УДК 004.2

**Р. А. ХИСАМУТДИНОВ**

# СИСТЕМАТИЗАЦИЯ АРХИТЕКТУР ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Рассмотрены некоторые известные классификации вычислительных систем и предложена новая таксономия архитектур, базирующаяся на концепции сигналов. Приведены примеры описания некоторых известных архитектур. *Архитектура вычислительных систем*; *классификация*; *обработка информации*

## ВВЕДЕНИЕ

Рост производительности вычислительной техники впечатляет, а повышение плотности компоновки интегральных микросхем позволяет увеличивать объем памяти и разрядность процессоров. Но какой совершенной не была бы технология, существует естественный способ увеличения производительности без изменения элементной базы — параллельной обработки данных, — в которой задействовано множество однотипных обрабатывающих устройств. Известно множество архитектурных решений параллельных вычислительных систем и множество названий таких архитектур, обозначающих общие принципы обработки — массивно-параллельные, симметрично-многопроцессорные, кластерные, векторные и многие другие.

Но создание более «быстрых» компьютеров не всегда позволяет пропорционально уменьшить время, затрачиваемое на решение задачи. Это связано с тем, что различные архитектурные особенности, такие как способ доступа к памяти, разделение команд и данных, организация межпроцессорной связности, приводят к специализированности параллельных вычислительных систем, и эта специализация заключается в адекватности архитектуры какому-то классу алгоритмов. Другими словами, для уменьшения времени счета конкретного алгоритма необходимо исполнять его на максимально «подходящей» архитектуре. Широко известны лишь некоторые частные правила, например, использование векторных систем для матричных операций или высокая эффективность кластерных вычислительных систем на алгоритмах с большим временем счета отдельных гранул и малым объемом межпроцессных пересылок.

Современные алгоритмы могут включать фрагменты различного кода — обработки общих данных, операций над векторами, передачу сообщений. Такое многообразие делает определение адекватности алгоритма сложной задачей, требующей детализации как алгоритма, так и архитектуры вычислительной системы.

Описание архитектуры, включающее количественные параметры, очень важно для адекватного представления алгоритма, подготовленного к исполнению на данной архитектуре. Но только достаточно простые алгоритмы можно представить в виде одноуровневой схемы. Как правило, в кодированном алгоритме (программе) можно выделить несколько уровней вложенности. Программа состоит из подпрограмм (гранул), гранулы могут включать в себя другие гранулы, и такое включение может быть многоуровневым. Самые маленькие гранулы состоят из отдельных команд языка программирования, которые преобразуются транслятором в команды процессора. Команды процессора тоже раскладываются на микрокоманды, которые и исполняются соответствующими функциональными блоками процессора. Адекватность сложного многоуровневого алгоритма архитектуре может быть только в том случае, если вычислительная система является многоуровневой. Так, к примеру, архитектуру известного суперкомпьютера Earth Simulator можно представить как кластер из 640 узлов, в котором каждый узел есть SMP-система, состоящая из 8 процессорных модулей. В каждом процессорном модуле есть суперскалярный процессор и векторный процессор, обрабатывающий векторы до 256 элементов.

Целью данной работы является разработка системной классификации архитектур вычислительных систем, позволяющей оценивать системы с точки зрения эффективности процессов обработки информации. В первой части на основе [1, 2] будут рассмотрены наиболее известные классификации архитектур

вычислительных систем. Вторая часть в достаточно компактной форме излагает новый способ классификации архитектур систем обработки информации (и, в частности, вычислительных систем).

## ИЗВЕСТНЫЕ КЛАССИФИКАЦИИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Поиск подходящей архитектуры — нетривиальная задача, в первую очередь, из-за того, что все многообразие компьютерных архитектур плохо систематизировано.

В 1972 г. М. *Флинн* (*M. Flynn*) опубликовал [3] вариант классификации вычислительных систем, который базировался на понятиях потоков команд и потоков данных. Поток — последовательность команд или данных, которая может быть обработана одним устройством. Флинн выделил 4 возможных класса вычислительных систем по количеству используемых потоков:

- SISD — Single Instruction stream, Single Data stream;
- SIMD — Single Instruction stream, Multiple Data stream;
- MISD — Multiple Instruction stream, Single Data stream;
- MIMD — Multiple Instruction stream, Multiple Data stream.

Недостаток такой классификации — в ее общности. Некоторые виды архитектур не четко относятся к тому или иному классу. С другой стороны, класс MIMD машин явно «перегружен». К этому классу можно отнести вычислительные системы с принципиально разной архитектурой, а это не способствует лучшему пониманию.

Работа Флинна положила начало целому направлению в теории вычислительных систем — computer taxonomy.

Наиболее известны следующие попытки доработать классификацию Флинна:

- K. *Hwang* (Ванг), *F. A. Briggs* (Бриггс) (1984) [4]. В классе SISD были выделены 2 подкласса — с одним и несколькими функциональными устройствами; в классе SIMD — с пословно-последовательными и разрядно-последовательными устройствами; в классе MIMD — с распределенной и общей памятью;
- R. *Hockney* (Хокни) (1986) [5]. Класс MIMD был расширен на *переключаемые* (с общей и с распределенной памятью), *конвейерные* и *сети* (регулярные решетки, гиперкубы, иерархические структуры и с изменяемой конфигурацией);
- *E. E. Johnson* (Джонсон) (1988) [6]. В класс MIMD вводятся четыре подкласса, основанные на двух характеристиках — общая/распределенная память и разделяемые переменные/обмен сообщениями;
- R. *Duncan* (Дункан) (1990) [7]. 4 класса, введенные Флинном, были дополнены еще двумя — классом синхронных систем, включающим в себя векторные, систолические, а также SIMD системы матричного типа или с ассоциативной памятью, и классом систем, использующих идеи MIMD (dataflow, reduction, wavefront и MIMD/SIMD). Класс «чистых» MIMD систем Дункан разделил на системы с общей и с распределенной памятью.

В то же время предлагались варианты классификации вычислительных систем на принципах, отличных от принципов Флинна. В числе наиболее известных классификации:

- T. *Feng* (Фенг) (1972) [8]. Определены 4 класса, созданные из 2-х основных характеристик — разрядности машинного слова и числа одновременно обрабатываемых слов;
- *J. Shore* (Шор) (1973) [9]. Выделены 6 классов архитектур на базе фиксированного числа функциональных блоков — устройства управления, АЛУ, памяти команд и памяти данных;
- A. *Basu* (Базу) (1987) [10]. Предложена классификация на основе последовательности решений, принятых при проектировании системы, а именно: уровень параллелизма, метод реализации алгоритма, параллелизм исполнения инструкций, способ управления. На основании этих характеристик Базу предложил 10 возможных архитектурных решений;
- *E. V. Krishnamurthy* (Кришнамарфи) (1989) [11] фактически перенес идеи Базу на другую иерархию, выделив степень гранулярности, способ реализации алгоритма, топологию и природу связи процессоров, способ управления. Этот способ классификации дает возможность описать и «нетрадиционные» архитектуры, такие как систолические матрицы и wavefront;
- *D. Scillicorn* (Скилликорн) (1989) [12]. Наиболее оригинальный подход, позволивший ввести 28 классов систем, используя 4 основные компоненты системы — процессор команд, процессор данных, иерархию памяти и один из 4-х типов переключателей;
- *S. Dasgupta* (Дазгупта) (1990) [13]. Базируясь на идеях Скилликорна, Дазгупта предложил уточнения, касающиеся кэш-

памяти и описания межпроцессорных коммуникаций.

## О НОВОЙ ТАКСОНОМИИ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

У всех приведенных выше классификаций есть недостатки. Общий заключается в том, что они описывают *цифровые электронные вычислительные машины* и к тому же почти все классификации носят описательный характер, т. е. базируются на современных (к моменту создания классификации) представлениях о существующих ЦЭВМ. Некоторые варианты классификации описывают, по сути, топологию многопроцессорной системы. Топология системы есть способ соединения элементов системы, а не принцип ее функционирования. Это приводит к тому, что за пределами оказывается принцип обработки и обмена информацией. Это не способствует лучшему пониманию функционирования той или иной компьютерной архитектуры и, конечно же, не приближает нас к ответу на основной вопрос — насколько эффективно использование конкретного алгоритма на данной архитектуре или для каких алгоритмов какие архитектуры наиболее приспособлены.

Исходя из этого, при систематизации архитектур следует исходить из следующих положений:

- Систематизация должна быть концептуальной, а не описательной;
- Систематизация должна охватывать все системы обработки информации, а не только ЦЭВМ;
- Систематизация должна опираться на принципы информационного обмена, а не на физическую топологию системы;
- Систематизация должна быть иерархической.

В данной работе предлагается другой взгляд на системы обработки информации, базирующийся на типах и направлениях сигналов.

Обработка информации есть процесс изменения входной информации согласно определенным правилам. В общем случае система обработки информации должна содержать устройство обработки и хранилище правил обработки. Для обобщения устройств обработки информации, как технических, так и природных, примем, что информация представлена в виде сигналов — знаков, физических процессов или явлений, несущих сообщение о каком-либо событии, состоянии объекта либо передающих команды управления или оповещения.

При рассмотрении системы обработки информации, например, ЦЭВМ, аналоговой вычислительной машины, биологической нейросистемы, можно исходить из следующих принципов:

1) Система принимает входной сигнал, преобразует его в выходной сигнал.

2) Правила обработки сигнала находятся в хранилище.

3) Обрабатывающее устройство при обработке входного сигнала руководствуется правилами из хранилища.

4) Для кратковременного хранения промежуточной информации (при необходимости) также используется хранилище информации.

Переходя на описание обрабатывающего устройства (например, процессора ЦЭВМ), мы обнаружим, что эта же модель полностью соответствует устройству. В этом случае обрабатывающим устройством будет арифметико-логическое устройство с устройством управления, а хранилищем информации будут являться блок микропрограммного управления и блок регистров. Углубляясь в детализацию, можно достичь уровня отдельного транзистора, и тогда входным сигналом будет являться сигнал на базе транзистора, выходным – сигнал на эмиттере, а «хранилищем информации» — характеристики $p$–$n$ перехода. Другими словами, при создании $p$–$n$ перехода закладывается информация о том, как обрабатывать входящий сигнал, т. е. когда открывать переход. Такая же аналогия прослеживается и при рассмотрении нейрона.

Вся информация, как входящая в систему, так и циркулирующая внутри системы, может быть определена двумя классами:

1) Информация, подлежащая обработке, — данные;

2) Информация, определяющая процесс обработки, — команды.

Информация 2-го класса может в системе не присутствовать в следующих случаях:

- устройство обработки информации способно производить только одну определенную функцию; такие устройства в дальнейшем будем называть *простыми*;
- функция обработки информации определяется содержанием информации (обычно это называется управлением потоком данных).

В процессе обработки информации информационный обмен происходит в двух основных контурах:

1) Между устройствами обработки и устройствами хранения,

2) Между устройствами обработки (если таковых несколько).

Предполагается, что непосредственный информационный обмен между устройствами хранения не происходит.

В первом контуре существуют 3 вида сигналов:

- M_RI — запрос команды из памяти для исполнения или передачи;
- M_RD — запрос данных из памяти для обработки или передачи;
- M_SD — запись данных (результата обработки) в память.

В этом контуре мы не вводим дополнительный сигнал записи команды. Обрабатывающее устройство может иметь множество входных сигналов, но имеет только один выходной сигнал для передачи результата. Подразумевается, что команда, требующая сохранения, есть результат исполнения какой-либо команды, т. е. относится к данным.

Во втором контуре (контуре обработки) существует 2 вида сигналов:

- P_I — управляющие сигналы между обрабатывающими устройствами;
- P_D — передача данных.

Связность между контурами определяется в следующих группах сигналов:

- M_RI – P_I — получение команд из памяти и распространение этих команд между устройствами обработки;
- M_RD – P_D — получение данных из памяти и распространение этих команд между устройствами обработки;
- P_D – M_SD — сбор данных с устройств обработки и передача их в устройства хранения.

Количественные характеристики для устройств обработки или хранения следующие:

- 1 — устройство присутствует в системе в единственном экземпляре;
- $n$ — в системе присутствует несколько подобных устройств.

Передача сигналов требует наличия специальных устройств — соединителей. Для каждого типа сигнала определим свой уникальный соединитель. Каждый соединитель имеет три характеристики — тип передаваемого сигнала, количество подключенных устройств на передающей стороне и количество подключенных устройств на принимающей стороне. Для сопряжения устройств определим следующие числовые характеристики соединителей:

1 — соединитель подключается только к одному устройству;

$k$ — соединитель подключается к нескольким (не всем!) устройствам;

$n$ — соединитель подключается ко всем устройствам в группе.

Можно определить несколько типов соединителей, которые показаны в табл. 1, где


— хранилище информации,

— устройство обработки.

Несложно подсчитать количество возможных вариантов при использовании всех видов соединителей. Но многие конфигурации являются нереализуемыми в силу следующих основных причин:

1) каждое обрабатывающее устройство должно иметь доступ не менее чем к одному устройству хранения; доступ может быть непосредственным или обеспечиваться транзитом через другое обрабатывающее устройство;

2) в системе не должно быть незадействованных устройств.

В результате применения топологических правил связности можно выделить около 1300 допустимых архитектур. Некоторые распространенные архитектуры могут быть описаны следующим образом:

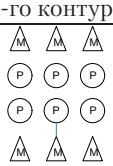
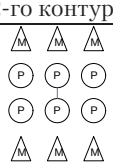
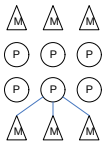
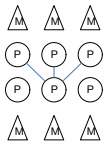
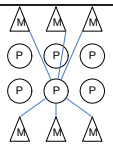
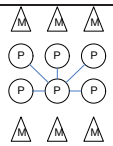
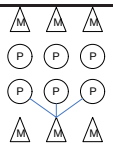
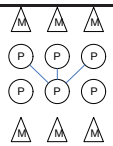
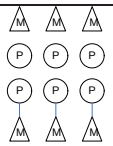
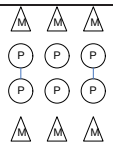
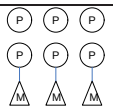
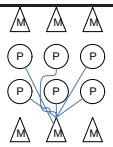
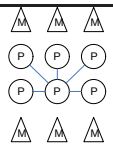
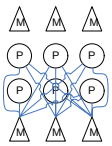
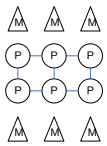
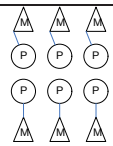
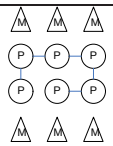
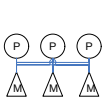
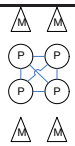
1) Классическая фон-неймановская архитектура:

- устройств обработки — 1;
- устройств хранения — 1;
- чтение команд из хранилища — L1-1;
- чтение данных из хранилища — L1-1;
- запись данных в хранилище — L1-1;
- передача команд между устройствами обработки — нет;
- передача данных между устройствами обработки – нет.

2) SMP-система на базе процессоров Intel x86 с одноканальной памятью:

- устройств обработки — $n$;
- устройств хранения — 1;
- чтение команд из хранилища — Ln-1;
- чтение данных из хранилища — Ln-1;
- запись данных в хранилище — Ln-1;
- передача команд между устройствами обработки — нет;

Таблица 1

| Тип | Описание | Пример 1-го контура | Пример 2-го контура |
|---|---|---|---|
| L1-1 | Соединение 2-х устройств. Обеспечивает соединение типа «точка-точка» |  |  |
| L1-k | Соединение одного устройства с несколькими. В 1-м контуре определяется, что сигнал данного типа считывается (или записывается) из/в нескольких устройств хранения. Примером могут служить системы гарвардской архитектуры, в которых память команд и данных разделена по разным банкам. В контуре «обработка-обработка» определяется, что сигнал передается только нескольким обрабатывающим устройствам, при этом остаются устройства, не получившие данного сигнала |  |  |
| L1-n | Соединение одного устройства со всеми. В контуре «обработка-хранение» определяет что сигнал может быть считан (или записан) из любого устройства хранения. Схема характерна для современных однопроцессорных систем с многоканальной памятью. В контуре «обработка-обработка» подразумевается, что один обработчик передает сигнал всем другим обработчикам |  |  |
| Lk-1 | Соединение нескольких устройств с одним. В первом контуре предполагается, что сигнал может быть инициирован несколькими обработчиками. Во втором контуре наличие данного соединителя предполагает задействование только нескольких устройств |  |  |
| Lk-k | Соединение посредством $k$ соединителей типа L1-1 |  |  |
| Lk-n | Соединение аналогично Lk-$k$ при том, что задействуются все устройства хранения, т. е. $k$ (обработчиков) = $n$ (хранилищ). Для второго контура нереализуемо |  | |
| Ln-1 | Соединение всех устройств с одним |  |  |
| Ln-k | В первом контуре наличие такого соединителя предполагает $n$ (обработки) = $k$ (хранения), т. е. у каждого устройства обработки для отправки данного сигнала есть «свое» устройство хранения, а общее количество устройств хранения больше, чем количество устройств обработки. Во втором контуре такой соединитель может быть представлен в виде $n(1 - k)$, т. е. каждое устройство обработки может отправлять данный сигнал нескольким другим устройствам (локальная связанность) |  |  |
| Ln-n | В первом контуре наличие такого соединителя предполагает равное количество устройств обработки и устройств хранения и реализует модель соединения $n(L1-1)$. Логическое распространение модели $n(L1-1)$ на второй контур приводит к модели последовательного соединения устройств обработки |  |  |
| Lnxn | Конфигурация с полным доступом. В первом контуре означает возможность посылки сигнала любым обработчиком любому хранилищу. Во втором контуре — возможность посылки сигнала любому обработчику |  |  |

• передача данных между устройствами обработки — нет.

3) SMP-система на базе процессоров AMD Opteron 2xx с 2-канальной памятью:

• устройств обработки — $n$;

• устройств хранения — $n$;

• чтение команд из хранилища — Ln-$n$;

• чтение данных из хранилища — Ln-$n$;

• запись данных в хранилище — Ln-$n$;

• передача команд между устройствами обработки — Ln-$k$;

• передача данных между устройствами обработки — Ln-$k$.

4) Dataflow-система:

• устройств обработки — $n$;

• устройств хранения — $n$;

• чтение команд из хранилища — нет;

• чтение данных из хранилища — Lk-$k$;

• запись данных в хранилище — Lk-$k$;

• передача команд между устройствами обработки — Ln-$k$;

• передача данных между устройствами обработки — Ln-$k$.

Таким же способом можно описать архитектуры и других систем обработки информации.

## ЗАКЛЮЧЕНИЕ

Предложенная классификация решает основную задачу — позволяет классифицировать системы обработки информации по основным критериям их функционирования. Это позволяет минимизировать разрыв между описанием алгоритма и описанием вычислительной системы и упрощает оценку адекватности алгоритма и системы.

Несомненным преимуществом предложенной классификации является возможность использовать ее и для описания архитектур. Предварительные эксперименты показывают, что, используя предложенный способ классификации для описания системы, можно идентифицировать системы на том уровне, который мы хотим использовать для оценки своего алгоритма, и оценивать пригодность синтезированной архитектуры для исполнения алгоритма.

## СПИСОК ЛИТЕРАТУРЫ


1. **Воеводин, В. В.** Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. СПб. : БХВ-Петербург, 2002. 608 с.
2. **http://www.parallel.ru/computers/taxonomy/** [Электронный ресурс].
3. **Flynn, M.** Some Computer Organizations and Their Effectiveness / M. Flynn // IEEE Trans. Comput. 1972. Vol. C-21. P. 94.
4. **Hwang, K.** Computer Architecture and Parallel Processing / K. Hwang, F. A. Briggs. 1984. P. 32–40.
5. **Hockney, R.** Parallel computers: architecture and performance / R. Hockney // Proc. of Int. Conf. Parallel Computing'85. 1986. P. 33–69.
6. **Johnson, E. E.** Completing an MIMD multiprocessor taxonomy / E. E. Johnson // Computer Architecture News. 1988. V. 16. № 2. P. 44–48.
7. **Duncan, R.** A survey of parallel computer architectures / R. Duncan // Computer. 1990. V. 23. № 2. P. 5–16.
8. **Feng, T.** Some characteristics of assotiative parallel processing / T. Feng // Proc. 1972 Sagamore Computing Conf. 1972. P. 5–16.
9. **Shore, J. E.** Second thoughts on parallel processing / J. E. Shore // Comput. Elect. Eng. № 1. P. 95–109.
10. **Basu, A.** Parallel processing systems: a nomenclature based on their characteristics / A. Basu // Proc. IEE (UK). № 134. 1987. P. 143–147.
11. **Krishnamurthy, E. V.** Parallel processing principles and practice / E. V. Krishnamurthy. Addison-Wesley Pub. Company, 1989. P. 208–246.
12. **Skillicorn, D.** A taxonomy for computer architectures / D. Skillicorn // Computer. 1988. V. 21. № 11. P. 46–57.
13. **Dasgupta, D.** A Hierarchical taxonomic system for computer / S. Dasgupta // 1990. V. 23. № 3. P. 64–74.