ПРИЛОЖЕНИЕ НА TL-VERILOG ЗА МОДЕЛИРАНЕ НА КОНВЕЙЕРНА МИКРОПРОЦЕСОРНА АРХИТЕКТУРА

Петър Минев, Валентина Кукенска, Илиан Върбов, Матьо Динев

Технически университет – Габрово
{ivarbov, vally, pminev, mdinev}@tugab.bg

APPLICATION OF TL-VERILOG FOR MODELLING A PIPELINED MICROPROCESSOR ARCHITECTURE

Petar Minev, Valentina Kukenska, Ilian Varbov, Matyo Dinev

Technical University of Gabrovo
{ivarbov, vally, pminev, mdinev}@tugab.bg

Abstract

This paper presents pipelined processor architectures and techniques for modelling them at a high level of abstraction, using the hardware description language TL-Verilog. A model of a classic 5-stage pipeline microarchitecture is developed. The basic steps for implementation and visualization of the model are proposed. The model is validated through simulation and visualization of its operation. Freely available online tools for open-source development, including RISC-V toolchain, and the online Makerchip IDE, were used for a model implementation and validation. The necessity of its application in the educational process of undergraduate students is substantiated.

Keywords: TL-Verilog; HDL; modelling; pipeline; microarchitecture.

ВЪВЕДЕНИЕ

Стандартната методика за проектиране на цифров хардуер включва описание или моделиране на хардуера на регистрово ниво чрез езици за хардуерно описание, като Verilog и VHDL. Описанието на ниво регистри (Register Transfer Level) позволява на проектантите да управляват, на сравнително ниско ниво, структурата и времевите характеристики на схемата. Същевременно, паралелната семантика, използвана при съставяне на RTL моделите може да затрудни тяхното разбиране, както и валидирането им.

В съвременните процесори изпълнението на последователните потоци от инструкции е паралелизирано чрез конвейери. Освен това за повишаване на пропускателната способност се прилагат различни решения за оптимизация на микроархитектурата. Такива са: пренасочване, спекулативно изпълнение и изпълнение с пренареждане. Процесът на проектиране на конвейерен процесор включва изграждането на коректен RTL модел. В модела трябва да се реализира паралелното изпълнение на операциите в различните части от схемата на конвейера, като същевременно се отчита последователната семантика на системата инструкции (ISA). За да се улесни процесът по създаване на коректен RTL модел на процесор, изпълняващ множество взаимодействащи си инструкции едновременно, може да се използва езикът TL-Verilog.

ИЗЛОЖЕНИЕ

А. Език за описание на хардуер TL-Verilog

Езикът за описание на хардуер на високо ниво TL-Verilog възниква като разширение на SystemVerilog. В него се използва методология на проектиране, основана на „транзакции“. Транзакцията е универсална проектна единица. Тя може да бъде машинна инструкция, операция четене/запис при достъп до паметта или мрежов пакет. Транзакцията се управлява и насочва, чрез универсални структури като конвейери, опашки или превключващи/арбитриращи схеми. Движението на транзакцията може да е независимо от логиката, с която се извършва нейната обработка [5]. Това дава възможност за моделиране на конвейерно изпълнение в цифровите изчислителни структури, включително и моделиране на паралелизма на ниво инструкции в микроархитектурата на процесорните ядра.

Моделирането с транзакции е по-високо ниво на абстракция спрямо моделирането с наложилата се от десетилетия абстракция „трансфер между регистри“, използвана при традиционното проектиране с VHDL или Verilog. Това води до подобряване четимостта на кода, намаляване на грешките и съкращаване на времето за разработка на TL-Verilog модели [3].

Моделите с TL-Verilog се трансформират до RTL ниво под формата на оптимизиран Verilog и SystemVerilog код, чрез компилатор. Първият компилатор е разработен в Intel през 2014 г. Към настоящия момент се използва компилаторът SandPiper на компанията Redwood EDA. Достъпен е безплатно под формата на микроуслугата SandPiper SaaS. За да се направи симулация, SystemVerilog кода се трансформира до C++/SystemC модел, чрез инструмента Verilator [6].

Посоченото до тук показва, че за моделиране с TL-Verilog не е необходимо да се изучават езици за описание на хардуер, като Verilog, SystemVerilog, SystemC и други. Техният произход е свързан с езиците за програмиране C/C++, а първоначалното им предназначение е само за симулация. Освен присъщите им последователни оператори, са добавени и паралелни такива, което води до усложняване на моделите [2].

Студенти, ентусиасти или професионалисти могат в сравнително кратък срок да се запознаят и да започнат да използват TL-Verilog за разработка на собствени модели на цифрови системи [3]. Наличието на свободно достъпната онлайн среда за разработка Makerchip също е предпоставка за бързото внедряване на езика и практическото му приложение.

Б. Система инструкции RISC-V

Първата стъпка при проектиране на процесор е изборът на неговата система от инструкции (ISA). В този случай за създаването на модел на конвейерна процесорна архитектура е използвана, добилата популярност през последните години системата инструкции RISC-V.

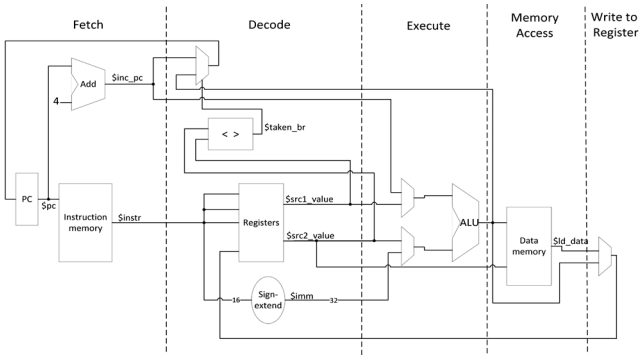
Системата инструкции RISC-V е разработена преди повече от десетилетие в Калифорнийският университет в Бъркли [8]. Тя е разширяема и свободна за използване без заплащане на лицензионни такси. За RISC-V е наличен пълен комплект софтуерни инструменти (компилатори, дебъгери, асемблери, операционни системи и симулатори), които също са достъпни за свободно използване. Налични за свободно използване са и множество HDL модели на RISC-V процесорни ядра за реализация в специализирани интегрални схеми или програмируема логика [4]. Това е предпоставка, изследванията и проектирането на реални процесори с тази архитектура бързо да нарастват. В допълнение за RISC-V има разработени платени и свободнодостъпни образователни ресурси под формата на онлайн курсове и материали. Съществуват и множество цялостни университетски или фирмени учебни програми, използващи RISC-V за основа в практическото обучение.

Разработена 30 години след появата на първите системи инструкции от вида RISC, в RISC-V са заложени основните характеристики на класическите RISC архитектури – голям набор от регистри и прости инструкции, подходящи за конвейерно изпълнение. Същевременно се избягват някой от допуснатите пропуски или грешки. Показателно е, че повече от 60 компании са се присъединили към фондацията RISC-V, включително: AMD, Google, HP Enterprise, IBM, Microsoft, Nvidia, Qualcomm, Samsung, и Western Digital.

В. Реализация на модел на микроархитектура на процесор с конвейерно изпълнение на инструкциите

След определяне на инструкциите, които процесорът трябва да изпълнява е необходимо да се определи, как ще бъде организирано тяхното изпълнение. За тази цел отделните стъпки при изпълнението на инструкциите се разпределят в няколко конвейерни етапа. На фиг. 1 е представено разпределението на микроархитектурните елементи в конвейер с пет етапа на изпълнение на инструкции [1].

На този етап следва да се преобразува моделът на неконвейерната архитектура представена в [7] в конвейерен процесор, като се използват съответните лексикални особености на TL-Verilog, отнасящи се до моделиране на конвейерни изчислителни структури [5].



***Фиг. 1**. Конвейерна микроархитектура на процесор, изпълняващ инструкции от системата RISC-V*

Макар, че конвейерното изпълнение в посочения модел не е явно реализирано, отделните етапи при изпълнението на една инструкция са лесно разграничими. Първо инструкцията се извлича от паметта, като за целта се използва програмният брояч, следва декодиране на двоичния код на инструкцията, записана в регистъра на инструкции, както и прочитането на операндите от регистрите. Едва след това може да се пристъпи към изпълнение на операцията, заложена в инструкцията и в крайна сметка изпълнението да завърши, а полученият резултат да се запише в указания целеви регистър. На тази основа, разграничавайки кои елементи от модела в кой етап от изпълнението на инструкцията участват, програмният код може лесно да се модифицира според TL-Verilog конструкцията, показна на фиг. 2.

```
|<pipe-name>
   @<pipe stage>
      Instructions present in this stage
   @<pipe_stage>
      Instructions present in this stage
```

Фиг. 2. *Конструкция в TL-Verilog за моделиране на изчислителен конвейер*

При реализацията на модела с конвейерна микроархитектура е необходимо да се използва друга библиотека, в която макросите за паметта за инструкции и данни, регистровият файл и визуализацията са съобразени с конвейерното изпълнение. Използваните в това обучение библиотеки са записани в публично достъпни хранилища в github [9].

На фиг. 3 е показано, в кой конвейерен етап ще се използва съответният макрос за четене и запис. Параметризацията на макросите е направена с цел да не се затруднява тяхното използване.

Г. Валидация на модела на конвейерна RISC-V микроархитектура

Чрез онлайн хранилище в github [10] студентите получават достъп до необходимите библиотеки и файлове, от които да стартират изграждането на моделите си. Вариант на завършения модел на RISC-V процесорно ядро с пет степенна

конвейерна микроархитектура също е наличен в посоченото хранилище. Трябва да се отбележи, че проектът все още не е документиран и организиран по подходящ начин, тъй като засега не е предназначен да бъде използван от незапознати с предназначението му потребители.

```
|cpu
   m4+imem(@1)    // Args: (read stage)
   m4+rf(@2, @5)   // Args: (read stage, write stage)
   m4+dmem(@4)    // Args: (read/write stage)
   m4+cpu_viz(@5) // For visualization,
   // argument should be equal to the last stage
```

Фиг. 3. *Инстанциране на макросите за: регистровия файл, паметите за данни и инструкции и визуализацията.*

В основата на демонстрацията на работата на модела е код на кратка тестова програма на асемблер, която реализира алгоритъм, представен чрез псевдокода от фиг. 4.

```
1       sum = 0;
2       for i = 1 to 10
3           begin
4               sum = sum + 2;
5           end
```

Фиг. 4. *Псевдокод на програмата за тестване на TL-Verilog модела на RISC-V микроархитектурата*

За реализация на програмата от фиг. 4 е използван езикът C. Готовата програма е компилирана с крос-компилатора от инструментите в RISC-V GNU Toolchain за Linux. След компилацията машинният код се дисасемблира. Така се получава кода на тестовата програма, който се залага в TL-Verilog модела. На фиг. 5 е показан начинът, по който отделните инструкции се записват в TL-Verilog код. Мнемоничният код на инструкцията се записва като параметър в макроса m4_asm(). При обработката на този макрос от препроцесора M4, преди същинската компилация със SandPiper, инструкциите се преобразуват в техния еквивалентен двоичен машинен код.

Наличието на зависимости между инструкциите в програмата от фиг. 5 позволява, при тестването на конвейера, да се демонстрира, че тяхното преодоляване е възможно, благодарение на техниката за пренасочване на резултата от изпълнението на операцията, реализирана в модела. Чрез пренасочване не може да се преодолее единствено съществуваща зависимост между инструкция load и първата съседна след нея. В този случай между двете инструкции трябва да се вмъкне друга независима инструкция, която да забави с един процесорен цикъл зависимата. Такъв случай съществува между инструкциите на редове 18 и 20 в програмата от фиг. 5. Описаната зависимост, и преодоляването ѝ е демонстрирано на фиг. 6.

```
1       m4_asm(ADDI, r8, r0, 0)
2       m4_asm(SW, r8, r0, 1000)
3       m4_asm(ADDI, r15, r0, 1010)
4       m4_asm(SW, r8, r15, 1100)
5       m4_asm(ADDI, r15, r0, 1)
6       m4_asm(SW, r8, r15, 100)
7       m4_asm(JAL, r0, 1110)
8       //loop:
9       m4_asm(LW, r15, r8, 1000)
10      m4_asm(ADDI, r15, r15, 10)
11      m4_asm(SW, r8, r15, 1000)
12      m4_asm(LW, r15, r8, 100)
13      m4_asm(ADDI, r15, r15, 1)
14      m4_asm(SW, r8, r15, 100)
15      //start:
16      m4_asm(LW, r15, r8, 100)
17      m4_asm(ADDI, r14, r15, 0)
18      m4_asm(LW, r15, r8, 1100)
19      m4_asm(ADD, r0, r0, r0)
20      m4_asm(BGE, r15, r14, 1111111011000)
21      m4_asm(ADDI, r15, r0, 0)
22      m4_asm(ADDI, r10, r15, 0)
```

Фиг. 5. *Програмата за тестване, записана в TL-Verilog кода на модела*

Работата на модела в средата за разработка Makerchip се визуализира чрез времедиаграми, показващи изменението на двоичните цифрови сигнали във времето. Входните въздействия се генерират автоматично, като генерираните стойности са произволни. Това освобождава студентите от задължението да изготвят тестова установка и да указват подходящи стойности за входни въздействия. При сложни цифрови системи, като представяното тук процесорно ядро е изключително трудно да се проследи поведението на системата, чрез преглед на времедиаграмите на цифровите сигнали, тъй като те са значителен брой. Изготвеният

модел на ядро с петстепенен конвейер, например, съдържа около 100 сигнала.

Fetch - F	Decode - D	Execute - E		Memory - M		Write - W				
		Processor Cycles								
Instruction		1	2	3	4	5	6	7	8	9	10
load r15, r8, 4		F	D	E	M	W					
addi r14, r15, 0			F	D	E	M	W				
load r15, r8, 12				F	D	E	M	W			
add r0, r0, r0					F	D	E	M	W		
bge r15, r14, -40						F	D	E	M	W	
addi r15, r0, 0							F	D	E	M	W

Фиг. 6. *Диаграма демонстрираща състезанията с инструкцията load при конвейерното изпълнение*

В HDL модела може да се добавят изрази за извеждане на текстови съобщения, които да показват текущото състояние на симулирания модел. Но и извеждането на текстови съобщения с данни за модела не дава пълноценна възможност за представяне на неговото поведение. Ето защо в езика TL-Verilog и в системата за разработка Makerchip са добавени нови методи за визуализиране на поведението на симулирания модел, чрез графични елементи или най-общо казано, чрез визуална структура, която се променя според постъпващите данни от модела при неговото симулационно изпълнение.

Визуалната структура се вгражда във файла с TL-Verilog модела. Тя се представя, чрез JavaScript код и библиотеката за графичен дизайн Fabric.js. Вграденият JavaScript има достъп до сигналите в модела и техните стойности. Тази нова система за графична визуализация на поведението на модели на цифрови системи е използвана и за да се представи по подходящ начин работата на RISC-V микроархитектурата, обект на разглеждане в настоящия доклад (фиг. 7).

Фиг. 7. *Програмата за тестване, записана в TL-Verilog кода на модела*

Изобразяват се инструкциите с техния мнемоничен код, съдържанието на паметта за инструкции, регистровия файл и паметта за данни. Посочва се текущата инструкция в последната фаза от нейното изпълнение в конвейера, заедно с нейните операнди. Посочва се целевият регистър или клетка от паметта, където се записва резултатът. Постъпковото изпълнение се контролира от бутони „Step“, „Step Back“ и „Jump to Start“ в горния десен ъгъл на прозореца за визуализация в Makerchip.

ЗАКЛЮЧЕНИЕ

В този доклад бяха представени възможностите на езика за описание на хардуер TL-Verilog и онлайн средата за разработка Makerchip за моделиране и валидиране на конвейерни процесорни микроархитектури. От представеното в доклада може да се заключи, че TL-Verilog е достъпен за използване и изучаване език, чрез който цифровите изчислителни системи могат да се моделират на по-високо ниво на абстракция в сравнение с традиционното регистрово или RTL ниво. Това опростява моделите и намалява обема на необходимия програмен код. Сложни системи, като конвейерни микропроцесорни ядра, могат да се моделират и валидират по-лесно и бързо, ако се използва моделирането с транзакции в TL-Verilog. В моделите създадени с TL-Verilog могат да се добавят блокове с Verilog и SystemVerilog код. Това позволява да се използват езикови конструкции, които все още липсват в TL-Verilog спецификацията. Например в TL-Verilog не могат да се дефинират масиви, което затруднява моделирането на памети и регистри.

Възможността за създаване на визуална симулация на работата на TL-Verilog моделите, чрез вграждане на JavaScript код за графичен дизайн на симулацията е нововъведение в проектирането на цифров хардуер, което заслужава отделно представяне в самостоятелен доклад.

Представеният модел на RISC-V процесорно ядро с конвейерна микроархитектура и езикът TL-Verilog могат да се използват в практическото обучение по дисциплината Компютърни архитектури. Авторите считат, че в условията на намаляващ хорариум по дисциплините в бакалавърските учебни планове, разработката на модели на процесорни ядра, чрез TL-Verilog и свързаните с езика инструменти, е подходящ подход не само за задълбочено разбиране на преподаваните концепции в конвейерните микроархитектури, чрез демонстриране на тяхната работа и съпоставката им, но и за субективизиране на принципите на проектиране на компютърни системи като задължителен елемент в изграждането на всеки бъдещ компютърен инженер.

БЛАГОДАРНОСТИ

Този доклад е подготвен и осъществен като част от проект № 2209Е „Виртуална лаборатория за обучение по проектиране на цифров хардуер“, финансиран от средствата по бюджета за научни изследвания на Технически университет – Габрово.

REFERENCE

[1] Hennessy J. and D. Patterson, Computer Architecture, Sixth Edition: A Quantitative Approach (6th. ed.). San Francisco, CA: Morgan Kaufmann Publishers, 2017.

[2] Coussy P., M. Meredith, A. Takach, D. Gajski, “An Introduction to High-Level Synthesis,” in IEEE Design & Test of Computers, vol. 26, no. 04, pp. 8-17, 2009.

[3] Hoover S. and A. Salman, "Top-Down Transaction-Level Design with TL-Verilog", CoRR, vol. abs/1811.01780, 2018, [online] Available: http://arxiv.org/abs/1811.01780.

[4] Höller R., D. Haselberger, D. Ballek, P. Rössler, M. Krapfenbauer and M. Linauer, “Open-Source RISC-V Processor IP Cores for FPGAs — Overview and Evaluation,” 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2019, pp. 1-6, doi: 10.1109/MECO.2019.8760205.

[5] Hoover S., “Timing-Abstract Circuit Design in Transaction-Level Verilog,” 2017 IEEE International Conference on Computer Design (ICCD), Boston, MA, USA, 2017, pp. 525-532, doi: 10.1109/ICCD.2017.91.

[6] Snyder W., "Verilator and SystemPerl". North American SystemC Users' Group, Design Automation Conference, June 2004.

[7] Varbov I., V. Kukenska, P. Minev, M. Dinev, “Application of TL-Verilog for modeling components of microprocessor architectures,” International Scientific Conference UNITECH´23, Gabrovo, 17 – 18 November 2023, *In Print*.

[8] Waterman A., Y. Lee, D. A. Patterson, and K. Asanović, “The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA,” Technical Report UCB/EECS-2011-62, EECS Department, University of California, Berkeley, May 2011.

[9] Hoover S., “Stevehoover - Overview,” GitHub, https://github.com/stevehoover/ (accessed Jun. 12, 2023).

[10] Minev P., “MIPS pipeline processor models repository,” https://github.com/pminev23/mipsI (accessed Jun. 28, 2023).