УДК 681.5

АЛГОРИТМ ТРАНСЛЯЦИИ ТАБЛИЧНОГО КОДА THDL В СИНТЕЗИРУЕМЫЙ ТЕКСТОВЫЙ HDL-КОД

Малиновский М. Л., д.т.н.,
Конищева А. П., инж.
Харьковский национальный технический университет сельского хозяйства им. П.Василенка
Аленин Д. А., инж.
ООО НПП «Стальэнерго»
Тел.: 057-712-35-37

Аннотация: **проанализированы табличные инструментальные средства описания цифровых устройств, разработан транслятор из табличного формата на языке THDL в язык Verilog.**

Ключевые слова: **проектирование цифровых устройств, транслятор, THDL, Verilog.**

*Постановка задачи. С*уществующие среды проектирования цифровых устройств (ISE, Quartus) поддерживают инструментальные средства, которые можно разделить на текстовые (языки описания аппаратуры VHDL, Verilog и др.) и графические (средства структурного и поведенческого описания). Графические средства наглядны и компактны. Их использование упрощает взаимодействие и улучшает взаимопонимание в коллективе разработчиков. Такие программы лучше поддаются анализу и коррекции.

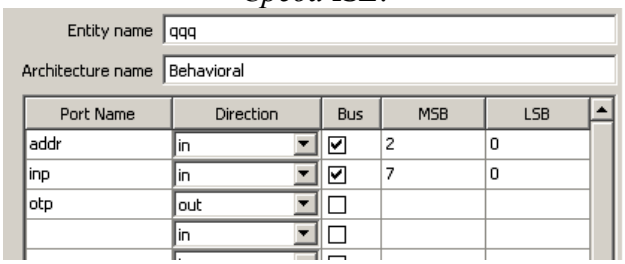
Большинство разработчиков в качестве основного используют текстовый инструментарий. Такое положение дел объясняется как субъективными, так и объективными факторами, в том числе тем, что текст не привязан к редактору, текстовые средства обладают большими возможностями и гибкостью при описании устройств, требующих использования вложенных конструкций, параметризируемых устройств с регулярной архитектурой и т.д.

Существует противоречие между наглядностью одних средств и универсальностью других. По мнению авторов, табличные средства способны в значительной степени разрешить данное противоречие.

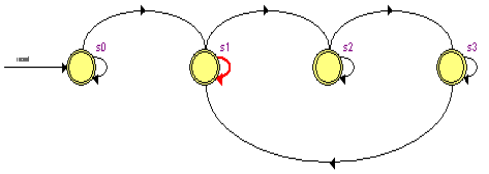
Примеры использования табличных форм имеются в существующих средах проектирования. В частности, при создании нового компонента в среде ISE описание интерфейсной части возможно пу-

тем заполнения таблицы входов-выходов. В редакторе Quartus поддерживается инструментарий табличного описания автоматов. В текстовом языке AHDL имеется удобная табличная конструкция TABLE, позволяющая описать логические зависимости (рис. 1).

Среда ISE:



Среда Quartus:



	Source State	Destination State	Condition
1	s0	s0	(!input)
2	s0	s1	(input)
3	s1	s1	(!input)
4	s1	s2	(input)
5	s2	s2	(!input)
6	s2	s3	(input)
7	s3	s1	(!input)
8	s3	s3	(input)

Язык AlteraHDL:

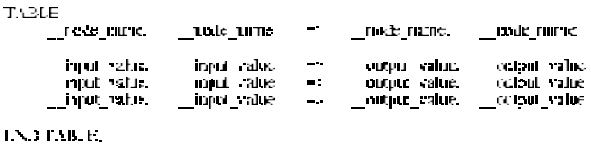


Рис. 1. Примеры использования табличных форм.

Универсальные табличные языки, которые позволили бы в полной мере использовать все преимущества табличных форм для описания цифровых устройств, в настоящее время отсутствуют.

В связи с этим актуальной является задача разработки табличных инструментальных средств описания цифровых устройств и трансляции табличного описания в поддерживаемые существующими компиляторами текстовые форматы.

Анализ публикаций. Подходы к методам трансляции представлены в [2], описание табличных языков проектирования – в [1], [3], [5]описание языка Verilog в [4]. Анализ публикаций подтверждает наличие предпосылок применения табличных языков и актуальность задачи разработки транслятора с табличного языка в компилируемый текстовый язык.

Целью исследования. Обеспечение возможности практического применения гибкого и универсального табличного языка для описания цифровых устройств.

Основные материал исследования. THDL (Tabular Hardware Description Language) – табличный язык описания цифровых устройств. В программе на THDL выделяются две части: 1) интерфейсная часть – объявление имени компонента и описания сигналов; 2) логическая часть – описание логики работы устройства.

Основными табличными конструкциями первой части являются COMPONENT и BINARY SIGNALS. На рисунке 2 приведен пример использования данных конструкций. Слева при помощи языка THDL объявлен компонент test_thdl и описаны его сигналы различных типов и разрядности. Справа – те же данные, описанные на языке Verilog.

COMPONENT
test_thdl

	BINARY SIGNALS				
	TYPE	NAME	MSB	LSB	VALUE
	CLOCK	CLK			
	RESET	RST			
	IN	BYTE	7	0	
		DATA_IN	2	0	
		LOAD			
		UP_DN			
	OUT	DATA_OUT	7	0	
	OUT REG	Q	2	0	"001"

```
module
test_thdl(CLK,RST,BYTE,DATA_IN,LOAD,U
P_DN,DATA_OUT,Q);
input CLK, RST, LOAD, UP_DN;
input [7:0] BYTE;
input [2:0] DATA_IN;
      output [7:0] DATA_OUT;
      output [2:0] Q;
wire CLK;
wire RST;
wire [7:0] BYTE;
wire [2:0] DATA_IN;
wire LOAD;
wire UP_DN;
      reg [7:0] DATA_OUT;
      reg [2:0] Q;
```

Рисунок 2 – Описание интерфейсной части на THDL и Verilog

а)

3	ACTION
	Q = Q + 1
	DATA_OUTP = !BYTE

```
always @(*)
      begin
      DATA_OUTP<= !BYTE;      end
always @(posedgeCLK)
      begin
      Q<= Q + 1;        end
```

б)

2	PRIORITY CONDITION	ACTION
	LOAD	Q = DATA_IN
	UP_DN	Q = Q + 1
	ELSE	Q = Q - 1

```
always @(posedge CLK)
begin
      if (LOAD)
      begin
            Q <= DATA_IN;     end
      else if (UP_DN)
      begin
            Q <= Q + 1; end
      else
      begin
            Q <= Q - 1; end
end
```

в)

7	PRIORITY ARGUMENT		FUNCTION	
	addr_ld	addr_in	w_match	r_match
	1	"1010"	1	0
		"1011"	0	1
		else	0	0
	ELSE		0	0

```
always @(posedge CLK)
begin
      if (addr_ld)
      begin
            if (addr_in == "1010")
            begin
                  w_match<= 1;
                  r_match<= 0;      end
            else if (addr_in == "1011")
            begin
                  w_match<= 0;
                  r_match<= 1;      end
            else
            begin
                  w_match<= 0;
                  r_match<= 0;      end
      else
      begin
            w_match<= 0;
            r_match<= 0;            end
end
```

Рис. 3. Описание логической части устройства
на THDL и Verilog: конструкции ACTION,
PRIORITY CONDITION и PRIORITY ARGUMENT.

В логической части используются следующие конструкции:

– ACTION, FUNCTION – конструкции, описывающие действия;

– CONDITION, PRIORITYCONDITION, ARGUMENT, PRIORITYARGUMENT – конструкции, описывающие условия, при которых выполняются определенные действия.

Пример использования конструкций логической части приведен на рисунке 3 (слева - THDL-описание, справа – Verilog-описание):

а) Конструкция ACTION – инкрементируется значение регистрового сигнала Q и присваивается значение комбинационному сигналу DATA_OUTP;

б) Конструкция ACTION дополнена условием с приоритетом PRIORITYCONDITION. Если получен сигнал LOAD, выполняется действие Q <= DATA_IN. Если сигнал LOAD не получен, а получен UP_DN, сигнал Q инкрементируется. При неполучении ни одного из перечисленных сигналов значение Q декрементируется;

в) Конструкции PRIORITY ARGUMENT и FUNCTION. При получении сигнала addr_ld проверяется значение аргумента addr_in. По результатам проверки функциям w_match и r_match присваиваются соответствующие значения.

Разработанный транслятор обеспечивает возможность ввода описания в табличном формате на языке THDL и трансляции этого описания в язык Verilog. Алгоритм трансляции будем рассматривать на примере реализации счетчика по модулю 6 (рис. 4), которую можно описать при помощи табличных и текстовых конструкций (рис.5).

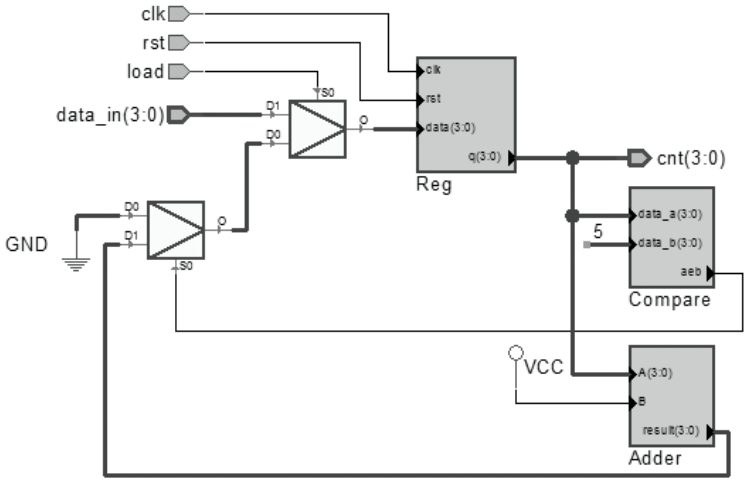


Рис. 4. RTL-схема счетчика по модулю 6.

Трансляция заключается в преобразовании табличных конструкций в текстовые, при этом схемная реализация устройства должна оставаться идентичной. Как упоминалось выше, любая программа состоит из двух частей – интерфейсной и части, описывающей логику работы устройства. Алгоритм трансляции интерфейсной части зави-

сит от синтаксиса языка и не вызывает сложностей. Рассмотрим алгоритм трансляции части, описывающей логику работы.

COMPONENT
mod_6_counter

	BINARY SIGNALS				
	TYPE	NAME	MSB	LSB	VALUE
	CLOCK	clk			
	RESET	rst			
	IN	data_in	3	0	
		load			
	OUT REG	cnt	3	0	

	PRIORITY CONDITION	ACTION
	load == 1	cnt = data_in
	cnt == 5	cnt = 4'b0000
	ELSE	cnt = cnt + 1'b1

END
mod_6_counter

```
Modulemod_6_counter     (
Input wireclk,
Input wirerst,
Input wireload,
Input wire [3:0]data_in,
Output reg [3:0]cnt     );
always @ (posedgeclk)
begin
      if (rst)
            cnt<= 4’b0000;
      else
      begin
            if(load)    cnt<= data_in;
      else
            if (cnt == 5)
                  cnt<= 4’b0000;
            else
                  cnt<= cnt + 1’b1;
            end
      end
end module
```

Рис. 5. Табличное и текстовое описания счетчика по модулю 6.

Поскольку табличные конструкции этой части описывают только логику и не учитывают тип сигнала (комбинационный или регистровый), удобно разбить схему на два блока – комбинационный и регистровый (рис. 6) и далее для каждой из этих частей формировать описание в текстовом формате.

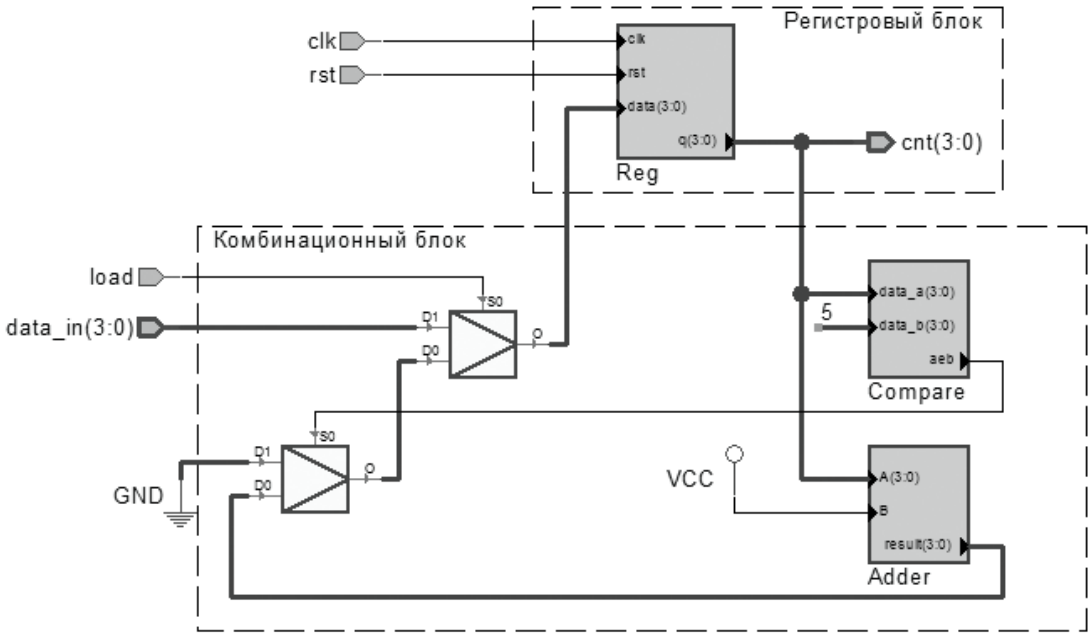


Рис. 6. Регистровый и комбинационный блоки RTL-схемы.

Все вычисления и присваивания будем производить в комбинационном блоке, а сохранение результатов – в регистровом.

Для этого необходимо для каждого регистрового сигнала создать вспомогательный сигнал такой же разрядности. Присвоим ему имя основного сигнала с приставкой next_. Таким образом, для регистровых сигналов появится два значения – сохраненное (будем его использовать в условиях и в правой части выражений присваивания) и текущее – с приставкой next_ (будем его использовать в регистровом блоке в левой части выражений присваивания).

В комбинационном блоке каждую табличную конструкцию будем разбирать в соответствии с заданным шаблоном (табл. 1). Для сохранения результатов в регистровом блоке регистровым сигналам необходимо присвоить значение, определенное в комбинационном блоке (присвоенное сигналу с приставкой next_).Таким образом, последовательность заполнения шаблона текстового описания соответствует диаграмме, приведенной на рис.7.

Таблица 1 – Соответствие табличных конструкций текстовым

Табличная конструкция	Текстовая конструкция

CONDITION
ARGUMENT
	if (condition_1)
	…
	if (condition_n)
	…
	if (!condition_1 and…and !condition_n)

PRIORITYCONDITION
PRIORITYARGUMENT
	if (condition_1)
	…
	else if (condition_2)
	…
	else if (condition_n)
	…
	else

Для проектирования цифровых устройств с использованием табличных конструкций создан редактор EditorTHDL. Интерфейс редактора приведен на рисунке 8. Начинать работу необходимо с создания проекта (а) и компонента в данном проекте (б). Далее из списка существующих табличных конструкций выбирают требуемую (в). Программа будет представлять собой перечень необходимых табличных конструкций (г). После создания табличного описания разработчик запускает трансляцию и получает Verilog-код (д).

Рис. 7. Последовательность действий при трансляции табличного описания текстовое.

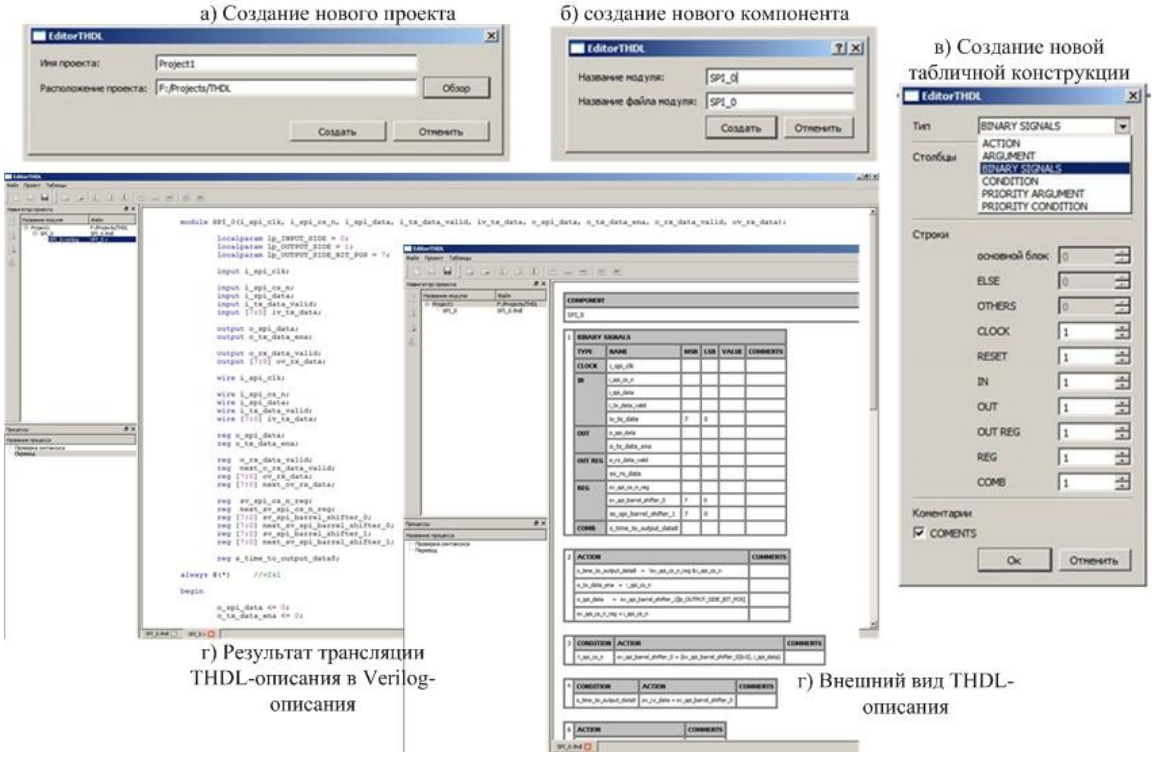


Рис. 8. Интерфейс редактора EditorTHDL.

Вывод. Полученный в результате исследования табличный язык THDL является универсальным средством проектирования цифровых устройств, а также позволяет повысить качество и снизить трудозатраты на создание программного обеспечения.

Разработанный алгоритм трансляции и табличный редактор EditorTHDL дает возможность использовать язык THDL при решении практических задач, связанных с проектированием цифровых устройств на основе заказных и программируемых интегральных схем.

Литература

1. Методы и средства табличного описания аппаратуры цифровых устройств / Малиновский М.Л., Конищева А.П., Сидоренко А.В., Казинов И.А. // Інформаційно-керуючі системи на залізничному транспорті – УкрДАЗТ. – 2011. - №4. – С. 69-72
2. Свердлов С.З. Языки программирования и методы трансляции: Учебное пособие. – СПб.: Питер, 2007. – 638с.: ил.
3. Languages and General Software Aspects for Telecommunication Systems. The Tree and Tabular Combined Notation. v.3 07/2001 ITU
4. IEEE Standard Verilog. Hardware Description Language 03/2001
5. Концепция создания табличных языков описания аппаратуры / *М.Л. Малиновский, И.А. Фурман, А.Ю. Аллашев, А.П. Конищева, А.В. Святобатько* // Радіоелектронні і комп'ютерні системи, 2010, №6 (47). - С. 289-291.

АЛГОРИТМ ТРАНСЛЯЦІЇ ТАБЛИЧНОГО КОДУ THDL У СИНХРОНІЗОВАНИЙ ТЕКСТОВИЙ HDL-КОД

М. Л. Малиновський , А.П. Коніщева , Д. О. Алєнін

Анотація

Проаналізовані табличні інструментальні засоби представлення цифрових пристроїв, розроблено транслятор з табличного формату на мові THDL у мову Verilog.

ALGORITHM A CONVERTING TABULAR CODE THDL IN SYNTHESIZED HDL-TEXT CODE

M. Malinovsky, A. Konicheva, D. Alenin

Summary

Analyzed tabular tools description of digital devices designed converter from a table format in the language THDL in the Verilog language.