Chapter

5

Abordagem Gamificada para Introdução em Arquiteturas de Hardware Evolutivo

Bernardo G. P. Cunha, Carlos Augusto P. da S. Martins

Abstract

In recent years, there has been a search for adaptive systems in a diversity of fields, such as in industries, control systems, aerospace, automobiles, among others. Many approaches were made to reach feasible, adaptive and scalable solutions, and one of them is Evolvable Hardware (EHW). It has the advantages hardware brings along, including high parallelism level, low response time - evolvable systems implemented in hardware can have evolution time of microseconds. The objective of this course is to introduce Evolvable Hardware to computer scientists, computer architects and engineers in a "hands on" approach, first playing a conceptual game, then hacking a developed system and then competing to discover who developed the architecture with best/fast response. After the event, the participants may have a glance at how they can apply hardware computation in their researches or even EHW in their projects.

Resumo

Recentemente há uma procura por sistemas adaptativos ou evolutivos em diversos campos da indústria, em sistemas de controle, aeroespacial, automotiva, entre outras. Muitas abordagens foram desenvolvidas para se chegar a soluções adaptativas, escaláveis e práticas, sendo uma delas o Hardware Evolutivo (EHW). Além de apresentar um elevado nível de paralelismo, sistemas evolutivos implementados em hardware podem apresentar tempo de evolução na ordem de microssegundos. O objetivo deste minicurso é introduzir EHW a cientistas da computação, arquitetos de computação e engenheiros, de maneira "hands on", primeiro introduzindo os conceitos de computação evolucionária como um game, depois hackeando um sistema de EHW já desenvolvido, e por último competindo entre si para descobrir qual arquitetura obteve melhores resultados. Após o minicurso, os participantes terão obtido uma visão geral sobre o tema, e poderão aplicar computação em hardware em suas pesquisas ou até mesmo EHW em seus projetos.

1.1. Descrição do Minicurso - Apresentação

Este minicurso consiste em 3 partes, de aproximadamente 50 minutos. Inicialmente, haverá exposição dos conceitos básicos para compreensão do tema, utilizando abordagem expositiva. Na segunda parte, os participantes colocarão estas novas informações em prática, "hackeando" o sistema evolutivo, experimentando variações nos parâmetros de entrada e de configuração, de uma forma gamificada, em equipes e por turno. Por último, será escolhida a melhor abordagem a ser implementada em um próximo encontro (virtual), e demonstrando como o sistema selecionado pode ser implementado utilizando um dispositivo de computação reconfigurável (FPGA). Aqueles que desejarem pesquisar mais sobre o assunto, pode acessar [**Cunha**, **2020**] para mais conteúdo.

Figure 1.1. Contexto envolvendo EHW. Autores, 2023.

1.2. Estrutura do texto

A estrutura do texto se dispõe em cinco etapas (subcapítulos): uma introdutória, contendo um breve resumo do estado da arte e origem de Evolvable Hardware; em seguida a apresentação dos conceitos necessários para uma melhor compreensão da área; um subcapítulo sobre implementação e obtenção de resultados a partir das ferramentas utilizadas; uma quarta, com os resultados, análises e possíveis modificações, de modo a gerar um sistema que chega à um resultado com melhor valor, de acordo com as métricas escolhidas; e por fim um subcapítulo com conclusões, considerações finais e próximos passos que os interessados podem utilizar para aprofundar em suas pesquisas em EHW.

1.3. Hardware Evolutivo

Evolvable Hardware (EHW) ou Hardware Evolutivo é uma abordagem baseada em três áreas do conhecimento, Engenharia Eletrônica, Ciências da Computação e Biologia. Em computação e engenharia, EHW pode ser utilizado para uma gama de aplicações, como em comunicação de dados [**Grochol**, **2013**], áreas como processamento de imagens digi-

tais [**Vasicek**, **2013**] [**Almeida**, **2018**] [**Salvador**, **2013**], para a criação de circuitos adaptativos [], detecção de falhas [**Wang**, **2014**], computação aproximada [**Vasicek**, **2015**]

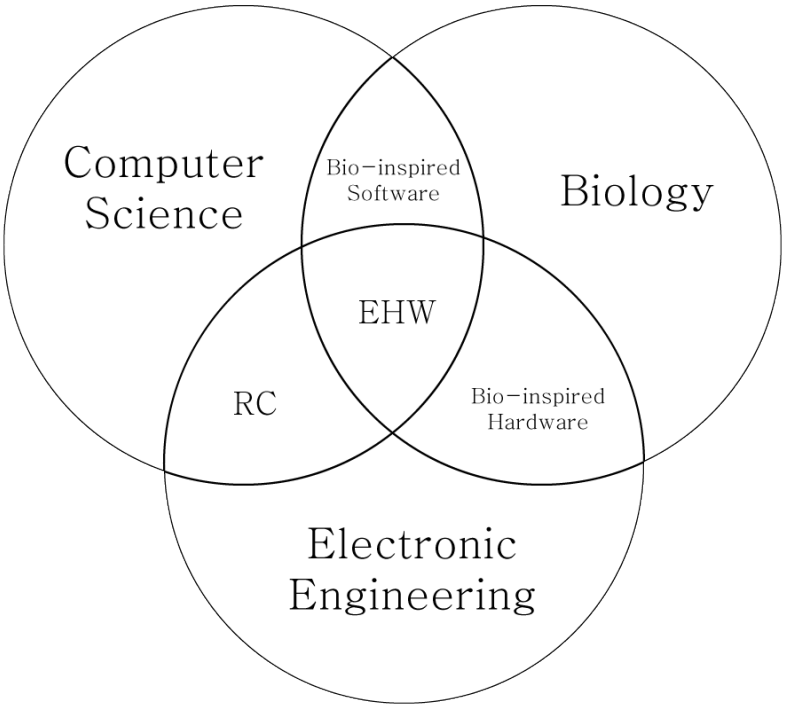

Figure 1.2. Diiagrama explicativo sobre as áreas que compõe Hardware Evolutivo.[Bentley, 2002]

1.4. Computação Reconfigurável

A Computação Reconfigurável (CR) é uma área idealizada na década de 1960, mas só foi fisicamente implementada na década de 1990. Um dos dispositivos digitais mais comuns na CR é o FPGA (Field Programmable Gate Array) [**Salvador**, **2016**]. Nas últimas décadas, esse campo tem crescido devido à demanda por sistemas computacionais de alto desempenho e flexibilidade, entre outros requisitos. Os FPGAs são tradicionalmente configurados por meio de linguagens alfanuméricas de descrição de hardware ou diagramas de circuito. As linguagens mais utilizadas são o Verilog e o VHDL (Very High-Speed Integrated Circuit).

Uma característica inerente à Hardware Evolutivo (EHW) é a reconfigurabilidade, uma vez que a evolução exige diversas alterações na topologia do hardware. O mecanismo responsável por reconfigurar o hardware torna-se um ponto crítico no EHW. Se o processo de modificação da topologia do hardware levar muito tempo, consequentemente, o tempo necessário para evoluir uma solução será prolongado.

No contexto de FPGAs e EHW, existem duas técnicas principais de reconfigu-

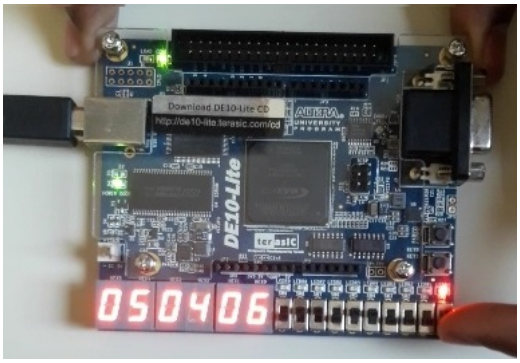

Figure 1.3. FPGA utilizado para testes do EHW. FONTE: Intel, 2019.

ração dinâmica implementadas: Circuito de Reconfiguração Virtual (VRC) e Reconfiguração Dinâmica e Parcial (DPR). Na literatura, é dito que existem compensações no uso de ambas as técnicas. O VRC basicamente utiliza o paralelismo espacial controlado por multiplexadores para criar vários cenários desejados, e os bits de seleção dos multiplexadores reconfiguram o hardware. O DPR consiste em reconfigurar apenas parte do FPGA enquanto o restante está em operação, de forma dinâmica, por meio da alteração do bitstream do dispositivo. O DPR é um recurso que apenas alguns dispositivos FPGA possuem, geralmente a um alto custo financeiro. [**Yao**, **2018**]

1.4.1. VRC

O cerne da solução pretendida foi batizado de *Evolvable Nucleic Architecture* (ENA), com base no conceito de Circuito de Reconfiguração Virtual (VRC) [**Srivastava**, **2014**]. Este módulo é chamado de "ENA" porque é a base, o alicerce (o núcleo) da Arquitetura Evolutiva. Foi projetado com três camadas, ou seja, três circuitos interconectados, compostos por portas AND, OR e NOT, dispostas em uma estrutura semelhante ao CGP (*Carthesian Genetic Programming*). As entradas estão conectadas à primeira camada, que está ligada à seguinte e, em seguida, à última camada. O genótipo é um vetor de bits que configura a topologia do circuito e representa um ponto no espaço de busca da solução. Cada alelo no genótipo configura um multiplexador que seleciona um operando ou operação em uma camada.

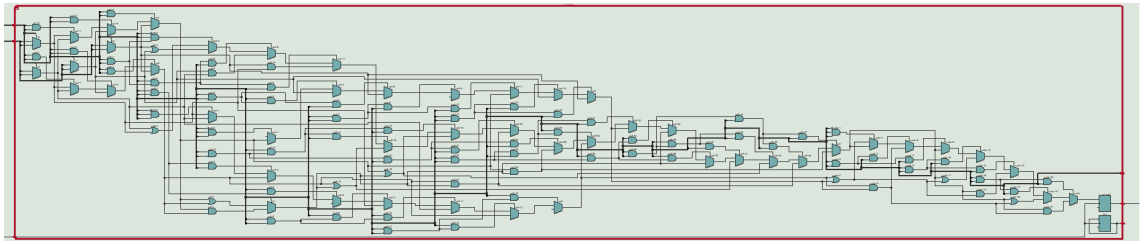

Figure 1.4. Modelo de exemplo de ENA elaborado para genótipos com 22 bits. Cunha, 2020.

Como estudo de caso, neste capítulo é apresentado um genótipo de nove bits. Quando o genótipo sofre uma variação, mesmo em um período muito curto (na ordem dos nanossegundos), um novo circuito é montado. Deve ser destacado que dois genótipos

diferentes podem resultar em um circuito com a mesma funcionalidade (cada linha de sua tabela verdade corresponde à mesma tabela de referência), mas a topologia do circuito pode diferir no número e na disposição das portas lógicas. No entanto, genótipos iguais têm a mesma topologia de circuito, mas podem diferir no valor da avaliação da aptidão, porque a tabela de referência (indicadora da solução desejada) pode ser diferente em ambas as situações.

O modelo utilizado implementa em hardware a ENA baseada em camadas para circuitos de duas entradas e uma saída. Esta foi projetada para ser modular, escalável e explorar o espaço de busca o máximo possível. Embora seja de fato um circuito combinacional, para estar sincronizado com toda a arquitetura, a instância VHDL teve que incluir uma entrada de *clock* para sincronizar a organização do circuito com as outras etapas do AE.

O módulo ENA for projetado especificamente para o objetivo do problema. Dependendo da complexidade do problema ou de sua área de origem, a estrutura da ENA pode variar. No entanto, independentemente da natureza do problema combinacional, é impossível gerar uma solução que não possa ser implementada, uma vez que os blocos de construção básicos são apenas portas lógicas. No entanto, existem soluções que devem ser evitadas: aquelas que fazem uso excessivo de portas lógicas, o que é feito pelo Motor Evolutivo.

Este minicurso escolheu a abordagem através de VRC, uma vez que é mais universal do que o DPR e porque a reconfiguração é mais rápida (para o requisito de desempenho).

1.5. Computação Evolucionária

Computação Evolucionária (CE) é uma área da Ciência da Computação que utiliza conceitos e abstrações inspirados na natureza para desenvolver estratégias otimizadas na exploração de um espaço de busca em um domínio específico de problemas. Esse espaço de busca consiste em um conjunto de soluções possíveis para um problema dado, sendo que cada solução é representada por um indivíduo, que é expresso como um cromossomo. Esse indivíduo possui um fenótipo, que funciona como um código para a interação do cromossomo com o ambiente, ou seja, ele se adapta de acordo com o meio em que está inserido. Neste trabalho, considera-se que cromossomo e genótipo são termos intercambiáveis. O indivíduo é composto por uma combinação do genótipo e do fenótipo.

O genótipo pode ser dividido em genes e alelos, sendo que os alelos são as menores divisões do genótipo, representando um único bit. Um gene é um grupo de alelos que desempenham uma função específica no genótipo. Na estrutura de VRC abordada pelo minicurso, existem 9 alelos e 3 genes.

A avaliação do indivíduo é realizada de acordo com uma função de aptidão (*fitness*), que quantifica o quão adequado o indivíduo é para resolver o problema em questão. A função de aptidão pode ter um único critério, mas geralmente é aplicada em cenários com dois ou mais critérios (otimização multicritério). No entanto, este minicurso não aborda algoritmos de otimização multicritério com mais de 2 objetivos.

Um conjunto de genótipos analisados em um determinado período de tempo é

chamado de população. Cada ciclo de operações do algoritmo é denominado geração. À medida que o tempo avança, a tendência do algoritmo é encontrar indivíduos mais aptos, cujo valor de aptidão é maior do que os das gerações anteriores, até encontrar a solução ótima global (ou ótimas).

Algorithm 1: Algoritmo Evolutivo Genérico

Entrada: Tamanho da população N, Número de gerações G, Taxa de mutação p_m, Taxa de cruzamento p_c
Saida : Solução ótima

1 Inicialize uma população aleatória com tamanho N;
2 Avalie a aptidão de cada indivíduo na população;
3 $geracao \leftarrow 1$ Ate G Selecione pais para cruzamento;
4 Crie uma nova população por meio do cruzamento;
5 Aplique mutações na nova população com probabilidade p_m;
6 Avalie a aptidão de cada indivíduo na nova população;
7 Selecione os melhores indivíduos para a próxima geração;
8 Selecione o melhor indivíduo da população final como a solução ótima;

Como o algoritmo foi implementado em hardware, algumas alterações foram realizadas, principalmente buscando uma alta performance em desempenho do hardware.

1.5.1. Exploração de espaço de projeto

A Exploração do Espaço de Projeto (EEP), de acordo com Kang, Jackson e Shulte (2011), é definida como "a atividade de descobrir e avaliar alternativas de design durante o desenvolvimento de sistemas". É usada para prototipagem rápida, otimização ou integração de sistemas com outros. Um dos principais objetivos deste minicurso é entregar uma arquitetura de hardware evolutivo de alto desempenho e fornecer ferramentas para abordar circuitos combinacionais mais complexos. A EEP é um meio potencial para alcançar as características desejadas. [**Kang**, **2011**]

O objetivo deste minicurso em relação à EEP é obter uma compreensão mais profunda da "caixa-preta", descobrindo quais parâmetros e níveis são mais adequados para um melhor desempenho esperado. O uso da EEP pode ser eficaz quando o sistema desejado possui diversas configurações e possibilidades. Este é o caso do projeto do VRC, que é uma estrutura combinacional dependente de um conjunto de parâmetros em três níveis diferentes.

O principal problema ao especificar uma Arquitetura de Hardware Evolutivo é que o design como um todo possui um grande número de graus de liberdade (Torresen, 2004), representado por um espaço multidimensional. Mesmo quando reduzimos o escopo ou limitamos os parâmetros (como a tecnologia alvo ou qual AE será utilizada), ainda existem vastas possibilidades a serem consideradas no design.

1.5.2. Motor Evolucionário

O Motor Evolutivo (ME), aqui definido como uma arquitetura que implementa as Operações Evolucionárias e a Avaliação da Aptidão em hardware, é, de fato, um módulo

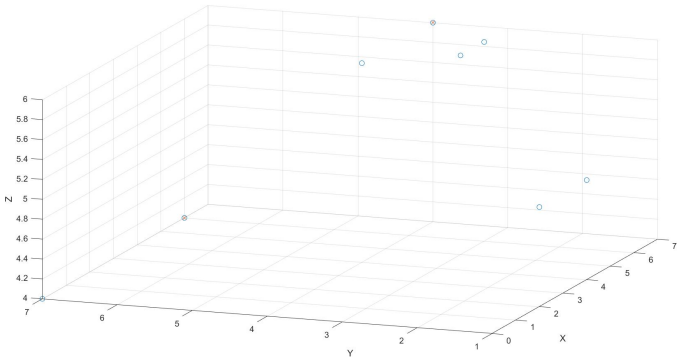

Figure 1.5. Visualização 3D dos genótipos ótimos (círculos azuis preenchidos de vermelho) e sub-ótimos (apenas círculos com bordas azuis) para o problema da porta XOR. Cunha, 2020.

computacional que otimiza o circuito desejado e encontra uma solução que corresponde totalmente à solução do problema, ao mesmo tempo que minimiza o número de portas lógicas necessárias para essa solução. A principal razão para o desenvolvimento de um Motor Evolutivo em vez de um Algoritmo Evolutivo tradicional (como Algoritmos Genéticos ou Programação Genética Cartesiana)[**Babu**, **2016**] é que o Motor Evolutivo é uma implementação mais adequada de Algoritmos Evolutivos para Hardware Evolutivo (EHW). Cancare et al. [**Cancare**, **2011**] menciona que abordagens canônicas genéricas devem ser evitadas nas implementações de hardware de AE.

O ME projetado tem como objetivo oferecer alto desempenho à solução de EHW, pois cada etapa e operação foram planejadas para serem executadas em hardware. Como não há software envolvido na evolução, não há atraso de comunicação entre um processador e o hardware em evolução. A EHW proposta é composta por duas partes principais: o Motor Evolutivo e o Hardware Operacional, que recebe a configuração da solução evoluída criada pelo ME. Na maioria dos trabalhos relacionados a EHW, algumas etapas do ME (ou todas) são implementadas em software, enquanto o Hardware Operacional é implementado em hardware. No entanto, essa abordagem costuma ser o gargalo da solução de EHW.

As operações envolvendo a Arquitetura Evolutiva apresentada neste capítulo incluem a Geração do Genótipo, Avaliação da Aptidão, Mutação, Cruzamento e Seleção. No entanto, cada uma dessas operações envolve diversos aspectos relacionados ao desenvolvimento de hardware, podendo estar relacionada a um ou mais parâmetros ou níveis. A Geração do Genótipo será feita

O AE implementado difere dos algoritmos comuns na literatura, e uma das razões para isso é que todo o processo ocorre simultaneamente, em paralelo. Por exemplo, enquanto o Gerador de Genótipos está fornecendo novos genótipos para a população inicial, eles estão sendo avaliados e outras operações estão sendo aplicadas a eles.

1.5.2.1. Geração de Genótipos

Como os genótipos são números que mapeiam o problema no espaço de solução, o algoritmo evolucionário precisa de um conjunto de indivíduos para criar uma população, e assim selecionar os indivíduos mais adaptáveis à situação-problema para uma próxima geração.

Para isso, é necessário gerar esses números, que aqui são representados por um conjunto de bits. Para tal, a arquitetura do Hardware Evolutivo conta com circuitos geradores de números pseudo-aleatórios (PRNGs), de 9 bits, chamados de LFSRs (Linear-Feedback Shift Registers).

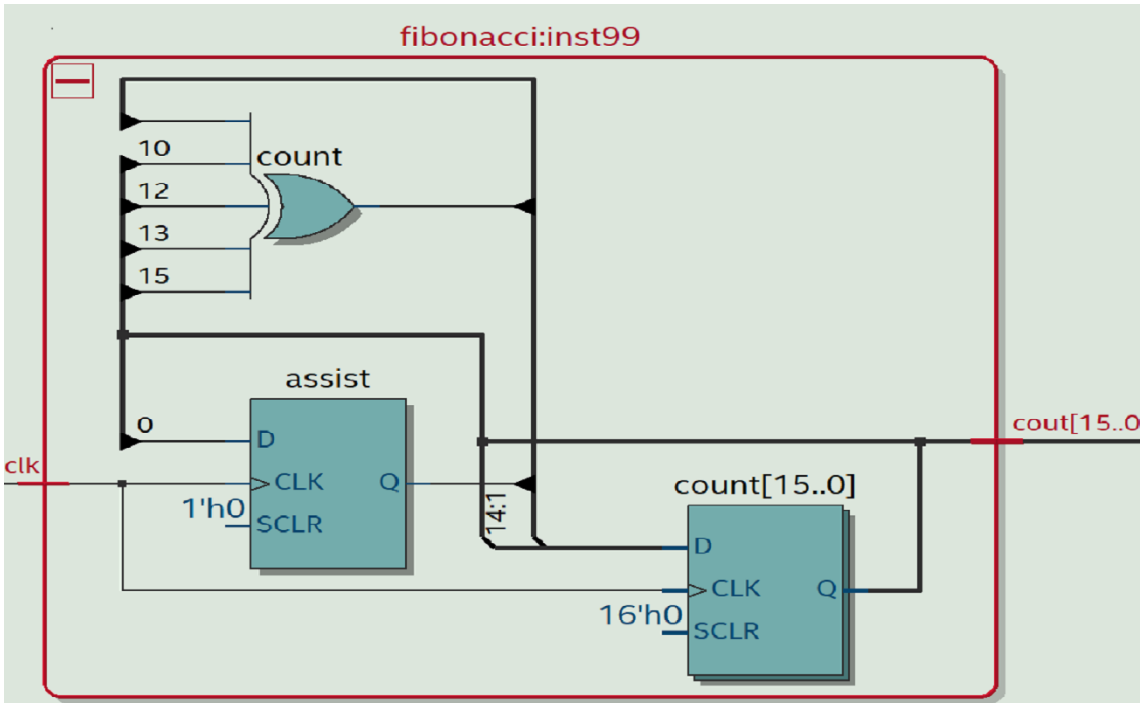

Figure 1.6. Imagem da extração do RTL de um LFSR Fibonacci presente no módulo gerador de PRNGs. Fibonacci é um tipo de circuito com uma retroalimentação (*feedback*) específica. Cunha, 2020

1.5.2.2. Função de Avaliação

A Avaliação de Aptidão (AA) é conhecida por ser a operação mais demorada na implementação de hardware de um AE [**Yao**, **1999**]. No entanto, neste caso, leva apenas um ciclo para avaliar uma solução candidata. Vamos considerar sistemas de dois entradas e uma saída, que podem ser descritos com quatro linhas de uma tabela verdade. Nesse caso, são gerados quatro pares de sinais digitais (chamados de Entradas Virtuais) que estimulam quatro instâncias do módulo ENA, representando as quatro possibilidades de combinações de entradas, simultaneamente.

A saída de cada instância do ANE é verificada com o vetor de referência correspondente pelo módulo de AA. A comparação é feita com um arranjo de portas XNOR

(quatro no total, uma para cada uma das quatro possibilidades), detectando semelhanças.

O Fenótipo é o resultado obtido quando o genótipo configura o circuito de base do EHW e sofre excitação de entrada. É um vetor do mesmo tamanho que o de referência e nos diz qual saída é encontrada pelo circuito para cada combinação de entradas.

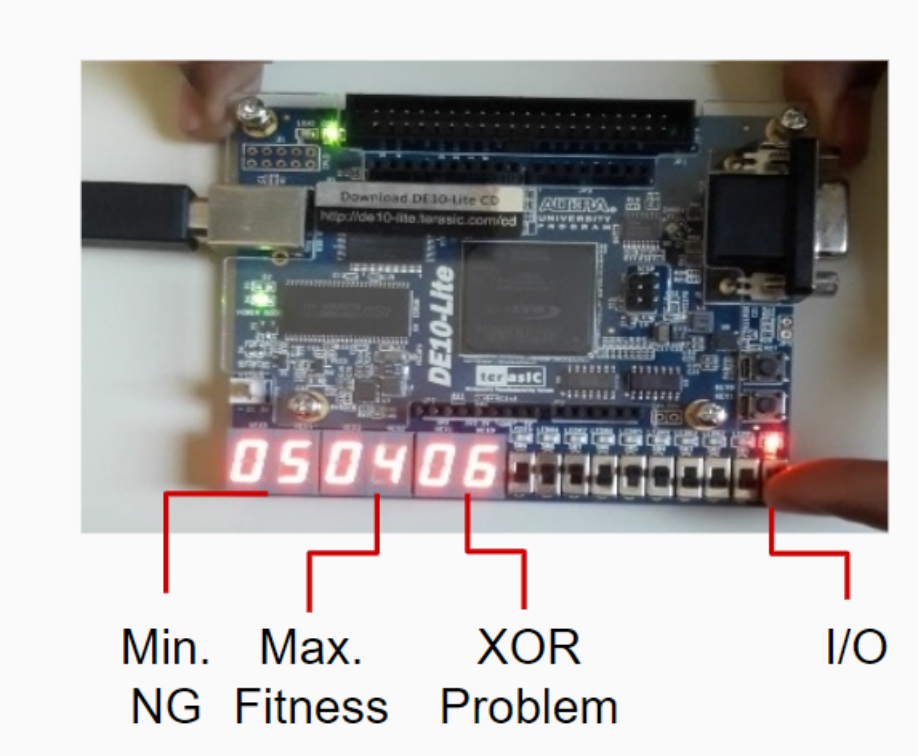

Figure 1.7. FPGA exportando nos displays e LED as saídas do EHW. Autores, 2023.

1.5.2.3. Mutação

A mutação é uma operação evolutiva que aumenta ligeiramente o poder de busca em um AE [**daSilva**, **2018**]. Consideramos esta operação aqui porque ela busca, nas proximidades de um genótipo, por outro que seja mais adequado do que o indivíduo atual. Genericamente, a operação de mutação é usada para evitar pontos de mínimo local. Como o contexto deste minicurso trata de circuitos digitais, a maior parte de seu projeto é representada por notações e cálculos binários. Para uma melhor compreensão de "proximidade" ou "closeness", usamos o seguinte exemplo: dois números podem ser completamente próximos em uma representação inteira e, em binário, ser extremamente distantes (como exemplo, 511 e 512, 0111111111 e 1000000000, respectivamente). É extremamente difícil gerar uma solução candidata 512 a partir de uma operação de mutação sobre 511, ou vice-versa.

A mutação consiste em "flippar" um alelo (bit) do genótipo, ou seja, alterar seu estado de ’1’ para ’0’ ou vice-versa. A ideia de proximidade tem a ver com o conjunto de genótipos que possuem semelhança em seus alelos, por exemplo, 8 em 9 bits são iguais.

1.5.2.4. Cruzamento

Cruzamento é outra operação evolutiva, e esta considera geralmente dois indivíduos que "trocam" entre si um conjunto de bits, gerando indivíduos novos compostos por bits de gerações anteriores.

Nas experiências realizadas em hardware, o cruzamento é aplicado a dois indivíduos de origens diferentes, com uma posição de corte aleatória, e exporta o melhor resultado, e a operação de cruzamento os combina em busca de candidatos a soluções ainda melhores.

1.5.2.5. Seleção e Elitismo

Essas duas operações estão basicamente conectadas nesse projeto. A Seleção é vista neste trabalho como um recurso onde uma solução candidata é escolhida em um grupo de várias outras, de acordo com uma função de avaliação de aptidão. E o Elitismo é um mecanismo que garante que as próximas gerações não terão um genótipo com uma avaliação pior. Elas são implementadas em uma conexão em cascata: primeiro, a Seleção escolhe o melhor resultado dos módulos anteriores, e então o Elitismo só registra a solução candidata se for melhor que o melhor indivíduo atual, de acordo com o código abaixo. Note que a linha 11 implica que a minimização dos portões lógicos só começa quando o requisito funcional é atendido, ou seja, um valor de aptidão igual a quatro ("100" em notação binária).

1.6. Hardware Evolutivo como um *Game*

Fazendo uma analogia com um recurso digital que é de interesse de várias faixas etárias, os jogos digitais, podemos analisar alguns parâmetros de um algoritmo evolucionário como um elemento de um jogo: a função de avaliação ou de *fitness* é como se fosse um *score* ou pontuação que cada jogador (ou genótipo possui). Quando avaliamos um "jogador", comparamos com o histórico de "jogadores" e, se sua avaliação ou score for a mais alta, selecionamos este jogador e dizemos que seu *score* foi a maior pontuação (*high score* ou melhor global).

Os outros genótipos gerados em uma geração são como NPCs (Non Playable Character) que competem com o "personagem principal", genótipo. Se algum dos NPCs tem *score* maior que o jogador, para ele em si, é *game over*. Na próxima geração, teremos o indivíduo que possui o *high score* das últimas gerações, e podemos alterar apenas algumas *features* desse jogador para serem os NPCs dessa geração. Podemos combinar *features* entre os melhores jogadores da rodada passada (cruzamento), modificar um pouquinho uma *feature* do melhor global, podendo resultar em um indivíduo ainda mais forte, ou não. Tudo depende de como a divisão dos bits for feita, sob quais bits dos genótipos originais.

Pensando nessa analogia, a abordagem realizada no minicurso é gamificada porque houve a divisão em equipes, que utilizaram o VRC como uma "caixa preta", e cada equipe tentava descobrir um genótipo que possui nota de avaliação melhor, cada rodada melhor que a da rodada anterior, e melhor que a outra equipe. Os números/genótipos eram argumento de uma função em MATLAB que devolvia a equação booleana, as saídas do

circuito configurado pelo genótipo, o fitness e o número de portas lógicas.

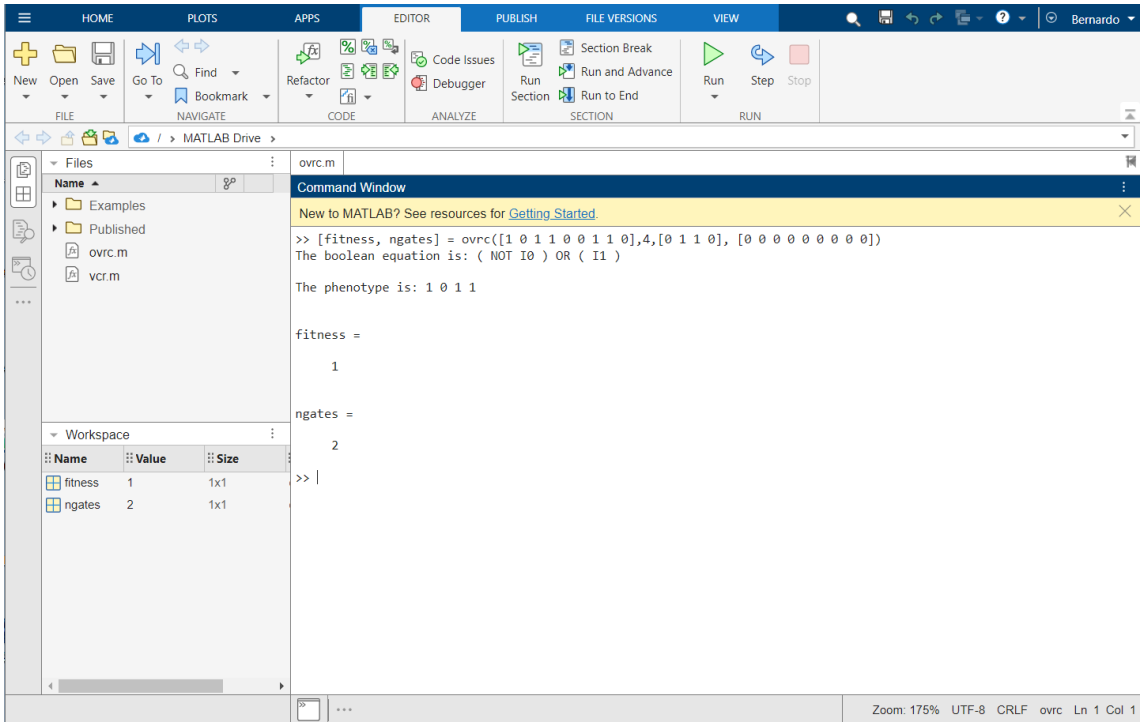

Figure 1.8. Tela de interface do MATLAB online com o console rodando a função OVRC, que devolve a avaliação do genótipo) para o problema da porta XOR. Autores, 2023.

O link para o código para verificação do genótipo está em [**github**, **2023**]

References

[1] P. Bentley, J and T. Gordon, G.W, "On Evolvable Hardware," in Soft Computing in Industrial Electronics, S. Ovaska and L. Sztandera, Eds. Heidelberg,Germany: PhysicaVerlag, 2002.

[2] F. Cancare, S. Bhandari, D. B. Bartolini, M. Carminati and M. D. Santambrogio, ”A bird’s eye view of FPGA-based Evolvable Hardware,” 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), San Diego, CA, 2011, pp. 169-175. doi: 10.1109/AHS.2011.5963932

[3] R.Salvador. ”Evolvable Hardware in FPGAs: Embedded tutorial”. 11th International Conference on Design Technology of Integration on Nanoscale Era (DTIS), 2016.

[4] Z.Vasicek, M.Bidlo, L.Sekanina, ”Evolution of efficient real-time non-linear image filters for FPGAs”. Soft Comput (2013) 17: 2163.https://doi.org/10.1007/s00500-013-1040-8

[5] R. Salvador, A. Otero, J. Mora, E. De La Torre, T. Riesgo, L. Sekanina, "Self-Reconfigurable Evolvable Hardware System for Adaptive Image Processing", IEEE Transactions on Computers, vol. 62, no. 8, pp. 1481-1493, Aug. 2013

[6] Yao, X., Higuchi, T. (1999). Promises and challenges of evolvable hardware. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 29(1), 87-97.

[7] Huang, X., Wu, N., Zhang, X., Liu, Y. (2015). An evolutionary algorithm based on novel hybrid repair strategy for combinational logic circuits. IEICE ELECTRONICS EXPRESS, 12(22). https://doi.org/10.1587/elex.12.20150765

[8] Dobai, R., Sekanina, L. (2015). Low-Level Flexible Architecture with Hybrid Reconfiguration for Evolvable Hardware. ACM TRANSACTIONS ON RECONFIGURABLE TECHNOLOGY AND SYSTEMS, 8(3). https://doi.org/10.1145/2700414

[9] M.A. Almeida; E.C. Pedrino. "Hybrid Evolvable Hardware for automatic generation of image filters". INTEGRATED COMPUTER-AIDED ENGINEERING, 2018, V.25, no.3, pp. 289-303

[10] R. Yao, P. Zhu, J.J. Du, M.Q. Wang, Z.H. Zhou. "A General Low-Cost Fast Hybrid Reconfiguration Architecture for FPGA-Based Self-Adaptive System". IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS, 2018

[11] Vasicek, Z., Sekanina, L. (2015). Evolutionary Approach to Approximate Digital Circuits Design. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 19(3), 432–444. https://doi.org/10.1109/TEVC.2014.2336175

[12] Sekanina, L., Vasicek, Z. (2013). Approximate Circuit Design by Means of Evolvable Hardware. In PROCEEDINGS OF THE 2013 IEEE INTERNATIONAL CONFERENCE ON EVOLVABLE SYSTEMS (ICES) (pp. 21–28). 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE.

[13] Babu, K. S., Balaji, N. (2016). Approximation of Digital Circuits Using Cartesian Genetic Programming. In PROCEEDINGS OF THE 2016 INTERNATIONAL CONFERENCE ON COMMUNICATION AND ELECTRONICS SYSTEMS (ICCES) (pp. 381–386). 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE.

[14] Kazarlis, S., Kalomiros, J., Kalaitzis, V., Balouktsis, A., Bogas, D. (2015). Intrinsic Evolution of Digital Circuits Based on a Reconfigurable Hyper-Structure. In E. Haase, J and Kakarountas, A and Grana, M and Fraile-Ardanuy, J and Debono, CJ and Quintian, H and Corchado (Ed.), IEEE EUROCON 2015 - INTERNATIONAL CONFERENCE ON COMPUTER AS A TOOL (EUROCON) (pp. 340–345). 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE.

[15] da Silva, J. E. H., Manfrini, F. A. L., Bernardino, H. S., Barbosa, H. J. C. (2018). Biased Mutation and Tournament Selection Approaches for Designing Combinational Logic Circuits via Cartesian Genetic Programming. In Anais do XV Encontro Nacional de Inteligencia Artificial e Computacional (pp. 835–846). Porto Alegre, RS, Brasil: SBC. https://doi.org/10.5753/eniac.2018.4471

[16] Stepney, S., Adamatzky, A. (2018). Inspired by Nature: Essays Presented to Julian F. Miller on the Occasion of his 60th Birthday (Vol. 28). https://doi.org/10.1007/978-3-319-67997-6

[17] Grochol, D., Sekanina, L., Zadnik, M., Korenek, J., Kosar, V. (2016). Evolutionary circuit design for fast FPGA-based classification of network application protocols. APPLIED SOFT COMPUTING, 38, 933–941. https://doi.org/10.1016/j.asoc.2015.09.046

[18] Wang, J., Liu, J. (2017). Fault-Tolerant Strategy for Real-Time System Based on Evolvable Hardware. JOURNAL OF CIRCUITS SYSTEMS AND COMPUTERS, 26(7). https://doi.org/10.1142/S0218126617501110

[19] Cunha, B.G.P., Ferreira, F.M.F., MARTINS, C. A. P. S. "A complete evolvable hardware architecture based on virtual reconfiguration: evolving combinational circuits". 2020. Dissertação (Mestrado em Engenharia Elétrica) - Pontifícia Universidade Católica de Minas Gerais.

[20] Srivastava, A. K., Gupta, A., Chaturvedi, S., Rastogi, V. (2014). Design and Simulation of Virtual Reconfigurable Circuit for a Fault Tolerant System. In 2014 RECENT ADVANCES AND INNOVATIONS IN ENGINEERING (ICRAIE). 345 E 47TH ST, NEW YORK, NY 10017 USA: IEEE.

[21] Mahdavi, S. J. S., Mohammadi, K. (2010). Reliability enhancement of digital combinational circuits based on evolutionary approach. MICROELECTRONICS RELIABILITY, 50(3), 415–423. https://doi.org/10.1016/j.microrel.2009.11.016

[22] Swarnalatha, A., Shanthi, A. P. (2014). Complete hardware evolution based SoPC for evolvable hardware. APPLIED SOFT COMPUTING, 18, 314–322. https://doi.org/10.1016/j.asoc.2013.12.014

[23] Liang, H., Luo, W., Li, Z., Wang, X. (2010). Designing Combinational Circuits with an Evolutionary Algorithm Based on the Repair Technique. In Tempesti, G and Tyrrell, AM and Miller, JF (Ed.), EVOLVABLE SYSTEMS: FROM BIOLOGY TO HARDWARE (Vol. 6274, pp. 193–201). HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY: SPRINGER-VERLAG BERLIN.

[24] KANG, E.; JACKSON, E.; SCHULTE, W. An Approach for Effective Design Space Exploration. In: Calinescu, R and Jackson, E. (Ed.). FOUNDATIONS OF COMPUTER SOFTWARE: MODELING, DEVELOPMENT, AND VERIFICATION OF ADAPTIVE SYSTEMS. HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY: SPRINGER- VERLAG BERLIN, 2011. (Lecture Notes in Computer Science, v. 6662), p. 33–54. ISBN 978-3-642-21292-5. ISSN 0302-9743

[25] TORRESEN, J. An evolvable hardware tutorial. In: Becker, J and Platzner, M and Vernalde, S. (Ed.). FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS, PROCEEDINGS. HEIDELBERGER PLATZ 3, D-14197 BERLIN, GERMANY: SPRINGER-VERLAG BERLIN, 2004. (Lecture Notes in Computer Science, v. 3203), p. 821–830. ISBN 3-540-22989-2. ISSN 0302-9743

[26] https://github.com/bguerrapc/EHW/blob/master/MATLAB/FUNCTIONS/ovrc.m