УДК 621.38
doi:10.31799/1684-8853-2019-4-9-18

Научные статьи
Articles

Отказоустойчивая и энергоэффективная система обработки информации и управления на кристалле

А. М. Грузликов[a], *канд. техн. наук, начальник отдела, orcid.org/0000-0001-8814-0726*
Н. В. Колесов[a], *доктор техн. наук, профессор, orcid.org/0000-0003-3287-7504, kolesovnv@mail.ru*
Д. В. Костыгов[a], *аспирант, orcid.org/0000-0003-4379-5803*
М. В. Толмачева[a], *канд. техн. наук, старший научный сотрудник, orcid.org/0000-0003-0795-7617*
[a]*АО «Концерн «ЦНИИ «Электроприбор», Малая Посадская ул., 30, Санкт-Петербург, 197046, РФ*

Постановка проблемы: *большинство реальных сложных систем проектируются с учетом требований отказоустойчивости, однако при этом все известные подходы имеют целью лишь повышение надежности.* ***Цель:*** *разработка подхода для проектирования отказоустойчивых систем на кристалле, нацеленного не только на повышение надежности, но и на снижение потребляемой системой мощности.* ***Результаты:*** *предложен двухэтапный подход к построению отказоустойчивых систем на многоядерном кристалле. На первом этапе формируется энергоэффективная архитектура проектируемой системы. При этом для каждого используемого в системе ядра определяется оптимальное в рамках существующих ограничений число дополнительных ядер. Критерием оптимальности при этом служит минимум потребляемой системой мощности. Предложенный алгоритм формирования энергоэффективной архитектуры основан на зависимости потребляемой в системе мощности от значений напряжения питания и тактовой частоты. На втором этапе разрабатывается процедура диагностирования и восстановления системы, использующая принципы распределенного диагностирования, предполагающего взаимные проверки между ядрами системы. Процедура позволяет децентрализовать процесс диагностирования и восстановления системы после отказов. Дополнительно в статье исследуется организация коммуникационной подсистемы, основу которой составляет общая память. Исследование опирается на проведенное моделирование в целях оценивания времени принятия решения об отказе в системах типа решетка, тор и гиперкуб.* ***Практическая значимость:*** *предложенный подход позволяет обеспечивать при проектировании системы необходимые значения для двух важнейших ее характеристик — отказоустойчивости и энергоэффективности. При этом обеспечивается децентрализация при принятии решений об отказе и восстановлении и, как следствие, повышение надежности системы .*

Ключевые слова *— отказоустойчивость, рем-модель, децентрализованная система, энергоэффективность, система на кристалле.*

Для цитирования: Грузликов А. М., Колесов Н. В., Костыгов Д. В., Толмачева М. В. Отказоустойчивая и энергоэффективная система обработки информации и управления на кристалле. *Информационно-управляющие системы*, 2019, № 4, с. 9–18. doi:10.31799/1684-8853-2019-4-9-18

For citation: Gruzlikov A. M., Kolesov N. V., Kostygov D. V., Tolmacheva M. V. Fault-tolerant and energy-efficient MCSoC for information processing and control. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2019, no. 4, pp. 9–18 (In Russian). doi:10.31799/1684-8853-2019-4-9-18

Введение

Разработка сложных систем обработки информации и управления всегда сопряжена с необходимостью удовлетворения целому набору технических требований и ограничений. Понятно, что в случае реализации этой системы на кристалле уровень ограничений лишь возрастает. В настоящей работе рассматривается последняя проблема с фокусированием на вопросах создания распределенной вычислительной отказоустойчивой системы реального времени. Очевидно, что проблема носит существенно комплексный характер. Как будет видно из дальнейшего изложения, кроме аспектов отказоустойчивости и энергопотребления, будут затронуты вопросы организации процесса диагностирования.

История исследований вопросов разработки отказоустойчивых систем насчитывает несколько десятилетий. Однако данная проблематика по-прежнему остается актуальной и широко освещается в современной литературе [1–5]. При этом традиционный подход обычно связывают с использованием того или иного варианта резервирования. Все эти варианты хорошо и давно известны специалистам. К числу их общих недостатков можно отнести, в частности, то, что построение осуществляется без учета уровня энергопотребления будущей отказоустойчивой системы. Эта проблема является одной из ключевых при проектировании вычислительных систем. В последние десятилетия ей уделяется особое внимание, в том числе и в связи с обсуждением ее в отношении систем на кристалле [6–9], когда снижение энергопотребления (рассеиваемой мощности) достигается за счет снижения тактовой частоты и напряжения питания. О признании практической значимости этого направления свидетельствует и тот факт, что разработчики процессоров в целях снижения энергопотреб-

ления стали предусматривать в своих проектах встроенные средства для варьирования тактовой частотой и напряжением питания.

Заметим, что необходимым элементом отказоустойчивой системы являются средства диагностирования, выполняемые на основе техник тестового и функционального диагностирования [10–14]. По их сигналам принимается решение о появлении отказа в системе, указывается его место и запускается процедура восстановления. Для сложных многопроцессорных вычислительных систем известен подход распределенного диагностирования, названный PMC-моделью (предложенной Ф. П. Препаратом, Г. Метцом и Р. Т. Ченом (F. P. Preparata, G. Metze and R. T. Chein)) [15–17], который основан на взаимных проверках, осуществляемых процессорами системы. За прошедшие десятилетия данный подход неоднократно уточнялся, однако при этом средства диагностирования, как правило, реализовались по централизованной схеме, когда выделяется некоторая часть их аппаратуры, в отношении которой делается предположение об априорной исправности. В результате в системе возникает «узкое место», отказ которого является катастрофическим для системы.

Основной вклад и практическая значимость настоящей статьи состоят в разработке подхода к построению децентрализованных отказоустойчивых и энергоэффективных систем, т. е. систем, лишенных «узких мест». Проблема рассматривается применительно к многоядерным системам на кристалле, что не исключает возможности применения полученных результатов и по отношению к другим типам многопроцессорных систем.

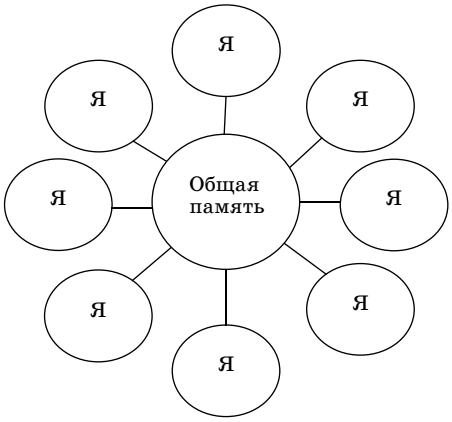
Материал излагается в следующем порядке. Обсуждается формирование архитектуры распределенной системы. При этом в ее состав так вводятся дополнительные ядра, чтобы снизить потребляемую системой мощность. На основе полученной архитектуры формулируется концепция отказоустойчивой системы, исследуются особенности коммуникационной системы и диагностического эксперимента.

Энергоэффективный алгоритм определения архитектуры системы

Постановка проблемы. Многоядерная гомогенная система на кристалле (рис. 1) включает совокупность одинаковых ядер *Я*, взаимодействующих через общую память. Будем считать, что каждое ядро оснащено устройством для параметрического управления его тактовой частотой и напряжением питания. Код настройки в это устройство может поступать как извне (начальная настройка), так и от самого ядра-хозяина в процессе функционирования.

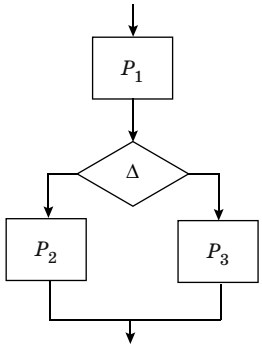
Рассмотрим проблему для такой системы реального времени с одним заданием. Сразу отметим, что, как будет пояснено в дальнейшем, случай с несколькими заданиями не слишком сильно отличается от рассматриваемого. Гипотетический пример граф-схемы задания показан на рис. 2. В системе реального времени это задание выполняется с периодом, равным периоду съема информации с датчиков. Предположим, что каждая операторная вершина P_1–P_3 задания реализуется отдельным программным модулем и выполняется на отдельном ядре. Таким образом, архитектуру A системы составляют три ядра. В дальнейшем каждое ядро исходной реализации системы будем называть стадией. Предположим, что на кристалле существуют дополнительные неиспользуемые ядра. Проблема определения архитектуры сводится к поиску такого перераспределения вычислительной нагрузки между всеми ядрами, чтобы мощность P, потребляемая системой, была минимальной:

$$A = \arg\min_{A} P(A). \qquad (1)$$



■ ***Рис. 1.*** Упрощенная архитектура многоядерного кристалла

■ ***Fig. 1.*** A architecture of MCSoCs



■ ***Рис. 2.*** Граф-схема задания

■ ***Fig. 2.*** Flowgraph of a task

Алгоритм определения архитектуры. Обсудим соотношения, описывающие потребляемую системой мощность P, поскольку именно она составляет основу критерия оптимизации на данном этапе.

Сразу отметим, что случай с минимизацией энергии вместо мощности аналогичен рассматриваемому ниже, поскольку энергия, потребляемая системой, равна произведению мощности на время работы системы. Известно [8], что мощность имеет две составляющие — динамическую P_d и статическую P_s. Выражения, описывающие эти составляющие без излишней для данного изложения детализации, имеют вид

$$P_d = aNV^2 f;\ P_s = bN, \qquad (2)$$

где a, b — коэффициенты пропорциональности, зависящие от свойств кристалла; N — число процессоров (ядер) в системе; V — напряжение питания; f — тактовая частота. Поскольку вклад статической мощности в суммарную потребляемую мощность невелик, далее будем учитывать лишь динамическую составляющую. Для ее анализа полезна приближенная формула, определяющая задержку, вносимую схемой при напряжении питания V [8]:

$$D = cV, \qquad (3)$$

где c — коэффициент пропорциональности, зависящий от свойств кристалла.

При снижении частоты тактовых импульсов в обратной пропорции возрастает их период, ограничивающий допустимое время для переходных процессов, возникающих в системе при каждом срабатывании. При исходном значении напряжения питания фактическое время переходных процессов будет мало по отношению к новому увеличенному значению периода, а значит, возникает возможность пропорционально снизить напряжение питания с увеличением задержки в рамках периода тактовых импульсов в соответствии с (3). В результате выполнения этих двух шагов может быть достигнуто существенное снижение потребляемой мощности (2). Действительно, пусть как частота, так и напряжение питания снижены в k раз. Тогда в соответствии с (2) динамическая мощность снижается в k^3 раз. При этом, правда, и время работы ядра увеличивается в k раз. Если для сохранения его на прежнем уровне увеличить в k раз число ядер, а вычислительную нагрузку разделить между всеми ядрами поровну, то в результате потребляемая мощность по отношению к исходному варианту уменьшится в k^2 раз. Описанный факт положен в основу предлагаемого подхода к определению архитектуры системы.

Суть подхода состоит в последовательном определении для каждой стадии количества реализующих ее ядер. Причем предполагается, что все ядра стадии работают на одной частоте и при одном напряжении питания, а также что для системы известен допустимый исходный вариант значений параметров f_0, V_0. Безусловно, проектируя систему на кристалле, разработчик всегда ограничен не только по потребляемой мощности, но также по площади s_0 кристалла, отведенной для реализации вычислительной системы:

$$s \le s_0, \qquad (4)$$

по напряжению питания

$$V \ge V_0 \qquad (5)$$

и частоте

$$f \le f_0. \qquad (6)$$

Зная размер площади s_k, занимаемой одним ядром, можно пересчитать ограничение по площади кристалла в допустимое число n_d дополнительных ядер:

$$n_d = \frac{s_0 - s_k n}{s_k}.$$

В общем случае, когда стадий несколько, возникает вопрос, как наилучшим образом с точки зрения минимизации потребляемой мощности распорядиться дополнительными ядрами (запасом по площади) в рамках ограничений (5) и (6). На него отвечает приводимый ниже алгоритм определения энергоэффективной архитектуры. Будем называть ядра исходной реализации системы «исходными», процесс добавления в i-ю стадию $n_{d,i}$ дополнительных ядер «расщеплением i-го исходного ядра», а образованное в результате этого расщепления множество ядер — «расщепленным множеством i-го ядра». Кроме того, будем считать расщепленное множество i-го ядра предельным, если исключение из него одного ядра делает его среднюю по множеству потребляемую мощность максимальной среди стадий системы. Интуитивно, по-видимому, ясно, что экономия мощности будет максимальной, если разгружаться будут наиболее загруженные ядра, а распределение нагрузки между ядрами в расширенном множестве будет осуществляться сбалансированным образом, т. е. равномерно. По этому принципу работает предлагаемый алгоритм. В дальнейшем этот принцип будет обоснован. Заметим, что идеальная сбалансированность при распределении нагрузки, когда нагрузка делится на равные части между ядрами

стадии, на практике в общем случае невозможна. В связи с этим речь идет лишь о приближенной сбалансированности. Архитектуру системы представим вектором состава $\mathbf{A} = (a_1\ a_2\ \ldots\ a_n)$, где a_i — число ядер в расщепленном множестве i-го ядра, и вектором средней потребляемой мощности $\overline{\mathbf{P}} = (\overline{P}_1\ \ \overline{P}_2\ \ \cdots\ \ \overline{P}_n)$, где $\overline{P}_i$ — средняя по расщепленному множеству i-го ядра потребляемая мощность. Итак, предлагается следующий простой алгоритм определения энергоэффективной архитектуры.

Шаг 1. Сделать начальные присвоения: $M = n_d$ (допустимое число дополнительных ядер), $\mathbf{A} = (1\ 1\ \ldots\ 1)$, $\overline{\mathbf{P}} = (\overline{P}_1\ \ \overline{P}_2\ \ \cdots\ \ \overline{P}_n)$.

Шаг 2. Выбрать в $\overline{\mathbf{P}} = (\overline{P}_1\ \ \overline{P}_2\ \ \cdots\ \ \overline{P}_n)$ компоненту с максимальным значением $\overline{P}_{\max}$. Пусть ее номер равен l. Ввести дополнительное ядро в l-ю стадию. Произвести между ядрами l-й стадии приближенно сбалансированное перераспределение нагрузки. Пересчитать параметры алгоритма: $\overline{\mathbf{P}}$, $a_l := a_l + 1$, $M := M - 1$. Если $M \neq 0$, то повторить шаг 2, иначе конец.

Проведем качественный анализ приведенного алгоритма. Нестрогость дальнейшего анализа связана с предположением о делимости нагрузки стадий при расщеплении ядер на необходимое для реализации шагов алгоритма число частей. В результате мы сможем говорить о возможности строго сбалансированного назначения задач при расщеплении ядер.

Лемма 1. Если упорядочить стадии системы по убыванию потребляемой мощности, то соответствующая оптимальная последовательность, составленная из постадийных мощностей множеств используемых ядер, будет невозрастающей.

Доказательство: Запишем выражение для значения ΔP сэкономленной мощности

$$\Delta P = \sum_i P_i - \sum_i \frac{P_i}{n_i^2}.$$

Поскольку мощность безызбыточной системы $\sum_i P_i$ задана, для максимизации ΔP следует минимизировать слагаемое $\sum_i \frac{P_i}{n_i^2}$. Эту сумму можно рассматривать как произведение двух числовых последовательностей $\left\{P_i \mid i = \overline{1,n}\right\}$ и $\left\{\frac{1}{n_i^2} \mid i = \overline{1,n}\right\}$. Упорядочим первую последовательность по убыванию значений. Тогда известно [18], что рассматриваемая сумма будет минимальной, если соответствующая вторая последовательность будет неубывающей. Неубывание членов последовательности означает в данном случае невозрастание мощности множеств ядер, используемых на разных стадиях. ■

Данное утверждение подтверждает направленность предложенного алгоритма на разгрузку наиболее загруженных ядер.

Теорема 1. Алгоритм является «жадным», а именно, каждый его шаг является оптимальным по критерию $A = \arg\min_A P(A)$.

Доказательство: Покажем, что получаемый на каждом шаге алгоритма вклад в сэкономленную мощность будет максимальным. Запишем выражение для сэкономленной мощности:

$$\Delta P_i = \frac{P_i}{n_i^2} - \frac{P_i}{(n_i+1)^2}.$$

Перейдем от дискретной функции $\frac{P_i}{(n_i+1)^2}$ к непрерывной и воспользуемся для ее представления двумя первыми слагаемыми из разложения в ряд Тейлора:

$$\Delta P_i \frac{P_i}{n_i^2} - \frac{P_i}{n_i^2} + 2\frac{P_i}{n_i^2}\frac{1}{n_i} = 2\frac{P_i}{n_i^2}\frac{1}{n_i}. \qquad (7)$$

Из выражения (7) видно, что прирост сэкономленной мощности равен удвоенной удельной мощности стадии. Следовательно, выбор по максимуму удельной мощности соответствует жадному выбору по максимуму сэкономленной мощности. ■

Теорема 2. Алгоритм поставляет архитектуру, оптимальную по критерию $A = \arg\min_A P(A)$, при заданном ограничении на число дополнительных ядер $n_d \le n_{d0}$.

Доказательство: Прежде всего заметим, что в оптимальной архитектуре все расщепленные множества системы являются предельными, кроме, возможно, множества с максимальной удельной мощностью. Действительно, предположим противное, а именно, что в оптимальной архитектуре существует непредельное расщепленное множество с немаксимальной удельной мощностью. Тогда из этой стадии можно исключить одно ядро и перенести его в стадию с максимальной удельной мощностью. При этом сэкономленная мощность увеличится, а значит, предположение о существовании такой оптимальной архитектуры неверно. Очевидно, что алгоритм на каждом шаге поставляет архитектуру, в которой все расщепленные множества системы являются предельными, кроме, возможно, множества с максимальной удельной мощностью. При этом он осуществляет перебор всех возможных таких архитектур в рамках заданного ограничения на $n_d \le n_{d0}$, который в конце концов приведет к оптимальному решению.

Поскольку приведенный алгоритм оперирует не с абсолютными значениями мощностей, а с их соотношением, то с инженерной точки зрения воз-

можен переход к оперированию не потребляемыми в стадиях мощностями, а вычислительными сложностями решаемых в стадиях задач. Итак, пусть планируется m заданий $\left\{\tau_j \mid j=\overline{1,m}\right\}$, каждое j-е из которых включает n задач $\left\{\tau_{j,i} \mid i=\overline{1,n}\right\}$. Введем понятие вычислительной сложности Ψ_j задания τ_j, под которой будем понимать число составляющих это задание операций. В относительных расчетах, проводимых далее в примере, в качестве оценок можно использовать его исходную длительность (до снижения частоты). Аналогично введем вычислительную сложность $\Psi_{j,i}$ для задачи $\tau_{j,i}$, а также вычислительную сложность Ψ^i для стадии i. При этом

$$\Psi_j = \sum_{i=1}^{n} \Psi_{j,i}; \;\; \Psi^i = \sum_{j=1}^{m} \Psi_{j,i}.$$

Выбирая в расщепленном множестве каждого i-го исходного ядра ядро с максимальной нагрузкой $\Psi^i_{\max}$, рассчитываем для стадии минимальную тактовую частоту как пропорциональную вычислительной сложности:

$$f^i_{\min} = f_0 \frac{\Psi^i_{\max}}{\Psi^i}. \qquad (8)$$

Далее в той же пропорции снижается напряжение питания:

$$V^i_{\min} = V_0 \frac{\Psi^i_{\max}}{\Psi^i}.$$

Подчеркнем, что выбор варианта размещения задач стадии целесообразно делать в пользу сбалансированного назначения, поскольку в этом случае вычислительная сложность максимального блока будет минимальной, а значит, и минимальной будет выбираемая тактовая частота стадии (8). Рассмотрим иллюстративный пример.

Пример 1. Пусть система содержит три ядра, на которых выполняются четыре задания τ_1, τ_2, τ_3, τ_4, имеющие длительности стадий, выраженные в условных единицах (табл. 1).

■ ***Таблица 1***. Длительности стадий
■ ***Table 1***. Stage executions

Задание	$\tau_{i,1}$	$\tau_{i,2}$	$\tau_{i,3}$
τ_1	3	2	1
τ_2	3	1	1
τ_3	3	1	1
τ_4	2	1	1
τ_5	11	5	4

В нижней строке таблицы приведены суммарные длительности для всех задач каждой из стадий. Далее они будут использованы как оценки вычислительной сложности и оценки потребляемой на каждой стадии мощности в условных единицах. Необходимо определить наилучшее расщепление для каждого ядра при условии, что запас по площади позволяет добавить в систему четыре ядра ($n_d = 4$).

Для решения задачи воспользуемся вышеприведенным алгоритмом с анализом длительностей стадий.

1. Делаем начальные присвоения $M = 4$, $\mathbf{A} = (1\ 1\ 1)$, $\bar{\mathbf{P}} = (11\ 5\ 4)$.

2. Расщепляем ядро первой стадии, вводя два дополнительных ядра. Распределяем нагрузку примерно равномерно между тремя ядрами: {5, 3, 3}. Вычисляем среднюю мощность, потребляемую ядрами расщепленного множества 1-й стадии: $\bar{P}_1 = 3{,}6 < 5$. Вычисляем количество запасных ядер $n_d = 4 \neq 0$. Переходим к следующему шагу расщепления.

3. Расщепляем ядро второй стадии, вводя одно дополнительное ядро. Распределяем нагрузку примерно равномерно между двумя ядрами: {3, 2}. Вычисляем среднюю нагрузку ядер расщепленного множества 2-й стадии: $\bar{P}_2 = 2{,}5 < 3{,}6$. Вычисляем количество запасных ядер $n_d = 1 \neq 0$. Переходим к следующему шагу расщепления.

4. Расщепляем ядро третьей стадии, вводя одно дополнительное ядро. Распределяем нагрузку равномерно между двумя ядрами: {2, 2}. Вычисляем среднюю мощность, потребляемую ядрами расщепленного множества 3-й стадии: $\bar{P}_3 = 2$. Вычисляем количество запасных ядер $n_d = 0$. Конец.

Таким образом, результирующая система содержит семь ядер и характеризуется следующим вектором средних нагрузок для стадий $\bar{\mathbf{P}} = (3{,}6 \quad 2{,}5 \quad 2)$. Оценим приближенно достигаемое при этом снижение потребляемой мощности. В качестве оценки мощности в условных единицах, потребляемой исходной системой, будем использовать, как уже отмечалось: $P_0 = \sum_j \bar{P}_{\Sigma,j}$.

При этом для преобразованной системы выражение будет иметь вид

$$P = \sum_j \frac{\bar{P}_{\Sigma,j}}{k_j^2},$$

где $k_j = \dfrac{\bar{P}_{\Sigma,j}}{\bar{P}_{j,\max}}$ — коэффициент снижения частоты (напряжения питания); $\bar{P}_{j,\max}$ — наибольшая средняя мощность, потребляемая ядрами расщепленного множества j-й стадии. В результате снижение мощности выражается величиной $l = 5{,}3$.

Безусловно, реальный выигрыш будет меньше, поскольку в проведенных вычислениях не учтено ограничение (5) по напряжению питания.

Концепция построения децентрализованной энергоэффективной отказоустойчивой многоядерной системы на кристалле

Концепция основана на рассмотренных выше принципах построения энергоэффективных систем, а также принципах распределенного диагностирования, предполагающего взаимные проверки между процессорными модулями и распределенную процедуру принятия решений. При построении энергоэффективной отказоустойчивой системы предлагается воспользоваться скрытой избыточностью описанной в предыдущем разделе энергоэффективной архитектуры. Действительно, при формировании архитектуры в состав системы вводились дополнительные ядра с последующим снижением тактовой частоты и напряжения питания. В результате сформировалась архитектура многоядерного кристалла (см. рис. 1), включающая множество ядер, обменивающихся информацией через общую память. Общая память разбита на области, через каждую из которых реализуется информационная связь между соответствующей парой ядер по принципу «точка-точка». Для обеспечения отказоустойчивости предлагается при возникающих отказах организовать обратный процесс слияния ядер, а именно, задачи с отказавшего ядра переносить на ядро, в результате расщепления которого образовалось отказавшее. Безусловно, этот процесс должен сопровождаться увеличением тактовой частоты и напряжения питания для ядра слияния. Таким образом, за отказоустойчивость система «расплачивается» энергоэффективностью. Следует отметить, что как уровень энергоэффективности, так и уровень отказоустойчивости, достигаемые в конкретном приложении, могут быть весьма различными и в общем случае невысокими. Многое при этом будет определяться имеющимся запасом по площади кристалла и, как следствие, количеством дополнительных ядер. Более того, даже при большом числе использованных избыточных ядер может оказаться, например, что достигается высокий уровень энергоэффективности и низкий уровень отказоустойчивости. Это возможно, когда запас по дополнительным ядрам истрачен для разгрузки исходно перегруженных ядер, а для всех остальных исходных ядер дополнительных попросту не хватило. При этом, конечно, возможен поиск компромиссных вариантов, однако эти вопросы в настоящей работе не рассматриваются.

Для создания энергоэффективной отказоустойчивой системы требуется дополнительно решить еще несколько вопросов. Среди них определение структуры коммуникационной системы, разработка процедур диагностирования отказавшего ядра и восстановления системы после отказа.

Применим при разработке отказоустойчивой энергоэффективной системы PMC-модель [17] с распределенным принятием решения об отказе. Это позволит реализовать децентрализованную отказоустойчивую систему, лишенную «узких» мест. Обсудим реализацию PMC-модели, предварительно определив модель отказа ядра как утрату функции обработки информации при сохранении функции ее трансляции со входа ядра на выход. Процесс диагностирования можно разбить на три этапа: проверки, сбора диагностической информации и принятия решения об отказе. На первом этапе предполагается, что ядра осуществляют взаимные проверки. Каждая из проверок характеризуется малой длительностью, поскольку нацелена на диагностирование лишь небольшой части объекта диагностирования, опираясь на которую объект может далее построить процесс самодиагностирования. Второй этап, очевидно, должен отличаться наибольшей длительностью, поскольку предполагает передачу результатов взаимных проверок и самодиагностирования от каждого ядра к каждому. Наконец, на третьем этапе в каждом из ядер происходит анализ результатов диагностирования и принятие решения об отказе.

В основу процедуры реконфигурации (восстановления) системы после отказа положим предположение о том, что в памяти каждого ядра есть таблицы назначения задач для номинальной ситуации, а также для ситуаций отказа любого из ядер. В результате после принятия решения об отказе данное ядро знает, где будут решаться задачи отказавшего модуля. При этом возможно, что какие-то из этих задач будет исполнять оно. Переход на повышенную вычислительную нагрузку потребует повышения тактовой частоты и напряжения питания путем направления в устройство параметрического управления соответствующих кодов настройки. Аналогична описанной процедура возвращения восстановленного ядра в состав системы. Действительно, ядра, которые взяли на себя после отказа исполнение дополнительных задач, убедившись в процессе взаимных проверок в работоспособности восстановленного ядра, прекращают исполнение дополнительных задач. Важно, что никакого влияния на другие ядра данное ядро не будет оказывать, а значит, и в случае если отказало именно оно, с его стороны не будет никакого неадекватного влияния.

Анализ диагностического эксперимента

В рамках PMC-модели могут быть реализованы различные подходы, но для всех них справедливы достаточные условия t-диагностируемости (локализации t отказавших ядер) в системе из n ядер [19, 20]:

1) $n \geq 2t + 1$;

2) не существует двух ядер, проверяющих друг друга;

3) каждое ядро проверяется минимум t ядрами.

Можно выделить два наиболее распространенных подхода. Первый рассчитан на случай $t = 1$ и предполагает выделение в графе межъядерных связей системы гамильтонова цикла (цикла, проходящего через все вершины системы, причем через каждую из них не более чем по одному разу). Далее каждое ядро проверяет лишь одно ядро системы, а именно следующее за ним в гамильтоновом цикле. Ясно, этот диагностический эксперимент удовлетворяет приведенным достаточным условиям. Во втором подходе гамильтонов цикл не выделяется, а каждое ядро проверяет всех своих непосредственных соседей. При этом, очевидно, число проверок зависит от числа соседей, т. е. геометрии связей, реализованной в коммуникационной системе.

Пример 2. Рассмотрим эксперимент, в котором используется гамильтонов цикл в системе из трех ядер.

В табл. 2 приведены синдромы проверок. Ее строки сопоставлены с отказами ядер, а столбцы — с проверками. В таблице «0» означает положительный, «1» — отрицательный, «x» — неопределенный результат проверки. Из таблицы видно, что при любом варианте ее доопределения строки таблицы не будут совпадать, а значит, отказы будут различимы.

Проанализируем требования, предъявляемые к коммуникационной системе. Как уже отмечалось, общая память системы коммуникаций разделена на области, соотносимые с информационными связями конкретных пар ядер. Понятно, что, с одной стороны, число таких связей необходимо сокращать в целях сокращения площади кристалла, отведенной под общую память. Однако, с другой стороны, число и геометрия этих связей влияют на эффективность диагностического эксперимента. Действительно, применяемая коммуникационная система должна не только допускать организацию в системе принятого диагностического эксперимента, но и не увеличивать чрезмерно его длительность, оставаясь в заданных ограничениях по площади кристалла. Для проектирования коммуникационной системы необходимо располагать значениями ее соответствующих характеристик. Для получения таких характеристик было проведено моделирование в среде YACSIM [21]. При этом исследовались коммуникационные системы трех типов: решетка, тор, гиперкуб [22, 23] (рис. 3, *а*–*в*).

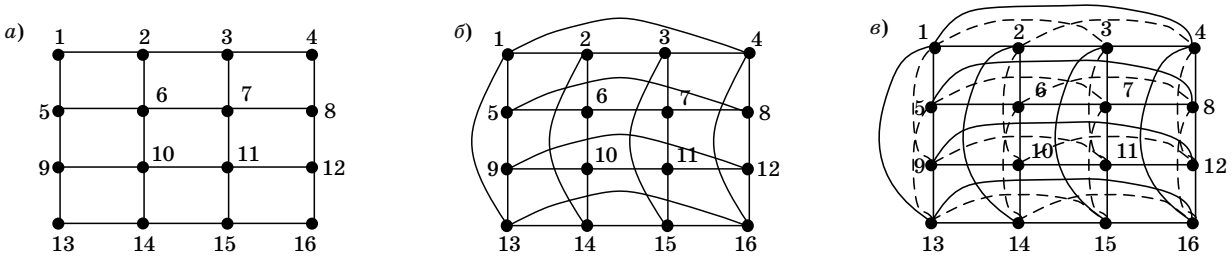
■ ***Таблица 2***. Синдромы проверок

■ ***Table 2***. Check results

Отказ	Проверка		
	1→2	2→3	3→1
1	x	0	1
2	1	x	0
3	0	1	x

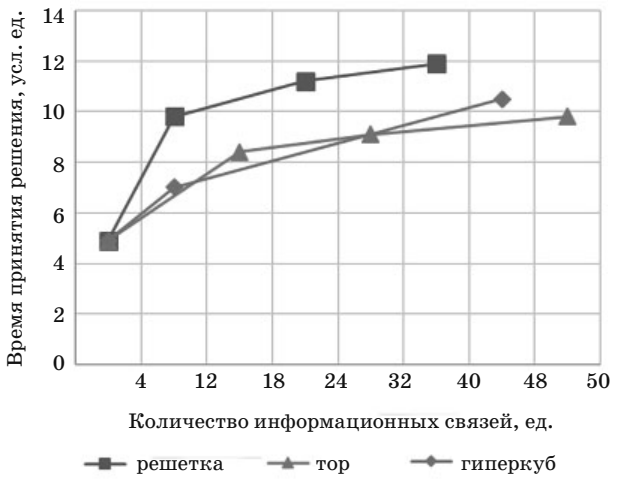
Целевой характеристикой моделирования была длительность диагностического эксперимента, включающего взаимные межпроцессорные проверки и сбор всех результатов в каждом из процессоров. Моделирование проводилось в предположении, что реализуется эксперимент с проверкой соседних ядер, причем время проверки одного ядра другим было принято равным 2,1 усл. ед., а время трансляции диагностической информации со входа ядра на его выход — равным 1,4 усл. ед.

Для каждого из трех типов на рис. 4 приведены зависимости этой характеристики от числа реализуемых информационных связей (объема используемой общей памяти). Последовательности



■ ***Рис. 3***. Графы межъядерных связей в системе на кристалле: *а* — решетка; *б* — тор; *в* — гиперкуб

■ ***Fig. 3***. Variants of intercore links in a SoC: *а* — grid; *б* — torus; *в* — hypercube



■ ***Рис. 4.*** Результаты моделирования
■ ***Fig. 4.*** Simulation results

расчетных точек на графиках соответствуют последовательности размерностей коммуникационной системы — 2×2, 3×3, 4×4, 5×5.

В ходе диагностического эксперимента получено, что наиболее эффективной коммуникационной системой с точки зрения времени принятия решений является схема типа тор.

Заключение

В настоящей работе предложен подход к построению децентрализованной отказоустойчивой и энергоэффективной системы на многоядерном кристалле. Подход включает, во-первых, определение энергоэффективной архитектуры за счет введения в состав системы дополнительных ядер, сопровождающегося снижением тактовой частоты и напряжения питания, а, во-вторых, разработку процедур диагностирования и реконфигурации системы.

Финансовая поддержка

Работа поддержана грантом РФФИ 19-08-00052.

Литература

1. **Dongarra J., Herault T., Robert Y.** *Fault Tolerance Techniques for High-Performance Computing*. In: *Fault-Tolerance Techniques for High-Performance Computing* / ed. by Thomas Herault, Yves Robert. Cham, Springer International Publishing, 2015. Pp. 3–85. doi:10.1007/978-3-319-20943-2
2. **Neelutpol G., Lakshmi P. S.** Fault tolerance in distributed real time environment: a survey. *International Journal of Computer Science and Mobile Computing*, 2015, vol. 4, iss. 6, pp. 829–836.
3. **Mengfei Y., Gengxin H., Yanjun F., Jian G.** *Fault-Tolerance Techniques for Spacecraft Control Computers*. Whiley, 2017. 344 p. doi:10.1002/9781119107392
4. **Blanke M., Kinnaert M., Lunze J., Staroswiecki M.** *Distributed Fault Diagnosis and Fault-Tolerant Control*. In: *Diagnosis and Fault-Tolerant Control*. Berlin, Heidelberg, Springer Berlin Heidelberg, 2016. Pp. 467–518. doi:10.1007/978-3-662-47943-8_10
5. **Karavai M. F., Podlazov V. S.** Extended generaized hypercube as fail-safe system network for multiprocessor systems. *Autom. Remote Control*, 2015, no. 2, pp. 336–352. doi:10.1134/S0005117915020137
6. **Cannella E., Stefanov T. P.** Energy eficient semi-partitioned scheduling for embedded multiprocessor streaming systems. *Design Automation for Embedded Systems,* September 2016, vol. 20, iss. 3, pp. 239–266. doi:10.1049/iet-wss.2011.0125
7. **Dorronsoro B., Nesmachnow S., Taheri J., Zomaya A., Talbi E.-G., Bouvry P.** A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustainable Computing: Informatics and Systems*, 2014, vol. 4, pp. 252–261. doi:10.1016/j.suscom.2014.08.003
8. **Panda P. R., Shrivastava A., Silpa B. V. N., Gummidipudi K.** *Power-Efficient System Design*. Springer US, 2010. 260 p. doi:10.1007/978-1-4419-6388-8
9. **Rubavani R., Saranraj S., Saranya S., Ranjani Devi R.** Power efficient scheduling for network on chip applications on multicore processor. *International Journal of Applied Engineering Research*, 2016, vol. 11, no. 7, pp. 4751–4757.
10. **Patton R. J., Frank P. M., Clark R. N.** *Issues in fault diagnosis for dynamic systems*. London, Springer-Verlag, 2000. 597 p. doi:10.1007/978-1-4471-3644-6
11. **Isermann R.** *Fault Diagnosis Application*. Heidelberg, Springer, 2011. 354 p. doi:10.1007/978-3-642-12767-0
12. **Kaldmuae A., Kotta U., Jiang B., Shumsky A., and Zhirabok A.** Measurement feedback disturbance decoupling in discrete-time nonlinear systems. *Automatica*, 2013, vol. 49, no. 9, pp. 2887–2891. doi:10.1016/j.automatica. 2013.06.013
13. **Gruzlikov A. M., Kolesov N. V., Tolmacheva M. V.** Event monitoring of parallel computations. *Int. J. Applied Mathematics and Computer Science*, 2015, vol. 25, no. 2, pp. 311–321. doi:10.1515/amcs-2015-0024
14. **Gruzlikov A. M., Kolesov N. V., Lukoyanov E. V., Tolmacheva M. V.** Test-based diagnosis of distributed computer system using a time-varying model. *IFAC PapersOnLine*, 2018, vol. 51, iss. 24, pp. 1075–1082. doi:10.1016/j.ifacol.2018.09.724

15. **Романкевич В. А.** Самотестирование многопроцессорных систем с регулярными диагностическими связями. *Автоматика и телемеханика*, 2017, № 2, с. 115–127. doi:10.1134/S0005117917020084
16. **Haider S., Nazir B.** Fault tolerance in computational grids: perspectives, challenges, and issues. *SpringerPlus*, 2016, vol. 5, no. 1, p. 1991. doi:10.1186/s40064-016-3669-0
17. **Teng Y.-H., Lin C.-K.** A test round controllable local diagnosis algorithm under the PMC diagnosis model. *Applied Mathematics and Computation*, 2014, vol. 244, pp. 613–623. doi:10.1016/j.amc.2014.07.036
18. **Cloud M. J., Drachman B. C., Lebedev L. P.** *Inequalities. With applications to engineering.* 2nd ed. Springer International Publishing, 2014. 239 p. doi:10.1007/978-3-319-05311-0
19. **Min X., Liangcheng Y., Jiarong L.** A *t*/*k* diagnosis algorithm on hypercube-like networks. *Concurrency and Computation: Practice and Experience*, 2017, vol. 30, e4358. doi:10.1002/cpe.4358
20. **Димитриев Ю. К.** Необходимые и достаточные условия *t*-диагностируемости многопроцессорных вычислительных систем для разных моделей ненадежного тестирования, полученные с помощью теоретико-графовой модели системы. *Автоматика и телемеханика*, 2016, № 6, с. 145–158. doi:10.1134/S0005117916060096
21. **Молдованова О. В.** Адаптивный алгоритм децентрализованной самодиагностики распределенных вычислительных систем. *Вестник СибГУТИ*, 2013, № 2, с. 22–30.
22. **Shuming Z., Limei L., li X., Dajin W.** The diagnosability of star graph networks. *IEEE Transactions on Computers*, 2015, vol. 64, no. 2, pp. 547–555. doi:10.1109/TC.2013.228
23. **Lidan Wang, Ningning Liu, Cheng-Kuan Lin, Tzu-Liang Kung, Yuan-Hsiang Teng.** A diagnosis algorithm on the 2D-torus network. *Proc. of the 12th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2018)*, pp. 620–625. doi:10.1007/978-3- 319-93554-6_60

UDC 621.38
doi:10.31799/1684-8853-2019-4-9-18

Fault-tolerant and energy-efficient MCSoC for information processing and control

A. M. Gruzlikov[a], PhD, Tech., Head of Department, orcid.org/0000-0001-8814-0726
N. V. Kolesov[a], Dr. Sc., Tech., Professor, orcid.org/0000-0003-3287-7504, kolesovnv@mail.ru
D. V. Kostygov[a], Post-Graduate Student, orcid.org/0000-0003-4379-5803
M. V. Tolmacheva[a], PhD, Tech., Senior Researcher, orcid.org/0000-0003-0795-7617
[a]Concern CSRI Elektropribor, JSC State Research Center of Russia, 30, Malaya Posadskaya St., 197046, Saint-Petersburg, Russian Federation

Introduction: The majority of real complex systems are designed with respect to fault tolerance requirements. However, all the known approaches are intended only to increase reliability. **Purpose:** An approach for designing fault-tolerant systems on a chip, aimed not only at increasing the reliability, but also at reducing the energy consumed by the system. **Results:** A two-stage approach to the design of fault-tolerant multicore systems-on-chip (MCSoCs) is proposed. At the first stage, an energy-efficient architecture of the designed system is formed. For each core used in the system, the optimal number of additional cores is determined within the framework of the imposed restrictions. The optimality criterion is the minimum power consumed by the system. The algorithm proposed for the formation of an energy-efficient architecture is based on the dependence of the power consumed in the system on the values of the supply voltage and the clock frequency. At the second stage, a procedure for diagnosing and repairing the system is developed which uses the principles of system-level diagnosis, involving mutual checks between the system cores. This procedure allows you to decentralize the process of diagnosing and restoring the system after a failure. Additionally, the article examines the organization of the communication subsystem based on shared memory. The study is based on a simulation conducted in order to estimate the time for making a decision about a failure in systems such as a lattice, torus and hypercube. **Practical relevance:** The proposed approach allows a system to provide the necessary values for its two most important characteristics: fault tolerance and energy efficiency. At the same time, decentralization is ensured when making decisions about a failure and restoration. As a result, the system becomes more reliable.

Keywords — fault tolerance, PMC model, decentralized system, energy efficiency, multicore system-on-chip (MCSoC).

For citation: Gruzlikov A. M., Kolesov N. V., Kostygov D. V., Tolmacheva M. V. Fault-tolerant and energy-efficient MCSoC for information processing and control. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2019, no. 4, pp. 9–18 (In Russian). doi:10.31799/1684-8853-2019-4-9-18

References

1. Dongarra J., Herault T., Robert Y. *Fault Tolerance Techniques for High-Performance Computing*. In: *Fault-Tolerance Techniques for High-Performance Computing*. Ed. by Thomas Herault, Yves Robert. Cham, Springer International Publishing, 2015. Pp. 3–85. doi:10.1007/978-3-319-20943-2
2. Neelutpol G., Lakshmi P. S. Fault tolerance in distributed real time environment: a survey. *International Journal of Computer Science and Mobile Computing*, 2015, vol. 4, iss. 6, pp. 829–836.
3. Mengfei Y., Gengxin H., Yanjun F., Jian G. *Fault-Tolerance Techniques for Spacecraft Control Computers*. Whiley, 2017. 344 p. doi:10.1002/9781119107392
4. Blanke M., Kinnaert M., Lunze J., Staroswiecki M. *Distributed Fault Diagnosis and Fault-Tolerant Control*. In: *Diag-*

nosis and Fault-Tolerant Control. Berlin, Heidelberg, Springer Berlin Heidelberg, 2016. Pp. 467–518. doi:10.1007/978-3-662-47943-8_10
5. Karavai M. F., Podlazov V. S. Extended generaized hypercube as fail-safe system network for multiprocessor systems. *Autom. Remote Control*, 2015, no. 2, pp. 336–352. doi:10.1134/S0005117915020137
6. Cannella E., Stefanov T. P. Energy eficient semi-partitioned scheduling for embedded multiprocessor streaming systems. *Design Automation for Embedded Systems,* September 2016, vol. 20, iss. 3, pp. 239–266. doi:10.1049/iet-wss.2011.0125
7. Dorronsoro B., Nesmachnow S., Taheri J., Zomaya A., Talbi E.-G., Bouvry P. A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustainable Computing: Informatics and Systems*, 2014, vol. 4, pp. 252–261. doi:10.1016/j.suscom.2014.08.003
8. Panda P. R., Shrivastava A., Silpa B. V. N., Gummidipudi K. *Power-efficient System Design*. Springer US, 2010. 260 p. doi:10.1007/978-1-4419-6388-8
9. Rubavani R., Saranraj S., Saranya S., Ranjani Devi R. Power efficient scheduling for network on chip applications on multicore processor. *International Journal of Applied Engineering Research*, 2016, vol. 11, no. 7, pp. 4751–4757.
10. Patton R. J., Frank P. M., Clark R. N. *Issues in fault diagnosis for dynamic systems.* London, Springer-Verlag, 2000. 597 p. doi:10.1007/978-1-4471-3644-6
11. Isermann R. *Fault Diagnosis Application*. Heidelberg, Springer, 2011. 354 p. doi:10.1007/978-3-642-12767-0
12. Kaldmuae A., Kotta U., Jiang B., Shumsky A., and Zhirabok A. Measurement feedback disturbance decoupling in discrete-time nonlinear systems. *Automatica*, 2013, vol. 49, no. 9, pp. 2887–2891. doi:10.1016/j.automatica. 2013.06.013
13. Gruzlikov A. M., Kolesov N. V., Tolmacheva M. V. Event monitoring of parallel computations. *Int. J. Applied Mathematics and Computer Science*, 2015, vol. 25, no. 2, pp. 311–321. doi:10.1515/amcs-2015-0024
14. Gruzlikov A. M., Kolesov N. V., Lukoyanov E. V., Tolmacheva M. V. Test-based diagnosis of distributed computer system using a time-varying model. *IFAC PapersOnLine*, 2018, vol. 51, iss. 24, pp. 1075–1082. doi:10.1016/j.ifacol.2018.09.724
15. Romankevich V. A. Self-testing of multiprocessor systems with regular diagnostic connections. *Automation and Remote Control*, 2017, vol. 78, iss. 2, pp. 289–299 (In Russian). doi:10.1134/S0005117917020084
16. Haider S., Nazir B. Fault tolerance in computational grids: perspectives, challenges, and issues. *SpringerPlus*, 2016, vol. 5, no. 1, p. 1991. doi:10.1186/s40064-016-3669-0
17. Teng Y.-H., Lin C.-K. A test round controllable local diagnosis algorithm under the PMC diagnosis model. *Applied Mathematics and Computation*, 2014, vol. 244, pp. 613–623. doi:10.1016/j.amc.2014.07.036
18. Cloud M. J., Drachman B. C., Lebedev L. P. *Inequalities. With applications to engineering*. 2nd ed. Springer International Publishing, 2014. 239 p. doi:10.1007/978-3-319-05311-0
19. Min X., Liangcheng Y., Jiarong L. A t/k diagnosis algorithm on hypercube-like networks. *Concurrency and Computation: Practice and Experience*, 2017, vol. 30, e4358. doi:10.1002/cpe.4358
20. Dimitriev Yu. K. Necessary and sufficient conditions for t-diagnosability of multiprocessor computer systems for various models of nonreliable testing established using the system graph-theoretical model. *Automation and Remote Control*, 2016, vol. 77, iss. 6, pp. 1060–1070 (In Russian). doi:10.1134/S0005117916060096
21. Moldovanova O. V. Adaptivnyj algoritm decentralizovannoj samodiagnostiki raspredelennyh vychislitel'nyh sistem. *Vestnik SibGUTI*, 2013, no. 2, pp. 22–30 (In Russian).
22. Shuming Z., Limei L., Li X., Dajin W. The diagnosability of star graph networks. *IEEE Transactions on Computers*, 2015, vol. 64, no. 2 pp. 547–555. doi:10.1109/TC.2013.228
23. Lidan Wang, Ningning Liu, Cheng-Kuan Lin, Tzu-Liang Kung, Yuan-Hsiang Teng. A diagnosis algorithm on the 2D-torus network. *Proc. of the 12th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2018)*, pp. 620–625. doi:10.1007/978-3-319-93554-6_60