UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE ENGENHARIA DE COMPUTAÇÃO

IAGO OLIVEIRA LIMA

EXPLORANDO VULNERABILIDADES EM PROJETOS COM FPGAS POR MEIO DA INSERÇÃO DE *TROJANS* DE *HARDWARE*

QUIXADÁ

2019

IAGO OLIVEIRA LIMA

EXPLORANDO VULNERABILIDADES EM PROJETOS COM FPGAS POR MEIO DA INSERÇÃO DE *TROJANS* DE *HARDWARE*

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Cristiano Bacelar de Oliveira

QUIXADÁ

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L698 Lima, Iago Oliveira.
Explorando vulnerabilidades em projetos com FPGAs por meio da inserção de trojans de hardware / Iago Oliveira Lima. – 2019.
47 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Computação, Quixadá, 2019.
Orientação: Prof. Dr. Cristiano Bacelar de Oliveira.

1. Dispositivo lógico programável. 2. Trojan de hardware. 3. Segurança computacional. 4. Sistemas de controle por realimentação. I. Título.

CDD 621.39

IAGO OLIVEIRA LIMA

EXPLORANDO VULNERABILIDADES EM PROJETOS COM FPGAS POR MEIO DA INSERÇÃO DE *TROJANS* DE *HARDWARE*

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: _____/_____/_____

BANCA EXAMINADORA

Prof. Dr. Cristiano Bacelar de Oliveira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. André Ribeiro Braga
Universidade Federal do Ceará (UFC)

Prof. Me. Roberto Cabral Rabêlo Filho
Universidade Federal do Ceará (UFC)

Aos meus pais, irmãos, namorada, amigos e professores.

AGRADECIMENTOS

A Deus, por ter colocado em minha vida pessoas tão especiais, que sempre me ajudaram e me apoiam nas adversidades do dia a dia.

Aos meus pais, Livaldo e Josivanda, pelo infinito amor e cuidado, por todos os ensinamentos que moldaram a pessoa que sou hoje.

Aos meus irmãos, Iana, Ítalo e Isli, por todo companheirismo e apoio.

À minha madrinha, Adriana, por toda preocupação, cuidado e suporte a mim dedicado desde os primeiros instantes da minha vida.

À minha tia, Lurdeci, por ter me acolhido como um filho durante todos esses anos, por todo apoio e carinho.

À minha tia, Lucivanda, por ser um dos pilares da família e me mostrar o quão grandioso e importante o papel que temos na vida uns dos outros.

À minha namorada, Fernanda, por sempre me encorajar a fazer o meu melhor, pela cumplicidade e por todo amor.

Ao meu amigo e orientador Prof. Dr. Cristiano Bacelar, por todos os conselhos, orientações, paciência, conversas aleatórias, e principalmente, por ser para mim exemplo de pessoa e de profissional.

Ao Prof. Dr. Paulo de Tarso Oliveira e a Profa. Dra. Viviane Menezes, pela preocupação com os educandos e com a educação, sendo exemplos não só de professores, mas também de humanos no melhor sentido que a palavra possa ter.

Aos demais professores da Universidade Federal do Ceará, Campus Quixadá, por todos os ensinamentos compartilhados.

À Universidade Federal do Ceará, em especial, ao Prof. Dr. Davi Romero e à Profa. Dra. Andréia Libório pela excelência administrava e acadêmica, e pelo enorme empenho em tornar o Campus Quixadá referência em pesquisa, tecnologia e inovação.

A todos os técnicos-administrativo do Campus Quixadá, em especial, Abdul-Hamid Moreira, Gerlysson Girão e Venício de Oliveira.

Aos meus amigos e amigas, Camila Diógenes, Felipe Silva, Iury Queirós, Marianna Pinho, Marisa Silva, Rafaella Fernandes, Raynara Lima, Robert Cabral e Robertty Freitas, por todas as contribuição nessa jornada acadêmica, por me ajudarem a ser uma pessoa melhor, dividirem comigo os pesos dessa longa e turbulenta caminhada da vida e por todos os momentos dentro e fora da universidade.

Por último mas não menos importante, ao time de produto da TrixLog do Brasil, em especial, ao Thiago Silva, Daniel Alves, Douglas Henrique, Jonas Chaves e Vicente Macambira por todo incentivo, pelas oportunidades, risadas e conselhos.

“Tudo o que acontece no universo tem uma razão de ser; um objetivo. Nós como seres humanos, temos uma só lição na vida: seguir em frente e ter a certeza de que apesar de as vezes estar no escuro, o sol vai voltar a brilhar.”

(Santa Dulce dos Pobres)

RESUMO

A segurança no meio digital sempre foi uma área muito preocupante, principalmente com o crescimento avassalador de novas tecnologias. Nas últimas décadas, as pesquisas na área de segurança de *hardware* vem crescendo, pois várias vulnerabilidades de segurança e ataques ao hardware foram descobertas em todo o mundo, como por exemplo o relatório de 2012 das Forças Armados do Senado dos Estados Unidos expôs diversos dispositivos falsificados em vários ramos das Forças Aéreas. Os relatos de trojans de hardware vem crescendo, identificar este tipo de trojan é uma tarefa que requer muito tempo. Este trabalho explora diferentes cenários fictícios envolvendo a inserção de *trojans* de *hardware*. Com objetivo de demonstrar o processo de ataque a *hardware* em projetos envolvendo FPGAs.

Palavras-chave: Computação Reconfigurável. FPGA. *Trojan* de *Hardware*. Segurança de *Hardware*. *LFSR*

ABSTRACT

Security in the digital environment has always been a very worrying area, especially with the overwhelming growth of new technologies. In the last decades, research in the field of hardware security has grown as a number of security vulnerabilities and hardware attacks have been discovered around the world, for example, the 2012 US Senate Armed Forces report exposed several counterfeit devices in various branches of the Air Forces. Reports of hardware trojans have been growing exponentially, identifying this type of trojan is a time-consuming task. This paper explores different fictional scenarios involving the insertion of hardware trojans. In order to demonstrate the process of hardware attack.

Keywords: Reconfigurable Computing. FPGA. Hardware Trojan. Hardware Security. LFSR

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

3PIP	*Third Party Intellectual Property*
ASIC	*Application Specific Integrated Circuits*
CAN	*Controller Area Network*
CEC	*Consumer Electronics Control*
CLBs	*Configurable Logic Blocks*
CMOS	*Semicondutor de óxido metálico complementar*
CPU	Unidade Central de Processamento
DARPA	*Defense Advanced Research Projects Agency*
DMA	*Direct Memory Access*
ECC	*Error Correcting Code*
FPGA	*Field Programmable Gate Array*
HT	*Trojan de Hardware*
I2C	*Inter-Integrated Circuit*
IP	*Intellectual Property*
IRIS	*Integrity and Reliability of Integrated Circuits*
LFSR	*Linear-feedback shift register*
LUT	*Look-Up Table*
MOS	Metal-Óxido-Semicondutor
MUX	Multiplexador
PUF	Funções Físicas não Clonáveis
RAM	*Random Access Memory*
ROM	*Read Only Memory*
SoC	*System on Chip*
SPI	*Serial Peripheral Interface*
TRNG	*True Random Number Generator*
UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*
VGA	*Video Graphics Array*

SUMÁRIO

1 INTRODUÇÃO

A segurança no meio digital tornou-se uma parte essencial do mundo eletrônico moderno. Assim, diversas áreas da tecnologia preocupam-se com esta questão em vários níveis de abstração. Por exemplo, a área de segurança de rede de computadores se concentra nos ataques a uma rede que conecta vários sistemas de computador e nos mecanismos para garantir sua usabilidade e integridade sob possíveis ataques (KUROSE, 2005). Já a segurança da informação concentra-se na prática geral de fornecer confidencialidade, integridade e disponibilidade de informações através da proteção contra acesso, uso, modificação ou destruição não autorizada (BHUNIA; TEHRANIPOOR, 2018).

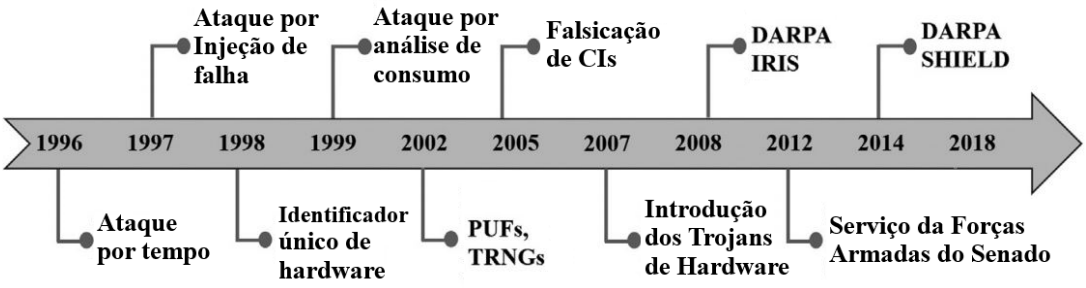
Considerando sistemas computacionais em geral, podemos pensar em questões de segurança que envolvem tanto *software* como *hardware*. De maneira geral, a segurança de *software* lida com ataques maliciosos ao *software*, os quais explorando diferentes vulnerabilidades, como *bugs* de implementação, tratamento inconsistente de erros e estouros de *buffer* (KOEUNE; STANDAERT, 2005). No contexto de segurança de *hardware*, por outro lado, os problemas são relativos à parte eletrônica dos sistemas, abrangendo arquitetura, implementação, validação ou mesmo distribuição e acesso aos equipamentos (TEHRANIPOOR; KOUSHANFAR, 2010). Nesse contexto são levados em consideração os próprios componentes de *hardware*, como circuitos integrados (ICs), componentes passivos (resistores, capacitores, indutores) e placas de circuito impresso (PCBs), bem como os segredos armazenados dentro desses componentes, como, por exemplo, chaves criptográficas, fusíveis programáveis, dados confidenciais do usuário, *software* instalado e dados de configuração.

A área de Segurança de *hardware* vem evoluindo e se tornando uma importante área no campo de Segurança de computadores. Nas últimas três décadas, o campo da segurança de *hardware* evoluiu rapidamente com a descoberta de muitas vulnerabilidades e ataques. A Figura 1 fornece uma breve linha do tempo para a evolução da segurança do *hardware*. Antes de 1996, havia apenas instâncias esporádicas de pirataria de IP de *hardware*, principalmente a clonagem de circuitos integrados, levando ao desenvolvimento de algumas técnicas de marca d'água de IP e outras técnicas antipirataria.

Em 1996, Kocher (1996) desenvolveu um método de ataque de *hardware* que visava extrair informações de um *hardware* criptográfico com base em uma análise sistemática do tempo de computação para diferentes operações. Em 1997, Voas (1997) relata as injeções de falhas como um vetor de ataque que pode levar a comprometer a segurança de um sistema. O ataque

se concentra na aplicação de falhas no sistema, a fim de forçá-lo a vazar dados confidenciais. O primeiro ataque de canal lateral (BHUNIA; TEHRANIPOOR, 2018) baseado em análise de potência foi introduzido em 1999 e é descrito no trabalho de Kocher *et al.* (1999). Este ataque concentrou-se em analisar as dissipações de energia em tempo de execução para recuperar segredos de um *chip* de criptografia. Em 2005, tiveram relatos (TEHRANIPOOR; KOUSHANFAR, 2010) sobre a produção e o fornecimento de circuitos integrados falsificados, incluindo chips clonados e reciclados, que criaram grandes preocupações de segurança e confiança.

Figura 1 – Linha do tempo dos principais acontecimentos na área de segurança de *hardware* nas últimas três década



Fonte: Adaptado de Bhunia e Tehranipoor (2018)

Conforme mostrado na Figura 1, o conceito de *trojan* de *hardware* foi introduzido em 2007. Isto revelou a possibilidade de se inserir circuitos maliciosos em um design de *hardware* com o objetivo de interromper o comportamento funcional normal, vazar informações confidenciais, conceder controle não autorizado ou, ainda, degradar o desempenho do sistema (TEHRANIPOOR; KOUSHANFAR, 2010).

Semelhante ao que ocorre no contexto de *software*, as contramedidas para ataques a *hardware* foram desenvolvidas de maneira reativa. Ao longo dos anos, muitas soluções de design e teste evoluíram para mitigar ataques conhecidos. A ideia de marcação de *hardware* foi introduzida em 1998, onde os circuitos integrados foram atribuídos com um identificador exclusivo. Primitivas de segurança de *hardware*, como Funções Físicas não Clonáveiss (PUFs) e *True Random Number Generator*s (TRNGs) foram introduzidas no início de 2000 para melhorar o nível de proteção contra ataques de *hardware* (BARBARESCHI *et al.*, 2015).

Em 2008, a *Defense Advanced Research Projects Agency* (DARPA) introduziu o programa de integridade e confiabilidade de circuitos integrados, do inglês *Integrity and Reliability of Integrated Circuits* (IRIS) para desenvolver técnicas para garantir a integridade e a confiabilidade do *hardware* por meio de análises destrutivas e não destrutivas. Em 2012, um

relatório publicado pelas Forças Armadas do Senado mostrou que um conjunto de dispositivos falsificados foi descoberto em diferentes ramos da Força Aérea dos EUA (U.S. SENATE COMMITTEE ON ARMED SERVICES, 2012), acentuando a gravidade do problema. O número total dessas falsificações excedeu 1 milhão, e a investigação foi concluída com uma emenda que aplica práticas de prevenção à falsificação. O programa Integridade de *Hardware* da Cadeia de Suprimentos para Defesa Eletrônica (SHIELD) foi introduzido pela DARPA em 2014 para desenvolver tecnologia para rastrear componentes eletrônicos à medida que se movem pela cadeia de suprimentos (BHUNIA; TEHRANIPOOR, 2018).

Dado o histórico apresentado, percebe-se que a segurança envolvendo os dispositivos físicos forma a base da segurança de todo o sistema computacional, fornecendo a âncora de confiança para os demais componentes do sistema. Assim, se um dado dispositivo tem sua funcionalidade comprometida, todos os outros elementos do sistema podem ser afetados, sobretudo os que lidam diretamente com tal dispositivo.

Uma vez que uma das formas de afetar o funcionamento de um sistema é com o uso de *trojans*, este trabalho explora diferentes cenários fictícios envolvendo a inserção de *trojans* de *hardware*. Com isto, pretende-se demonstrar como se dá o processo de ataque direto ao *hardware*, bem como analisar o nível de dificuldade na detecção do *trojan* em um sistema afetado.

Este trabalho está dividido da seguinte forma: no Capítulo 2 são apresentados os conceitos teóricos utilizados no desenvolvimento deste trabalho. No Capítulo 3 são mostrados os trabalhos na literatura relacionados à esta pesquisa. O Capítulo 4 mostra os procedimentos metodológicos utilizados no decorrer do desenvolvimento desta pesquisa. No Capítulo 5 são mostrados os experimentos e os resultados obtidos em cada um deles. Por fim, o Capítulo 6 discorre sobre as conclusões obtidas neste trabalho e seus próximos passos de pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 *System on Chip (SoC)*

Nolan (2019) definiu *System on Chip* (SoC) como um conjunto de componentes heterogêneos complexos, que geralmente possuem processadores programáveis, hardware dedicado, memórias *on-chip*, interfaces de entrada e saída e uma interface de comunicação. Logo, um SoC consiste de um hardware e no *software* que controla os núcleos, periféricos e interfaces do microcontrolador, microprocessador ou processador de sinal digital, analógico, de frequência de rádio, etc., variando de acordo com a necessidade do projeto.

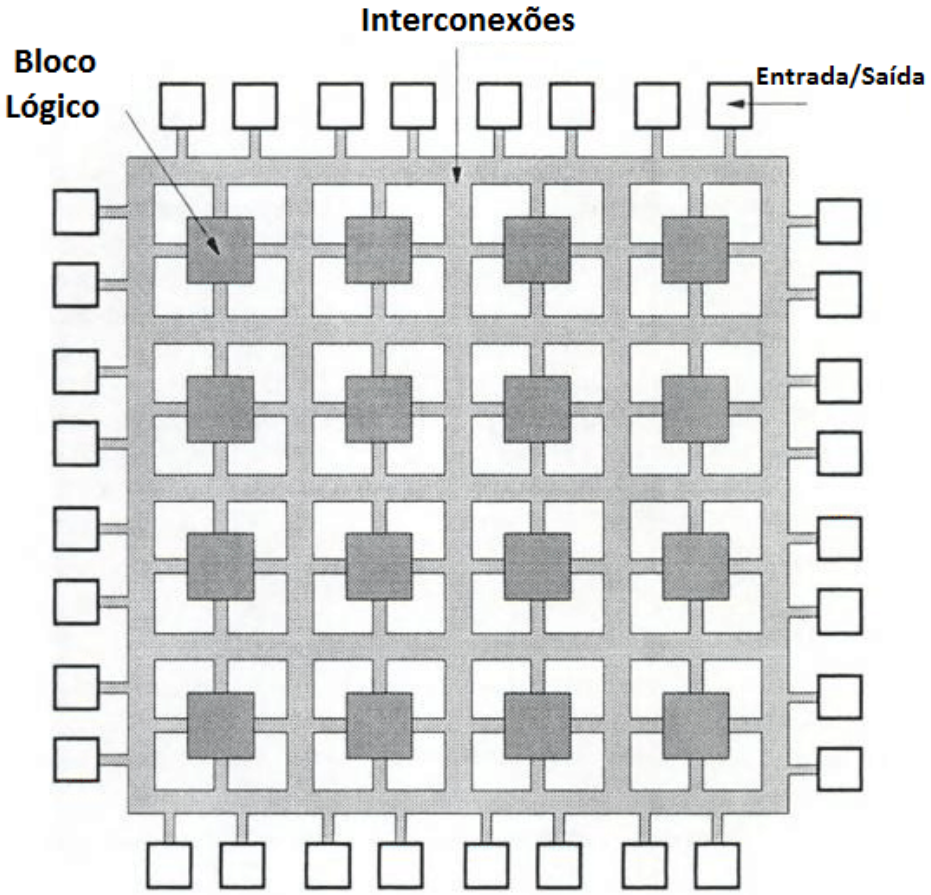
Como são integrados em um único *chip*, os SoCs tem um consumo de energia muito menor e ocupam muito menos área do que os projetos de vários chips com funcionalidade equivalente (BHUNIA; TEHRANIPOOR, 2018). SoCs geralmente são fabricados usando a tecnologia Metal-Óxido-Semicondutor (MOS) e são comumente usados em sistemas embarcados e Internet das Coisas, em projetos onde anteriormente seriam utilizados microcontroladores.

O fluxo de design de um SoC, conhecido como co-design da arquitetura, visa desenvolver o *hardware* e o *software* ao mesmo tempo, além de levar em conta otimizações e restrições. A metodologia de co-design possibilita que o projetista desenvolva um sistema com um equilíbrio de subsistemas de *hardware* e *software* de forma que, trabalhando juntos, consigam atingir um comportamento específico que atenda aos requisitos do projeto (GUIN *et al.*, 2014). É comum que SoCs usem em conjunto dispositivos heterogêneos, tais como CPUs, GPUs e FPGAs na criação de novas arquiteturas

2.2 *Field Programmable Gate Array* (FPGA)

O primeiro FPGA surgiu em 1985 como criação da empresa norte americana Xilinx Inc. com o objetivo de serem circuito programáveis compostos por um conjunto de blocos lógicos organizados na forma de uma matriz (OLIVEIRA, 2015). Segundo Wiśniewski (2009), um FPGA tem como principal objetivo a utilização da lógica programável para implementação de simples funções lógicas. A estrutura básica dos FPGAs podem variar de acordo com seus fabricantes e de família para família, mas no geral possuem três componentes principais: blocos lógicos configuráveis, do inglês *Configurable Logic Blocks* (CLBs), entradas e saídas (*input/output*) e blocos de interconexões programáveis (DUTRA, 2016), como mostrado na Figura 2.

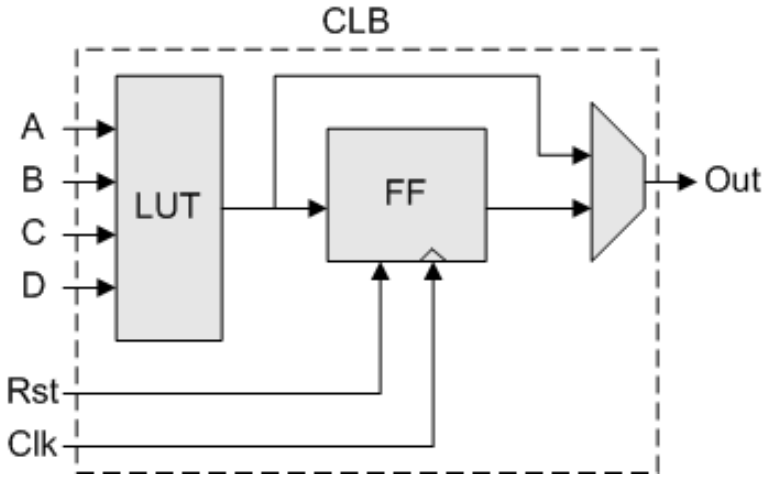
Figura 2 – Estrutura típica de um FPGA.



Fonte: Adaptado de Oliveira e Bastos-Filho (2011)

Os FPGAs utilizam como base estrutural os blocos lógicos configuráveis (CLBs). Cada CLB (Figura 3) dispõe de uma *Look-Up Table* (LUT) que pode ser programada para executar uma função lógica específica, além de um ou mais *Flip-Flop*s tipo D, que permitem a criação de máquinas sequenciais (OLIVEIRA; BASTOS-FILHO, 2011). Como cada bloco lógico pode ser programado para executar uma função lógica básica e as conexões podem ser definidas para a criação de interconexões entre os blocos, circuitos lógicos são implementados em FPGA dividindo o sistema em blocos lógicos pequenos e depois interconectando-os. Um FPGA pode ter milhares de CLBs de tipos distintos em um único dispositivo, tornado possível a criação de sistemas lógicos com alta complexidade em um único chip (WILSON, 2015).

Figura 3 – Estrutura básica de um CLB



Fonte: Bennis *et al.* (2009)

Normalmente, os FPGAs possuem um *clock* de execução mais lento se comparado aos processadores atuais, porém, o paralelismo que é implementado nos FPGAs permite que suas diferentes partes realizem diferentes funções no mesmo ciclo de *clock*, o que compensa a baixa frequência[1]. Como exemplo do potencial dos FPGAs, temos os dispositivos da família Stratix®10 DX 2800, fabricados pela Intel[2], com as características abaixo:

- *Tranceivers* com capacidade de até 57,8Gbps;
- Mais de 2,73 milhões de elementos lógicos;
- Blocos de memória interna distribuídos e proteção *Error Correcting Code* (ECC);
- Bloco de DSPs de alta precisão e desempenho de 9,2 TFLOPS em operações de ponto flutuante.

2.3 Módulo de Propriedade Intelectual (*IP Core*)

Com o avanço na utilização dos FPGAs a complexidade dos projetos que utilizam esses dispositivos também aumentou. Isto tornou impraticável criar todas as partes do design do zero, fomentado o reuso de componentes (ROUSE, 2011). Esses componentes são normalmente chamados de módulos de *Intellectual Property* (IP), ou apenas IPs. Um IP é um bloco de lógico que é usado para criar um circuito para uma aplicação específica[3] para um produto.

Como elementos fundamentais da reutilização do design, os IP fazem parte da crescente tendência da indústria no uso repetido de componentes já projetados e validados

[1] Modelos mais simples geralmente operam na ordem de MHz.
[2] https://www.intel.com/
[3] A literatura adota a expressão em inglês *Application Specific Integrated Circuits* (ASIC)

anteriormente. Idealmente, um núcleo de IP deve ser totalmente portátil, ou seja, capaz de ser facilmente inserido em qualquer outro design. A *Universal Asynchronous Receiver/Transmitter* (UART), o processador, os controladores Ethernet e controladores de barramentos PCI são exemplos clássicos de módulos IP.

Existem três categorias de módulos IP, são elas: *Hard cores, Firm cores* e *Soft cores*. Os *Hard IPs* são blocos pré-implementados, como núcleos de microprocessadores, interfaces de Ethernet, multiplicadores, somadores, entre outros. Esses blocos são projetados para terem a maior eficiência possível em termos de consumo de energia e desempenho. Cada família FPGA terá diferentes combinações de tais blocos, com diferentes quantidades de CLBs (MAXFIELD, 2008).

No outro extremo, o *Soft IP* refere-se a uma biblioteca no nível de fonte de funções de alto nível que pode ser incluída nos projetos dos usuários. Essas funções são normalmente representadas usando uma linguagem de descrição de *hardware* (HDL), como Verilog ou VHDL, no nível de transferência de registro (RTL) da abstração. Quaisquer funções *Soft IP* que os engenheiros de projeto decidem usar são incorporadas ao corpo principal do projeto - que também é especificado em RTL e posteriormente sintetizadas em um grupo de blocos lógicos programáveis.

O *Firm IP* é o meio termo entre o *Hard IP e o Soft IP*, que também vem na forma de uma biblioteca de funções em alto nível (MAXFIELD, 2008). Ao contrário dos *Soft IP*, essas funções já foram mapeadas, colocadas e roteadas para um grupo de CLBs, às vezes combinados com alguns blocos do tipo *Hard IP*, como multiplicadores, por exemplo.

Os IPs podem ser distribuídos como uma *netlist* sintetizada direcionada aos recursos fornecidos pelo FPGA de interesse. Podem ser desenvolvidos tanto pelo fabricante do FPGA, quanto por terceiros[4].

Os projetistas de IPs geralmente criptografam e ofuscam detalhes de implementação proprietários. Essas medidas são tomadas para ocultar detalhes do projeto e tentar preservar sua integridade (Portillo *et al.*, 2016). Os fornecedores de um módulo 3PIP têm controle total sobre o IP desenvolvido e podem inserir *trojans*, o que seria extremamente difícil, se não impossível, de detectar usando técnicas tradicionais de teste e verificação (BHUNIA; TEHRANIPOOR, 2018). No entanto, existem técnicas específicas e complexas para a exploração da implementação desses módulos 3PIP, de modo que sejam expostos problemas com sua integridade.

[4] Nesses casos chamados de *Third-Part IP*, ou 3PIP

2.4 Segurança de *Hardware*

A segurança da informação ou dos dados continua sendo uma questão de grande preocupação para os projetistas e usuários de sistemas desde o início dos computadores e redes. Consequentemente, a proteção de sistemas e redes contra várias formas de ataques, visando a alteração/vazamento de informações críticas e acesso não autorizado, tem sido amplamente investigada ao longo dos anos (ALKABANI; KOUSHANFAR, 2007). O estudo da segurança de *hardware*, por outro lado, é relativamente novo, pois o *hardware* é tradicionalmente considerado imune a ataques e, portanto, usado como âncora de confiança de um sistema (BHUNIA; TEHRANIPOOR, 2018).

Várias vulnerabilidades de segurança e ataques ao *hardware* foram relatados nas últimas décadas. Anteriormente, eles se concentraram principalmente nas vulnerabilidades dependentes da implementação em chips criptográficos, levando ao vazamento de informações (RAY *et al.*, 2018). No entanto, a massificação da produção de *hardware* eletrônico, como os SoCs baseados em IPs, deu origem a muitas preocupações crescentes de segurança. Um fator que contribui para isso é o controle reduzido de algum fabricante sobre uma das etapas de projeto, fabricação e distribuição dos componentes eletrônicos. Este tipo de falha facilita a inclusão de modificações maliciosas em ICs, também conhecidas como ataques de *trojan* de *hardware* (TEHRANIPOOR; KOUSHANFAR, 2010).

2.4.1 Trojan de Hardware

Por definição, um *trojan* de *hardware* é uma modificação intencional e maliciosa de um projeto de circuito que resulta em comportamento indesejado quando o circuito é implantado (XIAO *et al.*, 2016). Os SoCs atingidos por um *trojan* de *hardware* podem apresentar alterações em suas funcionalidades ou coletar informações confidenciais. Os *trojan* de *hardware* demonstram uma séria falha no processo de criação do *design* de *hardware* sendo implantado em uma operação crítica.

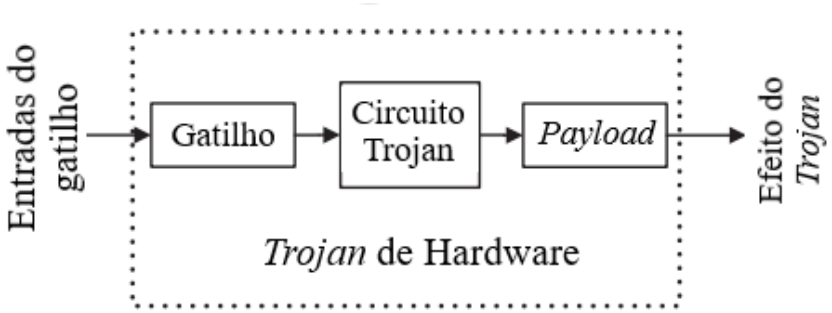
Pelo fato dos *trojans* de *hardware* serem inseridos a nível de *hardware*, as proteções utilizadas em nível de *software* são ineficientes no combate às ameaças trazidas pelo *trojan* de *hardware* (LI *et al.*, 2016). Além disso, a detecção de um *Trojan de Hardware* (HT) a nível de *hardware* é muito difícil, pois se torna inviável comparar um determinado design com um "design modelo" durante a verificação.

Uma maneira de identificar um *trojan* de *hardware* é ativá-lo e observar seus efeitos, mas o tipo, tamanho e localização de um Trojan são desconhecidos e sua ativação é, na grande maioria das vezes, um evento raro (BHUNIA; TEHRANIPOOR, 2018). Um *trojan* de *hardware* pode, portanto, estar oculto durante o funcionamento normal do *chip* e ser ativado somente quando uma condição de disparo é aplicada.

2.4.2 Estrutura de um trojan de hardware

A estrutura básica de um *trojan* em um *Third Party Intellectual Property* (3PIP) tipicamente iclui duas partes principais: um gatilho (*trigger*) e um *payload*[5] (NAHIYAN; TEHRANIPOOR, 2017). O gatilho permite que o *trojan* seja ativado de acordo com determinadas situações, podendo monitorar diferentes sinais ou uma série de eventos no circuito. O *payload* geralmente passa pelos sinais do circuito original (sem estar infectado por um *trojan*) e a saída do gatilho. Depois que o gatilho detecta um evento ou uma condição, o *payload* é ativado para executar um comportamento malicioso. Normalmente, espera-se que o gatilho seja ativado sob condições extremamente raras, portanto o *payload* permanece inativo a maior parte do tempo. Quando o *payload* está inativo, o sistema age como um circuito sem um HT, dificultando a detecção do HT no sistema (BHUNIA; TEHRANIPOOR, 2018). A Figura 4 mostra a estrutura mais comum de um *trojan* de *hardware*.

Figura 4 – Estrutura típica de um *Trojan* de *Hardware*



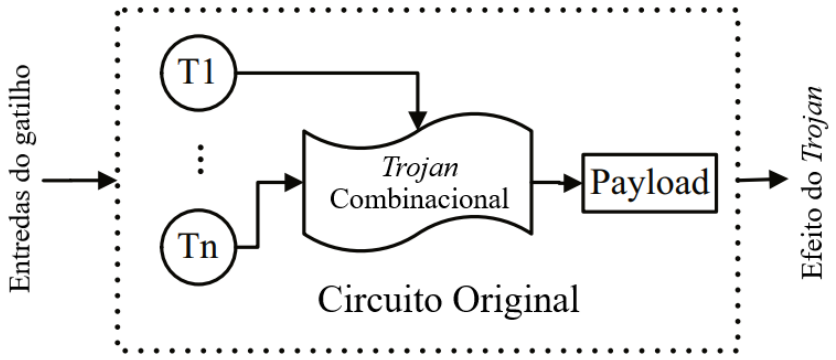
Fonte: Adaptado de Li *et al.* (2016)

Geralmente, os HTs podem ser categorizados em dois tipos: *trojan* combinacional e *trojan* sequencial, ilustrados nas figuras 5 e 6, respectivamente. Um *trojan* combinacional depende da ocorrência simultânea de um conjunto de sinais para desencadear um mau funcionamento. O *trojan* sequencial passa por uma sequência de eventos, cada um disparado por

[5] A tradução literal de *payload* é carga útil, porém, a literatura adota o termo em inglês, assim como este trabalho.

diferentes conjuntos de sinais, antes de ativar o *payload*.

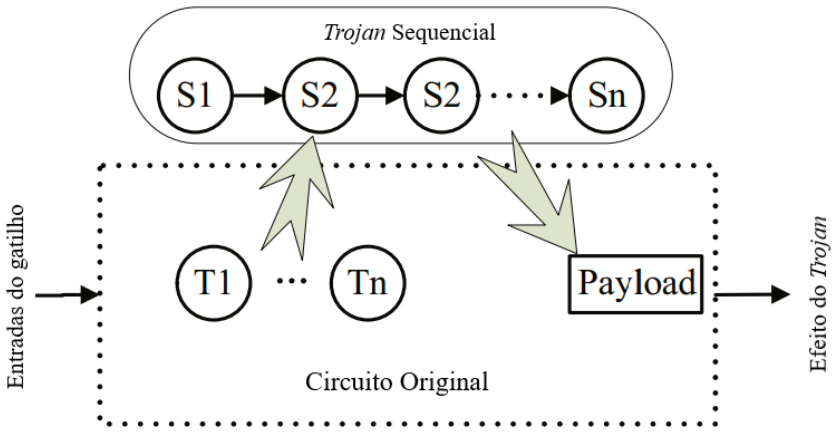
Figura 5 – Modelo para circuitos de *trojan* de *hardware* combinacional



Fonte: Adaptado de Li *et al.* (2016)

Baseado na condição de disparo, os *trojans* de *hardware* podem ser classificados em analógicos e digitais. Os *trojans* analógicos são ativados por condições analógicas, como temperatura, *delay* ou efeito de envelhecimento do dispositivo, enquanto os *trojans* digitais são acionados por algumas funções lógicas booleanas. Da perspectiva do *payload*, um *trojan* pode, por exemplo, causar uma quebra de confiança por meio de um sinal de rádio transmitido ou uma interface de porta de dados serial.

Figura 6 – Modelo para circuitos de *trojan* de *hardware* sequencial



Fonte: Adaptado de Li *et al.* (2016)

2.5 Outros conceitos utilizados

2.5.1 Ataque de canal lateral

Um ataque de canal lateral é qualquer ataque baseado em informações obtidas com a implementação de um sistema (ASHOKKUMAR *et al.*, 2016), em vez de fragilidades no próprio algoritmo implementado (por exemplo, análise de criptografia e *bugs* de *software*). Informações de tempo, consumo de energia, vazamentos eletromagnéticos ou mesmo som podem fornecer uma fonte extra de informações, que pode ser explorada. Alguns ataques de canal lateral exigem conhecimento técnico da operação interna do sistema, embora outros, como a análise de potência diferencial, sejam eficazes (YUAN *et al.*, 2017).

2.5.2 Gerador de números verdadeiramente aleatórios (TRNG)

Um gerador de números verdadeiramente aleatórios (do inglês *True Random Number Generator*) é um dispositivo que gera números aleatórios a partir de um processo físico, e não por meio de um algoritmo. Esses dispositivos geralmente são baseados em fenômenos microscópicos que geram sinais de "ruído"estatisticamente aleatórios de baixo nível, como ruído térmico, efeito fotoelétrico, envolvendo um divisor de feixe e outros fenômenos (LEE *et al.*, 2018). Esses processos estocásticos são, em teoria, completamente imprevisíveis, e as afirmações de imprevisibilidade da teoria estão sujeitas a testes experimentais (MARANDI *et al.*, 2012). Isso contrasta com o paradigma da geração de números pseudo-aleatórios comumente implementados em programas de computador.

3 TRABALHOS RELACIONADOS

Como trabalhos relacionados foram selecionados três pesquisas. No trabalho de Ender *et al.* (2017) são feitas alterações na frequência do *clock*, o que torna o sistema vulnerável ao furto de informações. Já no trabalho desenvolvido por Paar *et al.* (2017) um *trojan* de *hardware* é inserido em uma unidade USB comercial, onde os dados são criptografados usando o algoritmo AES-256. Neste trabalho são realizadas alterações no *bitstream* do FPGA na parte onde esta implementado o algoritmo do AES-256. No trabalho de Ghandali *et al.* (2019) é projetado um *trojan* para uma arquitetura TRNG, o acionamento do *trojan* esta relacionado à temperatura do sistema, quando a temperatura esta em níveis normais o TRNG funciona como esperado, porém em altas temperaturas o módulo é desativado, e a saída do módulo pode ser prevista.

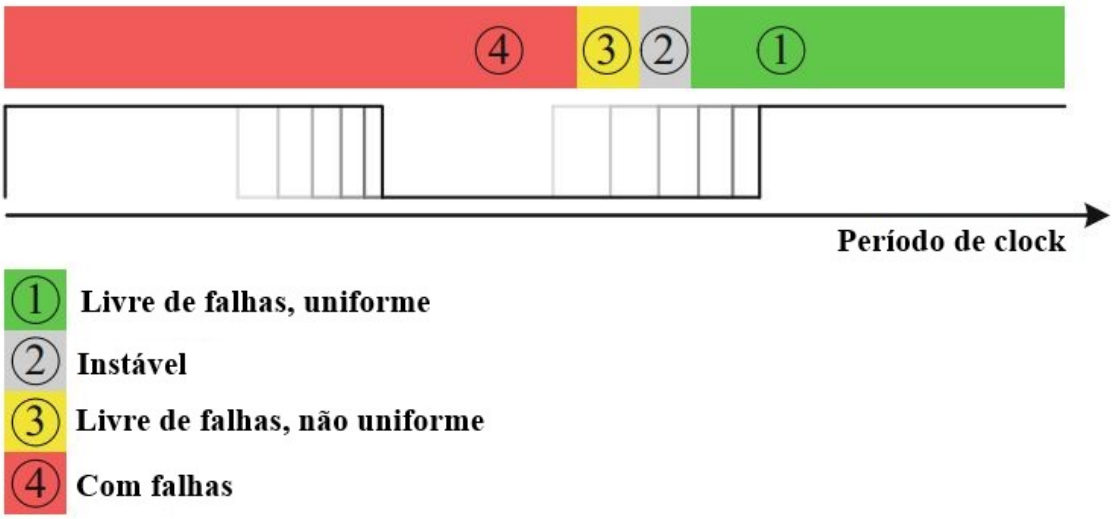
3.1 *The first thorough side-channel hardware trojan*

No trabalho de Ender *et al.* (2017), é apresentado um mecanismo de inserção de *trojans* de *hardware* extremamente furtivos em primitivas criptográficas equipadas com contramedidas de canal lateral de primeira ordem comprovadamente seguras. Depois que o *trojan* é acionado, o design malicioso mostra falhas exploráveis no canal lateral, levando a ataques bem-sucedidos de recuperação de chaves. Geralmente, esses *trojans* não requerem adições nem remoções de nenhuma lógica, o que torna extremamente difícil a detecção. Nos ASICs, ele pode ser inserido por manipulações sutis nos FPGAs, alterando o roteamento de sinais específicos, levando à sobrecarga lógica. O conceito subjacente baseia-se na modificação de uma implementação de *hardware* mascarada com segurança, de forma que a execução do dispositivo em uma frequência de *clock* específica viole uma de suas propriedades essenciais, levando a falhas exploráveis. Os autores aplicaram a técnica a uma implementação de limite da cifra de bloco PRESENT realizada em duas tecnologias *Semicondutor de óxido metálico complementar* (CMOS) diferentes e mostraram que o acionamento do *trojan* torna os protótipos ASIC vulneráveis.

Com base Figura 7, os status do dispositivo pode ser categorizado em quatro estados:

- Com uma frequência de *clock* baixa (período mostrado em verde), o dispositivo opera sem falhas e mantém a uniformidade;
- Aumentando a frequência do *clock* (no período mostrado em cinza), o circuito começa a

Figura 7 – Status do design com *trojan* em diferentes frequências de *clock*



Fonte: Adaptado de Ender *et al.* (2017)

ficar instável, quando as correções não cancelam completamente o efeito uma da outra, o tempo de espera ou o tempo de configuração dos registradores são violados;

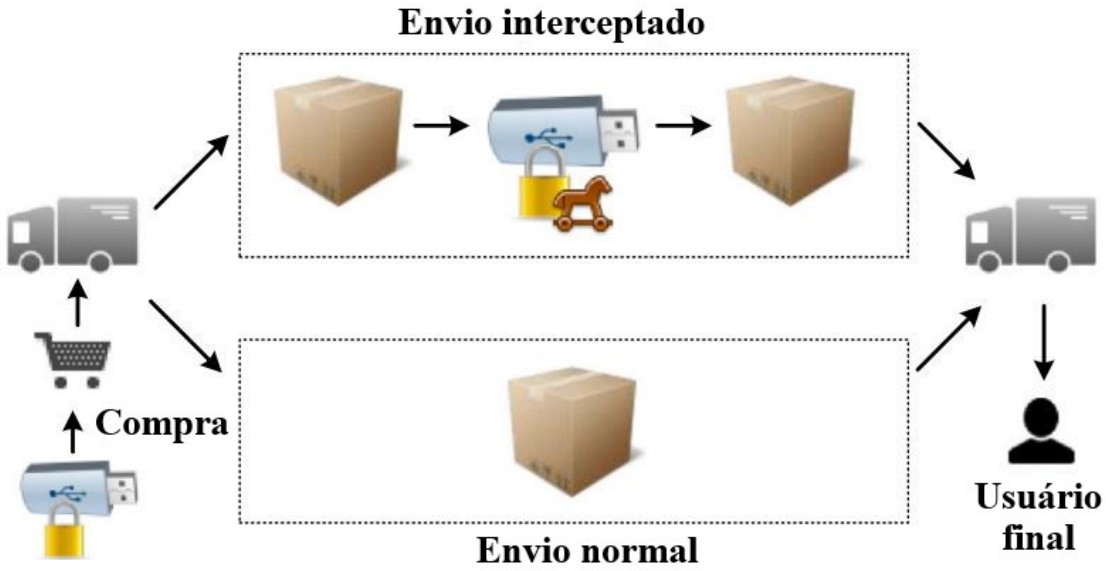
- Aumentando a frequência do *clock* (período em amarelo), o atraso de ambas as instâncias de correção é violado e os circuitos operam sem falhas, mas não mantém a uniformidade;
- Aumentando ainda mais a frequência do *clock* (área em vermelho), o período do *clock* se torna menor que o atraso do caminho crítico do restante do circuito e o dispositivo não funciona corretamente.

3.2 *Interdiction in practice – hardware trojan against a high-security USB flash drive*

Neste artigo, Paar *et al.* (2017) demonstra uma inserção bem-sucedida de *trojan* de *hardware* em um produto comercial. No dispositivo de destino, uma unidade *flash Universal Serial Bus* (USB) com certificado FIPS-140-2 nível 2 da Kingston, os dados do usuário são criptografados usando o AES-256 no modo XTS, e a encriptação/decriptação é processada por uma unidade baseada em SRAM pronta para uso em FPGA. Neste artigo são demonstradas duas etapas de engenharia reversa, relacionadas ao *bitstream* do FPGA proprietário e ao *firmware* da Unidade Central de Processamento (CPU) do ARM. No cenário de inserção de *trojan* apresentado no trabalho, a unidade USB é interceptada antes de ser entregue ao usuário, a Figura 8 ilustra o cenário proposto. A inserção física de *trojan* requer a manipulação do conteúdo da memória *flash* SPI, que contém o *bitstream* do FPGA e o código da CPU do ARM. A manipulação do

bitstream do FPGA altera o algoritmo AES-256 explorado de forma que ele se transforme em uma função linear que pode ser interrompida com 32 pares conhecidos de texto sem criptografia e texto com criptografia. Após o uso da unidade USB manipulada pela vítima, o invasor pode obter todos os dados do usuário dos textos cifrados. Neste trabalho são destacados os riscos à segurança e, especialmente, a relevância prática dos ataques de modificação de *bitstream* que se tornaram realistas.

Figura 8 – Cenário proposto no trabalho de Paar *et al.* (2017)



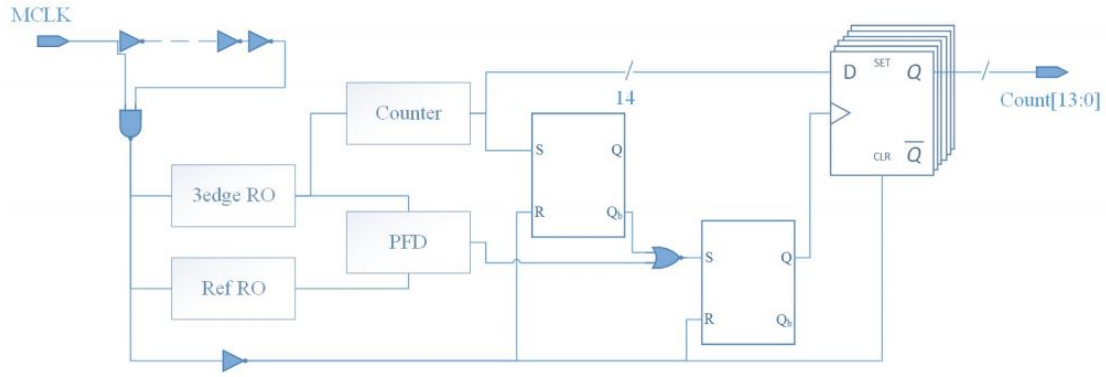
Fonte: Adaptado de (PAAR *et al.*, 2017)

3.3 *Temperature-based hardware trojan for ring-oscillator-based TRNGs*

Neste trabalho, Ghandali *et al.* (2019) apresentam um mecanismo para projetar um *trojan* de *hardware* paramétrico furtivo para uma arquitetura TRNG baseada em uma arquitetura de oscilador em anel proposta no trabalho de Yang *et al.* (2014). A Figura 9 mostra a arquitetura TRNG, baseada no tempo de colapso do oscilador em anel. Quando o *trojan* é acionado, o TRNG malicioso gera saídas não aleatórias previsíveis. Esse *trojan* não requer nenhuma lógica adicional e é puramente baseado em manipulações sutis no nível do sub-transistor. A ideia é desativar o módulo TRNG em alta temperatura para acionar o *trojan*, garantindo que o TRNG infectado por *trojan* funcione corretamente em condições normais. Os autores mostram como um ataque pode ser realizado com o design TRNG infectado por *trojan*, no qual o invasor usa

um modelo estocástico de Markov Chain para prever as saídas de entropia de forma reduzida.

Figura 9 – Diagrama de blocos do sistema TRNG



Fonte: Yang *et al.* (2014)

Neste capítulo foram mostrados trabalhos que se relacionam com a pesquisa desenvolvida aqui. O Quadro 1 faz uma comparação entre os três trabalhos mostrados nesta seção e a pesquisa desenvolvida neste trabalho.

Quadro 1 – Comparação entre os trabalhos relacionados e este trabalho

Trabalhos	**Contexto**	**Plataforma**
Ender *et al.* (2017)	Criptografia de canal lateral	FPGA
Paar *et al.* (2017)	Criptografia	CPU ARM / FPGA
Ghandali *et al.* (2019)	Gerador de números aleatórios	FPGA
Este trabalho	Gerador de números aleatórios e 3PIP	FPGA

Fonte: Elaborada pelo autor.

4 PROCEDIMENTOS METODOLÓGICOS

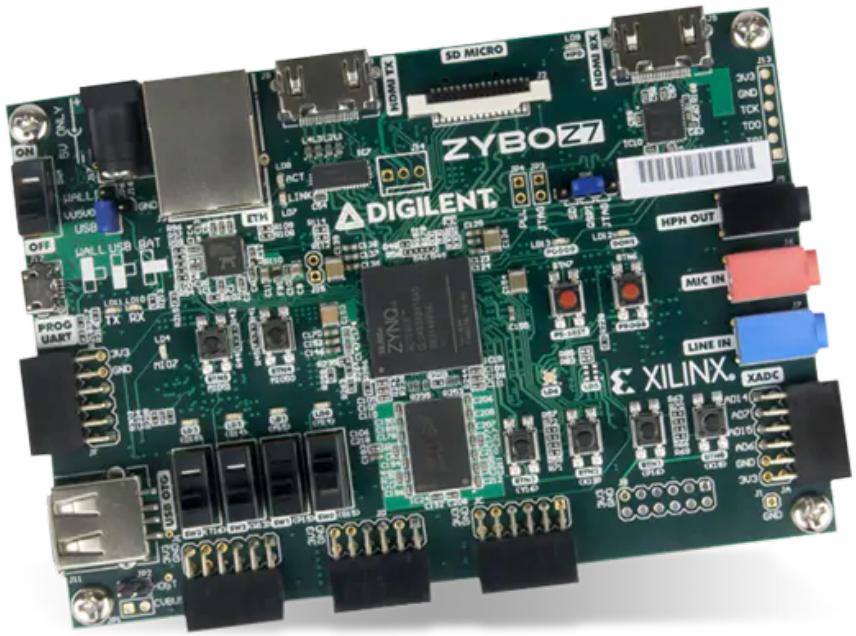
O sistema proposto foi desenvolvido na Universidade Federal do Ceará - Campus Quixadá, na cidade de Quixadá-CE. Como material foi utilizado o kit *Zybo Board*[1] (Figura 10), distribuído pela Digilent [2]. A placa possui as seguintes especificações:

- Processador ZYNQ
  - Processador de 667 MHz dual-core Cortex-A9 ;
  - Controlador de memória DDR3L com 8 canais *Direct Memory Access* (DMA) e 4 portas AXI3 de alto desempenho;
  - Controladores periféricos de baixa largura de banda: *Serial Peripheral Interface* (SPI), UART, *Controller Area Network* (CAN), *Inter-Integrated Circuit* (I2C);
  - Programável a partir de JTAG, flash Quad-SPI e cartão microSD;
  - Lógica programável equivalente ao FPGA Artix-7.
- Memória
  - 1 GB DDR3L com barramento de 32-bit e frequência de 1066 MHz;
  - Flash Quad-SPI de 16 MB com número aleatório de 128 bits programado na fábrica e identificador exclusivo de 48 bits compatível com a EUI-48/64^{TM};
  - *Slot* microSD.
- Alimentação
  - Alimentado por USB ou qualquer fonte de alimentação externa de 5V;
- USB e Ethernet
  - Gigabit Ethernet PHY;
  - Circuito de programação USB-JTAG;
  - USB 2.0 OTG PHY com suporte a *host* e dispositivo.
- Áudio e vídeo
  - Conector de câmera com suporte MIPI CSI-2;
  - Porta de origem HDMI (saída) com *Consumer Electronics Control* (CEC);
  - *Codec* de áudio com fone de ouvido estéreo, entrada de linha estéreo e conectores de microfone.

[1] https://www.xilinx.com/products/boards-and-kits/1-4azfte.html
[2] https://store.digilentinc.com/

- *Switches, Push-buttons* e *LEDs*
  - 6 *Push-buttons*;
  - 4 *Switches*;
  - 6 LEDs (1 LED RGB).
- 17,600 LUTs;
- 35,200 Flip-flops;
- 32 Pinos de entrada/saída.

Figura 10 – *Zybo Board*.



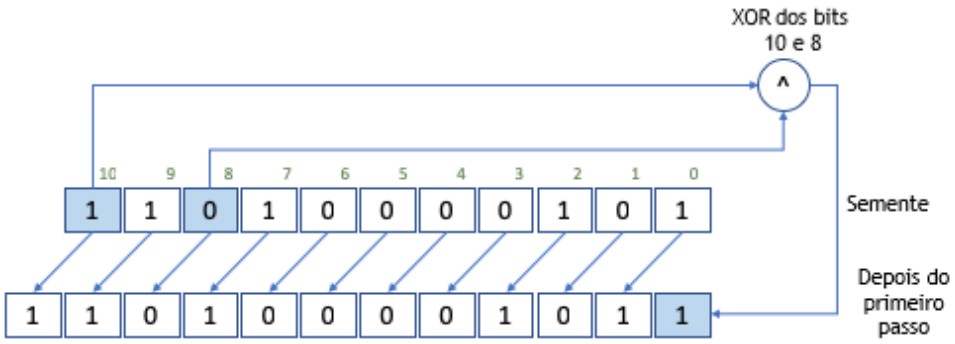
Fonte: (DIGILENT, 2019)

A plataforma de *software* utilizada para desenvolver este trabalho foi a plataforma Vivado, na versão 2016.4. Nesta plataforma foi implementada toda a arquitetura desenvolvida durante a realização deste trabalho. Foram utilizados alguns módulos IP educacionais, são eles: uma CPU, uma memória *Random Access Memory* (RAM), uma memória *Read Only Memory* (ROM), um Multiplexador (MUX) de duas entradas, um *debugger Video Graphics Array (VGA)*, um controlador VGA. O módulo LFSR foi implementado utilizando como base uma implementação disponível no site *https://www.nandland.com/*.

O módulo LFSR é um pseudo-gerador de números aleatórios, geralmente um registrador de deslocamento cujo bit de entrada é acionado pela porta lógica XOR de alguns bits do

valor geral do registrador de deslocamento. O número máximo possível de iterações de qualquer LFSR é dado por $2^{N\_BITS} - 1$, na Figura 11, temos $2^{11} - 1 = 2047$. A Figura 11 mostra um LFSR de 11-bits onde o bit menos significativo é substituído pela XOR dos bit 10 e bit 8, esta implementação permite gerar sequências de 2047 valores inteiros positivos distintos, no intervalo de [1,2047]. Como a operação do registrador é determinística, o fluxo de valores produzidos por ele é completamente determinado pelo seu estado atual (ou anterior). Da mesma forma, como o registrador possui um número finito de estados possíveis, ele deve, eventualmente, entrar em um ciclo de repetição.

Figura 11 – Funcionamento de um LFSR de 11 bits



Fonte: Elaborada pelo autor.

Foram implementados quatro cenários para demonstração da inserção de *trojans* em *hardware*. São eles:

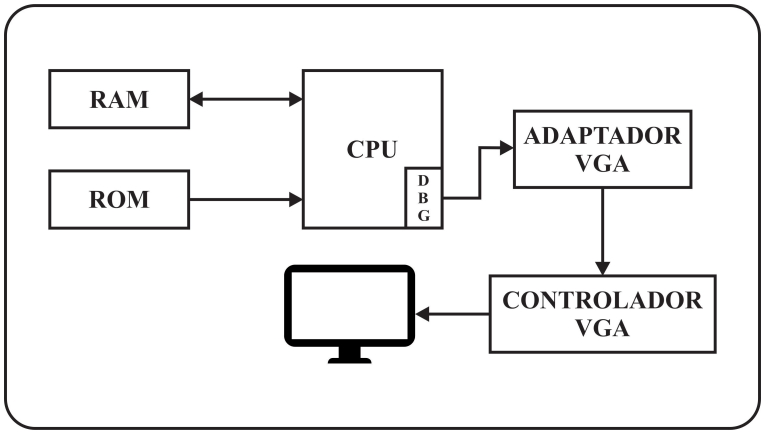
- **1A**: Execução de código em ROM, livre de *trojans*;
- **1B**: Cenário 1A modificado com a inserção de *trojan* para execução de código malicioso;
- **2A**: Geração de sequência aleatória com LFSR;
- **2B**: Cenário 2A modificado com a inserção de *trojan* para geração de sequência aleatória com menor nível de segurança.

Os cenários 1A e 2A são cenários de operação normal, que servem de base para comparação com os cenários 1B e 2B, respectivamente, onde há a inserção de *trojans* de *hardware*. O *trojan* inserido no cenário 1B, causa uma falha imediata no sistema, enquanto que o do cenário 2B atua de forma mais sutil, já a operação continua, porém com uma condição de menor segurança.

No cenário 1A, mostrado na Figura 12, há apenas a execução de um programa que

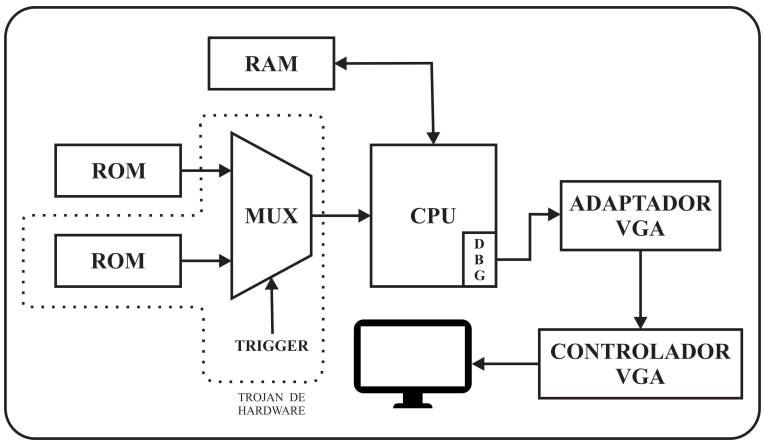
está na memória ROM. Já no cenário 1B (Figura 13) foi inserido mais dois blocos na arquitetura: uma memória ROM, com um novo programa e um MUX. Neste cenário então temos duas memória ROM com dois programas diferentes, sendo a primeira memória ROM onde esta gravado o mesmo programa do cenário 1A e uma segunda memória ROM onde está gravado um programa com um comportamento diferente do esperado pelo usuário. Na Figura 13 a memória ROM e o MUX que representam o *Trojan* de *Hardware* estão explicitados pelo bloco tracejado.

Figura 12 – Cenário 1A proposto no primeiro experimento



Fonte: Elaborada pelo autor.

Figura 13 – Inserção do *Trojan* de *Hardware* no cenário 1A ilustrado na Figura 12



Fonte: Elaborada pelo autor.

Nos experimentos realizados com os cenários 1A, 2A e 2B, o programa salvo na memória ROM executa as instruções descritas no trecho de código mostrado na Figura 14a. No cenário 1B, temos duas memórias ROMs uma executando as instruções mostradas em 14a e a outra, executando um código malicioso. Com o intuito de facilitar os testes em bancada, os

triggers dos *trojans* implementados nos cenários 1B e 2B estão de forma manual, geralmente são analisados de forma automática sinais que são ativados em circunstâncias raras, a fim de dificultar ainda mais a detecção do *trojan* .

Figura 14 – Programas salvo nas ROMs

```
1 NOP
2 MOV Rd, RM
3 MOV Rd, #Im
4 ADD Rd, Rm, Rn
5 STORE [Rm], Rn
6 STORE [Rm], #Im
7 LOAD Rd, [Rm]
8 LOAD Rd, [Rm]
9 ADD Rd, Rm, Rn
10 SUB Rd, Rm, Rn
11 MUL Rd, Rm, Rn
12 AND Rd, Rm, Rn
13 ORR Rd, Rm, Rn
14 XOR Rd, Rm, Rn
15 NOT Rd, Rm
16 HALT
```

(a) Código Original

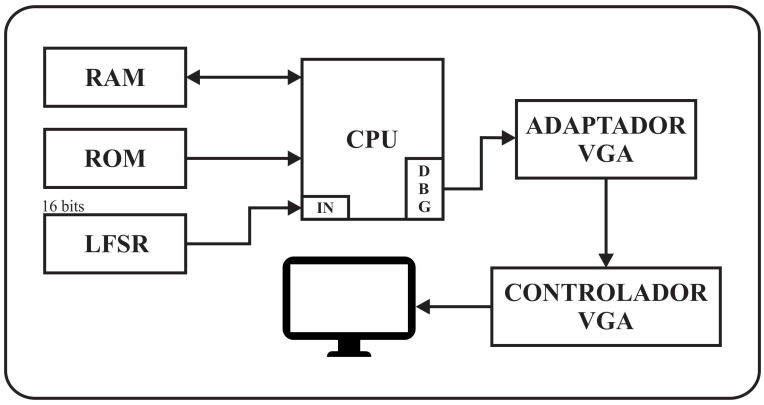
```
1 NOP
2 MOV Rd, RM
3 MOV Rd, #Im
4 ADD Rd, Rm, Rn
5 STORE [Rm], Rn
6 HALT
```

(b) Código Malicioso

Fonte: Elaborado pelo autor.

No cenário 2A (Figura 15), incluímos o bloco do LFSR como uma entrada de números pseudo-aleatórios na CPU de 16-bits, que pode ser usado em algum método de um sistema criptográfico. Neste trabalho o número gerado pelo LFSR é mostrado no monitor VGA.

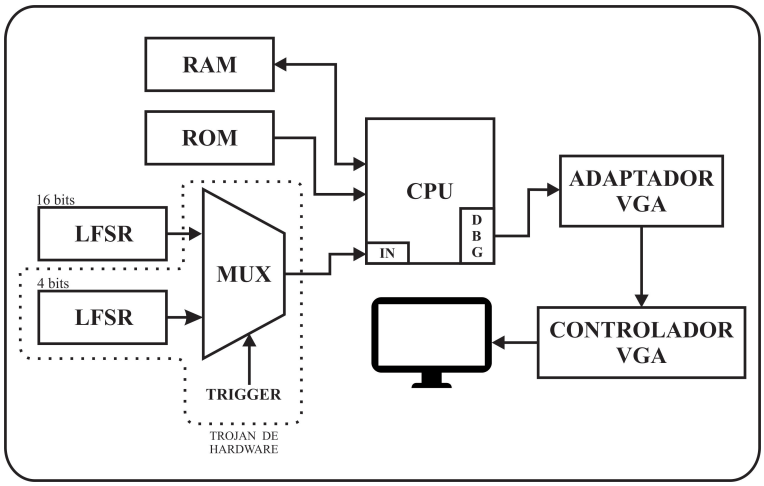
Figura 15 – Cenário 2A proposto no segundo experimento.



Fonte: Elaborada pelo autor.

No cenário 2B (Figura 16), foi realizada uma adaptação do cenário 2A (Figura 15), foi inserido o *Torjan* de *Hardware* que consiste no chaveamento, por meio de um MUX, entre o módulo LFSR de 16-bit e um módulo LFSR de 4-bits. Fazendo assim que a sequência de números geradas venha a se repetir em um espaço menor de tempo, diminuindo a confiabilidade do sistema e se tornando quase imperceptível ao usuário, sem que este realize um análise profunda no sistema. Os itens que compõem *Trojan* de *Hardware* que foi inserido são destacados pelo bloco tracejado.

Figura 16 – Inserção do *trojan* de *hardware* no cenário 2A ilustrado na Figura 15



Fonte: Elaborada pelo autor.

O Quadro 2 faz uma comparação entre os cenários propostos neste trabalho, mostrando seus objetivos, tipo e um resumo dos itens utilizados.

Quadro 2 – Comparação entre os cenários propostos

Cenários / Características	Resumo	Tipo	Objetivo
Cenário 1A	1 RAM 1 ROM 1 CPU 1 ADAPTADOR VGA 1 CONTROLADOR VGA 1 MONITOR	NORMAL	FLUXO NORMAL DE UM PROGRAMA
Cenário 1B	1 RAM 2 ROM 1 MUX 1 CPU 1 ADAPTADOR VGA 1 CONTROLADOR VGA 1 MONITOR	TROJAN INSERIDO	CHAVEAMENTO ENTRE MEMÓRIAS ROMs, COM DOIS PROGRAMAS DIFERENTES
Cenário 2A	1 RAM 1 ROM 1 LFSR DE 16 BITS 1 CPU 1 ADAPTADOR VGA 1 CONTROLADOR VGA 1 MONITOR	NORMAL	FLUXO NORMAL DE UM PROGRAMA
Cenário 2B	1 RAM 1 ROM 1 LFSR DE 16 BITS 1 LSFR DE 4 BITS 1 MUX 1 CPU 1 ADAPTADOR VGA 1 CONTROLADOR VGA 1 MONITOR	TROJAN INSERIDO	CHAVEAMENTO ENTRE DOIS MÓDULOS LFSR, COM COM TAMANHOS DIFERENTES

Fonte: Elaborada pelo autor.

5 EXPERIMENTOS E RESULTADOS

O cenário 1A foi implementado conforme Apêndice A. Os dados de síntese obtidos após a geração do *bitstream* são mostrados na Tabela 1. Neste cenário o código que esta salvo na memória ROM é um código *dummy*/exemplo.

Tabela 1 – Dados de síntese do cenário 1A

Recurso	Utilizados	Disponível	Percentual de utilização (%)
LUT	619	17600	3,52
LUTRAM	64	6000	1,07
Flip-Flops	289	35200	0,82
BRAM	1.50	60	2,50
DSP	1	80	1,25

Fonte: Elaborada pelo autor.

Conforme apresentado no Apêndice B, o cenário 1B foi implementado tendo como base o cenário 1A. Nele foi acrescentado uma memória ROM e MUX para selecionar qual das duas memórias ROM irão ser utilizadas. A Tabela 2 mostra os dados de síntese obtidos.

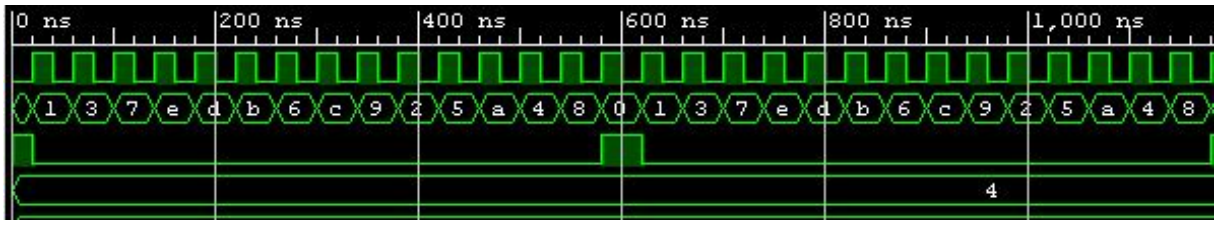
Tabela 2 – Dados de síntese do cenário 1B

Recurso	Utilizados	Disponível	Percentual de utilização (%)
LUT	633	17600	3,60
LUTRAM	64	6000	1,07
Flip-Flops	300	35200	0,85
BRAM	1,50	60	2,50
DSP	1	80	1,25

Fonte: Elaborada pelo autor.

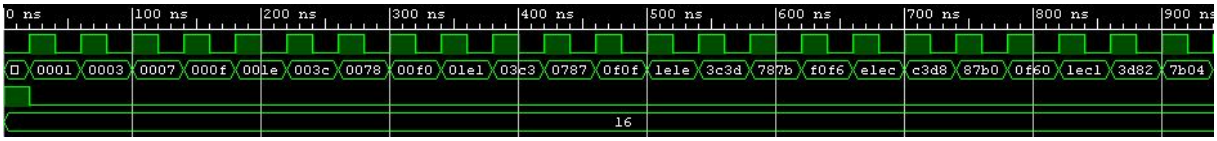
Para a criação dos próximo cenários foi implementado um bloco com o componente LFSR e realizada a simulação do comportamento na plataforma Vivado. Foram implementados e simulados LFSRs com dois tamanhos diferentes, sendo um com 16-bits e outro com 4-bits. As simulações podem ser vistas nas Figuras 17 e 18, onde os sinais são, de cima para baixo: *clock*, número gerado, *start* e número de *bits*. Na Figura 17 é visível a repetição da sequência gerada (a partir de 580 ns), já na Figura 18 não é possível visualizar a repetição, dado maior número de bits. Apesar de um LFSR de 16 bits não prover um alto nível de segurança, ele foi utilizado para questão de demonstração.

Figura 17 – LFSR 4 bits



Fonte: Elaborada pelo autor.

Figura 18 – LFSR 16 bits



Fonte: Elaborada pelo autor.

Um novo cenário base foi criado, como mostrado no Apêndice C. Neste cenário também foi utilizado o módulo LFSR de 16-*bits* implementado como uma entrada da CPU, os dados de síntese são mostrados na Tabela 3.

Tabela 3 – Dados de síntese do cenário 2A

Recurso	Utilizados	Disponível	Percentual de utilização (%)
LUT	646	17600	3,67
LUTRAM	64	6000	1,07
Flip-Flops	305	35200	0,87
BRAM	1,50	60	2,50
DSP	1	80	1,25

Fonte: Elaborada pelo autor.

No cenário 2B, foi desenvolvido tendo como base o cenário 2A, como apresentado no Apêndice D. Neste cenário foi adicionado um módulo LSFR de 4-*bits* e um MUX para ativação do *trojan*. Neste cenário o objetivo é não deixar perceptível o *trojan* inserido, pois diminuindo o número de *bits* do LFSR diminuímos o nível de segurança do sistema mas mantemos o comportamento do circuito aparentemente sem alterações. Os dados de síntese deste cenário são mostrados na Tabela 4.

Tabela 4 – Dados de síntese do cenário 2B

Recurso	Utilizados	Disponível	Percentual de utilização (%)
LUT	654	17600	3,72
LUTRAM	64	6000	1,07
Flip-Flops	321	35200	0,91
BRAM	1,50	60	2,50
DSP	1	80	1,25

Fonte: Elaborada pelo autor.

Se compararmos os dados das Tabelas 1 e 2, percebemos um pequeno aumento nos números de LUTs e *Flip-Flop*s, isso se dá ao fato de termos adicionado mais dois blocos em nossa arquitetura, um MUX e uma memória ROM. A porcentagem no aumento do uso de recuso entre os cenários foi de 0,08% no uso de LUTs e 0,03% no uso de *Flip-Flops*. Do mesmo modo, se compararmos as Tabelas 3 e 4, também é possível perceber um pequeno aumento na utilização dos recursos. O aumento foi de 0,05% na utlização de LUTs e de 0,04% na utilização de *Flip-Flop*s, isso se dá pois foram inseridos dois blocos na arquitetura, um LFSR de 4-*bits* e um MUX.

De forma similar ao que ocorre com a utilização de recursos, também há mudança em relação aos aspectos temporais de execução, que refletem diretamente na frequência de operação do circuito. Na Tabela 5 podemos comparar as variações percebidas nos relatórios de tempo gerado pela plataforma Vivado.

Apesar de tratarem-se de aumentos sutis, essas informações podem ser usadas como contra-medida para evitar ataques similares. Contudo, vale ressaltar que variações nessa ordem de magnitude podem ser geradas também por mudanças nas configurações das ferramentas de síntese ou mesmo por mudançasna plataforma utilizada, variando de FPGA para FPGA.

Tabela 5 – Comparação entre os relatórios de tempo gerados pelo Vivado.

	Timing	
Cenário/Métrica	**Worst Negative Slack (WNS)**	**Worst Hold Slack (WHS)**
Cenário 1A	12,290ns	0,219ns
Cenário 1B	12,755ns	0,179ns
Cenário 2A	11,227ns	0,172ns
Cenário 2B	11,811ns	0,061ns

Fonte: Elaborada pelo autor.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este trabalho teve como objetivo mostrar a inserção de *trojan* de *hardware* em diferentes cenários. Esse tipo de *trojans* é muito difícil de ser detectado, porque, em geral, não requer a remoção de nenhuma lógica do design. Assim, mesmo em um cenário de “caixa branca", o *trojan* dificilmente será detectado por uma verificação de *hardware* de rotina. O conceito implícito do trabalho é baseado no aumento da probabilidade de que uma sequência de números se repita quando o *trojan* é acionado. A inserção do *trojan* no sistema, foi realizada durante o desenvolvimento do design, em uma de suas partes mais críticas e vulneráveis, a criação de 3PIP. Esse *trojan* permite reduzir significativamente o nível de segurança, mesmo em implementações criptográficas altamente protegidas que estão relacionados ao *LFSR*.

Trabalhos futuros incluem a criação de novos cenários principalmente ligados a TRNGs, pois este tipo de gerador de sequência numérica é muito aplicado em sistemas criptográficos. Ainda como trabalhos futuros, há a implementação de *trojans* de *hardware* onde o acionamento por meio dos *triggers* seja feita de maneira automática e não de forma manual.

REFERÊNCIAS

ALKABANI, Y.; KOUSHANFAR, F. Active hardware metering for intellectual property protection and security. In: **USENIX Security Symposium**. Boston, MA: USENIX Association, 2007. p. 291–306.

ASHOKKUMAR, C.; GIRI, R. P.; MENEZES, B. Highly efficient algorithms for aes key retrieval in cache access attacks. In: **2016 IEEE European Symposium on Security and Privacy (EuroS P)**. Saarbrucken, Alemanha: IEEE, 2016. p. 261–275.

BARBARESCHI, M.; BAGNASCO, P.; MAZZEO, A. Authenticating iot devices with physically unclonable functions models. In: IEEE. **2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)**. Cracóvia, Polônia, 2015. p. 563–567.

BENNIS, A.; LEESER, M.; TADMOR, G. Implementing a highly parameterized digital piv system on reconfigurable hardware. In: **2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors**. Boston, MA, USA: Institute of Electrical and Electronics Engineers, 2009. p. 32–37.

BHUNIA, S.; TEHRANIPOOR, M. **Hardware security: A hands-on learning approach**. Cambridge, Reino Unido: Morgan Kaufmann, 2018.

DIGILENT. **Zybo Z7: Zynq-7000 ARM/FPGA SoC development board**. 2019. Disponível em: https://store.digilentinc.com/zybo-z7-zynq-7000-arm-fpga-soc-development-board/. Acesso em: 09 Dez. 2019.

DUTRA, N. G. Bacharelado em Engenharia Elétrica, **Estudo e implementação em FPGA de um microprocessador**. Belo Horizonte, Minas Gerais: Centro Federal de Educação Tecnológica de Minas Gerais, Departamento de Engenharia Elétrica, 2016.

ENDER, M.; GHANDALI, S.; MORADI, A.; PAAR, C. The first thorough side-channel hardware trojan. In: **Advances in cryptology – ASIACRYPT 2017**. Cham, Alemanha: Springer International Publishing, 2017. p. 755–780. ISBN 978-3-319-70694-8.

GHANDALI, S.; HOLCOMB, D.; PAAR, C. **Temperature-based hardware trojan for ring-oscillator-based TRNGs**. [S.l.]: arXiv preprint, 2019.

GUIN, U.; HUANG, K.; DIMASE, D.; CARULLI, J. M.; TEHRANIPOOR, M.; MAKRIS, Y. Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. **Proceedings of the IEEE**, IEEE, Auburn, Alabama, v. 102, n. 8, p. 1207–1228, 2014.

KOCHER, P.; JAFFE, J.; JUN, B. Differential power analysis. In: WIENER, M. (Ed.). **Advances in Cryptology - CRYPTO' 99**. Berlim, Heidelberg: Springer Berlin Heidelberg, 1999. p. 388–397. ISBN 978-3-540-48405-9.

KOCHER, P. C. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: **Advances in Cryptology - CRYPTO 96**. Berlim, Heidelberg: Springer Berlin Heidelberg, 1996. p. 104–113. ISBN 978-3-540-68697-2.

KOEUNE, F.; STANDAERT, F.-X. A tutorial on physical security and side-channel attacks. In: ALDINI, A.; GORRIERI, R.; MARTINELLI, F. (Ed.). **Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 78–108. ISBN 978-3-540-31936-8.

KUROSE, J. F. **Computer networking:** A top-down approach featuring the internet, 3/E. Amherst, Canadá: Pearson Education India, 2005.

LEE, K.; LEE, S.; SEO, C.; YIM, K. Trng (true random number generator) method using visible spectrum for secure communication on 5g network. **IEEE Access**, v. 6, p. 12838–12847, 2018. ISSN 2169-3536.

LI, H.; LIU, Q.; ZHANG, J. **A survey of hardware trojan threat and defense**. [S.l.]: Elsevier, 2016. 426–437 p.

MARANDI, A.; LEINDECKER, N. C.; VODOPYANOV, K. L.; BYER, R. L. All-optical quantum random bit generation from intrinsically binary phase of parametric oscillators. **Optics Express**, The Optical Society, v. 20, n. 17, p. 19322, 2012. ISSN 1094-4087.

MAXFIELD, C. **FPGAs: Instant access**. Oxford, Reino Unido: Elsevier, 2008. v. 1.

NAHIYAN, A.; TEHRANIPOOR, M. Code coverage analysis for ip trust verification. In: **Hardware IP security and trust**. Flórida, Estados Unidos da America: Springer, 2017. p. 53–72.

NOLAN, S. M. **Power management for internet of things (IoT) system on a chip (SoC) Development**. 2019. Disponível em: https://www.design-reuse.com/articles/42705/power-management-for-iot-soc-development.html. Acesso em: 05 Nov. 2019.

OLIVEIRA, C. B. d. **LALP+: um framework para o desenvolvimento de aceleradores de hardware em FPGAs** 2015. 164 f. Tese (Doutorado em Ciências - Ciências de Computação e Matemática Computacional) — Instituto de Ciências Matemáticas e de Computação, Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional, Universidade de São Paulo, São Carlos, São Paulo, 2015.

OLIVEIRA, M. C.; BASTOS-FILHO, C. J. A. Uma implementação em fpga de redes neurais de hopfield para roteamento em redes de comunicação. In: **10º Congresso Brasileiro de Inteligência Computacional**. Fortaleza, CE: Anais do 10º Congresso Brasileiro de Inteligência Computacional, 2011. p. 1–8.

PAAR, C.; FYRBIAK, M.; KOPPE, P.; MORADI, A.; SWIERCZYNSKI, P. Interdiction in practice – hardware trojan against a high-security usb flash drive. **Journal of Cryptographic Engineering**, Amherst, USA, v. 7, n. 3, p. 199–211, Sep 2017. ISSN 2190-8516.

Portillo, J.; John, E.; Narasimhan, S. Building trust in 3pip using asset-based security property verification. In: **2016 IEEE 34th VLSI Test Symposium (VTS)**. Las Vegas, NV, USA: Institute of Electrical and Electronics Engineers, 2016. p. 1–6. ISSN 2375-1053.

RAY, S.; PEETERS, E.; TEHRANIPOOR, M. M.; BHUNIA, S. System-on-chip platform security assurance: Architecture and validation. **Proceedings of the IEEE**, Flórida, EUA, v. 106, n. 1, p. 21–37, Janeiro 2018. ISSN 1558-2256.

ROUSE, M. **IP core (intellectual property Core)**. 2011. Disponível em: https://whatis.techtarget.com/definition/IP-core-intellectual-property-core. Acesso em: 26 de Nov. 2019.

TEHRANIPOOR, M.; KOUSHANFAR, F. A survey of hardware trojan taxonomy and detection. **IEEE Design Test of Computers**, v. 27, n. 1, p. 10–25, Janeiro 2010. ISSN 1558-1918.

U.S. SENATE COMMITTEE ON ARMED SERVICES. **Inquiry into counterfeit electronic parts in the department of defense supply chain**. 2012. Disponível em: https://www.armed-services.senate.gov/imo/media/doc/Counterfeit-Electronic-Parts.pdf. Acesso em: 07 Nov. 2019.

VOAS, J. Fault injection for the masses. **Computer**, IEEE, v. 30, n. 12, p. 129–130, 1997.

WILSON, P. **Design recipes for FPGAs: Using verilog and VHDL**. [S.l.]: Newnes, 2015.

WIŚNIEWSKI, R. **Synthesis of compositional microprogram control units for programmable devices**. Zielona Góra, Poland: University of Zielona Góra, 2009.

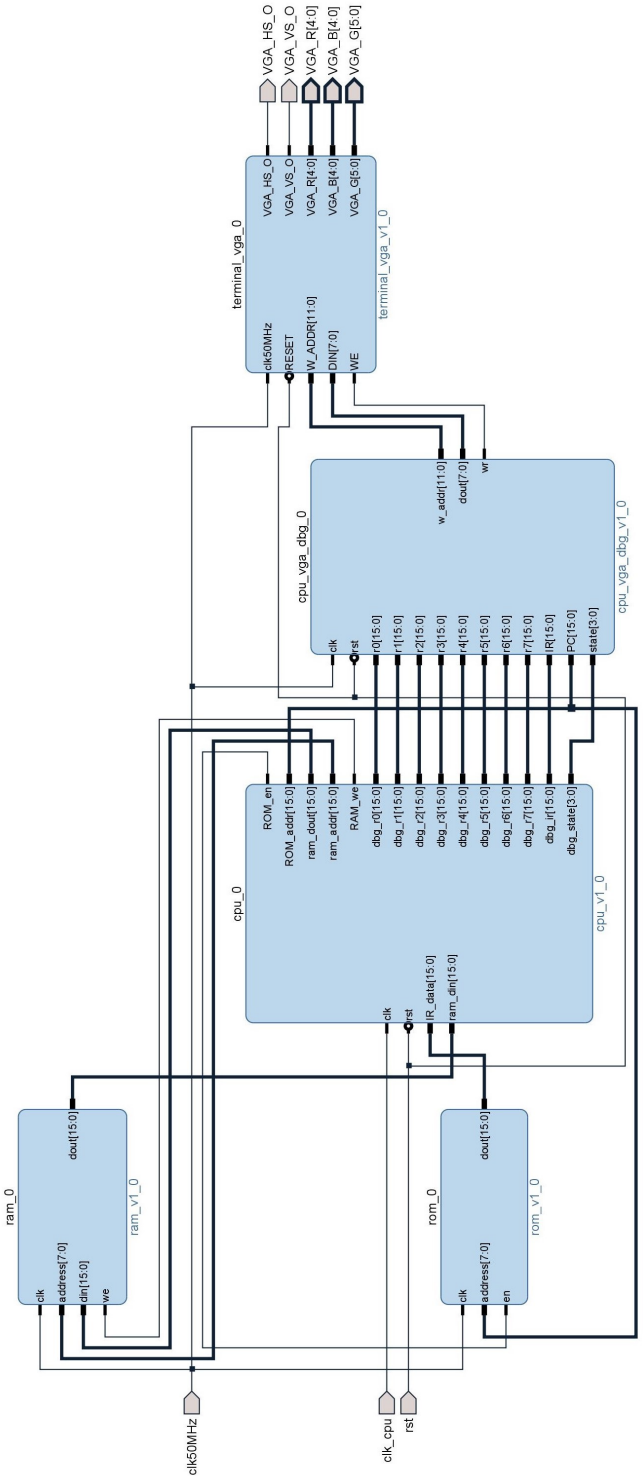
XIAO, K.; FORTE, D.; JIN, Y.; KARRI, R.; BHUNIA, S.; TEHRANIPOOR, M. Hardware trojans: Lessons learned after one decade of research. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM, v. 22, n. 1, p. 6, 2016.

Yang, K.; Fick, D.; Henry, M. B.; Lee, Y.; Blaauw, D.; Sylvester, D. 16.3 a 23mb/s 23pj/b fully synthesized true-random-number generator in 28nm and 65nm cmos. In: **2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)**. São Francisco, Califórnia: Institute of Electrical and Electronics Engineers, 2014. p. 280–281. ISSN 2376-8606.

YUAN, Y.; WU, L.; ZHANG, X.; YANG, Y. Side-channel collision attack based on multiple-bits. In: **2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)**. Xiamen, China: IEEE, 2017. p. 1–5. ISSN 2163-5056.

APÊNDICE A – DIAGRAMA DE BLOCOS NA PLATAFORMA VIVADO PARA O CENÁRIO 1A

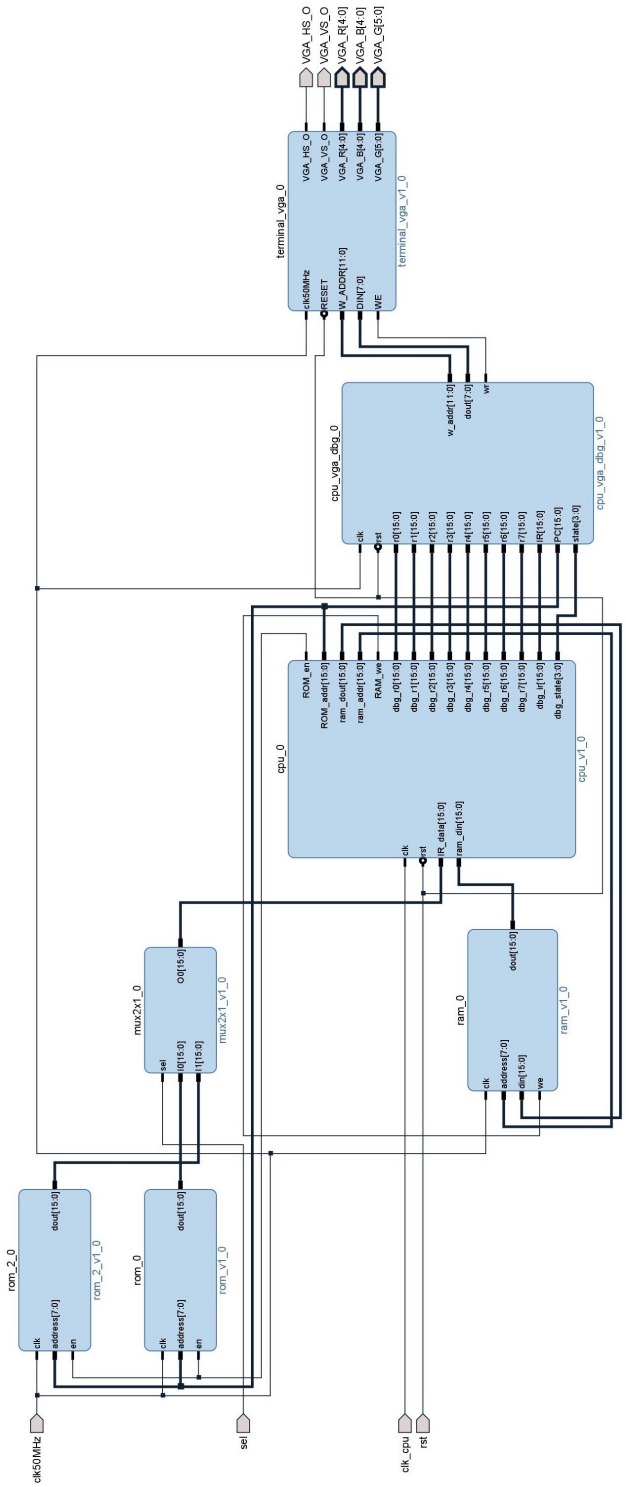
Figura 19 – Diagrama de blocos na plataforma Vivado do cenário 1A.



Fonte: Elaborado pelo autor.

APÊNDICE B – DIAGRAMA DE BLOCOS NA PLATAFORMA VIVADO PARA O CENÁRIO 1B

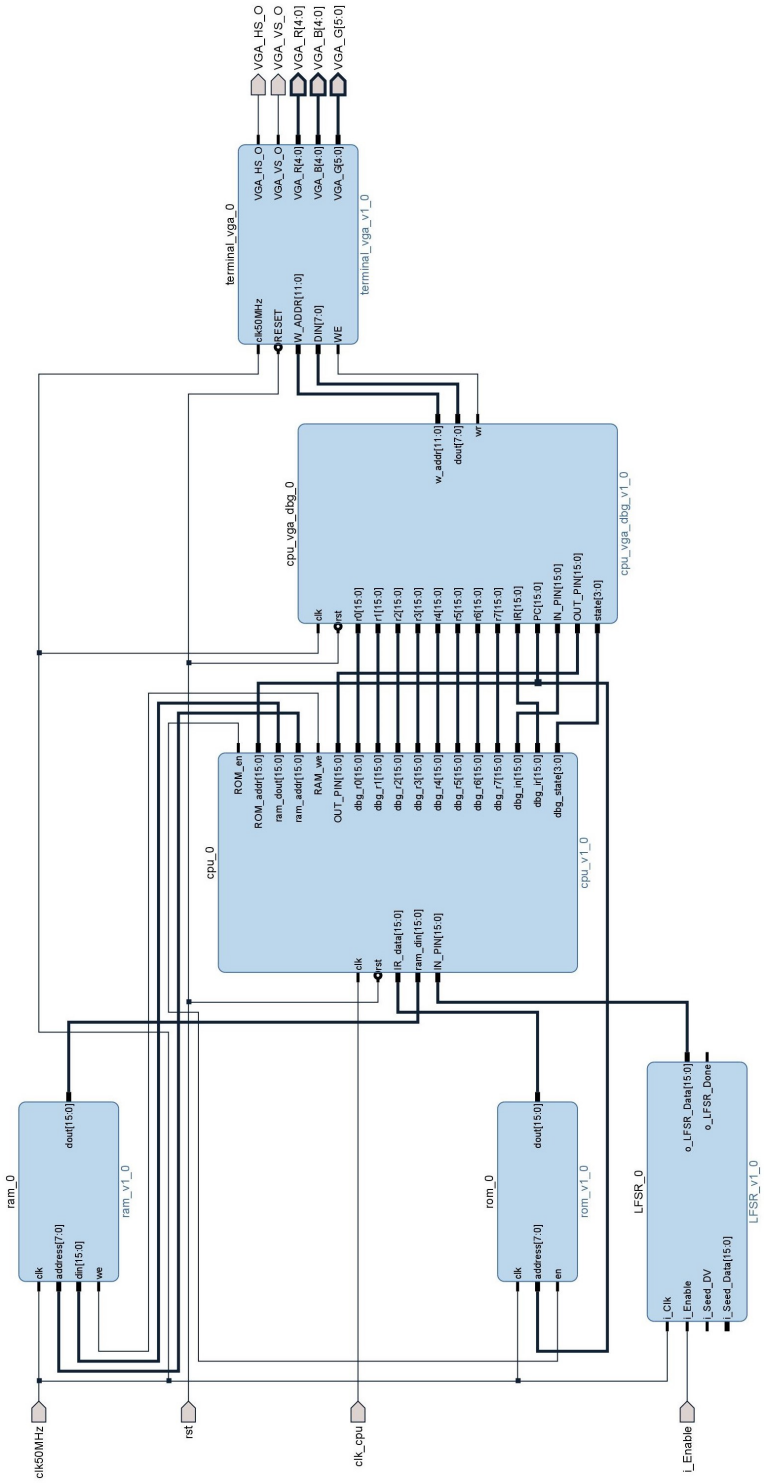
Figura 20 – Diagrama de blocos na plataforma Vivado do cenário 1B



Fonte: Elaborado pelo autor.

APÊNDICE C – DIAGRAMA DE BLOCOS NA PLATAFORMA VIVADO PARA O CENÁRIO 2A

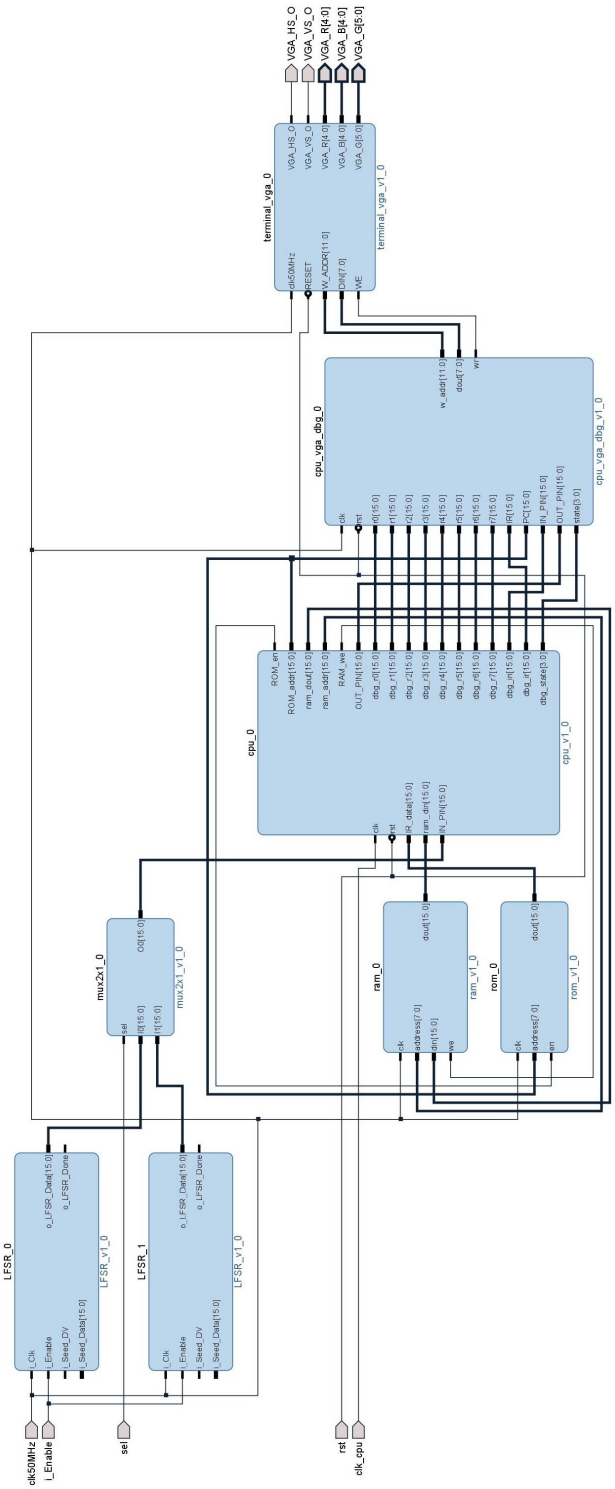
Figura 21 – Diagrama de blocos na plataforma Vivado do cenário 2A



Fonte: Elaborado pelo autor.

APÊNDICE D – DIAGRAMA DE BLOCOS NA PLATAFORMA VIVADO PARA O CENÁRIO 2B

Figura 22 – Diagrama de blocos na plataforma Vivado do cenário 2B



Fonte: Elaborado pelo autor.