УДК 510.64:004.052

Ю. ПРОХОРОВА[1], С. ОСТРОУМОВ[1], Е. ТРУБИЦЫНА[2], Л. ЛАЙБИНИС[2]

[1]*Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина*
[2]*Åbo Akademi University, Turku, Finland*

ПРИМЕНЕНИЕ EVENT-B ДЛЯ СОЗДАНИЯ СИСТЕМ НА ПРОГРАММИРУЕМОЙ ЛОГИКЕ

Приведены особенности создания формальной спецификации систем на программируемой логике. Рассмотрены примеры описания таких систем в Event-B. Предложены элементы методики формального описания и трансляции полученной спецификации в код на языке описания аппаратуры VHDL.

Ключевые слова: *формальные методы, Event-B, платформа Rodin, ПЛИС, VHDL.*

Введение

С развитием технологии ПЛИС растет сложность и объем решаемых задач. Следствием этого является увеличение времени на тестирование и верификацию проектируемых систем. При этом не уделяется должного внимания показателям надежности, в частности функциональной безопасности, что особенно важно для информационно-управляющих систем (ИУС) и систем обработки информации (СОИ). Использование методов формальной спецификации и верификации, например Event-B, позволяет существенно сократить время тестирования и повысить показатель безопасности.

В основе Event-B лежит описание свойств – инвариантов (invariants) и событий (events) системы и проверка путем математического доказательства того, что выполнение событий (в любых допустимых состояниях) не нарушит указанные свойства. Таким образом, в процессе детализации (refinement) создается полная и непротиворечивая спецификация системы [1].

После получения спецификации, если речь идет о программном обеспечении (ПО), ее можно транслировать в код на языке высокого уровня (например, C [2] или Java [3]) с помощью существующих инструментальных средств. Однако при разработке систем на программируемых логических интегральных схемах (ПЛИС) возникают вопросы сопоставления конструкций формальной спецификации Event-B и языка описания аппаратуры VHDL. Теоретические основы трансляции, рассмотренные в [4], и анализ элементов преобразования из разновидности формальной спецификации Action Systems, приведенный в [5], не позволяют сформировать общую концепцию трансляции, а подход к формированию блоков HDL из спецификации, описанный в [6 – 8], не нашел практического применения.

Таким образом, **целью статьи** является анализ возможности применения метода формального описания для систем на ПЛИС и разработка элементов методики преобразования спецификации Event-B в код VHDL для создания транслятора как модуля открытой платформы Rodin [9].

1. Формальная спецификация в Event-B

Event-B – метод формальной спецификации и верификации системного уровня. Основными преимуществами формальной спецификации в Event-B являются применение процесса детализации (уточнения) для представления системы на различных уровнях абстракции и использование математических доказательств для проверки логичности (последовательности) между уровнями детализации [10].

B-метод был разработан Жаном-Раймондом Абриалем (Jean-Raymond Abrial) [1]. Позже была создана компания под названием B-Core, которая сделала метод коммерческим продуктом и разработала начальный B-комплект инструментальных средств разработки. Также другой коммерческий комплект инструментальных средств разработки - Atelier-B [11] был разработан французской группой исследователей. В B-методе спецификации и коды написаны в AMN (Abstract Machine Notation). Формальная спецификация в Event-B представляет собой совокупность контекста (context) и машины (machine) на каждом уровне детализации. Контекст – необязателен, он содержит константы, множества, а также аксиомы и теоремы, которые определяют типы и ограничения заданных констант и множеств. Машина является обязательным элементом спецификации и, в общем случае, включает в

себя список переменных, событий, которые выполняются в соответствии с определенными условиями, и инвариантов. Инварианты определяют свойства переменных системы, которые всегда выполняются [12].

Процесс детализации подразумевает постепенный переход от абстрактного описания к более точному, с постепенным добавлением деталей системы. При этом указанные на ранних стадиях свойства системы сохраняются на всех этапах детализации. Сущность формализации заключается в том, что событиям дается формальное математическое определение и при добавлении свойств системы осуществляется доказательство их соответствия инвариантам. Таким образом, гарантируется корректное поведение системы на всех этапах ее разработки.

Такой подход нашел широкое применение при разработке программного обеспечения. В этом случае в B задается спецификация, а затем осуществляется одна или несколько промежуточных детализаций. Последняя детализация называется реализацией. Эта реализация может вызывать операции других машин (импорт). После полной проверки разработанной реализации B-метода возможна генерация кода программы на языке высокого уровня.

Таким образом, на протяжении всего жизненного цикла разработки программного обеспечения используется математический аппарат для доказательства соответствия ПО техническим требованиям.

Например, необходимо описать ИУС, ключевым параметром которой является температура – t. Система может находиться в одном из трех состояний – Чтение данных (ReadingState), Анализ данных (Analyzing) и Выключено (ShutdownState). Из состояния Анализ данных возможен переход как в состояние Чтение данных (AnalyzingReading), так и в Выключено (AnalyzingShutdown) в зависимости от значения температуры. Если значение температуры превысит заданное (в данном случае 50°C), то необходимо гарантировать, что система перейдет из состояния Анализ данных в состояние Выключено (рис. 1).

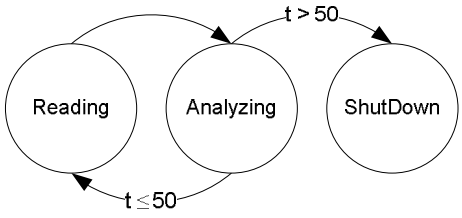

Рис. 1. Граф переходов-состояний ИУС

Машина ИУС включает в себя: переменные CurrMode (текущее состояние), PrevMode (предыдущее состояние), t (значение температуры); инварианты и события (рис. 2). Инварианты inv1–inv3 задают тип переменных, inv4–inv6 описывают разрешенные переходы между состояниями, а inv7 задает условие перехода в состояние Выключено.

```
MACHINE
    abs
SEES
    C_abs
VARIABLES
    CurrMode
    t
    PrevMode
INVARIANTS
    inv1  :  t ∈ ℤ
    inv2  :  CurrMode ∈ MODES
    inv3  :  PrevMode ∈ MODES
    inv4  :  CurrMode = Reading ⇒ PrevMode = Analyzing
    inv5  :  CurrMode = Analyzing ⇒ PrevMode = Reading
    inv6  :  CurrMode = ShutDown ⇒ PrevMode = Analyzing
    inv7  :  t > 50 ∧ PrevMode = Analyzing ⇒ CurrMode = ShutDown
EVENTS
INITIALISATION  ≙
BEGIN
        act1  :  t := 0
        act2  :  CurrMode := Reading
        act3  :  PrevMode := Analyzing
END
ReadingState  ≙
WHEN
        grd1  :  CurrMode = Reading
THEN
        act1  :  t :∈ ℤ
        act2  :  CurrMode := Analyzing
        act3  :  PrevMode := Reading
END
AnalyzingReading  ≙
WHEN
        grd1  :  t ≤ 50
        rd2   :  CurrMode = Analyzing
THEN
        act1  :  CurrMode := Reading
        act2  :  PrevMode := Analyzing
END
AnalyzingShutdown  ≙
WHEN
        grd1  :  t > 50
        grd2  :  CurrMode = Analyzing
THEN
        act1  :  CurrMode := ShutDown
        act2  :  PrevMode := Analyzing
END
ShutDownState  ≙
WHEN
        grd1  :  CurrMode = ShutDown
THEN
        skip
END
END
```

Рис. 2. Машина ИУС

Событие INITIALIZATION (инициализация) выполняется один раз при старте системы.

Контекст содержит множество состояний системы (рис. 3).

```
CONTEXT
  C_abs
SETS
  MODES
CONSTANTS
  Reading
  Analyzing
  ShutDown
AXIOMS
  axm1   :   MODES = {Reading, Analyzing, ShutDown}
  axm2   :   ¬ Reading = Analyzing
  axm3   :   ¬ Reading = ShutDown
  axm4   :   ¬ Analyzing = ShutDown
END
```

Рис. 3. Контекст ИУС

Таким образом, математически обосновывается и доказывается все множество переходов системы и гарантируется ее переход при определенном условии в безопасное состояние (Выключено). В этом случае нет необходимости проводить тестирование полученного после трансляции кода на предмет реакции системы при превышении заданного параметра.

2. Разработка спецификации для системы на программируемой логике

Для задания спецификаций систем на программируемой логике с последующей трансляцией в VHDL применяется методика, первым шагом которой является создание абстрактной машины, описывающей поведение системы в ее окружении, т.е. взаимодействие с датчиками, коммутаторами и т.п. Сама система представляет собой «черный ящик».

На следующем шаге детализации задаются возможные состояния системы и переходы между ними, а также производится анализ некоторых входных данных.

Количество уровней детализации зависит от сложности системы и решаемых задач. На каждом последующем шаге добавляются свойства системы, уточняющие ее поведение в описанных ранее состояниях, а также новые события, необходимые для реализации алгоритма управления.

В рамках проекта разработки блока управления противообледенительной системой самолета (БУ ПОС) была создана спецификация алгоритма управления в Event-B.

Данная спецификация имеет четыре уровня абстракции – абстрактная машина и три шага детализации (refinements). БУ ПОС имеет следующие состояния: Чтение данных (Reading), Анализ данных (Analyzing) и Управление (Controlling) (рис. 4).

На рис. 5 приведен пример абстрактной спецификации системы. Переменные с суффиксом _I определяют входы БУ, а с суффиксом _O – выходы (рис. 5, а). В случае, когда БУ выключен (входы Auto_I и Manual_I установлены в значение FALSE), необходимо установить выходные сигналы в состояние FALSE (рис. 5, б). Инварианты (рис. 5, в) описывают состояние выходов системы, при котором их значения одновременно не равны TRUE.

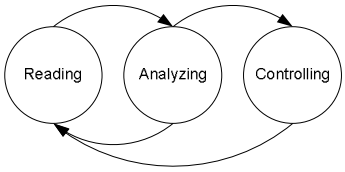

Рис. 4. Граф переходов-состояний БУ ПОС

```
MACHINE
    AbsEnv
VARIABLES
    Manual_I
    Auto_I
    Ice_I
    Temp_I
    Heater1_O
    Heater2_O
    Heater3_O
    HeaterKnife_O
```

а

```
Reset ≙
STATUS
    ordinary
WHEN
    grd1  :  Manual_I = FALSE ∧ Auto_I = FALSE
THEN
    act1  :  Heater1_O ≔ FALSE
    act2  :  Heater2_O ≔ FALSE
    act3  :  Heater3_O ≔ FALSE
    act4  :  HeaterKnife_O ≔ FALSE
END
```

б

```
inv12  :  Heater1_O = TRUE ⇒
          Heater2_O = FALSE ∧ Heater3_O = FALSE
inv13  :  Heater2_O = TRUE ⇒
          Heater1_O = FALSE ∧ Heater3_O = FALSE
inv14  :  Heater3_O = TRUE ⇒
          Heater1_O = FALSE ∧ Heater2_O = FALSE
```

в

Рис. 5. Абстрактная машина БУ ПОС:
а - переменные задающие входы и выходы;
б - пример взаимодействия входов и выходов;
в - инварианты, описывающие выходы

Переключение внешних сигналов Auto_I = TRUE, Manual_I = FALSE в Auto_I = FALSE, Manual_I = TRUE происходит через состояние Auto_I = FALSE, Manual_I = FALSE, которое является асинхронным сбросом системы. Этот случай описывается событием (рис. 6, а) и инвариантом (рис. 6, б).

```
EnvironmentReset ≙
STATUS
        ordinary
REFINES
        EnvironmentAssignment
BEGIN
        act1  :  Auto_I,Manual_I ≔ FALSE,FALSE
        act4  :  CurrMode ≔ Analyzing
END
```

а

```
        CurrMode = Reading ∧ PrevMode = Analyzing ⇒
inv11 :  Heater1_O = FALSE ∧ Heater2_O = FALSE ∧
        Heater3_O = FALSE ∧ HeaterKnife_O = FALSE
```

б

Рис. 6. Асинхронный сброс: а - событие; б – инвариант

На следующем шаге детализации были добавлены состояния БУ ПОС (рис. 7), разрешенная последовательность переходов между которыми задана инвариантами (рис. 8).

```
SETS
    MODES
CONSTANTS
    Reading
    Analyzing
    Controlling
AXIOMS
    axm7  :  MODES = {Reading, Analyzing, Controlling}
    axm8  :  ¬ Reading = Analyzing
    axm9  :  ¬ Reading = Controlling
    axm10 :  ¬ Analyzing = Controlling
```

Рис. 7. Описание состояний системы

```
inv20 :  CurrMode = Reading ⇒
         PrevMode ≠ Reading
inv22 :  CurrMode = Controlling ⇒
         PrevMode = Analyzing
```

Рис. 8. Инварианты допустимых переходов

Находясь в состоянии Reading, БУ ПОС считывает текущие значения датчиков и переключателей (рис. 9).

```
EnvironmentAssignment ≙
STATUS
        ordinary
REFINES
        EnvironmentAssignment
WHEN
        grd1  :  CurrMode = Reading
THEN
                 Auto_I,Manual_I :| Auto_I' ∈ BOOL ∧
        act1  :  Manual_I' ∈ BOOL ∧
                 (¬ Auto_I' = TRUE ∨ ¬ Manual_I' = TRUE)
        act2  :  Ice_I :∈ BOOL
        act3  :  Temp_I :∈ ℤ
        act4  :  CurrMode ≔ Analyzing
END
```

Рис. 9. Считывание входной информации

В состоянии Analyzing осуществляется сравнение параметра Temp_I (температура) с заданными значениями и определяется интервал (рис. 10), который влияет на режимы работы БУ в состоянии Controlling.

```
AnalyzingState1 ≙
STATUS
        ordinary
WHEN
        grd0  :  Manual_I = TRUE ∨ (Auto_I = TRUE ∧ Ice_I = TRUE)
        grd1  :  CurrMode = Analyzing
        grd2  :  Temp_I ≤ RightBorder ∧ Temp_I > MiddleBorder
THEN
        act1  :  Interval ≔ 1
        act2  :  CurrMode ≔ Controlling
END
```

```
AnalyzingState2 ≙
STATUS
        ordinary
WHEN
        grd0  :  Manual_I = TRUE ∨
                 (Auto_I = TRUE ∧ Ice_I = TRUE)
        grd1  :  CurrMode = Analyzing
        grd2  :  Temp_I ≤ MiddleBorder ∧
                 Temp_I > LeftBorder
THEN
        act1  :  Interval ≔ 2
        act2  :  CurrMode ≔ Controlling
END
```

Рис. 10. Анализ считанной информации

Для однозначного определения режима работы в состоянии Controlling были заданы соответствующие инварианты (рис. 11).

```
inv0 :  CurrMode = Controlling ⇒
        (Temp_I > RightBorder ⇔ Interval = 0)
inv4 :  CurrMode = Controlling ⇒
        (Temp_I ≤ RightBorder ∧
        Temp_I > MiddleBorder ⇔ Interval = 1)
inv5 :  CurrMode = Controlling ⇒
        (Temp_I ≤ MiddleBorder ∧
        Temp_I > LeftBorder ⇔ Interval = 2)
inv6 :  CurrMode = Controlling ⇒
        (Temp_I ≤ LeftBorder ⇔ Interval = 3)
```

Рис. 11. Инварианты связи значений параметра и интервалов

Два последних шага детализации описывают события и инварианты, задающие порядок (рис. 12) переключения выходов и длительность (рис. 13) их работы в зависимости от полученного интервала на этапе анализа данных.

```
inv9  :  CurrPhase = Phase1 ⇒ PrevPhase = PhaseEnd
inv11 :  CurrPhase = Phase2 ⇒  PrevPhase = Phase1
inv12 :  CurrPhase = Phase3 ⇒ PrevPhase = Phase2
inv13 :  CurrPhase = PhasePause ⇒ PrevPhase = Phase3
inv14 :  CurrPhase = PhaseEnd ⇒ PrevPhase = PhasePause
```

Рис. 12. Инварианты последовательности выходов

```
Counter01 ≙
STATUS
      ordinary
WHEN
      grd1  :  TimeCountEnable01 = TRUE
      grd2  :  TimeCountFlag01 = FALSE
      grd3  :  TimeCounter01 < TimeToHeat
      grd4  :  CurrMode = Controlling
THEN
      act1  :  TimeCounter01 := TimeCounter01 + 1
      act2  :  TimeCountFlag01 := TRUE
END
```

Рис. 13. Событие подсчета длительности

Данная спецификация была разработана с использованием открытой платформы Rodin [9], которая была создана в рамках проекта с одноименным названием.

3. Разработка транслятора

После получения реализации в виде Event-B спецификации возникает вопрос, каким образом транслировать полученную реализацию в код на языке описания аппаратуры VHDL. При этом основной сложностью при трансляции является определение соответствий конструкций спецификации и языка. Например, описание входов и выходов системы, заданных в абстрактной машине как окружение, в VHDL можно представить в виде сущности (ENTITY) (рис. 14, а), описание событий Event-B можно интерпретировать как процесс (PROCESS), а условия выполнения событий и их действия оператором IF … THEN … (рис. 14, б). Таким образом, можно определить соответствие простых конструкций.

Для использования библиотечных компонент систем автоматизированного проектирования необходимо указывать дополнительные атрибуты или использовать специализированные шаблоны.

Платформа Rodin, с помощью которой задается Event-B спецификация, поддерживает добавление шаблонов (patterns) и надстроек (plug-ins). Это позволяет добавить транслятор в платформу в виде надстройки, а библиотечные элементы и элементы с поддержкой отказоустойчивости в виде шаблонов.

Заключение

Использование методов формальной спецификации и верификации для создания систем на программируемой логике ведет к существенному уменьшению времени тестирования и верификации конечного продукта.

Рассмотренные примеры иллюстрируют преимущество использования спецификаций в Event-B при проектировании систем на ПЛИС, которое заключается в математическом доказательстве выполнения тех или иных свойств и сохранения их на последующих шагах детализации. Таким образом, результатом такого подхода является математически доказанная реализация системы, не требующая тестирования.

Направлениями дальнейших исследований являются определение четкого соответствия конструкций B-спецификации и языка описания аппаратуры и создание транслятора из Event-B в VHDL.

```
MACHINE
  AbsEnv
VARIABLES
  Manual_I
  Auto_I
  Ice_I
  Temp_I
  Heater1_O
  Heater2_O
  Heater3_O
  HeaterKnife_O
```

```
ENTITY AbsEnv IS
    PORT
    (
        Manual_I        : IN    STD_LOGIC;
        Auto_I          : IN    STD_LOGIC;
        Ice_I           : IN    STD_LOGIC;
        TEMP_I          : IN    STD_LOGIC;
        Heater1_O       : OUT   STD_LOGIC;
        Heater2_O       : OUT   STD_LOGIC;
        Heater3_O       : OUT   STD_LOGIC;
        HeaterKnife_O   : OUT   STD_LOGIC
    );
END AbsEnv;
```

а)

```
WaitingIce  ≙
STATUS
  ordinary
WHEN
  grd1   :   Auto_I = TRUE ∧ Ice_I = FALSE
THEN
  act1   :   Heater1_O ≔ FALSE
  act2   :   Heater2_O ≔ FALSE
  act3   :   Heater3_O ≔ FALSE
  act4   :   HeaterKnife_O ≔ FALSE
END
```

```
WaitingIce:PROCESS
BEGIN
    WAIT UNTIL __clk_signal = '1';
    IF Auto_I = '1' AND Ice_I = '0' THEN
        Heater1_O       <=  '0';
        Heater2_O       <=  '0';
        Heater3_O       <=  '0';
        HeaterKnife_O   <=  '0';
    END IF;
END PROCESS WaitingIce;
```

б)

Рис. 14. Соответствия описаний в Event-B и VHDL:
а - входов и выходов; б – событий

Литература

1. Abrial J.R. The B Book: Assigning Programs to Meanings. / J.R. Abrial. - Cambridge University Press, 1996.

2. Bert D. Adaptable Translator of B Specifications to Embedded C Programs / D Bert, S. Boulmé, M.L. Potet, A. Requet, L. Voisin1 // FME 2003: Formal Methods. – 2003. – Vol. 2805. – P. 94-113.

3. Yang L. Automatic Translation from Combined B and CSP specification to Java Programs. / L. Yang, M.R. Poppleton // 7th International B Conference. - 17-19 January 2007. - Besancon, France.

4. Abrial J.R. / Event Driven Electronic Circuit Construction / J.R. Abrial. - 2001.

5. Seceleanu T. Systematic Design of Synchronous Digital Circuits / T. Seceleanu // TUCS Dissertations, Turku Centre of Computer Science. – May 2001. – No 32.

6. Boulanger J.L. Formalization of digital circuits using the B method /J.L. Boulanger, A. Aljer, G. Mariano // Eighth International Conference on Computers in Railways. Computers in Railways VIII. – 2002. - P. 691-700.

7. Sørensen Ib Holm. Using B to specify, verify and design hardware circuits / Ib Holm Sørensen // ZUM '98: The Z Formal Specification Notation. Lecture Notes in Computer Science. – 1998. – Vol. 1493. – P. 60-65.

8. Zimmermann Y. Formal modeling of electronic circuits using event-B, Case Study : SAE J1708 Serial Communication Link / UML-B - Specification for Proven Embedded Systems Design, J. Mermet, editor. - KeesDA, Kluwer Academic Publishers. – 2004. – P. 211-226.

9. Event-B and the Rodin Platform [Электронный ресурс] – Режим доступа: http://www.event-b.org/.

10. Lano K. The B Language and Method: A Guide to Practical Formal Development / K. Lano. - Springer-Verlag, FACIT series, 1996.

11. Atelier B. [Электронный ресурс] – Режим доступа: http://www.atelierb.eu.

12. Schneider S. The B-Method: An Introduction / S. Schneider // Palgrave, Cornerstones of Computing series, 2001.

Поступила в редакцию 03.02.2009

Рецензент: д-р техн. наук, проф. В.С. Харченко, Харьковский национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.

ЗАСТОСУВАННЯ EVENT-B ДЛЯ ПОБУДОВИ СИСТЕМ НА ПРОГРАМОВАНІЙ ЛОГІЦІ

Ю. Прохорова, С. Остроумов, О. Трубіцина, Л. Лайбініс

Наведено особливості побудови формальної специфікації систем на програмованій логіці. Розглянуті приклади опису таких систем у Event-B. Запропоновані елементи методики формального опису та трансляції одержаної специфікації в код на мові опису апаратури VHDL.

Ключові слова: формальні методи, Event-B, платформа Rodin, ПЛІС, VHDL.

AN APPLICATION OF EVENT-B FOR DEVELOPMENT OF SYSTEMS ON PROGRAMMABLE LOGIC

Yu. Prokhorova, S. Ostroumov, E. Troubitsyna, L. Laibinis

This paper presents an approach for formal development of systems on programmable logic. The examples of the Event-B description of such systems are demonstrated. A new technique for formal description and translation of an obtained specification into VHDL code is proposed.

Key words: formal methods, Event-B, Rodin tool, FPGA, VHDL.

Прохорова Юлия – аспирант кафедры компьютерных систем и сетей Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: J.Prokhorova@csac.khai.edu.

Остроумов Сергей – аспирант кафедры компьютерных систем и сетей Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина, e-mail: S.Ostroumov@csac.khai.edu.

Трубицына Елена – канд. техн. наук, доцент кафедры информационных технологий Åbo Akademi University, Турку, Финляндия, e-mail: Elena.Troubitsyna@abo.fi.

Лайбинис Линас - канд. техн. наук, ст. научный сотрудник кафедры информационных технологий Åbo Akademi University, Турку, Финляндия, e-mail: Linas.Laibinis@abo.fi.