последовательностей применения разработок, предлагается новый подход, основанный на библиотеке стандартных последовательностей, осуществленных в виде шаблонов, а также на специальном наборе инструментов для модификации и проверки этих шаблонов. Разработанная библиотека шаблонов позволяет быстро спроектировать новые последовательности путем нахождения и переделывания частных примеров. Пользователь может расширить библиотеку путем добавления новых шаблонов. Подход формальной верификации, используемый в бизнес процессах, успешно применен к сконструированной библиотеке. Данное применение некоторых примерах последовательностей применения.проиллюстрированно на

На защиту выносятся следующие положения:

- Формальный язык для представления последовательностей применения и их шаблонов.
- Обобщенная библиотека последовательностей применения с основными шаблонами
- Подход для создания шаблонов последовательностей применения, основанный на комбинациях основных шаблонов.
- Проверка изменений шаблонов с использованием алгоритмов формальной верификации.
- Разработанные инструменты для осуществления данного подхода, а также эксперименты для подтверждения его правильности.

Основные результаты:

- Разработан формальный язык для последовательностей применения RTL компиляторов и их шаблонов.
- Сконструирована библиотека шаблонов последовательностей применения RTL компиляторов
- Показана возможность преобразования элементов библиотеки шаблонов последовательностей применения в конструкции формальных моделей бизнес процессов
- Разработаны инструменты для осуществления данного подхода.
- Эксперименты показали правильность описанного подхода.

Ծավալը՝ 24 էջ: Տպաքանակը՝ 100:
ՀՀ ԳԱԱ ԻԱՊԻ կոմպյուտերային պոլիգրաֆիայի լաբորատորիա:
Երևան, Պ. Սևակի 1

ՀՀ ԳԱԱ ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԱՎՏՈՄԱՏԱՑՄԱՆ ՊՐՈԲԼԵՄՆԵՐԻ ԻՆՍՏԻՏՈՒՏ

Խզարջյան Արամ Անդրանիկի

RTL կոմպիլյատորների շահագործման ավտոմատացման գործիքային, ծրագրային միջոցների հիմնավորում և մշակում

Ե.13.04 «Հաշվողական մեքենաների, համալիրների, համակարգերի և ցանցերի մաթեմատիկական և ծրագրային ապահովում» մասնագիտությամբ տեխնիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսության

Ս Ե Ղ Մ Ա Գ Ի Ր

Երևան – 2014

ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ И АВТОМАТИЗАЦИИ НАН РА

Хзарджян Арам Андраникович

Обоснование и разработка программных инструментальных средств для автоматизации эксплуатации RTL компиляторов

А В Т О Р Е Ф Е Р А Т

диссертации на соискание ученой степени кандидата технических наук
по специальности 05.13.04 – “Математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей”

Ереван – 2014

Ատենախոսության թեման հաստատվել է Հայաստանի Պետական Ճարտարագիտական Համալսարանում

Գիտական ղեկավար՝ ֆիզ.մաթ.գիտ.դոկտոր Ս.Կ. Շուքուրյան

Պաշտոնական ընդդիմախոսներ՝ տեխ.գիտ.դոկտոր Ա.Թ. Քուչուկյան
տեխ.գիտ.թեկնածու Գ.Է.Հարությունյան

Առաջատար կազմակերպություն՝ Սինոփսիս Արմենիա ՓԲԸ
Ուսումնական դեպարտմենտ

Պաշտպանությունը կայանալու է 2014թ. Հունիսի 17-ին, ժամը 15.00-ին ՀՀ ԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտում գործող 037 “Ինֆորմատիկա և հաշվողական համակարգեր” մասնագիտական խորհրդի նիստում հետևյալ հասցեով՝ Երևան, 0014, Պ. Սևակի 1:

Ատենախոսությանը կարելի է ծանոթանալ ՀՀ ԳԱԱ ԻԱՊԻ-ի գրադարանում:
Սեղմագիրն առաքված է 2014թ. Մայիսի 17-ին:

Մասնագիտական խորհրդի գիտական քարտուղար, ֆ.մ.գ.դ. Հ.Գ. Սարուխանյան

Тема диссертации утверждена в Армянском государственном инженерном университете

Научный руководитель: доктор физ. мат. наук С.К. Шукурян

Официальные оппоненты: доктор тех. наук А.Т.Кучукян
кандидат тех. наук Г.Э.Арутюнян

Ведущая организация: ЗАО "СИНОПСИС АРМЕНИЯ"
Департамент образования

Защита состоится 17-го июня 2014г. в 15.00 на заседании специализированного совета 037 “Информатика и вычислительные системы” Института проблем информатики и автоматизации НАН РА по адресу: 0014, г. Ереван, ул. П. Севака 1.

С диссертацией можно ознакомиться в библиотеке ИПИА НАН РА.
Автореферат разослан 17-го мая 2014г.

Ученый секретарь специализированного совета, д.ф.м.н. А.Г. Саруханян

Хзарджян Арам Андраникович

Обоснование и разработка программных инструментальных средств для автоматизации эксплуатации RTL компиляторов

ЗАКЛЮЧЕНИЕ

Каждый новый уровень полупроводниковых технологий приводит к новому уровню миниатюризации и увеличению производительности, тем самым позволяя повысить функциональность, которую могут обеспечить электронные продукты. Хотя конечный пользователь и выигрывает с добавлением новых функций, необходимость обеспечения процесса производства полупроводников более тонкой структуры и высокой плотности приводит к большей чувствительности чипов к дефектам. Современные субмикронные полупроводниковые технологии ниже 130нм достигают уровней чувствительности к дефектам, способных понизить выход продукции и надежности, и, следовательно, продлевают период нарастания производства.

Сверхсубмикронные технологии позволяют увеличить количество транзистров на кристалле и, таким образом, осуществить более сложные схемы на одном чипе. Система состоит из нескольких компонентов, которые изначально располагались на разных чипах. Эти компоненты можно расположить на одном чипе называемом SoC (system-on-chip). Основными достоинствами использования SoC являются: миниатюризация размеров устройств, сокращение стоимости продукта, сокращение времени вывода продукта на рынок. Обычно SoC включает в себя разные ядра от разных поставщиков интеллектуальной собственности (ИС). Ядра могут быть переделаны и повторно использованы во многих SoC проектах, требующих стандартные интерфейсы для этих ядер. IEEE 1500 является одним из таких известных стандартов.

Вычислительная мощность и размер встроенной памяти SoC должны быть достаточны для прикладных задач, требующих большие ресурсы памяти. Как следствие, встроенная память становится основной компонентой SoC, которая в 2014 году займет более 94% площади SoC. С точки зрения полезного выхода продукции, встроенная память более подвержена дефектам, чем какие-либо другие компоненты SoC. Существуют так называемые встроенные самотестирующие и восстанавливающие ИС ядра, предназначенные для разрешения этой проблемы. STAR Memory System (SMS) является одним из таких известных решений, которое представляет собой завершенный набор инфраструктурных ИС компиляторов для встроенных памятей и программного обеспечения. STAR Hierarchical System (SHS) - инфраструктурное ИС решение, предназначенное для механизма встроенного самотестирования и восстановления в ИС блоках. В настоящем, обе системы широко используются многими заказчиками, применяющими разнообразные методы разработок.

Для удовлетворения разнообразных запросов со стороны заказчиков возникает необходимость определения языка, предназначенного для какой-либо частной последовательности применения SMS и SHS разработок. Учитывая разнообразие

on a library of SMS/SHS standard use flows implemented in a form of templates and on a special toolset for modification and verification of these template. The implemented library of templates assists to design new flows quickly through retrieving and customizing specific examples. User can extend the library via insertion of new templates. The formal verification approach used already for business processes is successfully applied to the built library. The application is illustrated on some use flow examples.

The following aspects are provided in the thesis:

- Formal language of presentation of use flows and their tempalates.
- Generalized use flow library containing some basic templates
- An approach to construct new use flow templates based on various combinations of basic templates
- Verification of template changes using formal verification algorithms
- Developed tools to implement the suggested approach and experiments to prove its correctness.

The main results are:

- Formal language of RTL compiler use flows and their templates has been defined.
- Use flow template library of RTL compilers has been constructed.
- It has been shown that the elements of use flow template library can be transformed to a business process formal model constructs.
- It has been proved that one of business process formal verification algorithms can be applied for verification of use flow template changes.
- Tools are developed to implement the approach.
- Experiments have shown the correctness of the described approach.

ԱՇԽԱՏԱՆՔԻ ԸՆԴՀԱՆՈՒՐ ԲՆՈՒԹԱԳԻՐԸ

Թեմայի արդիականությունը

Յուրաքանչյուր կիսահաղորդչային տեխնոլոգիա ինտեգրալ սխեմաների (ԻՍ) տարրերի փոքրացման և արտադրողականության աճի հաշվին իր հետ բերում է նոր հնարավորություններ՝ ԻՍ ֆունկցիոնալության և արագագործության համար: Մյուս կողմից ֆունկցիոնալության աճը իր հերթին բերում է մի շարք նոր խնդիրներ, որոնց համար պահանջվում են նոր տիպի նախագծման և արտադրության պրոցեսներ: Այդ պրոցեսները դարձնում են ԻՍ-երին ավելի ընկալունակ տարբեր տեսակի արատների նկատմամբ[1]: Ժամանակակից ենթամիկրոնային կիսահաղորդչային տեխնոլոգիաները վերջին 10 տարում հասել են 130նմ-ից մինչև 14նմ և ավելի փոքր չափերի՝ իրենց հետ բերելով արատների զգայունության այնպիսի մակարդակ, որը հանգեցնում է արտադրության օգտակար ելքի ու հուսալիության կտրուկ նվազեցման և ԻՍ ֆունկցիոնալության ավելացման հետ մեկտեղ՝ նախագծման բարդության զգալի աճին[2]:

Միևնույն ժամանակ մրցակցային օրեցօր աճող պայքարը պարտադրում է նոր սերնդի սարքերի շուկա դուրս գալու ժամանակի անընդհատ կրճատում: Այդ տեսանկյունից նախագծողի համար արդիական է դառնում նոր ֆունկցիոնալությունը կարճ ժամանակում և որակով օգտագործողին մատուցելը՝ օգտագործելով ԻՍ-երի համար կիրառվող կիսահաղորդչային տեխնոլոգիական նոր մակարդակ[3]:

Համակարգ բյուրեղի վրա (ՀԲՎ) կոչվում են Մտավոր սեփականության (ՄՍ) բաղադրիչների հիման վրա ստեղծված ԻՍ-երը: Այն թույլ է տալիս արտադրողներին պատրաստի ՄՍ միավորներից արագ հավաքել ՀԲՎ: Այդպիսի միավորները կարող են լինել ինչպես ամբողջական ու անփոփոխ, այնպես էլ՝ որոշակի եղանակով պարամետրացվող RTL կոմպիլյատորների հիման վրա: ՄՍ-երը նախագծվում և ստուգվում են որոշակի մատակարար կազմակերպությունների կողմից: ՀԲՎ-ն ևս կարիք ունի ստուգվելու, որպեսզի համոզվենք, որ ՀԲՎ-ում ընդգրկված ՄՍ-ները ճիշտ են աշխատում միմյանց հետ՝ ապահովելով ՀԲՎ-ի ընդհանուր ֆունկցիոնալությունը: Ընդ որում ստուգելու համար հնարավոր է օգտագործել միայն ՀԲՎ-ի մուտքերը և ելքերը: Այդ նպատակին հասնելու համար նախ և առաջ բոլոր ՄՍ-ները պետք է բավարարեն նախամշակված ստանդարտի կամ պարփակվեն որոշակի պատյանով՝ այն ապահովելու համար: Այդպիսի ստարդարտի օրինակ է հանդիսանում IEEE1500 “Ներդրված միջուկների թեստավորման ստանդարտ”-ը[4]:

[1] Dupont, E., “Embedded Robustness IP”, IEEE Design & Test, Vol 19, No 3, May-June, 2002.

[2] Zorian, Y. ; Shoukourian, S., “Embedded-memory test and repair: infrastructure IP for SoC yield”, Design & Test of Computers, IEEE Volume: 20, Issue: 3 Digital Object Identifier: 10.1109/MDT.2003.1198687, Publication Year: 2003 , Page(s): 58-66

[3] Shoukourian, S.; Vardanian, V.; Zorian, Y, “SoC Yield Optimization via an Embedded-Memory Test and Repair Infrastructure”, IEEE Design & Test, v.21 n.3, p.200-207, May 2004

[4] http://grouper.ieee.org/groups/1500/

Հաջորդ քայլում ՀԲՎ-ի կառուցվածքում պետք է ընդգրկվեն թեստավորման և վերանորոգման ներկառուցվածքներ (ԹՎՆ): ԹՎՆ-ներն ապահովում են արդյունավետ թեստավորում և արատների վերանորոգում:

Քանի որ, ԹՎՆ-երը բազմաշերտ և բարդ համակարգեր են և նախատեսված են բազմաթիվ լուծումներ ապահովելու համար, ապա որոշակի ՄՍ օրինակ ստանալը նախագծողից պահանջում է այդ համակարգերի մանրամասն իմացություն և հմտություն: Հաշվի առնելով նախագծման ժամանակի սղությունը, նպատակահարմար է ունենալ այնպիսի լեզու, որը հնարավորություն կտա ավելի բարձր մակարդակում նկարագրել տվյալ կոնֆիգուրացիային համապատասխանող պարամետրերը ու հիերարխիան՝ որպես նախագծման աշխատանքային հոսք (ԱՀ), որը կանվանենք ԱՀ նկարագրման լեզու (ԱՀՆԼ)[5]: Բնական է տրամադրել նախագծողին այնպիսի միջավայր, որը թույլ կտա իրականացնել շարահյուսական վերլուծություն, ֆորմալ վերիֆիկացիա, ԹՎՆ-ի գեներացիա համապատասխան գործիքային միջավայրերի հրամանների հաջորդականությամբ և մոդելավորում: Փորձը ցույց է տալիս, որ գոյություն ունեն հաճախ օգտագործվող ԱՀ-եր, որոնք կարելի է ընդհանրացնել կաղապարների տեսքով: Սովորաբար առանձնացվում են մի քանի հիմնական կաղապարներ, որոնցից օգտվելով ՀԲՎ նախագծողը կարող է ստանալ իր պահանջներին համապատասխանող ընդհանրական ԱՀ՝ հիմք ընդունելով որոշ հենքային կաղապարներ և նրանց կոմբինացիաներ:

Հաշվի առնելով վերը նշվածը, ավելի քան կարևորվում է ունենալ հաճախ օգտագործվող ԱՀ-երի կաղապարների գրադարան: Ընդ որում օգտագործողը պետք է կարողանա կատարել փոփոխություններ ԱՀ-ների կաղապարներում, կառուցել նորերը, միավորել և ստուգել այդ ամենը՝ օգտվելով որոշակի գործիքային միջավայրից: Արդիական է այնպիսի արդյունավետ մեթոդների որոնումը, որը կկիրառվի ԱՀ կաղապարների փոփոխությունների ստուգման համար՝ ապահովելով ՀԲՎ նախագծողներին կայուն և ստուգված համակարգով:

Աշխատանքի նպատակն ու խնդիրները

Աշխատանքի նպատակն է.

- առաջարկել լեզու RTL կոմպիլյատորների ԱՀ-երի և նրանց կաղապարների նկարագրման համար,
- մշակել գործիքային միջավայր ԱՀ-երի և նրանց կաղապարների նկարագրման, շարահուսական վերլուծության, ֆորմալ վերիֆիկացիայի, ԹՎՆ գեներացիայի և օրինակի մոդելավորման համար,
- կատարել փորձարկումներ, որոնք կճշտեն առաջարկված մոտեցման կիրառման ոլորտը և կհաստատեն այդ մոտեցման արդյունավետությունը:

[5]http://www.synopsys.com/IP/SRAMandLibraries/TestandRepair/Pages/default.aspx

Aram Khzarjyan

Justification and development of software tools for RTL compilers usage automation

RESUME

Every new semiconductor technology node provides further miniaturization and higher performance, thus increasing the functions that electronic products could offer. Although adding such new functions do benefit the end-user, but they also necessitate finer and denser semiconductor fabrication processes, which make chips more susceptible to defects. Today's very deep-submicron semiconductor technologies of 130 nanometers and below are reaching defect susceptibility levels that result in lowering the manufacturing yield and reliability, and hence lengthening the production ramp-up period.

Very deep sub-micron technology makes it possible to increase the transistor count on a die and allows packing more complex circuits in one chip. A system is composed of several components that intiailly have been placed in different chips. Now it is possible to place these components in one chip called SoC. The main advantages of using SoC are miniaturization of the device size; product cost reduction, reduced Time to Market. Generally SoC contains various cores that could be designed by different IP vendors. Cores can be customized and reused in many SoC designs requiring standard interfaces for these cores. One of well-known standards is IEEE 1500.

SoC computational power and embedded memory size has to be increased to be enough for memory-hungry applications. As a result, embedded memory becomes the major component of SoC that will occupy more than 94% SoC area in the year 2014. From manufacturing yield point of view, embedded memories are more inclined to defects than other SoC components. There are IP cores for built-in self test and repair solutions to solve this problem. STAR Memory System (SMS) is one of well-known solutions which is a complete set of infrastructural IP compilers and supporting software tools. STAR Hierarchical System (SHS) is an infrastructural IP solution for built-in test and repair engines of IP blocks. Now both SMS and SHS are widely adopted by a variety of customers exploiting different development flows.

To cover the diversity of requests when providing the user support, it is necessary to define a language to be used to implement a custom use flow of SMS and SHS designs. Taking into account the diverisity of different development flows, a new approach is suggested based

- Կառուցվել է RTL կոմպիլյատորների հենքային օգտագործման ԱՀ-երի կաղապարների գրադարանը[5, 6],
- Ցույց է տրվել, որ RTL կոմպիլյատորների ԱՀ-երի կաղապարների գրադարանի տարրերը արտապատկերվում են ԱՀ ֆորմալ մոդելին [4, 5, 6],
- Հիմնավորվել է ԱՀ ֆորմալ վերիֆիկացիայի ալգորիթմի կիրառելիությունը ԱՀ-երի կաղապարների գրադարանի և նրանում կատարված փոփոխությունների ստուգման համար [5, 6],
- Մշակվել է գործիքային ծրագրային միջավայր ԱՀ-երի կաղապարների գրադարանի նկարագրման, նրա միջոցով նոր ԱՀ-երի կառուցման և նրանց ստուգման համար [1, 2, 3],
- Կատարված փորձակումները հաստատել են մոտեցման կենսունակությունը [4,5, 6]:

Թեմայի շրջանակներում հրապարակված աշխատություններ

1. Ա. Խզարջյան, “Ներդրված հիշողությունների կիրառման եվ շահագործման ավտոմատացման միջոցներ” ՀՊՃՀ Լրաբեր-75, էջեր 313-316, գիտական և մեթոդական հոդվածների ժողովածու մաս 1, Երևան 2008:
2. A. Khzarjyan, “A Tool for SoC Test Network Design Automation”, Published in Proceedings of "7th International Conference on Computer Science and Information Technologies" (CSIT’2009), pp. 443-446, September 2009, Yerevan, Armenia
3. A. Khzarjyan, “A Language for Building a Use Flow of STAR Memory System and Its Template Based Implementation”, Published in Proceedings of "8th International Conference on Computer Science and Information Technologies" (CSIT’2011), pp. 443-446, September 2011, Yerevan, Armenia
4. A. Khzarjyan, “AN APPROACH OF STAR MEMORY SYSTEM USE FLOW AUTOMATION AND ITS VERIFICATION”, ՀԳԱԱ ԵՎ ՀՊՃՀ ՏԵՂԵԿԱԳԻՐ էջեր 163-170, Տեխնիկական գիտությունների սերիա 2, հատոր LXVI, 2013
5. A. Khzarjyan, “Use-Flow Diversity and Toolset for Test and Repair of Embbeded Memory”, Published in Proceedings of "9th International Conference on Computer Science and Information Technologies" (CSIT’2013), pp. 345-348, September 2013, Yerevan, Armenia.
6. A. Khzarjyan, “A Toolset for Easy Development of Test and Repair Infrastructure for Embedded Memories”, IEEE Xplore digital library, ISBN 978-1-4799-2460-8, pp 1-7, Sept. 2013,http://dx.doi.org/10.1109/CSITechnol.2013.6710328

Հետազոտման օբյեկտները

Հետազոտության օբյեկտներ են հանդիսանում ՀԲՎ նախագծման ԱՀ-երը, RTL կոմպիլյատորների միջոցով ԹՎՆ ՄՍ-երի սպասարկման միջավայրերը և նրանց հետ փոխհամագործակցելու լեզվային միջոցները:

Հետազոտման մեթոդները

ՀԲՎ նախագծման մեթոդներ, ներդրված թեստավորման և վերանորոգման մեթոդները, RTL մակարդակի նախագծման մեթոդները, ֆորմալ վերիֆիկացիայի և գրաֆների տեսության մեթոդները:

Արդյունքների գիտական նորույթը

Աշխատանքի գիտական նորույթ է համարվում հետևյալը.

- Առաջարկվել է ԱՀՆԼ RTL կոմպիլյատորների ԱՀ-երի և նրանց կաղապարների նկարագրման համար,
- Մշակվել է ընդհանրական օգտագործման գրադարանը՝ բաղկացած հենքային ԱՀ-երի կաղապարներից,
- Առաջարկվել է հենքային ԱՀ-երի կաղապարների կոմբինացիայի միջոցով կառուցված ընդհանրական ԱՀ՝ որպես ԹՎՆ նախագծման ընդհանրական ԱՀ,
- Հիմնավորվել է, որ ԱՀ-երի ֆորմալ վերիֆիկացիայի ալգորիթմներից մեկը կարելի է կիրառել ԱՀՆԼ-ով սահմանված ԱՀ-երի ստուգելու համար:

Ստացված արդյունքների կիրառական նշանակությունը

Իրականացված է գործիքային միջավայր ԱՀ-երի կաղապարների գրադարանի նկարագրման, նրա միջոցով նոր ԱՀ-երի կաղապարների կառուցման և նրանց ստուգման համար: Այս գործիքային միջավայրի օգնությամբ հնարավոր է ստեղծել օգտագործողի սեփական պահանջներին համապատասխանող ստուգված ԱՀ-երի կաղապարների գրադարան: Ունենալով ստուգված ԱՀ-երի կաղապարների գրադարան, կաղապարում փոփոխություններ կատարելու հնարավորություն և ստուգելով ստացված ԱՀ-ի ճշտությունը, օգտագործողը ձևավորում է իր սեփական գրադարանը:

Տվյալ միջավայրը օգտագործվել է 60-ից ավել կազմակերպությունների՝նախագծերում:

Ներդրումներ

Աշխատանքի արդյունքների գործնական օգտագործումը և նրանց արժեքը արտացոլված են «Սինոփսիս Արմենիա ՓԲԸ» ընկերությունում ներդրման համապատասխան արձանագրությունում:

Պաշտպանությանը ներկայացվում են հետևյալ դրույթները

- RTL կոմպիլյատորների ԱՀ-երի և նրանց կաղապարների նկարագրման լեզուն,

- Ընդհանրական օգտագործման գրադարանը՝ բաղկացած հենքային ԱՀ-երի կաղապարներից
- Հենքային ԱՀ-երի կաղապարների կոմբինացիայով կառուցված ընդհանրական ԱՀ-երի և նրա ձևափոխման մոտեցումը՝ ԹՎՆ–ի կառուցման համար
- ֆորմալ վերիֆիկացիայի ալգորիթմներից մեկի կիրառելիությունը RTL կոմպիլյատորների ԱՀ-երի կաղապարների ստուգման համար:

Ստացված արդյունքների ապրոբացիան

Թեկնածուական աշխատանքի հիմնական արդյունքները և դրույթները քննարկվել և ներկայացվել են ՀՊՃՀ Լրաբեր-75 գիտական և մեթոդական հոդվածների ժողովածուի մաս 1-ում 2008թ (Երևան, Հայաստան), Համակարգչային գիտություն և ինֆորմացիոն տեխնոլոգիաներ կոնֆերանսին CSIT'2009 (Երևան, Հայաստան), Համակարգչային գիտություն և ինֆորմացիոն տեխնոլոգիաներ կոնֆերանսին CSIT'2011 (Երևան, Հայաստան), ՀԳԱԱ ԵՎ ՀՊՃՀ ՏԵՂԵԿԱԳԻՐ, Տեխնիկական գիտությունների սերիա 2, հատոր 67, 2013 (Երևան, Հայաստան), Համակարգչային գիտություն և ինֆորմացիոն տեխնոլոգիաներ կոնֆերանսին CSIT'2013 (Երևան, Հայաստան), IEEE ժողովածուում ընտրված լավագույն 40 հոդվածների շրջանակում:

Աշխատանքի հիմնական արդյունքները հրապարակված են 6 հոդվածներում, որոնց ցանկը բերված է սեղմագրի վերջում:

Աշխատանքի կառուցվածքն ու ծավալը

Ատենախոսությունը բաղկացած է առաջաբանից, չորս գլուխներից և օգտագործված գրականության ցանկից: Աշխատանքի ծավալը 109 էջ է, օգտագործված գրականության ցանկն ընդգրկում է 64 անուն:

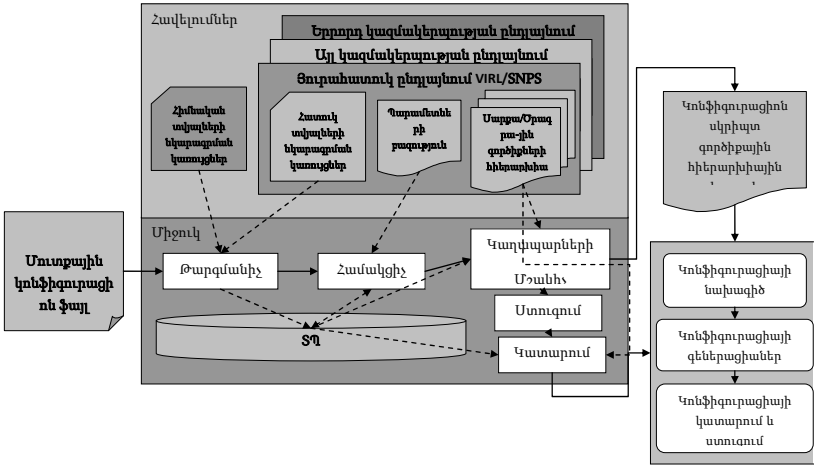
ԱՇԽԱՏԱՆՔԻ ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆԸ

Առաջաբանում հիմնավորված է արդիականությունը, ձևակերպված է աշխատանքի նպատակը: Բերված է աշխատանքի համառոտ նկարագրությունն ըստ գլուխների:

Առաջին գլխում նկարագրված է Համակարգ Բյուրեղի Վրա (ՀԲՎ) հասկացությունը, նրա նախագծման և վերիֆիկացիայի հետ կապված խնդիրները, RTL կոմպիլյատորների հասկացությունը և նրանց կիրառումը, RTL կոմպիլյատորների ԱՀ-երի հասկացությունը, նրանց նախագծման և կիրառման վերիֆիկացիայի խնդիրները:

§1.1–ում դիտարկված է ՀԲՎ-ների արտադրման հետ կապված խնդիրները, Մտավոր Սեփականության (ՄՍ) բաղադրիչների և ենթակառուցվածքային ՄՍ բաղադրիչների կիրառումը ՀԲՎ-ներում, ներդրված հիշողության ՄՍ-ների

§4.1- ում ներկայացվում է RTL կոմպիլյատորների ԱՀ-երի մշակման համակարգի ճարտարապետությունը (Նկար 14):



Նկար 14: Գործիքի ճարտարապետությունը

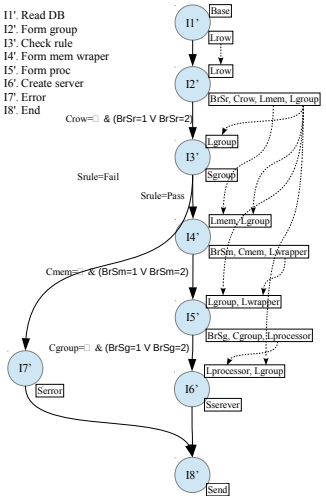
§4.2 ում ներկայացված է գործիքի կիրառման արդյունքները: Բազմաթիվ հաճախորդների մոտ ներդրման արդյունքում է 1-3 ամիս տևող գործընթացները կրճատվել են մինչև 1-3 շաբաթ: Հաճախորդներից մեկի օգտագործմանը համապատասխանեցված գեներացիայի ԱՀ-ի համար (Աղյուսակ 1):

Աղյուսակ 1

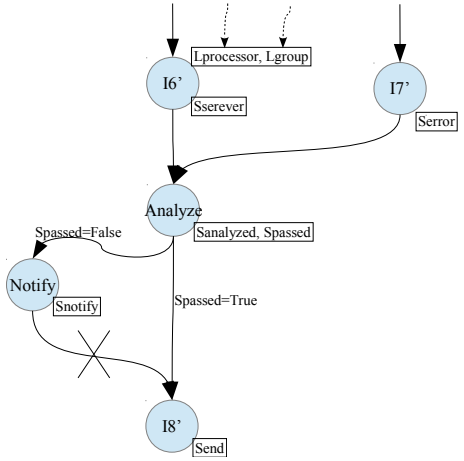
Գեներացիայի ԱՀ-ի օրինակ	Ֆորմալ վերիֆիկացիա	Կատարման տևողությունը	Սիմուլյացիա	Կատարման տևողությունը
GEN1	Correct	2.41 min	Correct	29.1 min
GEN2	Incorrect	2.53 min	Incorrect	32 min

Հիմնական արդյունքներն ու եզրակացությունները

- Սահմանվել է RTL կոմպիլյատորների ԱՀ-երի կաղապարների նկարագրման ԱՀՆԼ լեզուն [2, 3, 4],
- Լեզուն ընդլայնվել է, որ նկարագրի RTL կոմպիլյատորների շերտի վրա հիմնված որոշ ներդրված կիրառությունները [3, 4],
- Կառուցվել է ԱՀ-ի գեներացման ալգորիթմը ըստ ԱՀՆԼ նկարագրության և պարամետրերի արժեքների [4, 5, 6],



Նկար 12: Կրճատված ԱՀ-ը:



Նկար 13: Փոփոխված ԱՀ-ի հատվածը:

Կիրառելով ֆորմալ վերիֆիկացիայի ալգորիթմը կհամոզվենք, որ փոփոխված ԱՀ-ը կոռեկտ չէ[7]: Հեռացնելով Notify և End օպերատորների միջև առկա ղեկավարման կոնեկտորը՝ կհամոզվենք, որ այն կոռեկտ է հետևյալ փոփոխված կատարման վերջնապայմանի նկատմամբ.

PostC = (Serror = TRUE OR Sserver = TRUE) AND Sanalyzed =TRUE AND ((Spassed =TRUE AND Send = TRUE) OR (Spassed =FALSE AND Snotify = TRUE)):

Նշված ֆորմալ վերիֆիկացիայի ալգորիթմը կիրառվել է RTL կոմպիլյատորների ԱՀ-երի գրադարանի հիմնական ԱՀ-երի համար և ցույց է տրվել, որ դրանք կոռեկտ են:

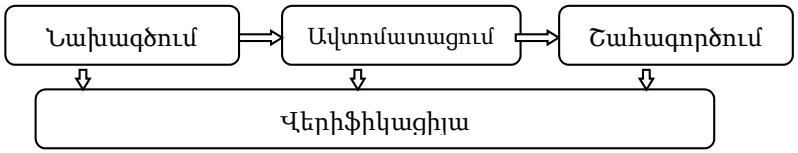
Չորրորդ գլխում նկարագրված է RTL կոմպիլյատորների ԱՀ-երի ավտոմատացված շահագործման և վերիֆիկացիայի համար մշակված գործիքային միջավայրերը: Ապացուցվել է նախագծման շաբլոնների ճշտությունը բազային ինվարիանտների նկատմամբ՝ մշակված գործիքային միջավայրի օգնությամբ: Կատարվել են փորձարկումներ գործիքների օգնությամբ:

ազդեցությունը ՀԲՎ-ների օգտակար ելքի վրա և այդ ազդեցության փոքրացման միջոցները:

§1.2-ում դիտարկված է ՀԲՎ-ում ներդրված հիշողության ՄՍ-ների թեստավորման ենթակառուցվածքային ՄՍ-ների տեսակները, RTL կոմպիլյատորների օգտագործումը ՄՍ-ների գեներացիայի համար, Սթար Հիշողության Համակարգը:

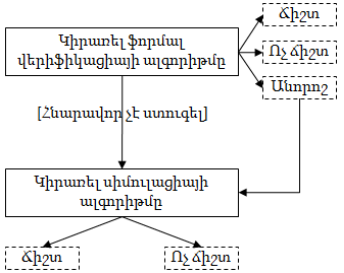
§1.3-ում դիտարկված է RTL կոմպիլյատորների դերը ՀԲՎ-ների նախագծման քայլերում, այդ քայլերի կատարման համար ԱՀ-երի մշակման խնդիրները:

§1.4-ում դիտարկված է RTL կոմպիլյատորների կիրառման և վերիֆիկացիայի խնդիրները: Նկարագրված է RTL կոմպիլյատորների ԱՀ-երի կենսացիկլը (Նկար 1) իր վերիֆիկացիայով և 3 փուլերով՝ նախագծում, ավտոմատացում, շահագործում:



Նկար 1: Վերիֆիկացիան RTL կոմպիլյատորների կենսացիկլում:

Նախագծման փուլում ՄՍ մատակարար կազմակերպությունները պետք է վերլուծեն և նախագծեն իրենց ԱՀ-երը: ԱՀ-երի մշակումը կատարվում է, հիմնականում, նախագծման կաղապարների գրադարանների հիման վրա: Հետևաբար, այս եղանակով աշխատելիս առաջնային է դառնում կաղապարների փոփոխությունների արդյունքում ստացված ԱՀ-երի վերիֆիկացիան: ԱՀ-երի ֆունկցիոնալ վերիֆիկացիան նախատեսված է ստուգելու համար ԱՀ-ի վարքագծային/ֆունկցիոնալ ճշտությունը, որը կարող է կատարվել ֆորմալ վերիֆիկացիայի կամ սիմուլացիայի միջոցով (Նկար 2): Աշխատանքում առաջարկվում է ֆորմալ վերիֆիկացիայի մոտեցումը և ցույց է տրվում, որ ֆորմալ վերիֆիկացիայի խնդիրը լուծելի է ԱՀ-երի կաղապարների դասի համար:



Նկար 2: ՄՍ ԱՀ-երի ֆունկցիոնալ վերիֆիկացիայի մոտեցումը

Այլ կերպ ասած, այդ պրոցեսների համար անհրաժեշտություն չի առաջանում կիրառել սիմուլյացիա:

Երկրորդ գլխում նկարագրված է RTL կոմպիլյատորների ԱՀ-րի լեզվային նկարագրման հիմնավորումը, նկարագրման հիմնական տարրերը, RTL կոմպիլյատորների ԱՀ-երի կիրառումը, լեզվի ձևայնական ներկայացումը և նկարագրման ու կիրառման օրինակներ:

§2.1-ում հիմնավորվում է RTL կոմպիլյատորների ԱՀ-երի նկարագրման լեզվի սահմանման անհրաժեշտությունը: Հիմնական պատճառը կայանում է նրանում, որ տարբեր RTL կոմպիլյատորների մատակարարների կողմից տրամադրված ՄՄ-ները պետք է օգտագործվեն մեկ ընդհանրական ԱՀ-ի շրջանակներում: Այդ իսկ պատճառով պետք է լինի միջոց որով կարողանանք նկարագրել ՄՄ բաղկացուցիչները և նրանց միջև կախվածությունները և կապերը: Քանի որ առկա ՄՄ բաղկացուցիչները բավականաչափ տարբեր են, ապա նպատակահարմար է ունենալ մեկ լեզվային միջոց այդ ամենը նկարագրելու և ինտեգրելու համար: Այդ լեզուն պետք է լինի հնարավորին չափ ընդհանրական, ճկուն և ընդլայնվելի:

Տվյալ պահանջներին բավարման տարբերակներից մեկն է այս աշխատանքի շրջանակներում առաջարկված ԱՀՆԼ: Այն ապահովում է ըստ ՄՄ մատակարարների RTL կոմպիլյատորների գրադարանների նկարագրություն, ԱՀ-երի նկարագրում և ձևափոխելիություն, հիերարխիկ տվյալների դասակարգում:

§2.2 - ում նկարագրվում է, թե ինչպես կարելի է ԱՀՆԼ-ի միջոցով սահմանել RTL կոմպիլյատորների հիմնական տարրերը:

ԱՀՆԼ-ի հիմնական սկզբունքն է՝ <<Ամեն ինչ տարր է>>: Ցանկացած տարր կարող է լինել պարզ (ատոմիկ) կամ բարդ, որը կարող է պարունակել պարզ և բարդ տարրերի ներդրված հիերարխիա առանց խորության կամ բարդության սահմանափակման: Նկար 3 ում բերված է բարդ տարրի նկարագրությունը

<*complex_element*> {<*simple_item*>} **‘{‘**
‘**class’** <*vendor*>’.’<*IP_compiler*> [(‘**;**’ | ‘**n**’)]
| {{<*simple_element*>}
| {{<*complex_element*>}}
‘}’ [(‘**;**’ | ‘**n**’)]

Նկար 3: Բարդ տարր

Ներմուծված դասակարգման հնարավորությունը թույլ է տալիս աջակցել միևնույն տեսակի, բայց տարբեր մատակարարների ՄՄ-ները իրենց առանձնահատկություններով: RTL կոմպիլյատորների հիերարխիան կարելի է ներկայացնել երկու ձևով: Առաջին տարբերակը գծային կառուցվածքն է, երբ

Որպես ֆորմալ վերիֆիկացիայի կիրառման օրինակ դիտարկենք Նկար 14-ում բերված գեներացիայի ԱՀ-ի կաղապարը, որն ամենահաճախ օգտագործվող ԱՀ-ն է: Վերիֆիկացիայի համար անհրաժեշտ է սահմանել համապատասխան սկզբնարժեքավորման նախապայմանը և կատարման վերջնապայմանը[13]՝ ելնելով ԱՀ-ի տրամաբանությունից: Այդպիսի հատկություններից է այն, որ գեներացիան չի կարող կատարվել առանց DB-ի: Նախապայմանը պետք է սահմանվի այն ապահովելու համար:

PreC = i(I1).Base ≠⊥ , որտեղ ⊥ նշանակում է անորոշ արժեք:

Գեներացիան պետք է ավարտվի և հաջող ավարտի, և սխալ ընթացքի դեպքում: I13 օպերատորի Serror ելքային փոփոխականի TRUE արժեքը նշանակում է, որ որևէ քայլում սխալ է տեղի ունեցել, իսկ I14 օպերատորի Sserver ելքային փոփոխականի PASS արժեքը նշանակում է, որ գեներացիան բարեհաջող ավարտվել է: Կատարման վերջնապայմանը պետք է ապահովի այդ տրամաբանության ստուգումը.

PostC = (Serror = TRUE OR Sserver = PASS) AND Send = TRUE

Ֆորմալ վերիֆիկացիայի ալգորիթմի կիրառումը կհայտնաբերի ինտերվալային ցիկլ, որը ալգորիթմի կիրառման առաջին քայլում[13] կկրճատվի ացիկլիկ ԱՀ-ի (Նկար 12): Հաջորդ քայլում կկիրառվի ացիկլիկ ԱՀՖՄ-ների ֆորմալ վերիֆիկացիայի ալգորիթմը[13], որը ցույց կտա, որ այն կոռեկտ է:

Այժմ ցույց տանք, թե ինչպես կարող է օգտագործողը փոխել ԱՀ-ը և կիրառել ֆորմալ վերիֆիկացիայի ալգորիթմը նրա համար: Դիցուք ավելացվել են Analyse և Notify լրացուցիչ օպերատորները՝ համոզվելու համար, որ սխալ կատարման դեպքում օգտագործողը կտեղեկացվի այդ մասին (Նկար 13): Նոր կատարման վերջնապայմանը պետք է ստուգի դա.

PostC = (Serror = TRUE OR Sserver = PASS) AND Sanalyzed=TRUE AND Spassed=TRUE AND Send = TRUE

միավորման և Cycle ցիկլիկ գործողությունները, որոնք օգտագործվում են ԱՀՖՄ նախագծման շաբլոնների կառուցման համար[11]: Պրիմիտիվները լինում են հետևյալը.

- *Պարզ պրիմիտիվ* u-ն մի օպերատորից բաղկացած պարզագույն պրոցեսն է:
- *Սինխրոնիզացիայի պրիմիտիվ* u-ն հանդիսանում է կոնեկտորների միավորման հանգույցը մոդելավորող պարզագույն պրոցեսը `
- *Ճյուղավորման պրիմիտիվ* u–ն հանդիսանում է ելքային կոնեկտորների ճյուղավորման հանգույցը մոդելավորող պարզագույն պրոցեսը`

Նշ. BU – պրիմիտիվների բազմությունը,

Սահմանում(RTL կոմպիլյատորների ԱՀ-ի կաղապար - RTL ԱՀԿ)

1. $\forall u \in BU$ RTL ԱՀԿ է:

2. $\forall P=Op_R(Op_{R-1}(\dots Op_1(BU')\dots)),R>0$, որտեղ $Op_i \in \{Concat,Merge\},1 \le i \le R \Rightarrow P \in$ RTL ԱՀԿ:

3. $P=Cycle_R(Cycle_{R-1}(\dots Cycle_1(P_1)\dots))$, $R>0$, որտեղ P1-ը ցիկլ չի պարունակում (միայն բավարարում է սահմանման 1. և 2. կետերին) => P RTL ԱՀԿ է:

4. Այլ RTL ԱՀԿ չկա:

Նշ. RT=<BU, Concat, Merge, ,Cycle>-ով բոլոր RTL ԱՀԿ-ների բազմությունը:

Սահմանում(բերելի բիզնես պրոցես)[7] Բիզնես պրոցեսը կանվանենք բերելի, եթե բավարարվում են հետևյալ պայմանները.

a. $\forall A \in N$ և $A' \in N$ օպերատորներն համաձայնեցված են
b. $\forall A \in N;\ |A^{\rightarrow}|>1 \Rightarrow |A^{\leftarrow}|=1$
c. $\forall A \in N;\ |A^{\leftarrow}|>1 \Rightarrow \forall A' \in A^{\leftarrow}(|A'^{\rightarrow}|=1 \ \& \ \forall A'' \in A'^{\leftarrow};\ |A''^{\rightarrow}|=1)$
d. $\forall \sigma \in \Omega, G \in K_\sigma, <G,D,t_1> \in E, D \notin K_\sigma \Rightarrow \exists e' \in E;\ t_2=\pi_3(e'),\ \pi_2(e') \in K_\sigma,\ t_1=t_2$

Պնդում 2: $\forall R \in RT$ RTL ԱՀԿ բերելի պրոցես է:

Պնդում 3: RTL կոմպիլյատորների ԱՀ-ի կաղապարների գրադարանի $\forall r \in R$ հանդիսանում է RTL ԱՀԿ:

Թեորեմ 1: $\forall$P RTL ՀԿ ֆորմալ վերիֆիկացիայի խնդիրը լուծելի է α-ի և β-ի նկատմամբ, որտեղ α-ն և β-ն հանդիսանում են P-ի սկզբնարժեքավորման նախապայմանը և կատարման վերջնապայմանը, համապատասխանաբար:

Թեորեմ 2: RTL կոմպիլյատորների ԱՀ-երի գրադարանի հիմնական ԱՀ-եր կոռեկտ են սկզբնարժեքավորման նախապայմանի և կատարման վերջնապայմանի նկատմամբ:

կախվածությունները իրականացված են հղումների միջոցով: Երկրորդ տարբերակը ներդրված կառուցվածքն է, երբ հիերարխիան ներդրված կառուցվածք է: Նկար 4-ում բերված է գծային կառուցվածքի օրինակ:

```
element IP_type1 {
        class vendor.ip_compiler1;
        element ip_type2;
        ...
};
element IP_type2 {
        class vendor.ip_compiler2;
        element ip_type3;
        ...
};
...
```

Նկար 4: RTL կոմպիլյատորի գծային կառուցվածք

```
ip_type ip_block_name {
        class vendor.ip_compiler_name;

        parameter1 value1;
        ...
};
```

Նկար 5: RTL կոմպիլյատորի օգտագործման օրինակ

§2.3 - ում սահմանվում են ԱՀՆԼ այն տարրերը, որոնց միջոցով հնարավոր է ներկայացնել RTL կոմպիլյատորների կամայական ԱՀ: Այդ ԱՀ-երի նկարագրման համար որպես հիմք վերցվում են նախօրոք սահմանված RTL կոմպիլյատորների կառուցվածքները:

Դիտարկենք այն ՄՄ-ների ներկայացման օրինակների համար (Նկար 5,6), որտեղ առաջին տեղում ՄՄ-ի տեսակն է այնուհետև անվանումը համապատասխանաբար ip_type և ip_block_name: Այնուհետև տվյալ ՄՄ-ի դասակարգումն է ըստ մատակարարի և RTL կոմպիլյատորի և հետո նրան առանձնահատուկ պարամետերը իր արժեքով:

Մյուս օրինակում բերված է RTL կոմպիլյատորների հիերարխիա, որտեղ ip_block_name1-ի ենթաբաղադրիչ է հանդիսանում ip_block_name2-ը: ՀԲՎ-ի ամբողջական նկարագրությունը իր մեջ ներառնում է բոլոր ՄՄ-ները (Նկար 7):

```
ip_type1 ip_block_name1 {
        class vendor.ip_compiler_name;

        subblock ip_block_name2;
        ...
};

ip_type2 ip_block_name2 {
        class vendor.ip_compiler_name;

        subblock ip_block_name3;
        ...
};
...
```

Նկար 6: RTL կոմպիլյատորների հիերարխիայի օրինակ

```
soc soc_name {
        ip_blocks {
                ip_block1 ...
                ip_block2 ...
                …
        }
        …
};
```

Նկար 7: ՀԲՎ-ի օրինակ

§2.4 - ում նկարագրվում է ներկայացվող լեզվի Բեկուս Նաուրի Ընդլայնված Ձևը (ԲՆՁ): Որտեղ բերված են հիմնական տարրերի ներկայացումը: Ամբողջական ներկայացումը բերված է ատենախոսության շրջանակներում:

§2.5 - ում նկարագրվում է RTL կոմպիլյատորների հիերարխիային կոնկրետ կիրառումը ներդրված հիշողության համար: Այս օրինակում սահմանված են տարբեր մատակարարներից երկու ներդրված հիշողության ՄՍ-ներ, որոնց համար սահմանված են համապատասխան ենթակառուցվածքային ՄՍ-ներ (Նկար 8):

```
memory mem1 {
    class vendor1.compiler1;
    NW 1024;
    NB 23;
    CM 8;
};
memory mem2 {
    class vendor2.compiler2;
    NW 512;
    NB 36;
    BK 16;
};
wrapper wr1 {
    class vendor3.compiler3;
    memory mem2;
};
wrapper wr2 {
    class vendor3.compiler4;
    memory mem1;
    FREQ 100MHZ;
};
processor proc1 {
    class vendor3.compiler4;
    wrapper {wr1 wr2};
};
server srv1 {
    class vendor3.compiler5;
    NVS efuse128;
    processor {{proc1 3}};
};
```

Նկար 8: RTL կոմպիլյատորների ԱՀ-ի նկարագրման օրինակ:

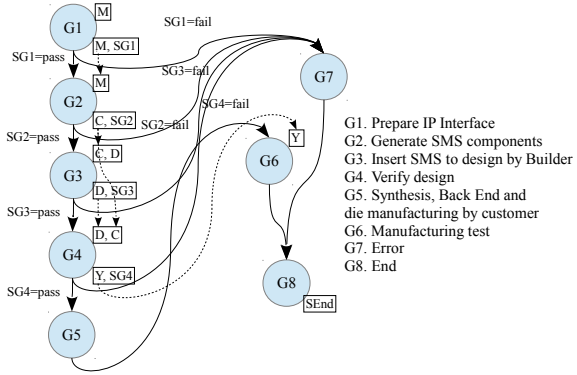
Երրորդ գլխում նկարագրված է ԱՀՆԼ-ի արտապատկերումը ԱՀ ֆորմալ մոդելին (ԱՀՖՄ), ինչպես նաև ԱՀՖՄ ֆորմալ վերիֆիկացիայի մի ալգորիթմի կիրառումը RTL կոմպիլյատորների ԱՀ-երի համար:

§3.1 - ում նկարագրվում է ԱՀՖՄ և RTL կոմպիլյատորների ԱՀ-երի արտապատկերման սկզբունքը այդ մոդելին:

Նկարագրենք IBM's MQSeries Workflow ացիկլիկ ԱՀՖՄ՝ օգտագործելով հետևյալ նշանակումները.[6]

- եթե $x=\langle x_1,\dots,x_n\rangle$ n-յակ է և $1\le i_1,\dots,i_k\le n$ ինդեքսների բազմություն է, ապա $\pi_{i1,\dots,ik}(x)$ –ով նշանակենք $\langle x_{i1},\dots,x_{ik}\rangle$ հավաքածուն:

[6] F., Roller,D. Leymann ”Production Workflow: Concepts and Techniques” Prentice Hall Press, 2000



Նկար 11: Ընդհանրական ԱՀ-ի կաղապար:

Նշ. R-ով RTL կոմպիլյատորների ԱՀ-երի կաղապարների գրադարանի տարրերի բազմությունը:

Պնդում 1: $\forall r\in R$ C-պրոցես է;

Ապացույցը բխում է այն ղեկավարման գրաֆի կառուցման ալգորիթմից, որը կառուցում է C-պրոցես ըստ սահմանման:

§3.3 - ում ձևակերպվում են RTL կոմպիլյատորների ԱՀ-ի կոռեկտության գաղափարը և լրացուցիչ սահմանումներ, որոնք անհրաժեշտ են RTL կոմպիլյատորների ԱՀ-ի ֆորմալ վերիֆիկացիայի խնդրի ձևակերպման և լուծման համար:

Սահմանում (RTL կոմպիլյատորների ԱՀ-ի կոռեկտություն): Դիցուք R-ն RTL կոմպիլյատորների ԱՀ է: Դիցուք **α** -ն պրեդիկատ է՝ կախված որոշակի մուտքային փոփոխականներից, իսկ **β** -ն պրեդիկատ է՝ կախված որոշակի ելքային փոփոխականներից (այսուհետև՝ ***հետևանք***): Կանվանենք **α**-ն և **β**-ն *նախապայման* և *վերջնապայման*, համապատասխանաբար:

R –ը կոռեկտ է α-ի և β-ի նկատմամբ $\Leftrightarrow$ [α(R)=1 և R-ն ավարտվում է $\Rightarrow$ β(R)=1]:

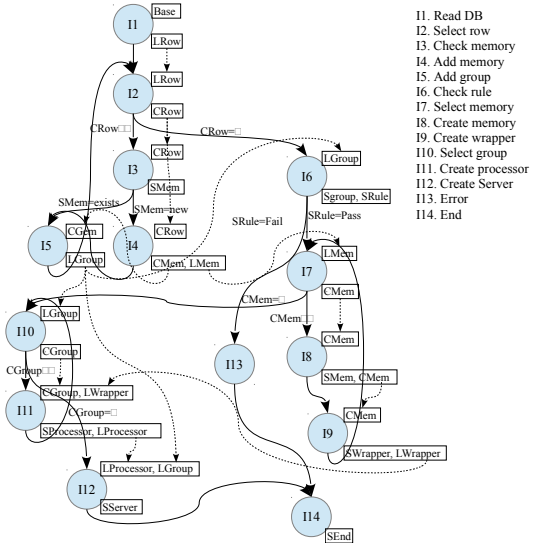
§3.4 - ում նկարագրվում է ԱՀՖՄ ֆորմալ վերիֆիկացիայի մի ալգորիթմի կիրառում RTL կոմպիլյատորների ԱՀ-երի կաղապարների համար: Այդ ալգորիթմը ացիկլիկ ԱՀՖՄ ֆորմալ վերիֆիկացիայի ալգորիթմի ընդլայնում է ցիկլիկ ԱՀՖՄ-ների համար: Այդ ալգորիթմը հիմնված է ցիկլիկ ԱՀՖՄ-ն ացիկլիկի բերման մոտեցման վրա:

RTL ԱՀ-ի կաղապարը ֆորմալ սահմանելու համար կօգտագործենք պրիմիտիվների և նրանց նկատմամբ կիրառվող Concat կոնկատենացիայի, Merge

գեներացիայի, ներմուծման, նախագծային և հետարտադրական վերիֆիկացիայի կաղապարները:

Նկար 10-ում պատկերված է SHS ստանդարտ ինտերֆեյսի ապահովման տարբեր ՄՍ-ների գեներացիայի Աշ-ի կաղապարը: Մնացած հիմնական Աշ-երի կաղապարները բերված են ատենախոսությունում:
Նշված Աշ-երի կաղապարները կարող են միակցվել մեկ Աշ-ի շրջանակում՝ բխելով հաճախորդի խնդրին հատուկ պահանջներից: Հիմնական Աշ-երից յուրաքանչյուրը կարող է դիտարկվել որպես առանձին, կամ որպես մեծ Աշ-ի բաղկացուցիչ: Խորհուրդ տրվող ընդհանրական Աշ-ի կաղապարը իր մեջ ներառում է բոլոր հիմնական Աշ-երը (Նկար 11):

Դիտողություն. Առաջարկված ընդհանրական Աշ-ի կաղապարը խորհուրդ է տրվում, բայց չի պարտադրվում: Օգտագործողը կարող է ձևափոխել այն՝ ավելացնելով նոր օպերատորներ կամ հեռացնելով արդեն գոյություն ունեցողները: Նշված Աշ-ն ունի 32 ձևափոխություն ամենաքիչը (5 հիմնական Աշ-ի կաղապար, յուրաքանյուր Աշ-ի կաղապար կարող է լինել կամ բացակայել):



Նկար 10: Գեներացիայի Աշ-ի կաղապար:

- X բազմության համար $\wp(X)$–ով նշանակենք X–ի բոլոր ենթաբազմությունների բազմությունը:

Սահմանում (ացիկլիկ պրոցես – A-պրոցես)

A-պրոցես հանդիսանում է $P=<T,N,C,V,E,\Phi,i,o,\Psi,\Delta>$ n-յակը, որտեղ

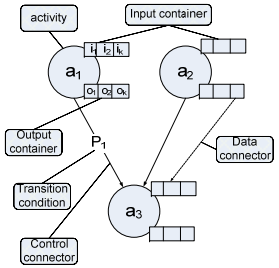
1. T -ն, N-ը, C -ն, V-ն, $E \subseteq N \times N \times C$ -ն համապատասխանաբար *տիպերի*, *օպերատորների*, *անցման պայմանների*, *փոփոխականների*, *ղեկավարման կոնեկտորների* վերջավոր բազմություններ են,
2. $\Phi: N \to F$ *միակցման պայմանների* արտապատկերում է,
3. $i: N \cup C \to V$ *մուտքային տվյալների* արտապատկերում է,
4. $o: N \to V$ *ելքային տվյալների* արտապատկերում է,
5. $\Psi: N \cup C \to E$
   a. $A \in N$, $\Psi(A): \times_{v \in i(A)} DOM(v) \to \times_{v \in o(A)} DOM(v)$:
   b. $p \in C$, $\Psi(p): \times_{v \in i(p)} DOM(v) \to \{0, 1\}$:
6. $\Delta : N \times (N \cup C) \to \cup_{A \in N,\ B \in N \cup C} \wp(o(A) \times i(B))$ *տվյալների կոնեկտորների* արտապատկերումն է

Պրոցեսի ամփոփ սահմանումը բերված է ատենախոսության մեջ:

Նշ. $A^{\to}=\{e \in E \mid \pi_1(e)=A\}$-ով և $A^{\leftarrow}=\{e \in E \mid \pi_2(e)=A\}$-ով $A \in N$ գագաթին համապատասխանաբար հաջորդող և նախորդող գագաթների բազմությունները՝ ըստ ղեկավարման գրաֆի (Նկար 9):

Ճանապարհը օպերատորների և նրանց միացնող ղեկավարման կոնեկտորների $<a_1,e_1, \ldots, a_{k-1},e_{k-1},a_k>$ հաջորդականությունն է, այնպիսին, որ $\pi_1(e_i)= a_i$ & $\pi_2(e_i)= a_{i+1}$, $1 \le i \le k-1$, $k>1$: Նշ. $path(a_1,a_k)=\{a_1,\ldots,a_{k-1},a_k\}$:

$<a_1,e_1,\ldots,,a_k,e_k,,a_1>$, $k>1$ ճանապարհը կանվանենք *ցիկլ*:



Նկար 9: Workflow պրոցեսի տարրերի գրաֆիկական ներկայացումը:

Նշ. **Ω** - P-ի բոլոր ցիկլերի բազմությունը, **Nc**-բոլոր ցիկլերի մուտքային գագաթների բազմությունը, $\mathbf{K_\sigma}$-$\sigma\in\Omega$ ցիկլին պատկանող օպերատորների բազմությունը: $\forall A\in N$ գործողության համար $\mathbf{A_\Delta^{\rightarrow}}$ ={ $D \in N \mid \Delta(A,D)\neq\varnothing$}:

Ցիկլերի հատումը սահմանվում է հետևյալ կերպ. $\sigma_1\cap\sigma_2\cap\ldots\cap\sigma_k \equiv K_{\sigma 1}\cap K_{\sigma 2}\cap\ldots\cap K_{\sigma k}$, որտեղ $\sigma_i\in\Omega$, $1<i\leq k\leq|N_C|$.

a_1 միակ մուտքային գագաթով $\sigma\in\Omega$ ցիկլը կոչվում է ***լոկալիզացված ցիկլ***, եթե $\forall\lambda\in\Omega$, $\lambda\neq\sigma$, $\lambda\cap\sigma\subseteq\{a_1\}$: (առաջին անգամ ներմուծվել է Ալենի կողմից՝ ծրագրերի օպտիմիզացիայի խնդրի դիտարկման ժամանակ[7])

$b\in N$ ***ճյուղավորման օպերատոր*** է $\leftrightarrow\exists\,\sigma\in\Omega$б $y\in$Уб иэ$\in T\setminus$Л$_\sigma$ (и$\in$Л$_\sigma$б $\pi_1(y)=$ иб $\pi_2(y)=$ иэ)Ж Նշ. BR_σ - $\sigma\in\Omega$ ցիկլի ճյուղավորման օպերատորների բազմությունը, իսկ BR - P-ի ճյուղավորման օպերատորների բազմությունը:

$\forall$ A, A' $\in$ N օպերատորները կանվանենք ***համաձայնեցված օպերատորներ***, եթե $\Delta(A, A')\neq\varnothing \Rightarrow A'\in A^{\rightarrow}$:

Յուրաքանչյուր ճյուղավորման օպերատորի համար կսահմանենք **վիճակի ռեգիստր**, որն ընդունում է հետևյալ արժեքներից մեկը.

1. 0 – ճյուղավորման օպերատորը դեռ չի կատարվել,
2. n; (n >= 1) – ճյուղավորման օպերատորը կատարվել է ճիշտ n անգամ:

Սահմանում (ցիկլիկ պրոցես – C-պրոցես)

W-պրոցեսը կանվանենք ցիկլիկ պրոցես կամ C-պրոցես, եթե

[7] A. Kostanyan, S. Shoukourian , A. Varosyan "Solvability of Formal Verification Problem for Business Process Templates", EAIA 2010, SpringSim'2010 pp. 66-74, 11-15 April 2010, Orlando, FL, USA

1. $\Omega\supseteq\varnothing$ և $\forall\sigma\in\Omega$ լոկալիզացված է,
2. $C^{c\leftarrow}(A) := \pi_3(\{e\in E \mid \pi_2(e)=A \;\&\; \pi_1(e)\in K_A\})$ – ներքին միավորման անցման պայմանների բազմությունն է
3. $C^{*\leftarrow}(A) := \pi_3(\{e\in E \mid \pi_2(e)=A \;\&\; \pi_1(e)\notin K_A\})$) –արտաքին միավորման անցման պայմանների բազմությունն է
4. $\forall A \in N_C$, $\Phi(A) \in \varphi_A^*$ ներքին միավորման պայման է, որտեղ

$$\varphi_A^* := \{ \bigwedge_{1\leq j\leq k} \bigvee_{1\leq i\leq l_j} p_i \mid p_i \in \{p', \sim p' \mid p' \in \{C^{*\leftarrow}(A)\}\}\}$$

5. $\forall A \in N_C$, $\Phi_c(A) \in \varphi_A^*$ արտաքին միավորման պայման է, որտեղ

$$\varphi_{AC} := \{ \bigwedge_{1\leq j\leq k} \bigvee_{1\leq i\leq l_j} p_i \mid p_i \in \{p', \sim p' \mid p' \in \{C^{c\leftarrow}(A)\}\}\}$$

: ԵՆթադրենք, որ $\forall A \notin N_C$, $\Phi_C(A)=0$:

6. $\forall A \in N_C$ օպերատորը կարող է կատարվել $\Leftrightarrow (\Phi(A)(i(p_1),\ldots, i(p_{nA})) = 1) \vee (\Phi_C(A)(i(q_1),\ldots, i(q_{mA})) = 1)$, $C^{*\leftarrow}(A) = \{p_1,\ldots,p_{nA}\}$, $C^{c\leftarrow}(A) = \{q_1,\ldots,q_{mA}\}$.
7. $\forall\sigma\in\Omega$, $\forall$ A, A' $\in K_\sigma \Rightarrow$ A և A' համաձայնեցված օպերատորներ են

Ցիկլի սահմանման հիմքում ընկած է LOOP կառուցվածքը[7]:

Այժմ նկարագրենք RTL կոմպիլյատորների ԱՀ-երի արտապատկերման սկզբունքը նկարագրված ԱՀՖՄ.

- <set of ip_type> -> T
- <list of ip_block> -> N
- <list of parameters for each ip_block> -> input container of an activity
- 2 օպերատորները կապված են ղեկավարման կոնեկտորով <-> համապատասխան ip_block-ներից մեկը հիերարխիկորեն պարունակում է մյուսին
- Տվյալների փոխանցումը իրականացվում է հիերարխիկորեն ներդրված բլոկին համապատասխան տվյալների փոխանցման կոնեկտորով

RTL կոմպիլյատորների ԱՀ-երի համար ղեկավարման գրաֆի կառուցման համար իրականացվել է ConstructGraph ալգորիթմը:

§3.2 - ում նկարագրվում են RTL կոմպիլյատորների հաճախ օգտագործվող ԱՀ-երի կաղապարների գրադարանի պրոցեսները՝ հիմված ԱՀՖՄ վրա:

Հաշվի առնելով հաճախորդի կողմից հաճախ օգտագործվող և խորհուրդ տրվող ԱՀ-երը՝ աշխատանքի շրջանակներում սահմանվել է RTL կոմպիլյատորների հաճախ օգտագործվող ԱՀ-երի կաղապարների գրադարանը, որը հնարավոր է ընդլայնել կամ ձևափոխել՝ օգտվելով համապատասխան գործիքային միջավայրից: Գոյություն ունեն հինգ հիմնական ԱՀ-երի կաղապարներ, որոնք են նախապատրաստման,