УДК 621.3.049.77:658.512.2

А.В. Лапин

ИСПОЛЬЗОВАНИЕ АДАПТАЦИИ СЕТЕЙ ПЕТРИ ДЛЯ МОДЕЛИРОВАНИЯ ЛОГИЧЕСКИХ СХЕМ, ПРЕДСТАВЛЕННЫХ НА ПОВЕДЕНЧЕСКОМ УРОВНЕ

Национальный исследовательский университет «МИЭТ», Москва, Россия

В логическом моделировании существует такое понятие как гонка сигналов – состояние, при котором несколько входных сигналов переключаются одновременно, создавая неоднозначность выходного состояния. Для устранения подобной неоднозначности использовать такой математический аппарат, который позволит достоверно моделировать параллельно происходящие переключения сигналов. В работе предложен новый алгоритм событийного функционального моделирования цифровых интегральных схем, основанный на использовании адаптированного математического аппарата сетей Петри. Описанный подход позволяет устранить неоднозначность переключений сигналов, происходящих в один момент времени, за счёт отказа от использования дельта-задержки. Это стало возможным благодаря механизму сетей Петри, в котором параллельные конструкции имитируются с помощью последовательных инструкций. Следовательно, нет необходимости разделять события с помощью дельта-задержки. На основе предложенного алгоритма реализована программа логического моделирования цифровых схем. Представлены результаты работы алгоритма на примере моделирования на вентильном и поведенческом уровнях ряда комбинационных и последовательностных схем. На основании полученных временных диаграмм, а также времени, затраченном на моделирование схем, можно утверждать, что предложенный алгоритм не уступает существующим средствам моделирования в плане достоверности и быстродействия.

Ключевые слова: цифровой симулятор, логическое моделирование, функциональное моделирование, алгоритм событийного моделирования, сети Петри.

Введение. При моделировании триггерных схем может возникнуть ситуация, при которой новое значение входного сигнала при переключении обрабатывается позже переключения синхросигнала. Это проявляется в неверной работе схемы, поскольку выход триггера переключается раньше положенного такта (Рисунок 1). Такое явление носит название «гонка сигналов».

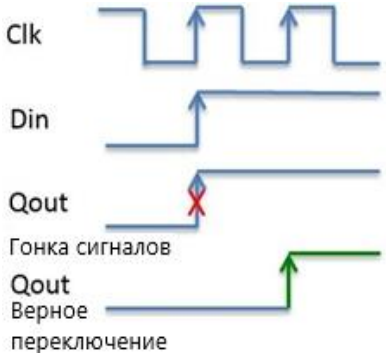

Рисунок 1 - Временные диаграммы, иллюстрирующие гонку сигналов.

Подобные неточности могут быть губительными, если возникают при моделировании, поскольку результаты моделирования схемы отличаются от результатов синтезированной логики [1]. Такая ситуация возникает за счёт нулевой задержки, при которой все входные сигналы меняют свои значения в один момент времени с синхросигналом. Современные логические симуляторы решают данную проблему назначением синхросигнала блокирующим воздействием в раннее временное окно, а функциональные сигналы назначаются неблокирующими воздействиями. Алгоритм симулятора определяет, как это делать, по способу генерации данных воздействий. Синхросигнал генерируется блокирующим назначением, в то время как логические выходы Q_{out} генерируются неблокирующим воздействием.

В данной работе для разрешения описанной проблемы предлагается использовать адаптированный алгоритм сетей Петри первого рода для событийного моделирования логических схем.

Адаптация алгоритма сетей Петри для логического моделирования. В работе [2] описываются основы применения сетей Петри в области логического моделирования. Вкратце, суть заключается в следующем.

1. Состояние представляет собой узел цифровой схемы, следовательно, в каждом состоянии не может находиться более одной метки. Наличие метки в состоянии эквивалентно значению уровня логической единицы в узле, отсутствие метки – уровню логического нуля. В случае реализации многозначной логики предлагается воспользоваться формализмом цветных сетей Петри [3] – вариантом сетей Петри, в котором каждая метка имеет некоторую внутреннюю характеристику, обычно обозначаемую цветом. В терминах многозначного логического моделирования цвет показывает состояние логического уровня, например, логический «0», логическая единица «1», неопределённое состояние «U», неизвестное состояние «X», высокоимпедансное состояние «Z» и другие значения логических уровней, применяемые в современных симуляторах при моделировании цифровых устройств.

2. Переход представляет собой логический вентиль. В простейшем случае для симулятора комбинационных схем переходы осуществляют функционирование в соответствии с заранее предопределённым набором встроенных логических примитивов: «НЕ», «ИЛИ», «ИЛИ-НЕ», «И», «И-НЕ», «ИСКЛЮЧАЮЩЕЕ ИЛИ».

3. Для срабатывания перехода не обязательно наличие метки во входных состояниях, всё зависит только от типа перехода. Так, для появления метки в выходном состоянии перехода, соответствующего логическому вентилю «НЕ», необходимо, чтобы на входном состоянии метки не было, что соответствует таблице истинности инвертора.

4. Факт наличия метки в состоянии на выходе перехода зависит не только от срабатывания перехода, но и от его типа. Наличие меток в состоянии на выходе перехода обуславливается типом логического вентиля и таблицей истинности, описывающей его работу (аналогично пункту 3).

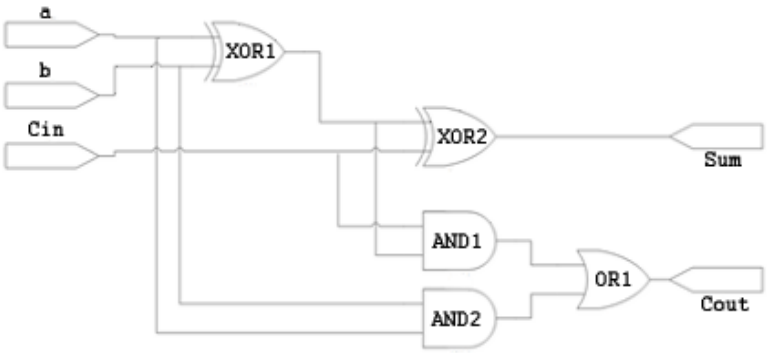

Рисунок 2 - Схема полного сумматора

Наглядно продемонстрировать описываемую адаптацию можно на примере схемы полного сумматора (Рисунок 2). С учётом предложенных преобразований схема примет вид, показанный на рисунке 3.

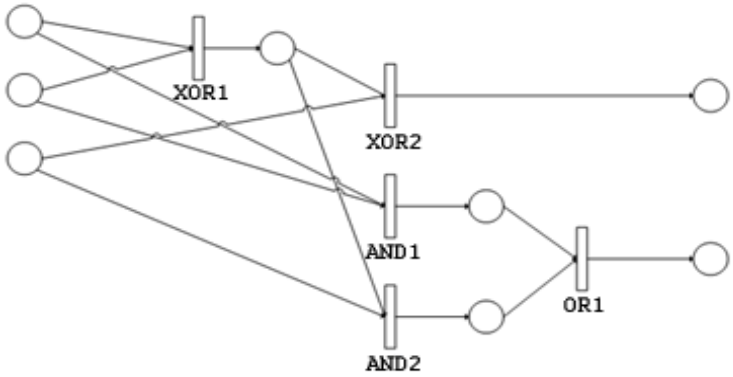

Рисунок 3 - Схема полного сумматора, описанная в терминах предлагаемой адаптации сетей Петри

При моделировании цифровых схем помимо чисто функционального моделирования также применяется временное моделирование, суть которого сводится к учёту времён задержек распространения сигналов в межсоединениях и задержек, обусловленных инертностью срабатывания самих элементов. Такие задержки разделяются на транспортные и инерциальные.

Независимо от типа, временные задержки по сути своей являются параметрами межсоединений, равно как логические значения на этих соединениях и логические функции вентилей. Последние, как было

показано ранее, могут быть преобразованы в параметры цветных сетей Петри. Аналогично, задержки также можно задать в виде параметра для места-состояния, определяя тем самым цветовую маркировку состояния в раскрашенной сети Петри.

Работа с поведенческим описанием схем. Трансляция регистрового и структурного описаний схем в термины сетей Петри не вызывает сложностей: базовые примитивы отражают выполняемые ими функции. При использовании же поведенческого описания подобный подход не может быть применён, так как описываемым элементом схемы может выполняться произвольная функция, описанная в его процедурном блоке. Значит, необходимо реализовать такой алгоритм трансляции поведенческого описания в термины сетей Петри, при котором структура сети не будет многократно усложняться её примитивами. Если в описании схемы присутствует несколько процедурных блоков, то каждый из них будет представлен в виде отдельного перехода предлагаемой адаптации сетей Петри. У таких переходов требуется задавать функционал в виде аналога машинного кода, поскольку лишь простейшими логическими примитивами в данном случае не обойтись.

Чаще всего при моделировании схем, описанных на поведенческом уровне, в списке чувствительности процедурных блоков указаны фронты сигналов, а не конкретные значения. Чтобы определить фронт при переключении сигнала, необходимо внести изменения в алгоритмическую базу предлагаемой адаптации сетей Петри, в частности, в момент работы сети t_. Именно на этом этапе происходит перенос фишек из состояний к переходам, или, выражаясь в терминах предлагаемой адаптации, происходит присвоение новых значений сигналов временным входам логического вентиля [2].

По аналогии с цветовой маркировкой для значений сигналов и величин задержек [2], вводится дополнительный параметр для состояния, отражающий форму изменения сигнала в данном состоянии. Для наглядности, а также удобства программной реализации, он назван «stability». Данный параметр может принимать три фиксированных значения, отражающих форму изменения сигнала: «/» и «\» для фронта и среза соответственно, «_» для стабильного, неизменного значения сигнала.

Представление процедурных блоков поведенческого описания в виде переходов предлагаемой адаптации сетей Петри подразумевает их работу в качестве виртуальных машин, которые при срабатывании должны будут выполнять определённы набор команд. Эти команды предлагается описывать на языке близком по синтаксису и стилистике к ассемблерному коду. По аналогии с ассемблером архитектуры x86 [3] определим набор команд, а именно:

- MOV – команда присвоения значения;

- CMP – команда сравнения двух значений;
- команды условных переходов JZ, JNZ, JG, JL, JGE, JLE, а также флаги состояний ZF, GF, LF;
- команда безусловного перехода JMP.

В первую очередь, зададим команду сравнения – CMP. Оператор сравнения CMP принимает два аргумента, являющиеся сравниваемыми значениями. Синтаксически конструкция имеет следующий вид: CMP(arg1, arg2). Аргументы в данном случае могут быть как числами, так и строками.

Таблица 1 - Флаги состояний

Флаг	Назначение
ZF	Zero Flag, флаг нулевого значения
GF	Greater Flag, флаг большего значения
LF	Less Flag, флаг меньшего значения

Сравнение происходит путём вычитания значения аргумента arg2 из значения arg1. Для определения результатов сравнения предусмотрим флаги состояния (Таблица 3.1). Флаг ZF принимает значение «1» лишь в том случае, когда оба аргумента команды cmp равны, т.е. остаток от вычитания одного аргумента из другого равен нулю. В противном случае ZF = «0». Флаг GF принимает значение «1» тогда и только тогда, когда флаг ZF принял нулевое значение, а остаток от вычитания аргументов больше нуля. Если же в результате вычитания аргументов было получено отрицательное значение, флаг LF установится в «1». Введение флагов состояний позволяет реализовать условные переходы.

Таблица 2 - Условные переходы

Команда	**Переход, если**	**Условие перехода**
JZ/JE	Нуль или равно	ZF=1
JNZ/JNE	Не нуль или не равно	ZF=0
JG/JNLE	Больше/не меньше и не равно	ZF=0 и GF=1
JL/JNGE	Меньше/не больше и не равно	ZF=0 и LF=1
JGE/JNL	Больше или равно/не меньше	ZF=1 или GF=1
JLE/JNG	Меньше или равно/не больше	ZF=1 или LF=1

Основным оператором перехода в ассемблере является команда JMP. В качестве аргумента она принимает ссылку на метку в коде, к которой

следует перейти. Однако, в зависимости от значений флагов состояний предусматриваются и отдельные операторы для каждого конкретного случая. Команда JZ/JE выполнит переход к указанной метке только в том случае, если флаг нулевого значения хранит ненулевое значение. Соответственно, команда JNZ/JNE сработает при обратном условии. В случае, если установлен флаг большего значения, следует использовать команду перехода JG/JNLE. Аналогично используется оператор перехода JL/JNGE в случае установления флага меньшего значения.

```
module inverter(out, in);
  input in;
  output out;
  always @in
    begin
      if(in == 1)
        out = 0;
      if(in == 0)
        out = 1;
    end
endmodule
```

Листинг 1. - Пример описания инвертора на поведенческом уровне.

Простейшие арифметические и логические операции задаются аналогичным образом на основе операторов ассемблера архитектуры Intel x86, однако, с небольшим изменением: результирующие значения, полученные в результате выполнения команд, хранятся в отдельно заведённых для этого регистрах, имена которых соответствуют выполняемым операциям.

Учитывая изложенные выше положения, исходный код инвертора, описанного на языке Verilog (Листинг 1), примет следующий вид:

```
cmp(in, *)
jnz @alwsend
cmp(in, 1)
jnz @if2
mov(out,0)
@if2:
cmp(in, 0)
jnz @if3
mov(out,1)
@if3
@alwsend
```

Листинг 2 - Машинный код, описывающий работу инвертора.

Для перехода в сети Петри данный набор команд будет представлен массивом инструкций, являясь параметром перехода (цветной маркировкой) tokens. Для облегчения обработки условных переходов также предусмотрен ассоциативный контейнер jumps, ключом в котором выступает имя метки, к которой осуществляется переход, а значением – индекс данной метки в массиве инструкций tokens.

При срабатывании перехода в сети Петри с таким набором команд, первым делом произойдёт сравнение текущего значения на входе in с контрольным значением, указанным во втором аргументе команды cmp. Если условием выполнения процедурного блока является не значение входного сигнала, а его состояние (фронт, срез), то вторым аргументом в команде cmp будут символы «/» или «\» в зависимости от состояния сигнала. Однако, если список чувствительности задан лишь именем входа без указания состояния (posedge или negedge), то вторым аргументом команды сравнения будет указан символ «*», обозначающий любое состояние входного сигнала. В том случае, когда условие срабатывания процедурного блока не выполнено, флаг ZF принимает нулевое значение, и команда «jnz @alwsend» производит переход к метке @alwsend, в противном случае происходит выполнение непосредственно команд самого процедурного блока.

Программная реализация алгоритма. Для проверки работоспособности предложенного алгоритма на языке C++ была разработана программа логического моделирования цифровых комбинационных схем, в которой в полной мере реализован описанный выше адаптированный аппарат сетей Петри.

В качестве входных данных в данной версии используется упрощённое подмножество языка Verilog.

Временные диаграммы, иллюстрирующие корректную работу схемы триггера (Рисунок 4), полученные в результате моделирования разработанным симулятором, показаны на Рисунке 5.

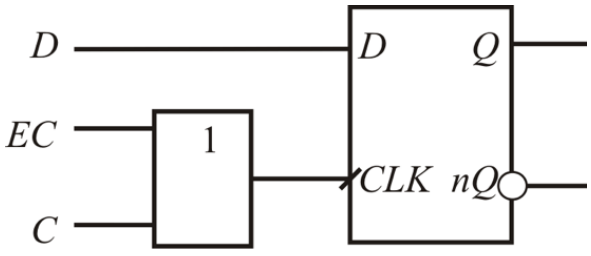

Рисунок 4 - Тестовая схема, иллюстрирующая проблему
выбора очерёдности переключений сигналов

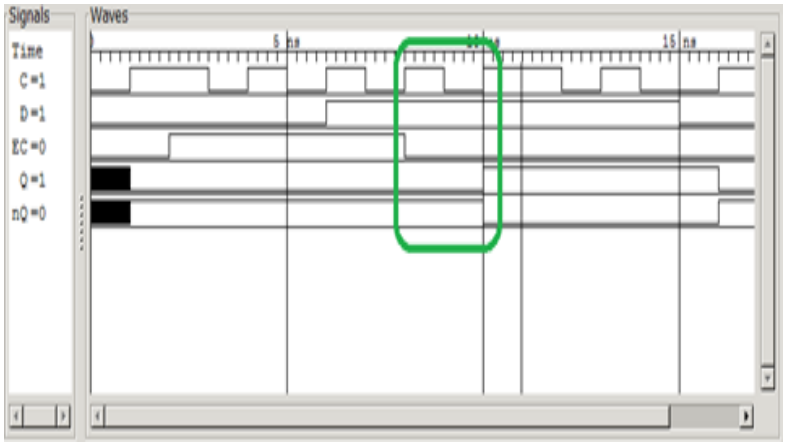

Рисунок 5 - Результат моделирования тестовой схемы с помощью симулятора, основанного на использовании адаптированного алгоритма сетей Петри

Заключение. Работоспособность разработанного алгоритма была проверена на следующих комбинационных и последовательностных схемах:

1. мультиплексоры, демультиплексоры различной разрядности, сумматоры;
2. сдвиговые регистры различной разрядности;
3. схемы из набора ISCAS'85.

Результаты моделирования схем с применением разработанного алгоритма соответствуют действительности.

ЛИТЕРАТУРА

1. Dan Joyce. 29 cost-effective gate-level simulation tips (pt 2) // deepchip.com. 2017. URL: http://www.deepchip.com/items/0569-03.html.
2. Булах Д.А., Казённов Г.Г., Лапин А.В. Использование модификации алгоритма работы сетей Петри для функционального моделирования логических схем, представленных на вентильном уровне// Изв. вузов. Электроника. - 2017. - Т. 22. - № 4. - С. 379-385. DOI: 10.214151/1561-5405-2017-22-4-379-385.
3. Kip R. Irvine. Assembly Language for x86 Processors (7th Edition) // Pearson. - 2014. ISBN 978-0133769401.

A.V. Lapin

APPLICATION OF ADAPTED PETRI NETS FOR SIMULATION OF LOGIC SCHEMES DESCRIBED ON BEHAVIOR LEVEL

National Research University of Electronic Technology, Moscow, Russia

In logical modeling there is such concept as a race condition – a state at which several input signals switch at the same time, creating ambiguity of an output state. For elimination of such ambiguity it is proposed to use such mathematical apparatus which will allow to

simulate parallel events. The main algorithm for simulation of digital circuits, applied in digital simulators, is the algorithm of the event driven simulation. However, the implicit choice of the sequence functioning of simultaneously switching signals may lead to an appearance of differences in the obtained simulation results. A new algorithm of the event driven functional simulation of digital integrated circuits, based on application of the adapted mathematical apparatus of the Petri nets, has been proposed. In the apparatus of the Petri nets the parallel constructions are imitated using the sequential instructions. Hence, there is no need to separate the events using delta-delays. It has been shown that the described approach enables to eliminate the ambiguity of switching signals, which occur at the same time due to failure because of using the delta-delay. The results of the algorithm work have been presented on an example of simulation on behavioral and the gate level of combinational and sequential schemes have been presented. The obtained time diagrams as well as the simulation time show that the algorithm proposed does not yield to existing simulation tools in terms of reliability and simulation time.

Keywords: logic simulation, event-driven simulation, Petri nets.

REFERENCES

1. Dan Joyce. 29 cost-effective gate-level simulation tips (pt 2). // deepchip.com. 2017. URL: http://www.deepchip.com/items/0569-03.html.
2. Bulakh D.A., Kazennov G.G., Lapin A.V. Use of Modification of Petri Nets Operation Algorithm for Functional Simulation of Gate Level Digital Circuits// Proc. of Universities. Electronics. - 2017. - Vol. 22. - No. 4. - P. 379-385. DOI: 10.214151/1561-5405-2017-22-4-379-385
3. Kip R. Irvine. Assembly Language for x86 Processors (7th Edition) // Pearson. - 2014. ISBN 978-0133769401.