Ferramenta de edição e configuração de código VHDL

Nuno Leal*, Filipe Moutinho*†, Anikó Costa*† and Luís Gomes*†
*NOVA School of Science and Technology, Monte de Caparica, Portugal
†UNINOVA, Centre of Technology and Systems, Monte de Caparica, Portugal
Emails: na.leal@campus.fct.unl.pt, fcm@fct.unl.pt, akc@fct.unl.pt, lugo@fct.unl.pt

***Resumo*—Este artigo tem como objectivo apresentar uma ferramenta que se encontra em desenvolvimento. A ferramenta em questão tem como objectivo facilitar a edição e configuração de código VHDL (VHSIC (*Very High Speed Integrated Circuits*) *Hardware Description Language*). Esta ferramenta oferece um conjunto de módulos descritos em VHDL em que o utilizador apenas terá de seleccionar o módulo que deseja e preencher um conjunto de campos auto-gerados que irão modificar o módulo escolhido. O público-alvo desta ferramenta em desenvolvimento são os estudantes, entusiastas e técnicos que não dominem linguagens de descrição de *hardware*.**

I. Introdução

Hoje em dia a maioria dos sistemas embutidos contém um ou vários micro-controladores, *Field Programmable Gate Arrays* (FPGA), sensores, entre outros. A complexidade dos sistemas tem aumentado e o que era antigamente maioritariamente analógico, hoje é maioritariamente digital. Para além disso, a pressão do mercado e competitividade obriga a que os produtos sejam tecnicamente superiores e que o ritmo de desenvolvimento dos produtos esteja em constante aceleração. De modo a satisfazer estas necessidades são criadas camadas de abstracção, satisfazendo assim os engenheiros e técnicos na necessidade de uma maior eficiência, como por exemplo as linguagens de descrição de *hardware* [1], que permitem maior nível de abstracção e portabilidade em comparação com os esquemáticos.

As linguagens de descrição de *hardware*, em comparação com linguagens de programação, são difíceis e com uma curva de aprendizagem longa. Estas são também um desafio para quem as utiliza esporadicamente, seja para realizar um projecto no âmbito pessoal ou profissional. São prova disso alguns artigos, como [2], que ilustram que após a introdução de algumas metodologias, de forma a melhorar a capacidade de aprendizagem e o gosto por linguagens de descrição de *hardware*, a percentagem de alunos que considera o VHDL complicado continua alta. Ao longo dos anos tem havido um crescente interesse na forma como se deve ensinar as linguagens de descrição de *hardware* e quais as ferramentas a ser desenvolvidas e aplicadas no ensino. A comunidade tem-se debruçado sobre quais são os melhores métodos e *softwares*, uma vez que é aceite pelo geral da comunidade que para além da dificuldade inerente a estas linguagens, os ambientes de desenvolvimento mais usados são complexos. Isto é visto como um entrave na configuração e desenvolvimento de circuitos, por exemplo para FPGA, para indivíduos que não dominem esta área, tal como engenheiros informáticos, mecânicos ou estudantes que tenham o seu primeiro contacto com FPGA e linguagens de descrição de *hardware* [3][4][5][6]. Também é importante referir que nas linguagens de descrição de *hardware*, ao contrário de linguagens como o C ou C++, é necessário ter em consideração o circuito digital, uma vez que o que se descreve em VHDL ou Verilog irá reproduzir-se num circuito que, sem o devido cuidado será um circuito defeituoso e pouco satisfatório. A adicionar a este problema surge também o facto de linguagens como o VHDL, dadas as suas regras e palavras-chave reservadas, terem uma sintaxe extensa e por vezes complicada, levando a más práticas e um tempo de pesquisa, em livros ou recursos *online*, elevado.

Face a estes impasses, e à semelhança de [6] o objectivo deste trabalho é desenvolver uma ferramenta que facilite a aprendizagem de VHDL. No entanto, enquanto que em [6] foi proposta uma ferramenta que permite compilar e executar o código VHDL no PC (*Personal Computer*), neste trabalho o objectivo é desenvolver uma ferramenta que simplifica a descrição de *hardware* em VHDL. A finalidade é tornar a edição de VHDL mais simples e intuitiva, de modo a que alunos, técnicos e engenheiros que não dominem linguagens de descrição de *hardware* possam desenvolver os seus projectos de forma mais simples e rápida. Para tal, a ferramenta irá disponibilizar modelos de código e irá permitir a sua configuração. Desta forma pretende-se que o utilizador consiga mais facilmente obter exemplos de código, e que os consiga configurar. Assim, o número de erros de sintaxe e de más práticas de descrição de *hardware* tenderá a diminuir.

Este documento é constituído por quatro secções. Na próxima secção são apresentados trabalhos relacionados, dificuldades na aprendizagem de VHDL e ferramentas comerciais. Na terceira secção é apresentada a ferramenta que está a ser desenvolvida. Por fim, a última secção é relativa às conclusões.

II. Trabalhos Relacionados

O VHDL, em geral, é uma linguagem de descrição de *hardware* considerada bastante difícil de aprender. A curva de aprendizagem é mais longa em comparação com outras linguagens, como o C++ ou Java. Em [2] leccionou-se VHDL e no final realizou-se um inquérito aos alunos, tendo 83% considerado o VHDL difícil de aprender, contrastando ao mesmo tempo com 70% dos alunos a referirem que gostaram de escrever código em VHDL. Para a realização deste inquérito foram realizadas dez aulas laboratoriais e um trabalho final. As aulas laboratoriais abordaram temas base como:

- Implementação de circuitos combinatórios;

- Aplicação de equações Booleanas, mintermos, max-termos e mapas de *Karnaugh*;
- Uso de *multiplexers* e *demultiplexers*, implementação de circuitos aritméticos de números de 8-*bits*, e implementação de *shift registers* de 16-*bits*;
- Implementação de máquinas de estado.

Para o trabalho final, era dado aos alunos a opção de escolha de um projecto entre quatro. Todos eles com aplicação directa de circuitos digitais.

Na realidade, apesar da dificuldade de aprendizagem do VHDL, é possível enumerar algumas razões que levam a uma curva de aprendizagem mais longa. Antes de tudo, é necessário ter em conta que as ferramentas e plataformas usadas nesta área são só por si complexas, pelo que é necessário um tempo significativo até que o utilizador as domine. Alguns exemplos são os ambientes de desenvolvimento disponibilizados pela *Xilinx*, *Altera*, e placas de circuitos integrados como as FPGA e componentes como conversores analógico-digitais e memórias DDR (*Double Data Rate*). Para além disso, é da opinião de vários autores que os livros sobre VHDL focam-se em demasia na sintaxe e formalismo da linguagem, secundarizando a relação entre os circuitos digitais e o VHDL [7]. Esta secundarização causa deficiências de aprendizagem, no sentido em que se torna difícil mapear e relacionar a que circuito e implementação de *hardware* corresponde cada excerto de código escrito em VHDL. Este nível de abstracção é preocupante para pessoas pouco experientes, uma vez que apesar da sintaxe ser bastante semelhante às linguagens de programação, o mesmo não se pode dizer em relação ao comportamento do sistema implementado [3].

A. Métodos de Aprendizagem Tradicionais

Os métodos tradicionais são bem conhecidos por toda a comunidade e também por pessoas que não sejam da área, mas que estejam familiarizadas com ela. Estes métodos assentam no uso de um ambiente de desenvolvimento que permita a síntese de VHDL e a sua implementação numa placa de circuito integrado, como uma FPGA. Com estas ferramentas e depois de algumas palestras relativas ao VHDL e circuitos digitais, os alunos são convidados a especificar os mais variados problemas, dependendo dos programas de ensino de cada instituição [5]. Segundo [8], alguns passos tradicionalmente usados para a implementação em VHDL, recorrendo a uma plataforma com dispositivos reconfiguráveis (FPGA), são:

- Definir requisitos e especificações;
- Desenvolver um modelo do sistema que atinja as especificações pretendidas;
- Especificar o modelo em VHDL;
- Simular o modelo especificado em VHDL de modo a determinar se se comporta como especificado inicialmente;
- Sintetizar, implementar e configurar a placa para protótipo com o circuito integrado reconfigurável;
- Verificar através de testes se o protótipo se comporta como desejado.

B. Métodos de Aprendizagem Alternativos

Os autores de [6] referem que a utilização de métodos visuais, como animações, facilita a aprendizagem face aos métodos tradicionais. O estudo do *hardware* envolve um grande número de conceitos abstractos que tipicamente são difíceis de visualizar para os menos experientes. Exemplo disso é a propagação de sinais digitais dentro de arquitecturas computacionais ou outras unidades funcionais. Estes autores criticam ainda o facto de se usarem ambientes de desenvolvimento comerciais que são demasiado complexos e desnecessários num ambiente que se quer introdutório e onde frequentemente apenas é necessário simular projectos simples em HDL, que muitas vezes contêm apenas um ficheiro. Para solucionar os problemas descritos, os autores de [6] sugerem algumas ferramentas como:

- Uma plataforma *Web* para visualização e animação de processos que consistem em portas lógicas. Inclui esquemas de circuitos combinatórios usando portas lógicas, *multiplexers*, *decoders* e máquinas de estado;
- GHDL, uma aplicação leve e *open-source* que permite que os utilizadores compilem e executem código VHDL diretamente usando apenas a linha de comandos do *Windows*, *Linux* ou *macOS*;
- GTKWave, para a visualização das formas de onda.

Outros autores [9], privilegiando uma atitude de desenvolvimento baseada em modelos, fazem referência a um ambiente de desenvolvimento *Web* chamado IOPT-Tools, disponível em http://gres.uninova.pt/IOPT-Tools. Composto por um conjunto de ferramentas, permite modelar sistemas de controlo digital através de redes de Petri. Algumas características interessantes das IOPT-Tools são:

- Edição gráfica interactiva de modelos de controladores. Neste caso houve o uso de tecnologias W3C (*World Wide Web Consortium*), como AJAX (*Synchronous Javascript and* XML (*Extensible Markup Language*)) e XLST (*Extensible Stylesheet Language Transformations*);
- Geração automática de código VHDL, C, etc., que permite, não só uma simplificação da implementação do controlador final ou da lógica inerente aos módulos, eliminando a necessidade de escrever código de baixo nível, mas também minimizar a complexidade e reduzir as dificuldades e a falta de conhecimento nesta área [10];
- Simulação e *model-checking* contribuem para a redução do tempo e custo de desenvolvimento, permitindo também a detecção de erros de modelação em fases precoces do projecto.

Já as IOPT-Flow, referidas pelos autores de [11], à semelhança das IOPT-Tools são também um ambiente de desenvolvimento baseado em ferramentas *Web*. Estas foram propostas para o desenvolvimento de sistemas embutidos e ciberfísicos, com maior ênfase no processamento de dados, quando comparadas com as IOPT-Tools. Como afirmado em [11], os modelos combinam redes de Petri com *dataflows*. Os autores sugerem que esta combinação é uma mais valia

no desenho de sistemas que ao mesmo tempo necessitem de máquinas de estado e de processar dados.

C. Ferramentas Comerciais

De modo breve importa também referir algumas das ferramentas comerciais mais conhecidas no mercado. As mais conhecidas são o *Vivado* e o *Quartus Prime*, que são também semelhantes. Ambas permitem a edição e simulação de projectos em HDL, bem como a sua síntese e implementação em dispositivos lógicos programáveis. Dado que são ferramentas comerciais e têm como público-alvo profissionais experientes, estas ferramentas são complexas e requerem algum treino [6].

A ferramenta *Simulink*, do *Matlab*, é um instrumento versátil para investigadores e engenheiros. Esta inclui um ambiente gráfico interactivo e um conjunto de bibliotecas disponíveis, que podem ser personalizadas, permitindo que o utilizador modele, simule e analise os sinais mais importantes para o seu projecto [12]. Modelos *Simulink*, que correspondem a circuitos digitais, podem ser convertidos para HDL, utilizando o HDL Coder.

O Active-HDL, da *Aldec*, é um ambiente de desenvolvimento, com uma interface gráfica intuitiva, totalmente integrado para o desenho de circuitos integrados, na qual os circuitos podem ser descritos em código C/C++ ou em HDL (VHDL e Verilog) [12]. O Active-HDL, à semelhança das IOPT-Tools e IOPT-Flow, permite a tradução de diagramas de estado em HDL. O Active-HDL permite também a tradução de código HDL em diagramas de blocos e vice-versa.

O Cadence Stratus HLS, que como o *datasheet* da ferramenta refere, permite que rapidamente se desenhe e se verifique implementações em RTL (*Register Transfer Level*) através de modelos descritos em C++, SystemC ou C, automatizando o desenho e verificação [13].

A Plataforma Xilinx PYNQ oferece uma interface de programação baseada em Python. Esta plataforma permite que os projectistas, através da linguagem de programação Python, explorem as capacidades de lógica reconfigurável das placas. O uso de Python, graças à sua natureza e às bibliotecas prontas para serem usadas, é uma vantagem face às HDL e outras formas menos abstractas de desenvolvimento de circuitos reconfiguráveis [14].

III. A Ferramenta em Desenvolvimento

A ferramenta em desenvolvimento está a ser implementada com recurso às tecnologias *Web* (JavaScript, HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) e JSON (*JavaScript Object Notation*)), permitindo uma interface gráfica simples, interactiva e intuitiva no navegador de *Internet* dos utilizadores. A arquitectura da ferramenta está descrita, esquematicamente, na figura 1.

A ferramenta disponibiliza um conjunto de módulos VHDL bastante comuns em electrónica digital, tendo sido escolhidos devido à sua baixa complexidade e inevitabilidade no desenho de novos sistemas por parte de utilizadores pouco experientes. Estes módulos são gravados em formato JSON na base de dados. O utilizador ao inicializar a página *Web* irá visualizar a lista de módulos disponíveis, como se pode ver do lado esquerdo da figura 2. Ao seleccionar um dos módulos, é feita

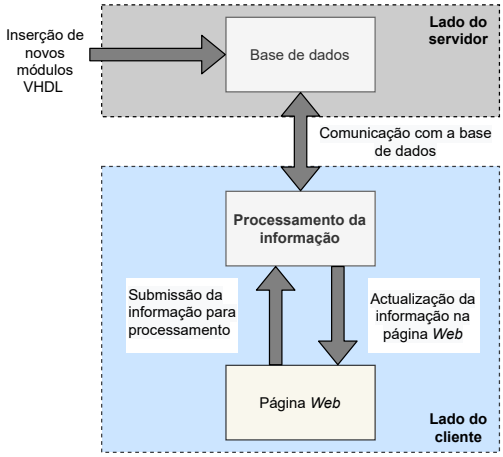

Fig. 1. Arquitectura da ferramenta em desenvolvimento.

a comunicação com o servidor e respectivo *download*, sendo carregado para uma área de texto, como se pode observar na parte central da figura 2.

O utilizador de seguida irá configurar o módulo através de um conjunto de campos auto-gerados, como o lado direito da figura 2 o ilustra. O valor introduzido/seleccionado nesses campos irá corresponder a um conjunto de comandos sinalizados pelos caracteres "<" e ">" que serão posteriormente substituídos aquando as ordens do utilizador.

Na zona central da figura 2 está presente o módulo correspondente a um contador síncrono cujas alterações desejadas pelo utilizador ainda não foram feitas, sendo possível reconhecer alguns comandos sinalizados pelos caracteres "<" e ">", como explicado anteriormente. Por exemplo, as *tags* <*NAME*> e <*OUTPUT*> visíveis no código VHDL irão ser substituídos pelos valores inseridos nos respectivos campos de entrada, visíveis no lado direito da figura 2. Importa realçar a *tag* <*EDGE_TYPE*> em que o utilizador terá de seleccionar uma das opções pré-definidas de uma lista (*rising_edge* ou *falling_edge*), em vez de escrever o nome desejado numa caixa de texto.

Adicionalmente, de entre mais algumas opções, é possível instanciar módulos. Para instanciar um módulo na ferramenta em desenvolvimento o utilizador terá de preencher os campos auto-gerados para o módulo seleccionado e clicar no botão que corresponde ao instanciar do módulo. O instanciar do módulo será automaticamente gerado, como se mostra na figura 3. De seguida o utilizador apenas terá de copiar e colar para o ficheiro VHDL do seu projecto.

IV. Conclusões

Este documento apresenta uma ferramenta em desenvolvimento que poderá ter um impacto significativo no ensino das linguagens de descrição de *hardware*. Características como a rápida configuração do código VHDL permite reduzir o tempo de pesquisa de exemplos de código e mitigar os erros de sintaxe, incentivando ainda boas práticas de descrição de *hardware* para FPGA.

Sendo que o público-alvo desta ferramenta será em grande parte estudantes, entusiastas e outros profissionais com pouca

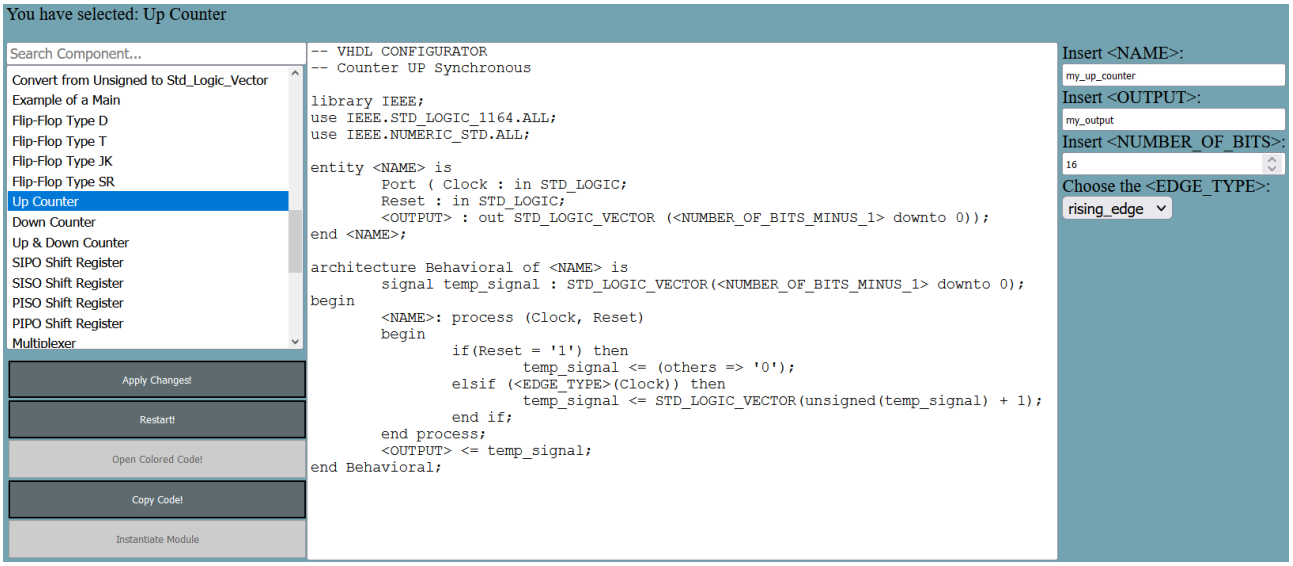

Fig. 2. Ferramenta em desenvolvimento com o módulo de contador síncrono seleccionado.

```
COMPONENT my_up_counter
Port ( Clock : in STD_LOGIC;
       Reset : in STD_LOGIC;
       my_output : out STD_LOGIC_VECTOR (15 downto 0));
END COMPONENT;

Inst_my_up_counter: my_up_counter PORT MAP(
       Clock => ,
       Reset => ,
       my_output => ,
);
```

Fig. 3. Módulo correspondente ao contador síncrono instanciado.

ou nenhuma experiência em linguagens de descrição de *hardware*, importa não descartar a sua potencial utilidade para profissionais nesta área. Esta ferramenta poderá ser vista como um complemento aos ambientes de desenvolvimento comerciais mencionados neste documento.

Como trabalhos futuros, pretende-se garantir a disponibilidade do ambiente para utilização aberta pela comunidade, assegurando-se a sua manutenção, através da sua expansão e correcção de erros, bem como da eventual extensão à linguagem Verilog e integração de novos módulos de maior complexidade.

AGRADECIMENTOS

Este trabalho foi parcialmente financiado por Fundos Nacionais através da Fundação para a Ciência e a Tecnologia (FCT) no âmbito do projecto UIDB/00066/2020.

REFERÊNCIAS

[1] Robert W. Nowlin e Rajeswari Sundararajan. A Vhdl Course For Electronics Engineering Technology. Em: *1997 Annual Conference*. 10.18260/1-2–6893. https://peer.asee.org/6893.Milwaukee, Wisconsin: ASEE Conferences, jun. de 1997.

[2] R. Rodriguez-Ponce e J. Rodriguez-Resendiz. Integrating VHDL into an undergraduate digital design course. Em: *Proceedings of 2013 IEEE International Conferenceon Teaching, Assessment and Learning for Engineering (TALE)*. 2013, pp. 172–177. doi:10.1109/TALE.2013.6654423.

[3] A. Wu. Interactive learning toolbox for logic synthesis with VHDL. Em: *Proceedings of International Conference on Microelectronic Systems Education*. 1997, pp. 77–78. doi:10.1109/MSE.1997.612555.

[4] A. Etxebarria, I. J. Oleagordia e M. Sanchez. An educational environment for VHDL hardware description language using the WWW and specific workbench. Em: 1 (2001), T2C–2. doi:10.1109/FIE.2001.963876.

[5] A. Madanayake et al. Teaching freshmen VHDL-based digital design. Em: *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2012, pp. 2701–2704. doi:10.1109/ISCAS.2012.6271865.

[6] Godofredo R. Garay et al. Visualization of VHDL-based simulations as a pedagogical tool for supporting computer science education. Em: *Journal of Computational Science* 36 (2019), p. 100652.issn: 1877-7503. doi:10.1016/j.jocs.2017.04.004. url:http://www.sciencedirect.com/science/article/pii/S187775031730385X.

[7] V. A. Pedroni. Teaching design-oriented VHDL. Em: *Proceedings 2003 IEEE International Conference on Microelectronic Systems Education. MSE'03*. 2003, pp. 6–7. doi:10.1109/MSE.2003.1205229.

[8] A. I. Hussein, D. M. Gruenbacher e N. M. Ibrahim. Design and verification techniques used in a graduate level VHDL course. Em: 2 (nov. de 1999), 13A4/28–13A4/31 vol.2.issn: 0190-5848. doi:10.1109/FIE.1999.841737.

[9] F. Pereira, F. Moutinho e L. Gomes. IOPT-tools — Towards cloud design automation of digital controllers with Petri nets. Em: *2014 International Conference on Mechatronics and Control (ICMC)*. 2014, pp. 2414–2419.doi:10.1109/ICMC.2014.7232002.

[10] F. Pereira e L. Gomes. Automatic synthesis of VHDL hardware components from IOPT Petri net models. Em: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. 2013, pp. 2214–2219. doi:10.1109/IECON.2013.6699475.

[11] F. Pereira e L. Gomes. The IOPT-Flow framework pairing Petri nets and dataflows for embedded controller development. Em: *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. 2016, pp. 4832–4837. doi:10.1109/IECON.2016.7794152.

[12] V. Boscaino et al. Modeling and simulation of a digital control design approach for power supply systems. Em: *2006 IEEE Workshops on Computers in Power Electronics*. 2006, pp. 246–249. doi:10.1109/COMPEL.2006.305638.

[13] Cadence, "Stratus High-Level Synthesis", https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-ds.pdf (Acedido de 24 Junho de 2021).

[14] L. Stornaiuolo, M. Santambrogio and D. Sciuto, "On How to Efficiently Implement Deep Learning Algorithms on PYNQ Platform," Em: *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 587-590, doi: 10.1109/ISVLSI.2018.00112.