SystemVerilog Vocabulary Extractor

Filipe C. Cavalcanti
Leandro de S. Albuquerque
Orientador: Tio Kat

11 de outubro de 2017

1 ABSTRACT

2 RESUMO

Desde a criação da primeira HDL até os presentes dias, cada vez mais o desenvolvimento de sistemas digitais se assemelha e aproxima-se a codificações de programas descritos em linguagem de programação

3 Introdução

Verilog foi uma das primeiras linguagens para descrição de hardware (HDL) a ser inventada em meados da década de 80. Até então o tamanho típico dos projetos era entre 5 a 10 mil portas lógicas. O método de concepção dos circuitos utilizava-se de esquema gráfico, e a simulação começava a ser ferramenta essencial para verificação [12]. Com a evolução da tecnologia de descrição e verificação de hardware, em 2002 surge SystemVerilog.

A partir disto, como a complexidade de sistemas digitais modernos aumentou exponencialmente, tanto que, o tamanho dos atuais projetos chega a ordem dos bilhões de portas lógicas. Assim as metodologias de projetos em sistemas digitais estão evoluindo extensivamente [7] e [4].

Com tal avanço, elevou-se o nível de abstração no desenvolvimento de hardware por meio de uma linguagem de descrição e verificação de hardware (HDVL), de tal forma que, o uso de ferramentas de análise de informações que antes eram somente do escopo da engenharia de software, pode ser estendido também para o desenvolvimento de sistemas digitais.

Umas das principais fontes de informações em um código fonte no âmbito da engenharia de software, é o vocabulário. Dentre suas principais utilidades listamos o seguinte:

1. Localização de bugs;
2. Identificação de uma arquitetura;
3. Métricas sobre o código fonte;
4. Identificação de especialista [11].

O vocabulário de software também denominado de léxico do código em [3], consiste no conjunto de termos repetidos ou únicos que compõem identificadores e que estão presentes no textos dos comentários [1].

Usando os princípios da engenharia reversa como uma coleção de metodologias e técnicas capazes de realizar a extração e abstração de informações [10], propõe-se neste trabalho a extração do vocabulário pertencentes a projetos de hardware descritos em SystemVerilog, e usando a definição formal de Santos em [11] sobre vocabulário de software para embasar e fundamentar o termo *Hardware Vocabulary*.

4 Background

Graças aos atuais projetos eletrônicos baseado em HDL, metodologias e ferramentas para simulação, síntese, verificação, modelagem física e teste pós-fabricação agora estão bem inseridos e são essenciais para designers digitais [9].

Nos últimos anos as linguagens de descrição e verificação de hardware tornaram-se tão importantes para a modelagem de sistemas digitais, quanto as linguagens de programação o são para a engenharia de software.

4.1 Software Vocabulary

Santos em [10], define que vocabulário de código fonte compreende as cadeias de caracteres que identificam os elementos estruturais e as palavras que compõem as sentenças dos comentários de um código fonte.

Dentro dos campos de estudo da engenharia de software outro termo bastante conhecido e isomorfo a *Software Vocabulary* é o *léxico do código* que em [3] são os elementos que nomeiam as entidades estruturais da linguagem além dos comentários escritos em linguagem natural.

Software Vocabulary é um multiconjunto de *Strings*, i.e., uma aplicação $V : \mathbb{S} \to \mathbb{N}$, que mapeia *Strings* para números naturais. Elementos de um vocabulário são chamados *termos*. Para qualquer termo t, $V(t)$ representa o número de ocorrências do termo t no vocabulário V. Se $V(t) > 0$ dizemos que t é um termo do vocabulário [11].

No paradigma de programação dominante atualmente OOP(*Object-Oriented Programming*), nomear os elementos estruturais da linguagem de forma concisa e representativa além de documentá-las, tem sido mais do que uma boa prática.

No desenvolvimento de grandes sistemas de software o vocabulário ou léxico, quando condizente ao problema, reduz o tempo de manutenção, facilita o entendimento do código e encontro de *bugs*.

O léxico de um programa representa um investimento substancial para uma empresa de software, portanto, sua importância deve preservada e elevada ao longo do tempo, para aproveitar ao máximo seus efeitos e benéficos na compreensão do programa [2].

4.2 Hardware Description Language (HDL)

Uma descrição HDL(*Hardware Description Language*) é uma representação precisa que pode ser usada para documentar, comunicar e simular o projeto [8].

As HDLs modernas são fundamentais para o desenvolvimento de sistemas digitais, possibilitando suas descrições de forma estrutural, comportamental e nos últimos anos, seguindo conceitos básicos de orientação a objetos [4], fornecendo assim um mecanismo efetivo para o desenvolvimento de projetos à medida que evoluem da abstração para a realidade [9].

Algumas HDls evoluíram de tal forma que, além de descrever hardware, passaram também a englobar todo um ambiente de verificação. Tais linguagens foram tipificadas de HDVL (Hardware Description and Verification Language).

Devido a grande quantidade de HDLs disponíveis, limitamos o escopo deste trabalho à análise de informações somente para projetos descritos em SystemVerilog.

A escolha dessa linguagem deve-se ao seu grande uso na industria VLSI (*Very Large Scale Integration* [6] e também por SystemVerilog ser uma HDVL [5], ampliando assim nosso horizonte de informações, uma vez que temos dados tanto sobre descrição quanto verificação de hardware.

4.3 O Hardware Como Um Software

A evolução no processo de desenvolvimento de software atrela-se ao fato do constante avanço dos paradigmas de programação, desde a programação procedural até a orientação a objetos.

De forma análoga, a evolução do processo de desenvolvimento de hardware, atrela-se também ao constante avanço dos métodos em desenvolver sistemas digitais. Indo desde esquemas gráficos à descrição baseada em orientação a objetos e verificação UVM (*Universal Verification Methodology*).

A aproximação (similaridade no processo de desenvolvimento) do software ao hardware surge da criação das HDLs. Neste ponto da evolução tecnológica, tanto hardware quanto software eram desenvolvidos por meio de uma representação textual unidemensional, seguindo uma determinada sintaxe e semântica.

Em [4] classes e módulos são coisas bastantes semelhantes, trazendo assim a possibilidade de usar classes dentro de um contexto de verificação de hardware. A partir desta similaridade, o inverso também é possível, ou seja, utilizar de conceitos e convenções de orientação a objetos na descrição de módulos.

4.4 Hardware Vocabulary

Baseado fortemente pela definição formal de vocabulário de software em [11], e embasado também por todo o levantamento feito sobre as semelhanças entre os processos de desenvolvimento entre hardware e software feitas na sessão anterior. Chegamos a primeira definição sobre vocabulário de hardware:

(1) Dada uma linguagem de descrição e (ou) verificação de hardware, temos que as sequencias de caracteres que nomeiam as entidades estruturais, assim como os blocos de comentários são chamados de *Vocabulário de Hardware*.

Complementando tal definição, o vocabulário de hardware refere-se também a como estes dados estão organizados, ou seja, a estruturação da informação refletindo as entidades a qual o léxico pertence, de modo que só com o vocabulário possa ser possível criar uma entidade complementar a original (entidade que seu vocabulário foi extraído).

Notando tal semelhança a definição de vocabulário de software e sabendo de seu isomorfismo com léxico do código(definição dada por Biggers em [3]), chegamos a conclusão que (1) também pode ser chamado de *léxico do hardware*.

Uma exemplificação é dada a partir de um trecho de código mostrado na *Listing 1*, onde seu vocabulário extraído, está numa representação VXL(Vocabulary XML).

Listing 1: Execerpt of code SystemVerilog

```
/* the module nextAddress returns the next address of the
2-bit offset program */
module nextAddress(
    input logic[7:0] regCounter,
    input logic[7:0] regAddress
    output logic[7:0] regPC);
  always_comb begin
    regPC <= (regCounter + regAddress) << 2;
  end
endmodule: nextAddress
```

Listing 2: Extracted Vocabulary of Listing 1

```
<mdl name= "nextAddress">
  <prm>
    <fld type="input logic[7:0]" name=
    "regCounter"/>
    <fld type="input logic[7:0]" name=
    "regAddress"/>
    <fld type="output logic[7:0]" name= "regPC"/>
  </prm>
  <cmt cmm="the module nextAddress returns the next
address of the 2-bit offset program"/>
</mdl>
```

Como pode ser notado no resultado da extração do vocabulário, a estruturação dos dados segue fielmente a modelagem original do problema, assim sendo a partir do vocabulário pode-se recriar uma entidade complementar a original.

Notamos que a definição formal de Santos em [11] sobre vocabulário de software, aplica-se igualmente ao vocabulário de hardware, assim chegamos a definição formal de vocabulário de hardware:

(2) *Hardware Vocabulary definition*: Nós definimos *Hardware Vocabulary* como um multiconjunto de *Strings*, isto é, uma aplicação $V : \mathbb{S} \rightarrow \mathbb{N}$, que mapeia *Strings* para números naturais. Elementos de um vocabulário são chamados *termos*. Para qualquer termo t, $V(t)$ representa o número de ocorrências do termo t no vocabulário V. Se $V(t) > 0$ dizemos que t é um termo do vocabulário.

Como um exemplo considere o trecho de código em *Listing 1*. De acordo com nossa definição o *Vocabulário de Hardware* pode ser expresso como o seguinte multconjunto de termos:

$$V : 2'regCounter + 2'regAddress + 2'regPC + 2'nextAddress$$

5 SystemVerilog Vocabulary Extractor

Com a necessidade de analisar e extrair o vocabulário de hardware pertencentes a projetos descritos em SystemVerlog, desenvolvemos o ferramental *Hardware Vocabulary Tool*.

A implementação desta ferramenta, baseia-se em entidades que são capazes de processar o vocabulário de hardware dos principais elementos estruturais básicos de SystemVerilog. Usando de delegação montam-se as estruturas mais complexas da linguagem (classes, interfaces, módulos e pacotes) Ex.: Uma classe SystemVerilog é uma entidade que possui um nome e compõem-se

principalmente de atributos, funções, tasks, comentários.

Afim de validarmos a ferramenta calculamos uma porcentagem de extracão que melhor represente a eficiência do software proposto, elaborando um design genérico com todas as estruturas possíveis da linguagem SistemVerilog. Os resultados obtidos são apresentados na tabela abaixo:

Tabela 1: Porcentagem de extração das principais entidades de SystemVerilog.

Entidade	Porcentagem de extração
module	90%
class	98%
interface	90%
function	100%
task	90%
fields	100%

Os resultados apresentados na tabela acima mostram que, calculando uma média aritmética, ou seja, deduzindo que o vocabulário de hardware pertencente a cada entidade é aproximadamente igual, o percentual de extração total atualmente é de 92% dado um código genérico.

Devido a dificuldade para encontrar projetos *opensource* realizamos testes com projetos obtidos em repositórios públicos no site *https://github.com*, afim de aferir o Tempo de Extração (TE), em milissegundo (ms), e a Quantidade de Memória(QM) usada durante a extração. Os resultados obtidos estão expostos na tabela abaixo:

Os resultados apresentados na tabela acima mostra que ...

6 Resultados e Discussões

Cada Desenvolvedor tem seu vocabulário, que é a acumulação de termos manipulados (adicionados, removidos e substituídos) durante suas contribuições no projeto [11]. Isso equivale à desenvolvimento de hardware, uma vez citada a similaridade entre as HDLs e as linguagens de programação.

Com a definição de *Vocabulário de Hardware* formalizada, abrimos então um novo campo de estudos para a engenharia de software. A análise e extração de informação pertencentes ao vocabulário de projetos de hardware descritos por alguma HDL.

Por fim, levantada todas as semelhanças do processo de desenvolvimento de hardware ao software, concluímos que toda analise, extração, métricas e conclusões feitas em cima do vocabulário de software são equivalentes quando aplicadas ao Vocabulário de Hardware.

Referências

[1] S. L. Abebe, S. Haiduc, A. Marcus, P. Tonella, and G. Antoniol. Analyzing the evolution of the source code vocabulary. *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, pages 189–198, 2009.

[2] G. Antoniol, Y. G. Guéhéneuc, E. Merlo, and P. Tonella. Mining the lexicon used by programmers during sofware evolution. *IEEE International Conference on Software Maintenance, ICSM*, pages 14–23, 2007.

[3] L. R. Biggers, B. P. Eddy, N. A. Kraft, and L. H. Etzkorn. Toward a metrics suite for source code lexicons. *IEEE International Conference on Software Maintenance, ICSM*, pages 492–495, 2011.

[4] V. Hahanov, D. Melnik, O. Zaharchenko, and S. Zaychenko. Overview of Object-Oriented Approach to HDL- Testbench Construction for System-on-Chips. pages 621–625, 2008.

[5] IEEE Computer Society and IEEE Standards Association Corporate Advisory Group. IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language. 2012(February):1315, 2013.

[6] P. Kumar. High Level Modeling Of Physical Layer Noise Parameters Using SystemC. pages 344–347, 2014.

[7] J. D. M. DAIGNEAULT. RAISING THE ABSTRACTION LEVEL OF HDL FOR CONTROL-DOMINANT APPLICATIONS Marc-Andre Daigneault and Jean

Tabela 2: Extração do *Vocabulário de Hardware* de outros projetos.

Nome do Projeto	TP(Bytes)	QL	TE (ms)
ahb_apb_bridge_uvm_tb https://github.com/mayur13/UVM/tree/master/projects/ahb_apb_bridge_uvm_tb	122.880	997	260
apb2_uvm_tb https://github.com/mayur13/UVM/tree/master/projects/apb2_uvm_tb	147.456	806	278
fpga_fast_serial_sort https://github.com/Poofjunior/fpga_fast_serial_sort	49.152	351	205
System-Verilog https://github.com/zricethezav/System-Verilog	49.221.632	4420	970
sha3_sv_tb https://github.com/mayur13/SystemVerilog/tree/master/projects/sha3_sv_tb	65.536	242	297
sha3_uvm_tb https://github.com/mayur13/UVM/tree/master/projects/sha3_uvm_tb	176.128	925	278
uvm-tutorial-for-candy-lovers https://github.com/cluelogic/uvm-tutorial-for-candy-lovers	700.416	11559	1.877
spi_uvm_tb https://github.com/mayur13/UVM/tree/master/projects/spi_uvm_tb	139.264	172	177
uvm-utest https://github.com/nosnhojn/uvm-utest	311.296	5281	1.379
zynq-sandbox https://github.com/swetland/zynq-sandbox	450.560	8671	546

Pierre David Department of Electrical Engineering , Ecole Polytechnique de Montreal. pages 515–518, 2012.

[8] D. L. Miller-Karlow and E. J. Golin. vVHDL: A Visual Hardware Description Language.

[9] Z. Navabi. HDLs Evolve as they Affect Design Methodology for a Higher Abstraction and a Better Integration. page 4799, 2015.

[10] K. D. F. Santos. Webservice De Extração De Vocabulário De Código Para Pesquisas Empíricas Em Engenharia De Software. 2009.

[11] K. D. F. Santos, D. D. S. Guerrero, and J. C. A. D. Figueiredo. Using Developers Contributions on Software Vocabularies to Identify Experts. *Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015*, pages 451–456, 2015.

[12] S. Sutherland, S. Davidmann, and P. Flake. *SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling*. 2006.