ДОДАТОК А

Графічний матеріал атестаційної роботи

Методи забезпечення функціонування обчислювальних комплексів на базі систем на кристалі

Атестаційна робота

другий (магістерський) рівень

Виконав:
студент гр. СПм-18-3
Дорошко А.П.

Керівник:
проф. каф. ЕОМ
Волк М.О.

Мета роботи та завдання

Метою роботи є дослідження методів створення архітектурних специфікацій систем на кристалі та їх верифікації.

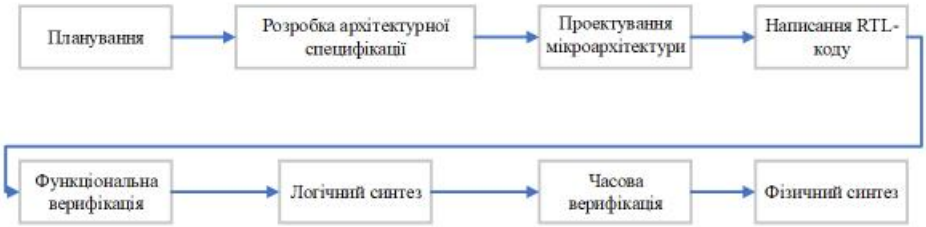
Об'єктом дослідження є методи забезпечення коректності функціонування обчислювальних комплексів на базі систем на кристалі.

Завдання:

- проведення аналізу ранніх стадій процесу розробки систем на кристалі для визначення основних наявних недоліків в цьому процесі;
- проведення порівняння різних форматів архітектурних специфікацій систем на кристалі;
- розробка методу створення та використання архітектурних специфікацій систем на кристалі.

Традиційний процес розробки ЗВІС

3



Способи вирішення вибору нових мов опису системного рівня та методології проектування

4

- ✓ Використання мов опису апаратури з деякими доповненнями, як, наприклад, розширення мови Verilog до SystemVerilog.
- ✓ Адаптація мов програмування високого рівня (C / C ++, Java) для опису архітектурних специфікацій систем на кристалі.
- ✓ Створення принципово нових мов системного рівня спеціально для проектування систем на кристалі.
- ✓ Розширення можливостей існуючих мов високорівневого моделювання та застосування їх разом з іншими мовами нижчого рівня для опису архітектурних специфікацій систем.

Сучасний процес розробки СнК

5

Проблеми сучасного процесу розробки СнК

6

- Процес розробки виконуваних моделей архітектури складних систем на кристалі загального призначення пов'язане зі значними труднощами, і доводиться починати розробку компонентів системи до готовності архітектурної моделі.
- Архітектурні специфікації в більшості своїй написані на природних мовах, і їм притаманні всі недоліки цих мов.
- Апаратне і програмне забезпечення системи найчастіше розробляється різними командами з вузькою спеціалізацією, і інтеграція їх починається тільки на пізніх етапах проектування системи. Ця ситуація веде до неможливості спільного аналізу апаратних і програмних компонентів системи на ранніх етапах проектування.
- При створенні архітектурних специфікацій системи необхідно строго стежити за інкапсуляцією компонентів системи і повним і строгим описом їх інтерфейсів.

7

Вибір відповідних форматів

1) Текст на природній мові.

2) Формальні мови.

3) Структурні діаграми.

4) Діаграми діяльності (UML-діаграми діяльності і нотації BPMN).

5) Діаграми взаємодії (UML-діаграми послідовності, комунікації, тимчасові діаграми і оглядові діаграми взаємодії, а також діаграми послідовності повідомлень).

6) Таблиці.

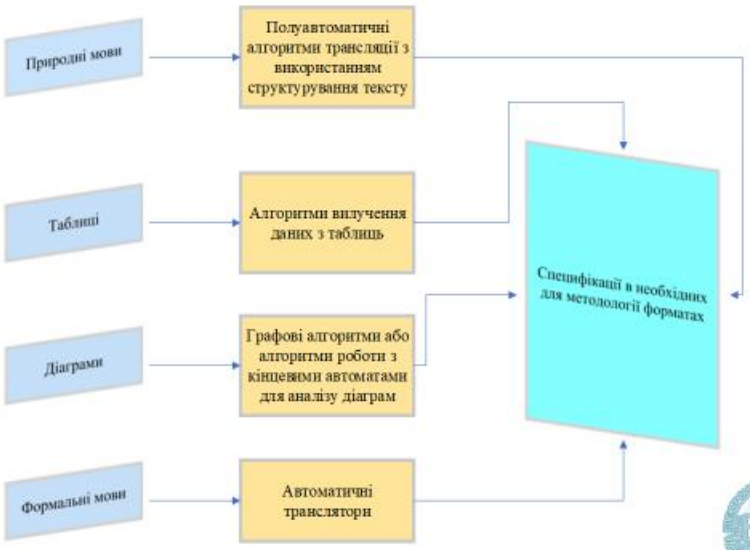
Результати аналізу форматів

8

- Більшість розглянутих форматів широко використовуються при створенні специфікацій обчислювальних систем, за винятком формальних мов. Однак, існують підходи до створення специфікацій і з використанням формальних мов.
- Складність створення специфікацій прямо пропорційна ступеню формалізації використовуваної мови. З цієї причини необхідно шукати компроміс між якістю специфікації, безпосередньо залежать від її формалізації, і складністю її створення.
- Можливості верифікації відсутні тільки у специфікацій, написаних на природних мовах, що і є однією з проблем, що вирішуються в даній роботі. Всі інші формати даних підтримують можливість перевірки коректності даних. У разі таблиць, однак, повноцінна верифікація вимагає створення таблиць по заздалегідь заданим шаблоном.
- Можливості трансляції на формальні мови також відсутні тільки у специфікацій на природних мовах. У разі таблиць для такої трансляції також необхідно заздалегідь знати структуру таблиці.
- Для опису структурних аспектів архітектури СнК підходять природні мови, формальні мови та в меншій мірі структурні діаграми.

Метод створення та використання архітектурних специфікацій СнК. Трансляція даних із специфікацій

9



Процес створення посилань між фрагментами специфікацій

10

11

Процес перевірки коректності специфікацій

12

Метод створення специфікацій та роботи з ними

1. Визначення складу і структури майбутньої специфікації.

2. Створення або вибір шаблонів даних для кожного фрагмента специфікації на основі використовуваних моделей даних.

3. Створення або вибір алгоритмів трансляції даних з існуючих форм специфікацій в використовуваній моделі даних. Цей крок необов'язковий, якщо специфікація створюється з нуля.

4. Опис класів зв'язків між елементами специфікації. Цей крок може виконуватися паралельно зі створенням специфікації.

5. Створення інструментарію для перевірки даних на цілісність і несуперечливість. Цей крок може виконуватися паралельно зі створенням специфікації.

6. Створення або вибір алгоритмів трансляції даних з використовуваних моделей на різні формальні мови. Цей крок може виконуватися паралельно зі створенням специфікації.

7. Трансляція даних з існуючих специфікацій в використовувані формати. Цей крок необов'язковий, якщо специфікація створюється з нуля.

8. Створення різних фрагментів специфікації.

9. Базова перевірка синтаксису специфікації.

10. Створення зв'язків між фрагментами специфікації.

11. Перевірка цілісності і несуперечності даних в специфікації.

12. Трансляція специфікації в різні формальні моделі.

13. Верифікація отриманих формальних моделей.

13

Результати роботи

```
SELECT bf.*, reg.ResetType FROM bitFields AS bf JOIN registers AS reg ON reg.RegisterType == bf.RegType
WHERE bf.FieldName != "-"
```

Register Field Definitions

						Reset
Comments						
FieldName	RegType	Bit Offset	Bit Width	Description	Access	ResetVal
-	PSTATE	13	51	Reserved	RO	0
tct	PSTATE	12	1	Trap on control transfer	RW	0
-	PSTATE	10	2	Reserved	RO	0
cle	PSTATE	9	1	Current little endian	RW	0
tle	PSTATE	8	1	Trap little endian	RW	0
-	PSTATE	6	2	Reserved	RO	0
-	PSTATE	5	1	Reserved (was red)	RO	0
pef	PSTATE	4	1	Enable floating-point	RW	1
am	PSTATE	3	1	Address mask	RW	0
priv	PSTATE	2	1	Privileged mode	RW	1
ie	PSTATE	1	1	Interrupt enable	RW	0
-	PSTATE	0	1	Reserved (was ag)	RO	0

Register Field Definitions

Comments							Reset
FieldName	RegType	BitOffset	BitWidth	Description	Access	ResetType	ResetVal
gl	TSTATE	40	2	Global level at previous trap level	RW	PowerUp	0
ccr	TSTATE	32	8	CCR at previous trap level	RW	PowerUp	0
asi	TSTATE	24	8	ASI at previous trap level	RW	PowerUp	0
pstate.tct	TSTATE	20	1	PSTATE.tct at previous trap level	RW	PowerUp	0
pstate.cle	TSTATE	17	1	PSTATE.cle at previous trap level	RW	PowerUp	0
pstate.tle	TSTATE	16	1	PSTATE.tle at previous trap level	RW	PowerUp	0
pstate.pef	TSTATE	12	1	PSTATE.pef at previous trap level	RW	PowerUp	0
pstate.am	TSTATE	11	1	PSTATE.am at previous trap level	RW	PowerUp	0
pstate.priv	TSTATE	10	1	PSTATE.priv at previous trap level	RW	PowerUp	0
pstate.ie	TSTATE	9	1	PSTATE.ie at previous trap level	RW	PowerUp	0
cwp	TSTATE	0	1	CWP from previous trap level	RW	PowerUp	0
tct	PSTATE	12	1	Trap on control transfer	RW	PowerUp	0
cle	PSTATE	9	1	Current little endian	RW	PowerUp	0
tle	PSTATE	8	1	Trap little endian	RW	PowerUp	0
pef	PSTATE	4	1	Enable floating-point	RW	PowerUp	1
am	PSTATE	3	1	Address mask	RW	PowerUp	0
priv	PSTATE	2	1	Privileged mode	RW	PowerUp	1
ie	PSTATE	1	1	Interrupt enable	RW	PowerUp	0

```
reg {
name = "Processor State";
regwidth = 64;
reg {
name = "Processor State";
regwidth = 64;
reg {
name = "Processor State";
regwidth = 64;
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 0;
desc = Address mask;
} am [4:3]
```

14

Висновки

В роботі розглянуті попередні стадії традиційного і сучасного процесу проектування систем на кристалі та виявлено недоліки в існуючому процесі: складність розробки повних виконуваних моделей архітектури системи на формальних мовах на основі наявної специфікації, неможливість ранньої спільної верифікації апаратного і програмного забезпечення, а також складність верифікації комплексних інфраструктурних протоколів СнК. В якості вирішення виявлених проблем запропоновано метод до розробки архітектурних специфікацій систем на кристалі і їх використання для створення високорівневих моделей системи. Для спрощення створення виконуваних моделей СнК в набір методів включений метод автоматизації перекладу специфікацій в виконувані моделі на формальних мовах; а для раннього спільного аналізу різних компонентів системи і протоколів її функціонування розроблений метод передбачає застосування різних методів верифікації даних на стадії створення специфікації.

ДОДАТОК Б

Фрагменти коду розробленої програми

```
“Ports”:
{
“field”
:
[ { “Name” : “PortName”, “Type” : “string”
},
{ “Name” :
“ShortDescription”, “Type” : “string”
},
{ “Name” : “Type”, “Type” :
“enum(Phys,Log)”
} ,
{ “Name” : “Required”,
“Type” : “bool”
},
{ “Name” : “MasterDirection”,
“Type” : “enum(in,out)”
} ,
{ “Name” : “SlaveDirection”,
“Type” : “enum(in,out)”
}
]
}
}
```

```
<Parameter
name="config_name"
default_value=""
type ="string"
is_prohibited_if="config == """>
<Description>Name of Configuration object, to identify correct
configuration in
the configuration folder.</Description>
</Parameter>
<xref name="iFlow_message_to_shape_parameters">
<description>message_shape_to_its_parameters</description>
<from type="shape">
<select>Prop.ElementType == "Connecting Object" &&
Prop.ConnectorType == "Message Flow"</select>
<key id="1">${Prop.Name}</key>
</from>
<to type="record">
<select>Template == "Message Attributes"</select>
<key id="1">${Name}</key>
</to>
</xref>
<assert>
<expr>v1 != v2</expr>
<var>
<name>v1</name>
<table_type>RegisterTable</table_type>
<column>Name</column>
<row>[1:-1]</row>
</var>
<var>
```

```
<name>v2</name>
<column>Name</column>
<row>[1:-1]</row>
</var>
</assert>
reg {
name = "Processor State";
regwidth = 64;
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 0;
desc = Interrupt enable;
} ie [2:1]
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 1;
desc = Privileged mode;
} priv [3:2]
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 0;
desc = Address mask;
} am [4:3]
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 1;
desc = Enable floating-point;
} pef [5:4]
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 0;
desc = Interrupt enable;
} ie [2:1]
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 1;
desc = Privileged mode;
} priv [3:2]
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 0;
desc = Address mask;
} am [4:3]
field {
hw = rw; sw = rw;
fieldwidth = 1;
resetsignal = PowerUp;
reset = 1;
desc = Enable floating-point;
} pef [5:4]
```