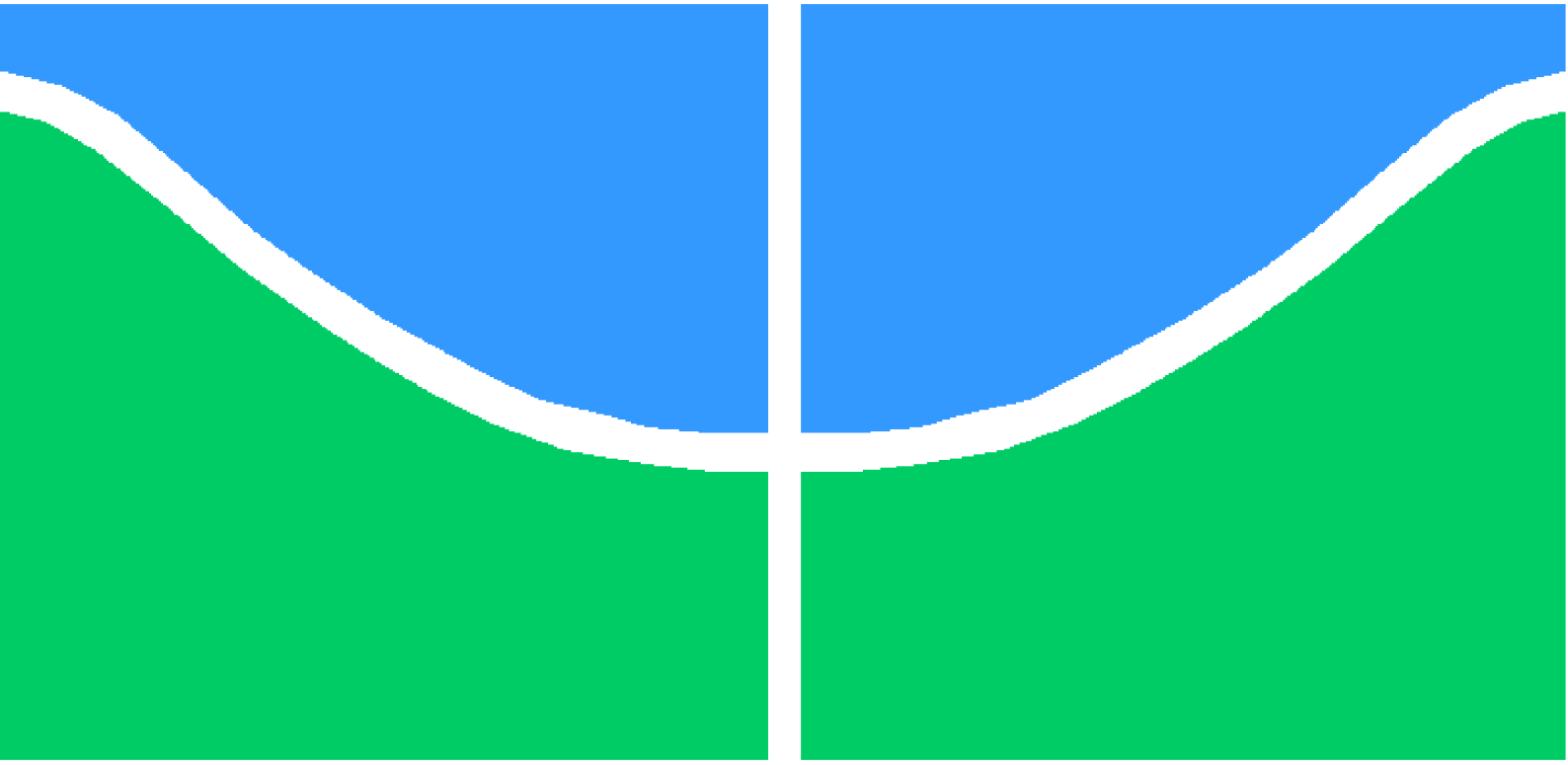

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Eletrônica

Simulação do controle de memória de uma tag passiva de RFID

Autor: Felipe Costa de Assis
Orientador: Prof. Dr. Gilmar Silva Bezerra

Brasília, DF
2018

Felipe Costa de Assis

Simulação do controle de memória de uma tag passiva de RFID

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Gilmar Silva Bezerra

Brasília, DF

2018

Felipe Costa de Assis

Simulação do controle de memória de uma tag passiva de RFID/ Felipe Costa de Assis. – Brasília, DF, 2018-

61 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Gilmar Silva Bezerra

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2018.

1. RFID. 2. Circuito integrado. I. Prof. Dr. Gilmar Silva Bezerra. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Simulação do controle de memória de uma tag passiva de RFID

CDU 02:141:005.6

Felipe Costa de Assis

Simulação do controle de memória de uma tag passiva de RFID

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Trabalho aprovado. Brasília, DF, 01 de junho de 2018:

Prof. Dr. Gilmar Silva Bezerra
Orientador

Prof. Dr. Wellington Avelino do Amaral
Convidado 1

Prof. Dr. Daniel Chaves Café
Convidado 2

Brasília, DF
2018

Dedico este trabalho à todos aqueles que me apoiaram em minha caminhada.

Agradecimentos

Agradeço aos meus pais, Geilson e Valéria, por sempre estarem presentes em todos os estágios de minha vida e serem a pedra fundamental de minha formação como cidadão; minha irmã Camila, por sempre alegrar meus dias com sua companhia incessante; meus maravilhosos avôs e avós Antonio, Dilo, Dione e Edíbia que sempre estiveram presentes em minha vida e são as bases de nossa família; além de todos os demais familiares.

Agradeço ao Prof. Dr. Gilmar Silva Beserra por me orientar neste trabalho e à todos os vários professores que moldaram minha formação e hoje são nada menos que modelos de profissionais que pretendo seguir pelo resto de minha vida acadêmica.

E claro, agradeço aos amigos que são fontes inestimáveis de carinho e companheirismo. Agradeço aos meus amigos de longa data, Davi, Diego, Henrique, Lucas, Nelson e Sérgio; aos companheiros de curso Alberto, Davi Gusmão, Johnson, Karoline, Matheus e Ricardo; e aos meus companheiros de intercâmbio aos quais eu considero como uma segunda família, André, Bruno, Leo, Mariana e Thiago.

"It may be that our role on this planet is not to worship God - - but to create him.
(Arthur C. Clarke)

Resumo

A tecnologia de RFID (*Radio-Frequency Identification*) tem sido utilizada em várias aplicações, tais como controle de acesso, pagamento de transporte, rastreamento de produtos, animais e veículos, pelo fato de oferecer uma alternativa de baixo custo para identificação de pessoas e objetos, sem necessitar de um operador para capturar os dados. Um sistema típico de RFID possui um leitor (*reader*) que envia um sinal para uma etiqueta (*tag*), recebe a resposta correspondente e armazena as informações em uma base de dados. Uma *tag* típica, por sua vez, contém antena, modulador, demodulador e memória, que pode, por exemplo, armazenar a informação de identificação do objeto a ser enviada para o leitor.

Trabalhos anteriores de graduação abordaram o projeto e implementação de uma *tag* passiva de RFID de 13,56 MHz, sendo que o bloco de *front-end* analógico/RF foi modelado em Verilog-AMS e implementado em tecnologia TSMC 0.18 um. Os blocos digitais foram modelados em Verilog e simulados isoladamente, considerando que a comunicação entre o leitor e a *tag* utiliza o protocolo ISO/IEC 14443 e que inicialmente a *tag* envia somente o número de identificação para o leitor. Considerando esse contexto, este projeto tem como objetivo geral a verificação dos blocos digitais da *tag* em conjunto com o sistema projetado nos trabalhos anteriores através de simulações mistas e análises comparativas de desempenho.

Palavras-chaves: RFID, Circuitos Integrados, Verificação, Simulação Mista.

Abstract

The RFID technology has been used in different applications, such as access control, transportation payment, tracking of products, animals and vehicles because it can offer a low-cost option in the field of identification of persons and objects without needing an operator to capture data. A typical RFID system is composed of a reader and a tag. The reader sends a signal to the tag, receives the corresponding answer and stores the information in a data bank. The tag is equipped with a memory chip, which stores the identification of the object. This information is read and sent back to the reader. A typical tag has an antenna, modulator, demodulator and memory chip.

Previous graduation work approached the project and implementation of a 13,56 MHz passive tag, with the analogic/RF front-end block being modelled in Verilog-AMS and implemented on 0.18 um TSMC technology. The digital blocks were modelled in Verilog and simulated, considering that the communication between the reader and the tag uses the ISO/IEC 14443 protocol and that initially the tag sends only the identification number to the reader. In this context, the aim of this work is the verification and implementation of the digital blocks that will be used in the system designed in previous work through mixed-signal simulations and comparative performance analysis.

Key-words: RFID, Integrated Circuits, Verification, Mixed-Signal Simulation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ASK	*Amplitude Shift keying*
FPGA	*Field Programmable Gate Array*
HDL	*Hardware Description Language*
IC	*Integrated Circuit*
ID	*Identification*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
RF	*Radio Frequency*
RTL	*Register Transfer Level*

Sumário

1 Introdução

RFID (*Radio Frequency Identification* ou Identificação por Rádio Freqüência) é uma tecnologia utilizada para identificar, rastrear e gerenciar desde produtos e documentos até animais ou mesmo indivíduos, sem contato e sem a necessidade de um campo visual.(COE C, 2005) Diversas situações comuns no dia a dia da população podem se tornar mais seguras ou ágeis com o uso da tecnologia de RFID, desde controles de acesso até gerenciamento de estoques. Por não necessitar de contato manual, a força de trabalho que, em situações sem o uso de tecnologias de identificação, estaria ocupada realizando estas tarefas, pode dedicar seu tempo a outras atividades, tornando processos empresarias ou industriais muito mais eficientes.

Apesar de o RFID ter algumas funções similares às do código de barras, cada vez mais se comprova que o RFID, com suas etiquetas inteligentes, completa o código de barras. A perspectiva é que uma grande revolução na gestão da cadeia de suprimentos será proporcionada através da larga adoção de RFID, fornecendo informações em tempo real para o seu gerenciamento.(COE C, 2005)

Atualmente, procedimentos de identificação automática se tornaram populares em muitas empresas de serviços, logísticas de compra e distribuição, indústria e manufaturas. Esses procedimentos existem para prover informações sobre pessoas, animais, bens e produtos em trânsito. (FINKENZELLER, 2010)

A presença universal de etiquetas de códigos de barra que proporcionou uma revolução nos sistemas de identificação algum tempo atrás está se tornando obsoleta em um número cada vez maior de situações. Apesar do baixo custo, sua pequena capacidade de armazenamento de dados e o fato de não serem reprogramáveis são pontos negativos. (FINKENZELLER, 2010)

As projeções futuras para este campo são extremamente motivadoras. A crescente necessidade de identificação imediata faz do uso dessa tecnologia algo não só preferível, como obrigatório para diversas aplicações. Em um futuro cada vez mais conectado, onde as informações devem ser recebidas e processadas o mais rápido possível e o não processamento de ditas informações em tempo hábil podem ser a diferença entre o lucro ou o prejuízo de um negócio, ou a vida e a morte de um animal, a tecnologia RFID possui um potencial enorme para impactar o futuro da sociedade, tal como ocorreu com outras tecnologias de identificação, como o código de barras.

1.1 Contextualização

O projeto da *tag* é um trabalho continuado, sendo temas de diferentes trabalhos de conclusão de curso e de alunos participantes de iniciação científica. Há blocos em diferente estágios de produção, alguns ainda em fase de projeto, de layout ou fabricado e caracterizados.

Em (GUIMARãES, 2013), foi feita a modelagem de duas arquiteturas propostas para um sistema de RFID usando a linguagem SystemC e sua extensão SystemC-AMS. Com isso, foi possível verificar a funcionalidade de cada arquitetura através de simulações em alto nível de abstração, incluindo as partes analógicas e de RF.

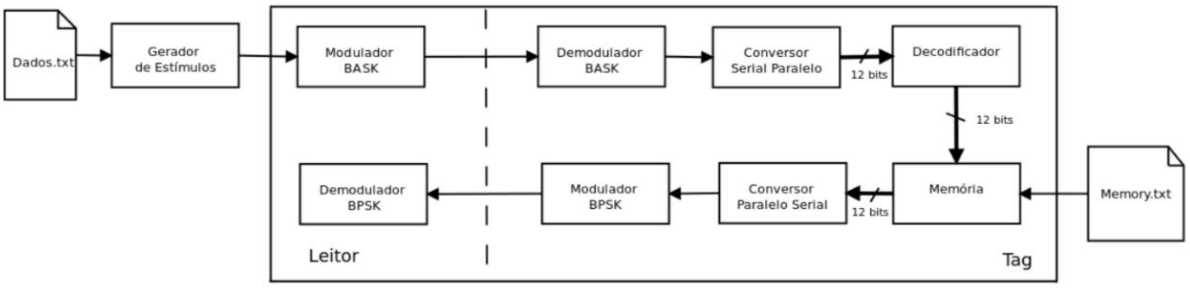

Figura 1 – Sistema modelado em SystemC-AMS.(GUIMARãES, 2013)

Em (FILHO, 2014), os blocos constituintes do *front-end* analógico de uma tag passiva de RFID para 13,56 MH, cujo diagrama de blocos está mostrado na Figura 2, foram desenvolvidos utilizando a linguagem Verilog-AMS. O projeto elétrico de um demodulador ASK foi realizado usando a tecnologia TSMC 0.18 um e utilizado em simulações mistas com os blocos descritos em Verilog-AMS. O layout do demodulador foi enviado para fabricação, juntamente com outros blocos individuais (AMARAL et al., 2014).

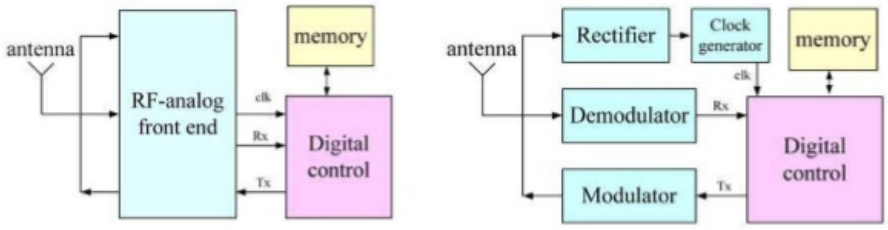

Figura 2 – Diagrama de blocos do sistema.(LI, 2009)

A caracterização do demodulador e do *bandgap* foi feita em (NETO, 2015), juntamente com o projeto e layout dos blocos individuais restantes do *front-end* analógico da tag. Para validar os circuitos projetados, foram realizadas simulações mistas entre blocos modelados em Verilog-AMS e esquemáticos em nível de transistores.

Por fim, para o projeto dos blocos digitais da tag, um protocolo anticolisão foi descrito em Verilog e simulado em (FARIA, 2017), considerando que a comunicação entre o leitor e a *tag* segue o protocolo ISO/IEC 14443 e que inicialmente a *tag* enviará apenas o ID para o leitor, sendo que poderá ocorrer colisão de dados quando o leitor acionar mais de uma *tag*. Não foi realizado o layout deste bloco projetado.

A figura 3 mostra em resumo o estado dos diferentes blocos que constituem a *tag*

	Componente	Projeto	Layout	Fabricação	Caracterização	Reprojeto
Demodulador	Detector de involtória					
	Filtro passa-baixa					
	Comparador					
Retificador	BandGap					
	LDO					
	Charge Pump					
	Modulador					
	Clock generator					
	Digital control					

?

Figura 3 – Estado dos diferentes blocos do projeto.

1.2 Objetivo Geral

Como continuação dos trabalhos anteriores a respeito do tema, este trabalho tem como objetivos gerais a verificação do sistema usando simulações mistas entre os blocos do *front-end* analógico e os blocos digitais, bem como a preparação para síntese lógica e física dos blocos do controle digital da *tag* passiva de RFID na frequência de 13,56MHz.

1.3 Objetivos Específicos

Para simular a *tag* completa, os blocos descritos em Verilog-AMS e Verilog serão gradativamente adicionados. Após a validação, será realizada a preparação dos arquivos para a geração do layout dos blocos digitais, na tecnologia TSMC 0.18 um, utilizando ferramentas de síntese automática da Cadence. Sendo assim, os objetivos específicos são:

- Modelagem e simulação dos blocos digitais da *tag*;
- Simulação mista dos blocos digitais com os blocos analógicos da *tag*;
- Adaptação dos scripts de síntese para os arquivos da biblioteca da TSMC 0.18um e para as ferramentas de síntese da Cadence;

1.4 Estrutura do Trabalho

O trabalho está dividido conforme a descrição a seguir. Inicialmente, é apresentada uma introdução geral ao tema, expondo a motivação, contextualização e os objetivos do trabalho. O Capítulo 2 contém uma fundamentação teórica sobre RFID, seus componentes e funcionamento. No Capítulo 3, é feito um estudo sobre projetos de circuitos integrados digitais, abordando o fluxo de projeto e ferramentas utilizadas. O Capítulo 4 contém a descrição da modelagem dos blocos digitais para a realização da simulação mista, e o Capítulo 5, os resultados da verificação. Por fim, o Capítulo 6 contém a conclusão, com considerações sobre os resultados obtidos e sugestões para trabalhos futuros. Como não foi possível realizar as sínteses lógica e física devido à falta de licença da ferramenta, a preparação dos arquivos e descrição dos passos necessários para gerar o layout se encontram no Apêndice A.

1.5 Aspectos Metodológicos

Este trabalho iniciou-se com uma revisão biblíográfica sobre a tecnologia de RFID, suas aplicações e sistemas. Em seguida, fez-se necessária a realização de diversos tutoriais para a aprendizagem do uso das ferramentas de simulações. No caso de simulações mistas,

o software utilizado foi o Virtuoso da Cadence, no qual foram reproduzidos os resultados de trabalhos anteriores contendo esquemáticos em nível de transistores e descrições em Verilog-AMS dos blocos de *front-end*. Neste contexto, o uso do Verilog-AMS é necessário pois o sistema possui sinais mistos, sendo necessário a conexão de blocos Verilog (para sinais discretos) e Verilog-A (para sinais analógicos). Nas simulações mistas, um ou mais blocos do *top level* são substituídos por implementações em outro nível de abstração e é verificado seu funcionamento no contexto do sistema, nestas simulações, é utilizada a metodologia *top-down*.

Após essa etapa, foi realizada a modelagem em Verilog dos blocos digitais a serem integrados aos blocos já existentes a fim de verificar seu funcionamento utilizando as simulações mistas para a tag completa. Os blocos modelados foram um conversor serial-paralelo, uma memória e um conversor paralelo-serial, bem como os sub-blocos que os compõem, através da metodologia *top-down*.

Após realizar as simulações dos blocos digitais isolados, a conexão entre os blocos digitais e analógicos e as simulações mistas, os resultados foram analisados utilizando como base os trabalhos anteriores para verificar se o funcionamento da tag estava de acordo com as especificações.

Por fim, a última etapa, que consistia na implementação dos blocos digitais na tecnologia TSMC 0.18 um utilizando ferramentas de síntese automática, não foi realizada devido a limitações de acesso às ferramentas, cuja licença não estava mais disponível. Sendo assim, foi elaborado um roteiro contendo os passos necessários para a preparação dos arquivos e realização das sínteses lógica e física das estruturas digitais validadas usando ferramentas da Cadence.

2 *Radio Frequency Identification* (RFID)

Identificação por radio frequência (RFID) é uma tecnologia de identificação automatizada que utiliza etiquetas (*tags*) para transmitir dados para leitores de RFID (LI, 2009) . Esta tecnologia possui diversas aplicações, sendo utilizada amplamente no mercado em cartões de identificação pessoal, identificação e controle de animais e processos migratórios, rastreamento de mercadorias em cadeias de fornecimento e aplicações gerais de monitoramento e identificação. Seu baixo custo de implementação e múltiplas utilidades fazem com que essa tecnologia venha crescendo de forma exponencial em diversos setores da sociedade.

A tecnologia RFID utiliza a radiofrequência para obter dados de uma *tag* de forma remota, sem a necessidade de ligações físicas e sem apresentar limitações para atravessar obstáculos como paredes ou encapsulamentos. A *tag* é fixada no objeto a ser monitorado e possui em sua memória os dados básicos sobre o objeto, como números de série. Esses dados são então recebidos pelo leitor quando a *tag* entra em seu campo magnético. Por fim, os dados são processados por uma central para que possa ser feita a identificação ou outras ações pertinentes. O diagrama da Figura 4 mostra o funcionamento geral do sistema completo.

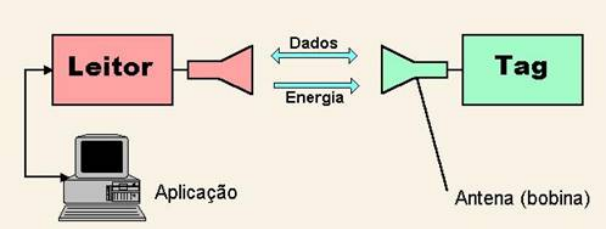

Figura 4 – Funcionamento geral do sistema RFID.(SANGREMAN, 2003)

A comunicação entre o leitor e a *tag* é controlada por protocolos, como por exemplo o ISO/IEC 14443, ISO/IEC 18001, etc. A escolha do protocolo depende de diversos fatores como frequência de operação, aplicação e o número de *tags* a serem identificadas. Os diversos protocolos existentes e suas particularidades serão abordados mais à frente.

2.1 Arquitetura de uma *Tag* de RFID

A *tag* é o componente fixado no objeto a ser monitorado e carrega todas as informações básicas que possam ser de interesse para o monitoramento, podendo ser uma *tag* apenas de leitura ou de leitura e escrita, dependendo da aplicação.

Este componente é a parte principal do sistema RFID e consiste basicamente em um *chip* e uma antena. O *chip* possui o bloco analógico e de RF, memória e um bloco de controle. Dependendo do tipo de tecnologia (ativa, semi-ativa ou passiva) e da frequência de operação, sua arquitetura para armazenamento de dados, comunicação e fonte de energia podem diferir. (MARKHAM, 2012)

No caso das *tags* passivas, sua fonte de alimentação é a própria onda eletromagnética incidente, não sendo necessária uma fonte de alimentação externa para a mesma. A *tag* absorve a energia da onda eletromagnética que carrega as informações enviadas pelo leitor.

O circuito da *tag* é formado por dois grandes blocos (controle digital e o bloco de *front-end* analógico) e uma memória (Figura 2). O circuito de *front-end* analógico, por sua vez, é formado pelos seguintes blocos(NETO, 2015):

- Retificador: bloco que retifica o sinal de RF de entrada e gera a tensão DC necessária para alimentar os outros blocos do sistema;
- Demodulador: Bloco receptor do sistema que detecta os comandos enviados pelo leitor. Também extrai o clock a partir do sinal RF recebido, que é necessário para sincronizar o RFID com o leitor.
- Modulador: Bloco transmissor do sistema que envia à *tag* o ID de identificação para o leitor RFID.
- Clock interno: Gerador de clock interno que fornece um clock gerado para a parte digital do circuito.

2.2 Frequências de funcionamento do sistema de RFID

Para a padronização do uso das *tags*, as normas técnicas disponíveis determinam os intervalos de frequência em que a *tag* deve funcionar para determinadas aplicações:

- *Low Frequency*: De 30 kHz até 300 kHz. As etiquetas desta faixa de frequências operam em 125 kHz ou 134,2 kHz. Geralmente, são etiquetas passivas e seu maior uso é na identificação de animais, na forma de brincos ou implantes subcutâneos, e de pessoas também na forma de implantes subcutâneos. Distância de leitura é de alguns centímetros;
- *High Frequency*: De 3 MHz até 30 MHz. Etiquetas construídas em 13,56 MHz, normalmente utilizadas como crachás para identificação individual ou então como meio de pagamento, por exemplo para o transporte público, tal como os bilhetes únicos. Também pode ser utilizado para identificar objetos individuais, como nas

lojas de departamento em sistemas antifurto. A distância de leitura chega a ser maior do que 10 cm;

- *Ultra High Frequency*: De 300 MHz até 1 GHz. Nesta faixa, as *tags* são fabricadas nas faixas de frequências de 433 MHz, para uso em rastreamento de cargas, tais como contêineres, vagões, caminhões e etc. Nessa faixa de frequências, as *tags* são ativas, ou seja, energizadas com baterias, e robustas. Outra faixa bastante utilizada é a de 868 MHz na Europa e de 915 MHz nos Estados Unidos e Brasil. Essas etiquetas também são empregadas em processos de rastreamento de ativos ou produtos em lojas, controle de estoque, inventários e identificação veicular, como por exemplo em pedágios eletrônicos;

- Microondas: Acima de 1 GHz. Duas frequências para RFID: 2,45 GHz e 5,8 GHz. Esta faixa de frequências é utilizada em aplicações industriais, científicas e médicas. No Brasil, a principal aplicação de *tags* nessa faixa de frequências é a de identificação veicular para o pedágio eletrônico. (PUHLMANN, 2015)

2.3 Protocolos de comunicação entre o Leitor e a *Tag*

A criação de padrões de comunicação entre o leitor e a *tag* é de extrema importância para que a tecnologia de RFID possa se consolidar no dia a dia da sociedade. Tais padrões asseguram que diferentes sistemas possam atuar em conjunto, eliminando problemas de compatibilidade e aumentando a segurança de ditos sistemas.

O desenvolvimento de padrões é de responsabilidade do comitê técnico de diferentes institutos de padronização. A ISO é uma união mundial de institutos de padronização e contribui com numerosos comitês e grupos de trabalho para o desenvolvimento de padrões para RFID(FINKENZELLER, 2010).

A Tabela 1 descreve as particularidades de alguns padrões.

ISO standard	**Descrição**
ISO 11784	RFID para animais - Estrutura de codigo
ISO 11785	RFID para animais - Concepção técnica
ISO/IEC 14443A,B	Cartões de identificação - Cartões de proximidade
ISO/IEC 18001, 15961, 15962, 15963	Gerenciamentos de diversos itens de RFID
ISO/IEC 18000-1 à 18000-4	Comunicação por ar para diversas frequências

Tabela 1 – Descrições de ISOS para RFID (MARKHAM, 2012)

ISO/IEC 14443 é um padrão internacional de quatro partes para cartões inteligentes sem contato operando à 13.56 MHz em proximidade do leitor e será utilizado neste trabalho.

Tais cartões destinam-se a operar a aproximadamente 10 cm da antena do leitor. Cada parte do padrão é responsável por certas características do cartão:

- Parte 1 [ISO/IEC 14443-1:2000(E)]: define o tamanho e as características físicas do cartão. Lista também as condições ambientais que o cartão deve ser capaz de suportar sem danos permanentes às suas funcionalidades.
- Parte 2 [ISO/IEC 14443-2:2001(E)]: define a potência de RF e a interface do sinal. São definidos na parte 2, dois esquemas de sinais, tipo A e tipo B. Ambos esquemas são *half-duplex* com 106 kbit por segundo de taxa de dados em cada direção. Os dados transmitidos pelo cartão são modulados com um *subcarrier* de 847.5 KHz. O cartão é alimentado pelo campo RF e não é dependente de baterias.
- Parte 3 [ISO/IEC 14443-3:2001(E)]: define os protocolos de inicialização e anticolisão para os cartões tipo A e B. Os comandos, respostas, *frame* de dados e o tempo do sistema de anticolisão serão definidos posteriormente. O esquema de inicialização e anticolisão é projetado para permitir a a construção de leitores de múltiplo protocolo capazes de se comunicar com placas de ambos tipos A e B.
- Parte 4 [ISO/IEC 14443-4:2001(E)]: define os protocolos de transmissão de alto nível para cartões de tipo A e B. Os protocolos citados são opcionais para o padrão ISO/IEC 14443, ou seja, os cartões de proximidade podem ser projetados com ou sem os protocolos.(ISO/IEC, 2005)

2.4 ASK - Amplitude Shift Key

Amplitude Shift Keying, no contexto de comunicações digitais, é um processo de modulação que atribui à uma senóide dois ou mais níveis discretos de amplitude. (ISLAND, 2012)

O sinal então modulado terá um valor zero para níveis lógicos baixos e o valor da saída do *carrier* para níveis lógicos altos.

3 Projeto de Circuitos Integrados Digitais

Neste capítulo, serão abordados os passos para o desenvolvimento de um projeto de circuitos integrados digitais, iniciando-se com a definição das especificações do projeto e validação de sua funcionalidade, e em seguida realizando-se as sínteses lógicas e físicas, finalizando o projeto com a verificação do layout obtido.

3.1 Especificação e Validação da Funcionalidade

Inicialmente, as áreas de interesse do projeto são identificadas para então se desenvolver planos de verificação (FILHO, 2014). As especificações devem ser totalmente respeitadas ao se modelar os blocos do sistema, ou seja, os formatos de dados, frequências de operação e todas as outras características de um bloco devem ser compatíveis com os demais blocos do sistema.

É neste ponto que as especificações determinadas por protocolos de comunicação são inseridas no sistema. Se essas regras forem seguidas, serão evitados problemas que poderiam atrasar o andamento do projeto (FILHO, 2014). Assim, ao se obter a descrição em alto nível do sistema, o mesmo pode ser validado por meio de simulações mistas em ambientes como o MatLab/Simulink ou em linguagens como C/C++, SystemC e SystemC-AMS.

3.2 Projeto RTL

Com as especificações definidas e a funcionalidade validada, o próximo passo é o projeto RTL. Nesta etapa, são usadas linguagens de descrição de *hardware* como o Verilog, SystemVerilog ou o VHDL para implementar os modelos funcionais obtidos na etapa anterior, utilizando-se componentes básicos como somadores e máquinas de estados.

É fundamental nesta etapa que os blocos funcionem sem nenhum erro e sigam à risca todas as especificações do projeto. Se estes blocos básicos não funcionarem como o requerido, todo o projeto pode apresentar falhas no fim. Por este motivo, as simulações de comportamento de cada bloco devem ser feitas e minuciosamente avaliadas.

3.3 Simulação Funcional

Para validar o funcionamento de cada bloco básico individual, é necessário realizar simulações comportamentais dos mesmos, analisando os sinais de saída de acordo com

diferentes sinais de entrada usados para testes.

É feita a analise de cada bloco básico individual, bem como dos blocos maiores que são formados a partir destes. No fim, o bloco de mais alto nível (também chamado de *top-level*) também deve ser simulado para que sua funcionalidade seja validada.

Essas simulações funcionais são feitas a partir de *testbenches*, que são códigos nos quais são adicionados sinais de entrada de teste e outros sinais pertinentes ao bloco para permitirem a análise dos sinais de saída. A figura 5 mostra um exemplo de *testbench* em Verilog.

```
1  module and_tb();
2
3    reg [3:0] a, b;
4    wire [3:0] out;
5
6  //declaração do módulo sob teste
7
8    basic_and #(.WIDTH(4)) DUT (
9      .a(a),
10     .b(b),
11     .out(out)
12   );
13
14 //Início do teste
15
16   initial begin
17     a = 4'b0000;
18     b = 4'b0000;
19     #20
20     a = 4'b1111;
21     b = 4'b0101;
22     #20
23     a = 4'b1100;
24     b = 4'b1111;
25     #20
26     a = 4'b1100;
27     b = 4'b0011;
28     #20
29     a = 4'b1100;
30     b = 4'b1010;
31     #20
32     $finish;
33   end
34
35 endmodule
```

Figura 5 – *Testbench* em Verilog.

É possível visualizar graficamente o comportamento dos sinais de entrada e de saída do módulo, como mostra a figura 6, em ferramentas como o NCSim, da *Cadence Design Systems*.

3.4 Síntese Lógica

Nesta etapa, o circuito comportamental que foi obtido na etapa anterior(em RTL) é sintetizado em termos de portas lógicas da biblioteca do fabricante, de acordo com a tecnologia na qual será implementado. Para esta tarefa uma ferramenta que pode ser

Figura 6 – Visualização dos sinais de entrada e saída.

usada é o RTL Compiler. Os arquivos necessários nesta etapa são os *scripts* de comandos, o código à ser sintetizado, os arquivos da tecnologia e as *constraints*. É então gerado um *netlist* (descrição estrutural) do que foi sintetizado, tal como relatórios de área, consume e atrasos.

3.5 Simulação Pós-Síntese Lógica

Nesta etapa, é feita a verificação da síntese. Uma ferramenta utilizada neste passo é o NCLaunch. É feita a compilação dos arquivos que contêm as descrições das células básicas da tecnologia, do arquivo SDF gerado na síntese lógica, do verilog da síntese lógica e de um *testbench* com os estímulos para a simulação. Então é realizada a verificação dos resultados e averiguado se o sistema está funcionando de acordo com o desejado.

3.6 Síntese Física

A síntese física se inicia com a *netlist* gerada pela síntese lógica. Essa *netlist* descreve as conexões lógicas entre os componentes físicos como portas lógicas, I/O *pins* e macro/IP *blocks*. A síntese física então gera uma nova *netlist* otimizada e um *layout* equivalente. Seu objetivo é de satisfazer uma combinação de tempo, área, energia e roteabilidade.(LI; ALPERT, 2011). A ferramenta *SoC Encounter* pode ser utilizada para este passo. Há vários passos a serem seguidos nesta etapa:

- ***Floorplanning***: Planta baixa do *layout*;
- ***Powerplanning***: Configuração da rede de distribuição de alimentação, com *rings* e *stripes*;
- ***Placement***: Distribuição de células no *core* do *layout*;
- ***Otimizações e Timing Analysis***: Melhorar atrasos e verificar caminhos críticos após o *placement*;
- ***Clock Tree Synthesisg***: Árvore de *clock*, que deve ser criada de forma a distribuir o sinal de forma homogênea entre vários pontos;

- ***Routing***: Interligação entre os pinos e blocos através de fios em várias camadas de metal;
- ***Filler Cells***: Preencher espaços vazios entre as células;
- ***Metal Fill***: Respeitar porcentagem de densidade de metal em cada uma das camadas utilizadas;
- ***Design Rule Check***: Verificar geometria;
- ***Layout Versus Schematic***: Verificação da conectividade entre os elementos do *layout* e comparação com o modelo no formato **spice**.

3.7 Simulação Pós-Layout

Na simulação Pós layout, uma ferramenta que poder ser utilizada é a NCLaunch. É realizada a compilação dos arquivos que contêm descrições das células básicas da tecnologia, o arquivo SDF, o verilog da síntese física e o *testbench* com os estímulos utilizados na simulação.

3.8 Verificação de*Layout*

A última etapa do projeto consiste em verificar a funcionalidade do *layout* obtido. Utilizando ferramentas computacionais, é necessário se certificar que a performance, tamanho, densidade e outros fatores estão dentro dos especificados na etapa inicial do projeto. Com um *layout* seguindo as especificações e não apresentando problemas elétricos ou de outra natureza, o *chip* está pronto para a fabricação.

4 Modelagem do Sistema

Como dito no capítulo 4, para se realizar a verificação do funcionamento do sistema, é necessária a simulação mista dos blocos analógicos com os blocos de controle digital, em específico um bloco de memória. Como a *tag* envia e recebe dados em série, e um bloco de memória digital necessita de tais dados em paralelo, é necessária a modelagem de um conversor serial-paralelo, e um paralelo-serial para integrar o sistema.

O sistema digital idealizado e que foi modelado respeita então o diagrama de blocos mostrado na figura 7.

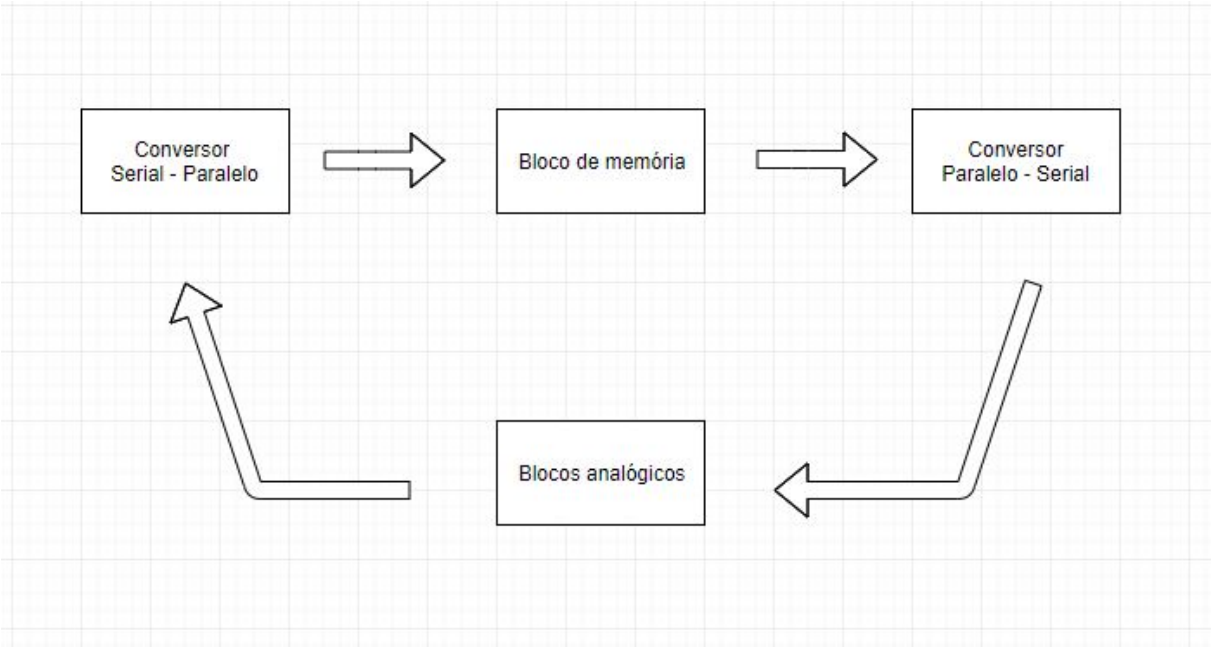

Figura 7 – Diagrama de blocos do sistema digital.

Neste capítulo, será abordada a modelagem destes blocos digitais e os procedimentos que foram realizados para a conexão dos mesmos com os blocos analógicos para validar o funcionamento da *tag*. A escolha do tamanho da palavra de dados é arbitrária, porém ela deve ser decidida antes da modelagem do sistema, pois todos os blocos devem ser devem usar um tamanho comum. Para a modelagem deste sistema, o tamanho da palavra de dados escolhida foi de quatro bits. Todas as modelagens foram realizadas e simuladas utilizando a ferramenta Virtuoso, da Cadence.

4.1 Modelagem do Conversor Serial-Paralelo

A figura 8 mostra a modelagem do bloco conversor serial paralelo em Verilog.

```
4   module serial_parallel (clk,rst,serial_data_in,ready,parallel_data_out);
5
6   input clk, rst;
7   input serial_data_in;
8   output reg ready;
9   output reg [3:0] parallel_data_out;
10
11  reg [3:0] s_parallel_data_out;
12  reg [2:0] i;
13
14  always @ ( posedge clk or posedge rst)
15
16  begin
17      if(rst)
18      begin
19          s_parallel_data_out <= 4'b0000;
20          ready <= 0;
21          i <= 0;
22      end
23
24      else
25
26      begin
27
28          s_parallel_data_out[3] <= serial_data_in;
29          s_parallel_data_out[2] <= s_parallel_data_out[3];
30          s_parallel_data_out[1] <= s_parallel_data_out[2];
31          s_parallel_data_out[0] <= s_parallel_data_out[1];
32
33          i <= i+1;
34
35          if(i == 3'b100)
36              begin
37
38                  i <= 0;
39                  ready <= 1;
40
41              end
42          else
43              ready <= 0;
44      end
45
46  parallel_data_out <= s_parallel_data_out;
47
48  end
49
50  endmodule
```

Figura 8 – Modelagem do bloco conversor serial paralelo.

Este bloco possui três entradas (**clk, rst e serial data_in**) e duas saídas (**ready e parallel_data_out**). São utlizados também dois sinais de controle interno: **s_parallel_data_out** e **i**.

Através do comando **always @ (posedge clk or posedge rst)**, realiza-se as operações designadas sempre que houver a borda de subida do sinal de clock ou de reset, em suas respectivas entradas. Caso o reset esteja em alto, o bloco voltará para o estado normal, zerando o sinal de entrada, os sinais de controle interno e a saída **ready**. Caso contrário, tem-se o funcionamento normal do bloco.

Para se realizar a conversão serial-paralelo, a cada subida de clock, ou seja, a cada bit do sinal serial enviado, há um deslocamento na palavra de saída, que recebe este bit de entrada, e um incremento no sinal de controle **i**. Este processo continua o número de vezes necessário para se adquirir todos os bits que formam a palavra desejada (no caso,

de 4 bits). Neste momento, o sinal de controle **i** é resetado, e o sinal **ready** sinaliza para o bloco de memória que a palavra paralela está pronta para ser adquirida. Esta palavra, então, será o endereço em que a memória irá buscar o dado solicitado.

4.2 Modelagem do Bloco de Memória

A figura 9 mostra a modelagem do bloco de memória em Verilog.

```
3   module memoria (oe,address,data_mem);
4
5       input oe;
6       input [3:0] address;
7       output reg [3:0] data_mem;
8
9   reg [3:0] mem[0:15];
10
11  always @(posedge oe)
12
13  begin
14      data_mem <= mem[address];
15  end
16
17  initial begin
18
19      mem[0] = (4'b1111);
20      mem[1] = (4'b1110);
21      mem[2] = (4'b1101);
22      mem[3] = (4'b1100);
23      mem[4] = (4'b1011);
24      mem[5] = (4'b000);
25      mem[6] = (4'b1001);
26      mem[7] = (4'b1000);
27      mem[8] = (4'b0111);
28      mem[9] = (4'b0110);
29      mem[10] = (4'b0101);
30      mem[11] = (4'b0100);
31      mem[12] = (4'b0011);
32      mem[13] = (4'b0010);
33      mem[14] = (4'b0001);
34      mem[15] = (4'b1010);
35  end
36
37  endmodule
```

Figura 9 – Modelagem do bloco de memória.

Este bloco é um bloco de memória simples, possui duas entradas (**oe** e **address**) e uma saída (**data_mem**). Possui também um sinal de controle interno (**mem**), onde são guardados dezesseis palavras de 4 bits.

Através do uso do comando **always @ (posedge oe)** , pode-se realizar o resgate do valor armazenado no endereço recebido pela entrada **address**, atribuindo então ao

sinal de saída (**data_mem**) o valor que foi resgatado na memória. Este comando será realizado sempre que houver uma mudança de estado de 0 para 1 no sinal de controle **oe** .

O próximo passo no processo é converter o dado resgatado da memória em um sinal serial, já que é necessário que o modulador receba estes dados neste formado. Para isso, é utilizado o bloco conversor paralelo-serial.

4.3 Modelagem do Conversor Paralelo Serial

A figura 10 mostra a modelagem do bloco conversor paralelo-serial.

```
2
3   module conv_par_ser(myout,clk,rst,out);
4
5   input clk,rst;
6   input [3:0] myout;
7   output reg  out;
8
9   reg [2:0]i;
10
11  always @(posedge clk or posedge rst) begin
12    if(rst) begin
13      out <= 0;
14      i   <= 0;
15   end
16   else begin
17     out <= myout[i];
18     i   <= i+1;
19    end
20      if(i==3'b100) begin
21      i <= 0;
22      end
23  end
24
25  endmodule
26
```

Figura 10 – Modelagem do bloco conversor paralelo serial.

Este bloco possui três entradas (**clk, rst e myout**) e uma saída (**out**), além de um sinal interno de controle chamado **i**.

Através do uso do comando **always @ (posedge clk or posedge rst**, realiza-se a conversão de paralelo para serial sempre que a borda de subida das entradas **clk** e **rst** são detectadas.

Caso a entrada de reset (**rst**) esteja ativa, a saída e o sinal de controle interno **i** irão para 0. Se não for este o caso, a saída receberá o bit indexado na posição **i** do vetor **myout** e o sinal **i** será incrementado. O processo será repetido até que o sinal **i** alcance o valor do tamanho predeterminado da palavra (no caso, quatro), e será então resetado para converter a próxima palavra.

4.4 Conexão Entre os Blocos Digitais e Analógicos

Após a modelagem dos blocos digitais, foi realizada a conexão aos blocos analógicos para validar o funcionamento da *tag*. A ferramenta utilizada para este procedimento foi o Virtuoso da Cadence.

Os blocos digitais foram incorporados ao projeto da *tag*, que possuía os blocos analógicos e cujo *símbolo* foi gerado automaticamente após validar e compilar seu código Verilog (figura 11).

Figura 11 – Vista símbolo gerada pelo Virtuoso.

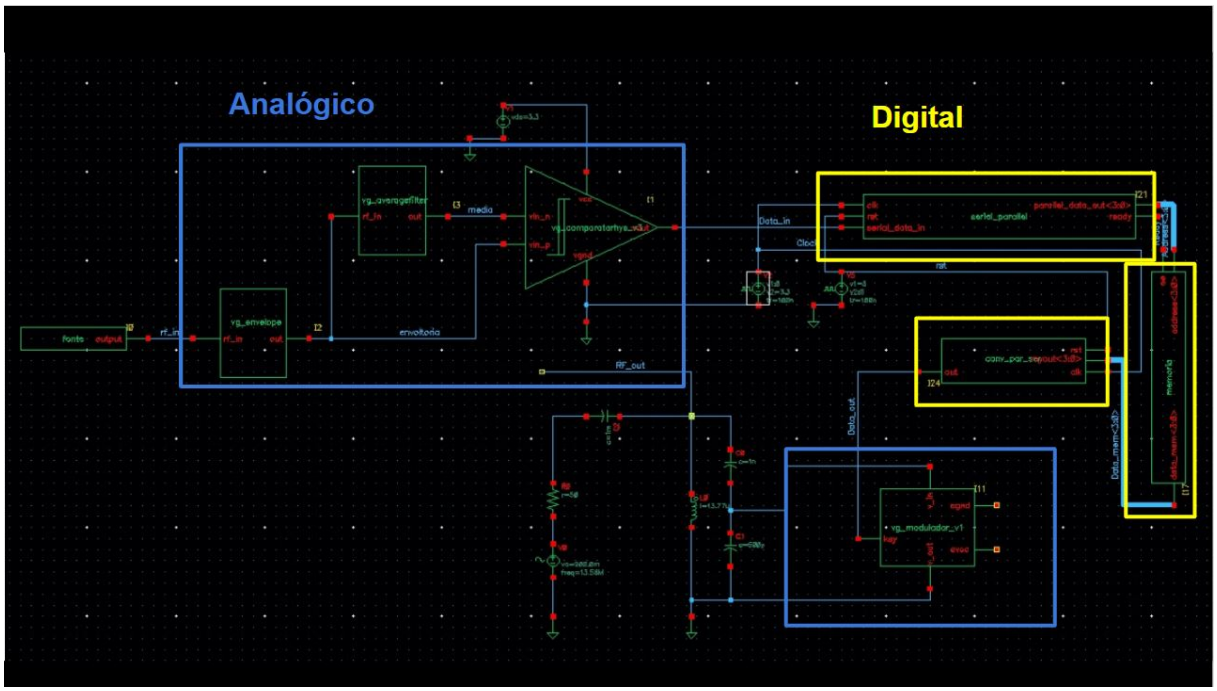

Figura 12 – Esquemático completo da *tag*.

No passo seguinte, abriu-se a vista de esquemático do projeto, utilizando a subferramenta *Schematic L*, e foram adicionados os blocos para se realizar a simulação com-

pleta da *tag*, a saber: blocos analógicos (modulador, demodulador e seus componentes), blocos digitais (conversor serial-paralelo, memória e conversor paralelo-serial), e bloco de fonte RF e outros componentes. A conexão completa de todos os blocos da *tag* pode ser vista na figura 12.

É importante frisar que os blocos analógicos foram instanciados com base em trabalhos anteriores, mais especificamente em (FILHO, 2014), sendo necessário então realizar as conexões internas do bloco analógico, as conexões internas do bloco digital e a conexão entre os dois blocos. Com o esquemático completamente conectado, é feita então a simulação mista final.

5 Resultados e Discussão

Neste capítulo serão apresentados e discutidos os resultados obtidos neste trabalho. Foram realizados dois níveis de simulações: simulação do bloco digital isolado e simulação mista do bloco digital com o bloco analógico. São apresentadas e comentadas também simulações realizadas em (FILHO, 2014) que tratam dos blocos analógicos isoladamente.

5.1 Simulação do Bloco Digital

Utilizando a ferramenta SimVision, foi realizada a conexão dos blocos individuais que compõem o bloco digital da *tag* para que pudesse ser feita a simulação deste bloco isoladamente. O bloco digital completo e suas conexões internas podem ser vistas na figura 13.

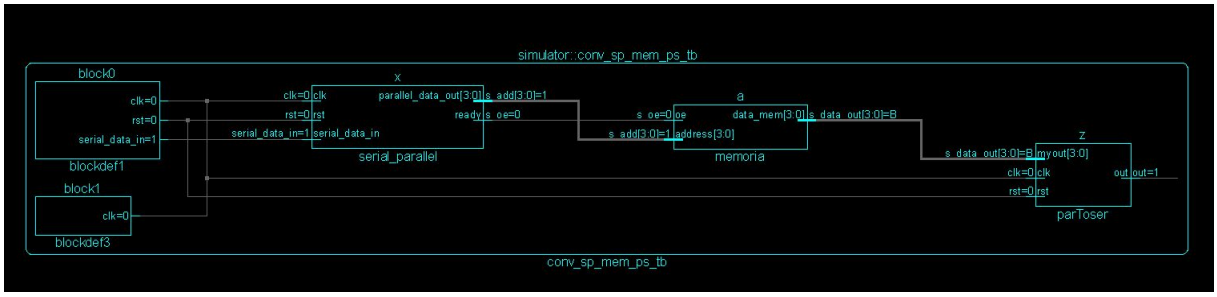

Figura 13 – Bloco Digital Completo.

Através de um arquivo *testbench*, foram adicionados estímulos conhecidos para realizar a simulação: A entrada **rst** começa em nível lógico alto para que seja feita a inicialização e, em seguida, fica em nível lógico baixo pelo resto da simulação. Na entrada **clk**, é adicionado um clock simétrico com um período de 100 ns. Finalmente, na entrada serial do bloco, foi adicionada uma sequência de valores lógicos para que se pudesse adquirir o endereço desejado na memória.

A figura 14 mostra o resultado da simulação. É possível observar que o funcionamento ocorre conforme o esperado: a cada subida de clock, após a entrada **rst** ir para nível lógico baixo, o contador interno é incrementado, e na quarta situação de subida de clock o sinal de controle **oe** alcança nível lógico alto. Neste momento, o bloco de memória recebe o endereço do dado requisitado, palavra esta que estava sendo recebida a cada subida de clock anteriormente (como pode ser visto no sinal **s_add[3:0]**).

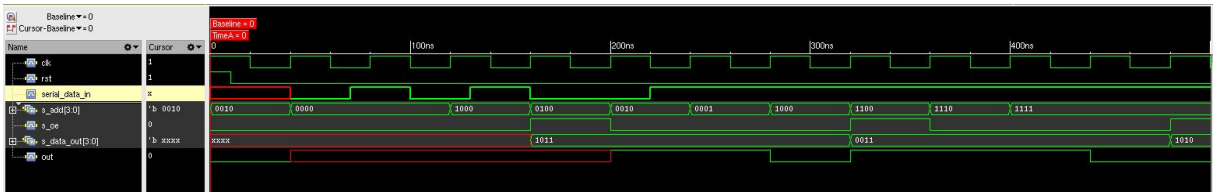

Figura 14 – Simulação do Bloco Digital Completo.

5.2 Simulação do Bloco Analógico

Simulações realizadas em (FILHO, 2014) demonstram que o bloco do demodulador funciona de forma satisfatória isoladamente. A modelagem em Verilog-AMS foi feita usando 3 sub-blocos: detector de envoltória, filtro de média e comparador com histerese. A Figura 15 mostra simulações do bloco demodulador realizadas em (FILHO, 2014), sendo possível observar as saídas intermediárias do detector de envoltória e do filtro de média.

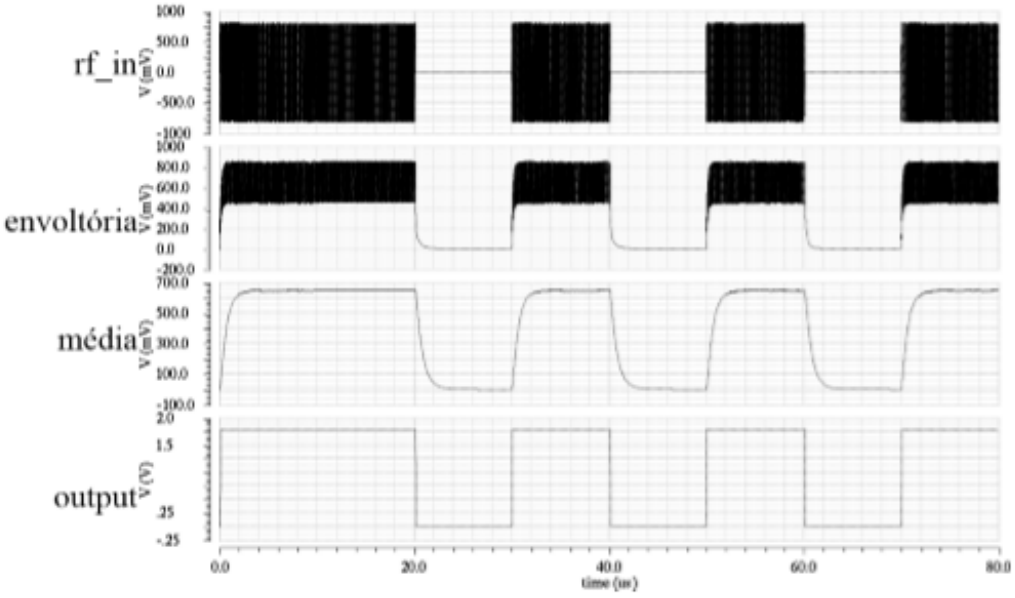

Figura 15 – Simulação do Demodulador ASK em Verilog-AMS(FILHO, 2014).

A simulação do bloco modulador também foi realizada com sucesso, como pode ser visto na figura 16. Com a amplitude em sintonia com 13,56 MHz, foi obtido 511,85 mV de pico enquanto que, descasando o circuito ressonante, foi obtido 413,64 mV de pico. Com esses resultados, o índice de modulação obtido foi de 10,61%, conforme ISO14443. (FILHO, 2014)

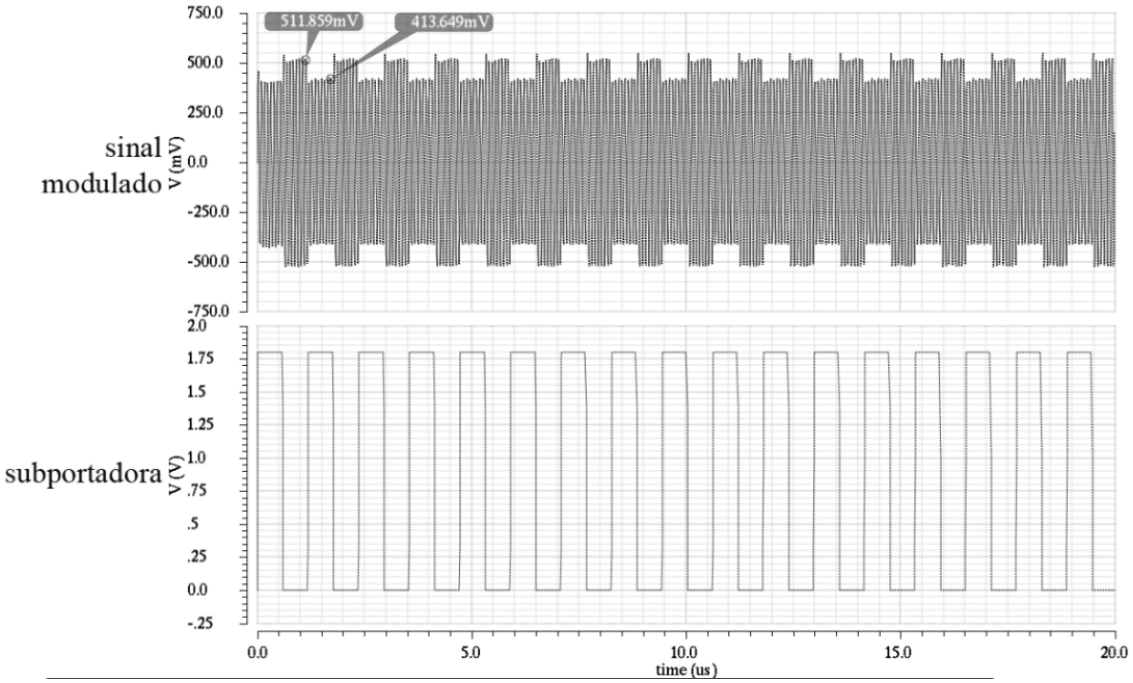

Figura 16 – Simulação do Modulador ASK em Verilog-AMS(FILHO, 2014).

5.3 Simulação Mista

A simulação mista, realizada com a virtuoso da *Cadence*, também produziu resultados satisfatórios, como pode ser visto na figura 17. A simulação se comporta de forma exatamente igual à do bloco digital isolado, com o acréscimo de sua entrada ser o sinal RF que foi demodulado no bloco demodulador, e sua saída sendo modelada com sucesso pelo bloco modulador, ambos blocos analógicos.

Na figura 17:

- /rf_in: Sinal de RF gerado por uma fonte, requisitando o dado da memória;
- /Data_in: Sinal de RF demodulado em um sinal discreto;
- /Clock: Sinal de clock síncrono utilizado como gatilho para os processos;
- /rst: Sinal de reset utilizado para inicialização da simulação;
- /Ready: Sinal de controle que autoriza a memória disponibilizar o dado requisitado;
- /Address: Sinal enviado para a memória que disponibilizará o dado existente neste endereço, já convertido para uma palavra em paralelo;
- Data_mem: Dado resgatado presente no endereço enviado para a memória;
- Data_Out: Dado convertido para uma palavra serial sendo enviado para o modulador;

- RF_out: Sinal requisitado modulado em um sinal análogico de RF, finalizando o processo.

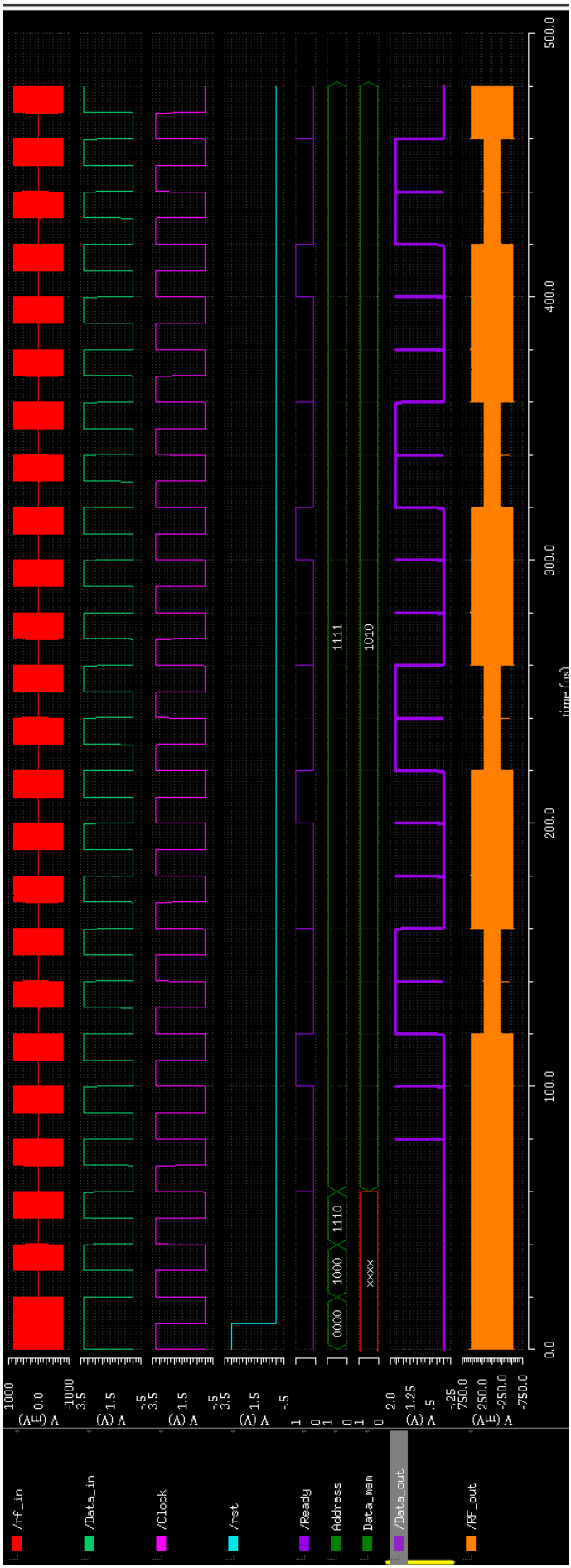

Figura 17 – Simulação Mista.

6 Conclusão

Neste trabalho, foi realizada a verificação dos blocos digitais de uma *tag* passiva de RFID de forma individual e em conjunto com seus blocos analógicos. Através do uso de simulações de sinais mistos, foi possível observar o completo funcionamento da estrutura da *tag* e a conexão entre blocos analógicos e digitais.

Na primeira parte deste trabalho, foram apresentados conceitos básicos sobre a tecnologia abordada e suas particularidades. Abordou-se informações sobre sistemas RFID, suas frequências de funcionamento e protocolos de comunicação, tal como formas de modulação de sinais. Foi realizado também um estudo sobre os passos para projetos de circuitos integrados digitais, desde a especificação básica, simulações e finalmente, sínteses e verificações finais.

Na segunda parte, foi feita a modelagem do sistema utilizado na simulação mista para verificar o funcionamento da t*ag* e as simulações individuais e mistas. Foram modelados os blocos que compõem o bloco digital da *tag*: conversor serial-paralelo, bloco de memória e conversor paralelo-serial. Em seguida, foi realizada a simulação deste bloco digital isoladamente e verificado seu funcionamento. Também foi realizada a análise das simulações dos blocos analógicos realizadas em trabalhos anteriores. Por fim, foi feita a simulação mista para verificar o funcionamento total da *tag*. Foi possível observar que o sistema modulado funciona como o esperado, e é capaz de realizar o resgate dos dados requisitados à memória.

Apesar dos objetivos de simulação e verificação deste trabalho terem sido alcançados, não foi possível realizar a síntese lógica e física da *tag* por conta da expiração da licença de uso das ferramentas necessárias para tal.

Como trabalhos futuros, sugere-se a adaptação da descrição Verilog do protocolo anticolisão para simulações mistas com o *front-end* analógico. A adição do retificador e a realização de simulações do sistema completo com blocos em Verilog-AMS, Verilog e em nível de transistores. A realização do layout dos blocos digitais modulados e da FSM e dos blocos que implementam o protocolo anticolisão utilizando células da biblioteca da TSMC 0.18 um e finalmente a integração dos blocos digitais com os analógicos e o envio para fabricação. Existe ainda a possibilidade de se usar um gerador de memória do fabricante para substituir o bloco de memória projetado. Por fim, para a realização de simulações, sugere-se o uso de *testbenches* automatizados com o propósito de facilitar e acelerar os processos de verificação de resultados.

Referências

AMARAL, W. et al. Design of a passive rfid tag for 13.56mhz. University of Brasília, 2014. Citado na página 24.

COE C, d. e. R. R. About rfid. <http://www.rfid-coe.com/_Ingles/AboutRFID.aspx>, 2005. Citado na página 23.

FARIA, H. Projeto de um processador banda base de uma tag passiva de rfid. Universidade de Brasília, 2017. Citado na página 25.

FILHO, M. P. Modelagem em verilog-ams de uma tag passiva de rfid e projeto elétrico do demodulador ask. Universidade de Brasília, 2014. Citado 7 vezes nas páginas 15, 24, 35, 44, 45, 46 e 47.

FINKENZELLER, K. *RFID Handbook - Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. [S.l.: s.n.], 2010. Citado 2 vezes nas páginas 23 e 31.

GUIMARãES, R. R. d. N. H. Modelagem de sistemas heterogêneos utilizando frameworks baseados em moc. Universidade de Brasília, 2013. Citado 2 vezes nas páginas 15 e 24.

ISLAND, U. of R. *Amplitude Shift Keying Frequency Shift Keying*. http://www.ele.uri.edu/Courses/ele436/labs/ASKnFSK.pdf, 2012. Citado na página 33.

ISO/IEC. *Undertanding the Requirements of ISO/IEC 14443 for Type B Proximity Contactless Identification Cards*. [S.l.], 2005. Citado na página 32.

LI, H. *Development and Implementation of RFID Technology*. Chinese Academy of Sciences, China, 2009. 1 p. Citado 3 vezes nas páginas 15, 24 e 29.

LI, Z.; ALPERT, C. What is physical synthesis? Disponível em: <https://pdfs.semanticscholar.org/fa4f/a72088b2f0b461c1bf3fc38776fa8046e53c.pdf/>, 2011. Citado na página 37.

MARKHAM. Understanding rfid (radio frequency identification). Canadá, p. 14,15, 2012. Citado 3 vezes nas páginas 17, 30 e 31.

NETO, J. Teste, projeto e verificação funcional de uma tag de rfid de 13,56 mhz. Universidade de Brasília, 2015. Citado 2 vezes nas páginas 24 e 30.

PUHLMANN, H. Introdução à tecnologia de identificação rfid. Disponível em: <https://www.embarcados.com.br/introducao-a-tecnologia-de-identificacao-rfid/>, 2015. Citado na página 31.

SANGREMAN, T. C. A. O que é rfid? utilidades práticas,como funciona. <http://www.gta.ufrj.br/grad/07_1/rfid/RFID_arquivos/Index.htm>, 2003. Citado 2 vezes nas páginas 15 e 29.

Apêndices

APÊNDICE A – Passos Para a Síntese Lógica e Física

A.1 Organização dos arquivos e pastas

Em primeiro lugar, é necessário deixar os arquivos organizados de acordo com a hierarquia de pastas para sínteses e simulações mostrada abaixo, dentro de uma pasta chamada **tag**.

1. **sim**

   Contém todos os arquivos .v a serem simulados.

2. **synth**

   Contém apenas os arquivos .v que serão utilizados na síntese lógica e o *script* para realizá-la. (**synth_rt_counter.tcl**)

3. **layout**

   a) *setup.sh*

   b) *Default.view*

   c) **scripts**

   Contém os *scripts* a serem utilizados para realizar a síntese física.

4. **sim_pos_synth**

   Contém os arquivos gerados pela síntese lógica e os arquivos necessários para realizar a simulação:

   - Verilog obtido da síntese lógica ***(synth/deliverables/tag_out_net.v)***
   - SDF gerado na síntese lógica ***(synth/deliverables/tag_out_sdf.sdf)***
   - Arquivos da biblioteca de tecnologia
   - *Testbench* ***(sim/tag_tb.v)***

5. **sim_pos_layout**

   Contém os arquivos gerados pela síntese física e os arquivos necessários para realizar a simulação:

- Verilog obtido da síntese física ***(layout/deliverables/tag.v)***
- SDF gerado na síntese ***(layout/deliverables/tag.sdf)***
- Arquivos da biblioteca de tecnologia
- *Testbench* ***(sim/tag_tb.v)***

A.2 Síntese Lógica

Para realizar a síntese lógica, basta executar o script **synth_rtl_counter.tcl** disponibilizado na pasta **synth**.

No diretório **synth**, abrir o **RTL Compiler** usando o comando:

rc -gui

No prompt do RTL Compiler, executar o comando:

source synth_rtl_counter.tcl

Os arquivos Verilog, SDF e SDC serão salvos na pasta **deliverables**, e os relatórios de atraso, área e potência na pasta **reports**.

A.3 Simulação Pós-Síntese Lógica

Copiar os arquivos que ainda faltam **(tag_out_net.v** e **tag_out_sdf.sdf)** para a pasta **sim_pos_synth** e abrir o **NCLaunch** usando o comando:

nclaunch &

OBS: Utilizar as opções **File -> Set Design Directory** para indicar o novo diretório e criar um novo arquivo **cds.lib**. Escolher **'*'** em *Filters* para que sejam mostrados todos os arquivos.

Compilar os arquivos na seguinte ordem:

1. Arquivos que contêm descrições das células básicas da tecnologia;
2. Arquivo SDF (informações sobre atrasos);
3. Verilog da síntese lógica **(tag_out_net.v)**;
4. Arquivo **tag_tb.v**.

Elaborar a célula do *testbench*. Em **Advanced Options » General**, setar **Default Timescale** para **1ns/1ps**.

Depois de elaborar o **testbench**, basta executar a simulação e verificar o resultado.

A.4 Síntese Física

Abrir o terminal na pasta **layout** e executar o *script* de setup:

sh setup.sh

Observar se as pastas foram criadas corretamente, se os arquivos foram copiados e se há alguma mensagem de erro.

Iniciar o **SoC Encounter** com o comando:

encounter

Na interface gráfica, ir em **File -> Import Design** e preencher os campos:

Verilog: selecionar o arquivo Verilog gerado pela síntese lógica na pasta VERILOG e selecionar **Auto Assign** em **Top Cell**.

LEF Files: selecionar os arquivos **.lef** na pasta do *Design Kit*

IO Assignment File: selecionar o arquivo **corners.io** na pasta **data**.

Power Nets: vdd!

Ground Nets: gnd!

MMMC View Definition File: selecionar o arquivo **Default.view** na pasta **data**.

Em seguida, clicar em OK.

Para executar todos os *scripts* de uma vez, basta carregar o *script* principal:

source scripts/main.tcl

Caso contrário, basta executar os *scripts* individuais seguindo a ordem apresentada no principal, conforme é descrito a seguir.

- Carregar as variáveis e setar os diretórios executando o *script* **variaveis.tcl** no *prompt* do terminal do **Encounter**:

source scripts/variaveis.tcl

- Executar o *script* **floorplan.tcl**.

source scripts/floorplan.tcl

- Executar o *script* **powerplan.tcl**.

source scripts/powerplan.tcl

Verificar a disposição das trilhas de alimentação no *layout*.

- Executar o *script* **placement.tcl**.

source scripts/placement.tcl

Verificar a disposição dos blocos no *layout*, e se há alguma violação.

- Executar o *script* c**locktree.tcl** para sintetizar a árvore de *clock*:

source scripts/clocktree.tcl

Verificar a distribuição do *clock* no *layout*, e se há alguma violação.

- Executar o *script* **postCTS.tcl** para realizar teste e otimizações pós-CTS:

source scripts/encounter/postCTS.tcl

Verificar se há violações.

- Adicionar as *FILLER* cells, executando o *script* **fillcore.tcl**:

source scripts/fillcore.tcl

- Executar o*script* **nanoroute.tcl** para realizar o roteamento:

source scripts/nanoroute.tcl

Verificar se há violações.

- Opcional: Executar o *script* **metalfill.tcl** para inserir os metais de preenchimento:

source scripts/metalfill.tcl

Se houver violações de conectividade, rodar o comando **trimMetalFill**.

- Verificar a conectividade, a geometria e erros de antena:

source scripts/verify.tcl

- Executar o *script* **generate.tcl** para gerar o bloco final:

source scripts/generate.tcl

A.5 Simulação Pós-Simulação Física

Para verificar se o bloco ainda está funcionando incluindo os dados da geometria do *layout*, deve ser feita uma simulação após a síntese física. A pasta **sim_pos_layout** deve conter os seguintes arquivos:

- Verilog obtido do Encounter **(/layout/deliverables/tag.v)**.

- Arquivo SDF gerado na síntese **(/layout/deliverables/tag.sdf)**.

- Arquivos da biblioteca de tecnologia.

– *Testbench* **(tag_tb.v)**

Basta então seguir os passos abaixo:

- Abrir o **NCLaunch** no diretório s**im_pos_layout** e compilar os arquivos na seguinte ordem:

  1. Arquivos Verilog da biblioteca de tecnologia;
  2. Arquivo SDF;
  3. Verilog da síntese física;
  4. **tag_tb.v**.

- Elaborar a célula do *testbench*. Em **Advanced Options » General**, setar **Default Timescale** para **1ns/1ps.**

- Depois de elaborar o *testbench*, basta executar a simulação e verificar a funcionalidade.