VHDL МОДЕЛЬ RISC ПРОЦЕССОРА ДЛЯ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Белецкий В.Н., Чемерис А.А. , Глушко Е., Витер В.
Институт проблем моделирования в энергетике НАН Украины,
Национальный технический университет Украины «КПИ»

Рассматривая проблемы построения систем автоматики и управления, а особенно встроенных систем для управления объектами с быстропротекающими процессами, становиться очевидным необходимость проектирования высокопроизводительных архитектур, цена которых значительно ниже существующих устройств при большей производительности. Современная технология предоставляет возможность проектирования быстродействующих устройств за короткое время. В частности, программируемые логические интегральные схемы (ПЛИС) представляют такой класс микросхем, которые позволяют проектировать устройства, работающие с частотой 200-300 МГц и динамически способны менять функциональные свойства. Работы авторов направлены на разработку систем автоматического проектирования устройств, реализованных в ПЛИС, и проектирование архитектур и их моделей, погружаемых в ПЛИС.

В данной статье рассматривается модификация RISC архитектуры, которая совмещает в себе особенности потоковой и фон-Неймановской. Она характеризуется тем, что команды программы располагаются в памяти по мере готовности операндов, образуя тем самым потоковый граф алгоритма задачи. Приведенная потоковая архитектура реализована как поведенческая модель на языке VHDL [1]. В качестве системного программного обеспечения разработан ассемблер, поддерживающий работу VHDL модели. RISC процессор разработан для решения задач диспетчерского управления в реальном масштабе времени.

Интерес к потоковым архитектурам обусловлен несколькими важными факторами, которые существенно отличают потоковые компьютеры от известных параллельных систем [2]. Прежде всего, потоковые компьютеры коренным образом отличаются от

архитектуры фон-Неймана, на основе которой построено большинство используемых компьютеров. В погоне за максимальной производительностью, уход от принципов построения архитектуры фон-Неймана должен основываться на максимальном совпадении алгоритма задачи и структуры вычислительной системы. Потоковые системы с этой точки зрения являются наиболее перспективными.

Разработанная архитектура совмещает в себе несколько принципов. Архитектура фон-Неймана определяет последовательную адресацию памяти, от RISC архитектуры взят длинный формат команды и ограниченный набор команд, потоковая архитектура определяет состав команд и метод их обработки. Формат командной ячейки приведен на рис. 1, где КОП – код операции; ОП1, ОП2 – соответственно, первый и второй операнды; П – признак продолжения рассылки; ЗУ – номер операнда (ОП1 или ОП2), куда поместить результат; АР – адрес рассылки, т.е. адрес команды куда следует поместить результат. Значения операндов располагаются непосредственно в команде в формате целого числа или числа с плавающей точкой.

КОП	ОП1	ОП2	П	ЗУ	АР
79 - 64	63 - 32	31 - 0	15	14 - 13	12 - 0

Рисунок 1 -Формат командной ячейки

Проектированию процессора предшествовал период моделирования различных вариантов потоковых многопроцессорных архитектур. Мы пришли к выводу, что «узким» местом не только потоковой, но и практически любой многопроцессорной системы является шина память-процессоры. Т.е. нет необходимости увеличивать количество процессорных элементов системы, необходимо найти способ их эффективно загрузить работой. Таким образом, на наш взгляд целесообразно ввести в систему два процессора. Первый работает с данными в целочисленном формате, а второй – с вещественными данными, которые представлены в формате IEEE. Основное внимание уделялось построению структуры и алгоритмам функционирования памяти и программному обеспечению, которое состоит из Си-транслятора, ассемблера и

загрузчика. Их основное назначение – транслирование исходного текста программы в последовательность внутренних команд процессора в порядке их выполнения.

Структурная схема RISC процессора приведена на рис.2.

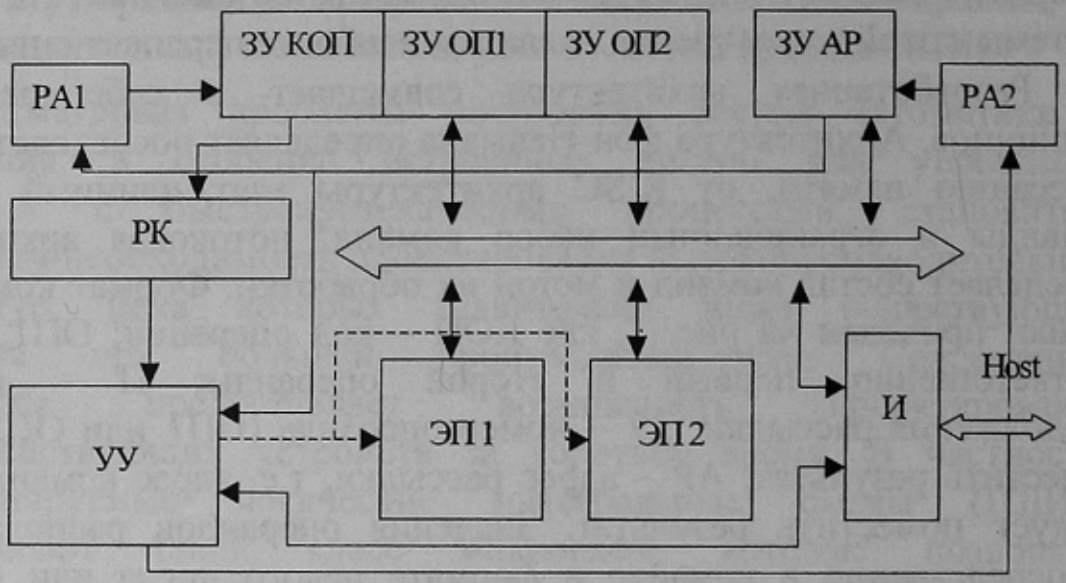

Рисунок 2 - Структурная схема RISC процессора

В качестве ядра процессора используется два элементарных процессора ЭП1 и ЭП2 для обработки целых и вещественных чисел, которые могут работать параллельно. Программа размещается в двух запоминающих устройствах (ЗУххх), разделенных физически и имеющих отдельные регистры адреса, соответственно, РА1 и РА2. Структура ЗУххх соответствует командной ячейке, приведенной на рис.1. На каждом такте работы код текущей, готовой к выполнению команды, записывается в регистр команд РК.

В зависимости от типа команды работает ЭП1 или ЭП2 и затем результат рассылается в соответствии с адресами рассылки, которые записаны в ЗУ АР. Потоковый процессор через интерфейсный блок И соединяется с хост-компьютером.

Для проектирования программ, выполняемых процессором, разработан ассемблер, который поддерживает следующую мнемонику команд программы.

<М>: <КК> <ОП1> [<ОП2>] <САР>

где М – метка; КК – код команды; ОП1, ОП2 – соответственно, операнд1 и операнд2; САР – список адресов рассылки. САР представляет из себя список меток соответствующих команд, куда требуется разослать результат выполнения команды.

Авторами разработана поведенческая модель потокового процессора на языке описания электронных схем VHDL, позволяющая проверить правильность функционирования разработанной архитектуры. Структура реализована в виде отдельных компонент, которые потом объединены в общую систему. Для связи VHDL-модели с разработанным компилятором создана специальная программа, генерирующая загрузочный код в виде, который воспринимается моделью. Организация вычислений в представленной архитектуре рассматривается на примере вычисления факториала числа *n*.

В качестве примера приведена программа умножения матриц.

```
.START
_A: .WORD 625
_B: .WORD 625
_C: .WORD 625
_m: .INT 0
_n: .INT 0
_r: .INT 0
.LISTIN _m, _n, _r, _A, _B
.LISTOUT _C
$mproduct:
L1: RDV OP1,#_r  1,L2 1,L5 1,L13
@j:
L2: DEC3 OP1,#_r  1,L2 2,L6 2,L14 1,L27
L3: RDV OP1,#_m 1,L4
@i:
L4: DEC3 OP1,&  1,L4 2,L5 2,L10 1,L26
L5: FIMUL3 OP1,& OP2,&  1,L6
L6: FIADD3 OP1,& OP2,&  2,L7
L7: FISUB3 OP1,#_C OP2,&  2,L25
L8: RDV OP1,#_n  1,L9 1,L10
```

```
L81: JOIN OP1,0  1,L23
@R:
L9: DEC3 OP1,&  1,L9 2,L11 1,L24 2,L13
L10: FIMUL3 OP1,& OP2,&  1,L11
L11: FIADD3 OP1,& OP2,&  2,L12
L12: FISUB3 OP1,#_A OP2,&  1,L20
L13: FIMUL3 OP1,& OP2,&  1,L14
L14: FIADD3 OP1,& OP2,&  2,L15
L15: FISUB3 OP1,#_B OP2,&  1,L21
L20: RDV OP1,&  1,L22
L21: RDV OP1,&  2,L22
L22: FMUL32 OP1,& OP2,&  2,L23
L23: FADD32 OP1,& OP2,&  1,L23 1,L25
L24: FISUB1 OP1,& OP2,0
L241: NEQLFI @R
L25: KOM OP1,& OP2,&
L26: FISUB1 OP1,& OP2,0
L261: NEQLFI @i
L27: FISUB1 OP1,& OP2,&
L271: NEQLFI @j
L30: STOP
.END
```

Особенностью программы являются операторы .LISTIN _m, _n, _r, _A, _B и .LISTOUT _C, в которых задаются соответственно входные параметры программы и ячейка памяти с результатом. В табл.1 приведены результаты моделирования в пакете Active-VHDL этой программы, а именно, время выполнения матричного умножения квадратных матриц в RISC- процессоре в зависимости от размерности n входных данных. На рис.3 эта информация представлена в графическом виде.

Таблица 1.

n	T, ms
5	0,406
10	1,752
15	5,331
20	12,239
25	23,572

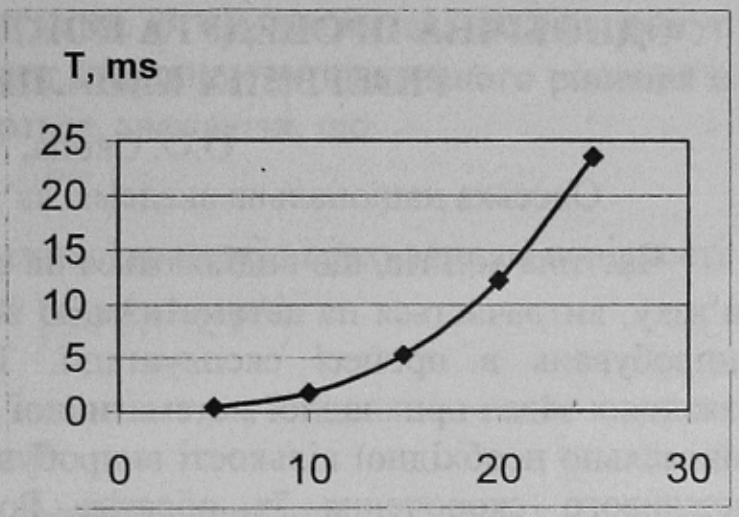

Рисунок 3 - Результаты моделирования

Выводы. Моделирование показывает, что нет необходимости в накапливании готовых к выполнению команд в отдельном запоминающем устройстве [3]. Увеличение количества процессоров не приносит существенного выигрыша, кроме как для задач, которые хорошо распараллеливаются. Таким образом, авторы остановились на двухпроцессорном варианте RISC процессора, в котором на предварительной стадии обработки решаемой задачи на основе потокового графа генерируется программа, в которой команды выстроены в порядке их готовности к выполнению. Введение такого детерминизма в потоковую архитектуру не сказывается на производительности вычислений, уменьшая при этом аппаратные затраты. Возможность увеличения производительности состоит в том, чтобы из таких RISC-процессоров проектировать многопроцессорные системы различной топологии. При этом требуется проектирование специальных планировщиков заданий и другое программное обеспечение.

Список источников

1. IEEE standard VHDL language reference manual. IEEE std. 1076-1993. The Institute of Electrical and Electronic Engineeres, Inc., 1994.

2. Dennis J. "Data flow ideas for supercomputers". IEEE Society, 28th international conference, San Francisco, 1984, pp. 15-19.

3. Белецкий В.Н., Ронто В.А., Крапивка В.И. Организация и моделирование потоковых ЭВМ. (Препринт ИПМЭ АН УССР 89-1)– Киев: Ин-т проблем моделирования в энергетике, 1989. – 44 с.