DOI:

Оптимизация параллельных вычислений бортовых систем реального времени

Часть 1

УДК 681.3.012 | ВАК 05.13.01

С. Назаров, д. т. н.[1], А. Барсуков, к. т. н.[2]

Развитие бортового оборудования космических, летательных и других объектов характеризуется постоянным увеличением числа решаемых задач, повышением их сложности, расширением интеллектуальных и адаптивных возможностей. Это приводит к усложнению бортовой вычислительной системы (БВС). На время решения задач, возлагаемых на БВС, накладываются жесткие временные ограничения. Выполнение этих требований возможно путем организации параллельных вычислительных процессов.

Актуальность тематики параллельных вычислений осознана достаточно давно как в связи с низкой надежностью и производительностью компьютеров, так и в связи с появлением многопроцессорных систем и многоядерных процессоров. Последние годы активно используются программируемые логические матрицы, и, видимо, недалеко то время, когда появятся программируемые матрицы процессорных элементов. Технология обеспечения надежности и высокой производительности на основе параллельных вычислений естественным образом стала преобладающей в бортовых вычислительных системах реального времени. На время решения большинства задач, возлагаемых на БВС, накладываются жесткие временные ограничения. Часто встает вопрос максимально возможного сокращения времени выполнения этих задач. Реализация этих требований приводит к необходимости организации параллельных вычислительных процессов. В данной работе представлена совокупность математических моделей, формулировок задач и подходов к их решению, позволяющих построить параллельный вычислительный процесс реализации информационно-связанных задач в минимально возможное время в условиях заданных ограничений по вычислительной среде. Математической основой решения являются теория графов, сетевого планирования и управления и расписаний.

Как правило, наборы программ, реализуемых современными бортовыми вычислительными системами (БВС), можно представить совокупностью информационно-связанных подпрограмм (заданным набором задач – ЗНЗ). Это позволяет представлять решаемые задачи в виде нагруженного графа и, соответственно, использовать для их анализа методы теории графов и сетевого планирования и управления. ЗНЗ обладает достаточным внутренним параллелизмом, который может быть удобно использован при реализации задач многопроцессорными (многоядерными) бортовыми системами. Основной недостаток этого подхода связан с тем, что архитектура аппаратных средств многопроцессорной системы зачастую фиксируется на этапе проектирования и остается неизменной на время выполнения программ. Это означает, что разработчик должен выбирать соответствующую систему для предполагаемых приложений. С этой целью разработчик должен разбить приложение на части, рассмотреть возможности параллелизма при выполнении приложения и выбрать соответствующую систему для своего приложения. Ограничением такого подхода является недостаток распространенных существующих многопроцессорных систем, заключающийся в отсутствии адаптивности на стадии разработки и во время выполнения программы [1].

Программируемая разработчиком логическая матрица предлагает более гибкое решение, потому что с ее помощью аппаратные средства можно переконфигурировать с новыми функциями и использовать повторно с различными приложениями. Некоторые поставщики программируемых пользователем логических матриц (компания Xilinx) предлагают специальную функцию, называемую динамическим и частичным переконфигурированием [2]. Это означает, что часть аппаратных средств системы может быть изменена во время выполнения программы, а другая часть остается действующей и неизменной. Программируемая логика от Xilinx позволяет применять интеллектуальные решения в широчайшем спектре отраслей промышленной, научной и потребительской деятельности человека. Это программируемые логические интегральные схемы ПЛИС (FPGA), 3D-микросхемы и системы на кристалле (SoC), которые устанавливают стандарты низкой стоимости, высокой производительности и пониженного энергопотребления. Архитектура Xilinx

UltraScale™ дает беспрецедентный уровень интеграции и возможностей, обеспечивая при этом производительность на системном уровне ASIC-класса для самых требовательных приложений, требующих высочайшей пропускной способности, быстродействия памяти, массовых потоков данных, DSP и производительности обработки пакетов [3].

ПОСТАНОВКА ЗАДАЧИ

Предположим: задана структура приложения, состоящего из некоторого множества информационно-связанных частей (задач) с известным (ожидаемым) временем выполнения каждой задачи заданного набора задач. Примем следующие ограничения и допущения по ЗНЗ:

- предполагается, что время выполнения каждой задачи набора определяется элементарным вычислителем (процессором или ядром) многопроцессорной бортовой вычислительной системы (БВС), и это время (в условных временных единицах) установлено в процессе отладки;
- каждая задача ЗНЗ (или часть задач) обладает внутренним параллелизмом. Время выполнения задачи сокращается пропорционально числу выделенных задаче вычислителей;
- задано максимальное количество вычислителей, которое может быть выделено задаче. Оно определяется уровнем параллелизма задачи и изменяется от одного (параллелизма в программе нет) до трех (для случая максимально возможного параллелизма);
- определено максимальное количество вычислителей, которое может быть использовано для выполнения ЗНЗ.

С учетом заданных ограничений необходимо определить следующие требования:

1. Минимально возможное время решения системой ЗНЗ – $T_{ЗНЗ}^{min}$.
2. Количество вычислителей V_{min}, необходимое для выполнения требования 1.
3. Количество вычислителей V_i, выделяемых каждой задаче ЗНЗ, необходимое для выполнения требования 1.
4. Количество вычислителей $V_{тр.}$, необходимое для выполнения ЗНЗ с заданным требованием по времени реализации – $T_{ЗНЗ}^{тр.}$.
5. Количество вычислителей V_i, выделяемых каждой задаче ЗНЗ, необходимое для выполнения требования 4.

В данной статье рассмотрим возможное решение, позволяющее на основе анализа приложения определить целесообразную структуру многопроцессорной системы с выполнением требований на время выполнения приложений. Все далее принимаемые решения

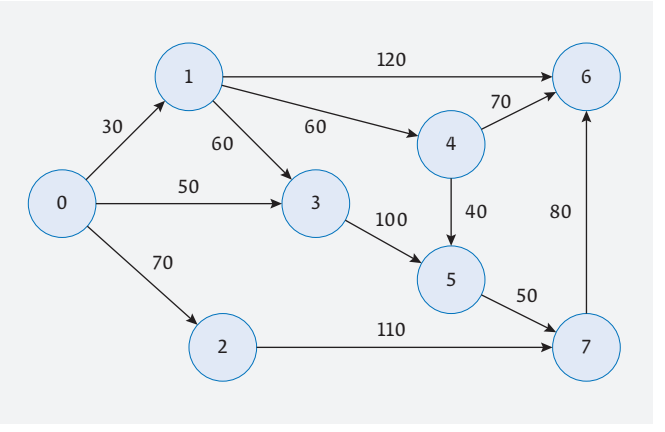


Рис. 1. Структура ЗНЗ

и разрабатываемые модели будем сразу иллюстрировать конкретными примерами.

Структуру подлежащего выполнению приложения удобно представить графом, например, как это показано на рис. 1, который будем далее использовать для иллюстрации решения поставленной задачи:

$$G=\{\langle z_i, z_j\rangle, t_{ij} \mid j>i,\ i=0,1,\ldots,M-2,\ j=1,2,\ldots,M-1\},$$

где z_i, z_j – номера событий (вершины) в графе; t_{ij} – время решения задачи (вес соответствующей дуги) при использовании одного вычислителя; M – количество задач в пакете G.

Для дальнейшей формализации задачи будем использовать понятия сетевого планирования и управления. В нашем случае работы представляются дугами графа $\langle z_i, z_j\rangle$ или просто «i,j», причем для любой дуги $j>i$. Обмен информацией между задачами инициируется событиями, например, событие z_1 инициирует завершение работы $\langle z_0, z_1\rangle$ длительность t_{01} – возможность запуска вычислений $\langle z_1, z_4\rangle$, $\langle z_1, z_5\rangle$ и $\langle z_1, z_6\rangle$. Организация вычислительного процесса при выполнении данного приложения сводится к определению временных параметров сетевого графика и его оптимизации по длительности выполнения всего комплекса задачи и затратам вычислительных ресурсов в соответствии с заданными требованиями.

РЕШЕНИЕ ЗАДАЧИ

Определение величины критического пути и резервов времени по отдельным вычислительным работам при выделении для каждой работы одного вычислителя

Критический путь T_{kr} – это полный путь, определяющий длительность всего комплекса вычислительных работ и имеющий наибольшую продолжительность. Для решения поставленной задачи необходимо найти длину критического пути и возможные резервы времени при

Таблица 1. Расчет параметров сетевого графика по рис. 1

Работа «i,j»	Продолжительность работы t_{ij}	Ранннее начало работы $t_{ij}^{\text{р.н.}}$	Раннее окончание работы $t_{ij}^{\text{р.о.}}$	Позднее начало работы $t_{ij}^{\text{п.н.}}$	Позднее окончание работы $t_{ij}^{\text{п.о.}}$	Общий резерв времени работы R_{ij}	Частный резерв времени работы r_{ij}
«0, 1»	**30**	**0**	**30**	**0**	**30**	**0**	**0**
«0, 2»	70	0	70	60	130	60	0
«0, 3»	50	0	50	40	90	40	0
«1, 3»	30	30	60	60	90	30	0
«1, 4»	**60**	**30**	**90**	**30**	**90**	**0**	**0**
«1, 6»	120	30	150	40	160	10	0
«2, 7»	110	70	180	130	240	60	0
«3, 5»	100	60	160	90	190	30	0
«4, 5»	40	90	130	150	190	60	0
«4, 6»	**70**	**90**	**160**	**90**	**160**	**0**	**0**
«5, 7»	50	160	210	190	240	30	0
«6, 7»	**80**	**160**	**240**	**160**	**240**	**0**	**0**

выполнении отдельных вычислительных работ. Определение длины критического пути можно проводить различными способами. Наиболее удобным способом расчета сетевого графика при небольшом количестве работ является расчет непосредственно на сети графика и заключается в нахождении критического пути и определении резервов времени для работ, которые не располагаются на этом пути.

При производстве расчетов сетевых моделей применяют следующие наименования его параметров [4, 5]:

- продолжительность работы t_{ij} (здесь i и j – номера соответственно начального и конечного событий);
- раннее начало работы $t_{ij}^{\text{р.н.}}$ – характеризуется выполнением всех предшествующих работ и определяется продолжительностью максимального пути от исходного события всей модели до начального события рассматриваемой работы;
- раннее окончание работы $t_{ij}^{\text{р.о.}}$ – определяется суммой раннего начала и продолжительности рассматриваемой работы;
- позднее начало работы $t_{ij}^{\text{п.н.}}$ – определяется разностью позднего окончания и продолжительности рассматриваемой работы;
- позднее окончание работы $t_{ij}^{\text{п.о.}}$ – определяется разностью продолжительности критического пути и максимальной продолжительности пути от завершающего события всей модели до конечного события рассматриваемой работы.

1. Общий резерв времени работы R_{ij} характеризуется возможностью роста продолжительности работы без увеличения продолжительности критического пути и определяется, как разность между поздним и ранним окончанием рассматриваемой работы. Общий резерв работы принадлежит не только первой работе, но и всем последующим работам данного пути. В случае использования на одной из работ общего резерва критический путь не изменит своей продолжительности, но все последующие работы окажутся критическими и лишатся резерва.

2. Частный резерв времени работы r_{ij} характеризуется возможностью увеличения продолжительности работы без изменения раннего начала последующей работы и определяется разностью между ранним началом последующей работы и ранним окончанием рассматриваемой работы. Частный резерв имеет место, когда одним событием заканчивается не менее двух работ. Отличие частного резерва от общего заключается в том, что частный резерв может быть использован только на рассматриваемой или предшествующих работах и не может быть использован на последующих.

3. Полным резервом некоторого пути в сетевой модели R называют разность между продолжительностью критического пути модели и продолжительностью рассматриваемого пути. Результаты расчета сетевой модели

выполнения ЗНЗ, представленного на рис. 1, приведены в табл. 1. Работы, не имеющие резерва времени и выделенные жирным шрифтом, образуют критический путь $T_{kr}=240$.

Если заданное значение времени $T_{ЗНЗ}^{тр.} \geq T_{kr}$ и разработчика это устраивает, то можно считать, что задача в части директивного времени решения ЗНЗ выполнена и предварительно определить требуемое количество вычислителей V_{min}. Полное время занятости вычислителей можно найти, просуммировав значения времени выполнения отдельных задач:

$$T_{ЗНЗ}=\sum_{i=0}^{i=6}\sum_{j=1}^{j=7}t_{ij}=810. \quad (1)$$

Минимальное количество вычислителей в этом случае можно определить, разделив вычислительную нагрузку на длину критического пути, то есть

$$V_{min} \geq T_з / T_{kr}=4. \quad (2)$$

Если выделить один вычислитель на выполнение задач, образующих критический путь, то оставшаяся вычислительная нагрузка потребует для своего выполнения еще не менее трех вычислителей. Знак «больше или равно» в соотношении (2) определяется тем фактом, что организация параллельного вычислительного процесса может потребовать дополнительного числа вычислителей.

Минимизация длины критического пути

Проведенные расчеты показывают, что при выделении для каждой задачи ЗНЗ одного вычислителя время решения заданного набора задач не может быть меньше, равного 240 условным временным единицам. Рассмотрим, какие имеются возможности для сокращения времени решения ЗНЗ, а в нашем случае – сокращения длины критического пути.

Повышение производительности вычислителей БВС

Наиболее простое решение – использование вычислителей более высокой производительности. Это – «лобовое» решение, которое, конечно, приведет к уменьшению времени критического пути. Однако оно, во-первых, не всегда возможно, во-вторых, может привести к неполному использованию вычислительных возможностей бортовой вычислительной системы и, наконец, удорожает всю систему.

Отметим еще одну особенность использования вычислителей БВС. Как все современные процессорные элементы, вычислитель БВС работает в режиме квантования процессорного времени, что позволяет организовать его мультипрограммную работу. Режим разделения времени позволяет рассматривать один физический вычислитель как совокупность виртуальных вычислителей (процессоров), которые могут выделяться отдельным задачам ЗНЗ. При этом суммарная производительность физического вычислителя, за исключением издержек мультипрограммирования (которыми далее пренебрегаем), равна сумме производительности его виртуальных вычислителей.

Использование имеющегося резерва времени вычислителей БВС

Имеющийся резерв времени по работам, не лежащим на критическом пути, свидетельствует о том, что имеется возможность изъятия некоторых временных ресурсов с целью добавления времени на работы, лежащие на критическом пути. Это приведет к уменьшению длины критического пути, то есть к сокращению всего цикла выполняемых вычислений. В нашем примере суммарный резерв времени составляет значение

$$R=\sum_{i=0}^{M-2}\sum_{j=1}^{M-1}R_{ij}=320.$$

Передача ресурса от какой-либо работы означает увеличение ее выполнения в зависимости от величины изъятого ресурса, добавление ресурса к другой работе, лежащей на критическом пути, означает уменьшение времени ее выполнения в соответствии с величиной добавленного ресурса. Например, изъятие половины резерва, то есть 30 единиц ресурса от работы «0, 2», приводит к увеличению времени ее выполнения до 100 единиц времени. Это означает, что для ее выполнения нужен будет виртуальный процессор со следующим значением доли физического процессора

$$V_{02}=\frac{t_{02}}{t_{02}+0{,}5\cdot R_{02}}=\frac{70}{70+0{,}5\cdot 60}=0{,}7.$$

Изъятый ресурс можно добавить с целью сокращения критического пути, например, к работе «6, 70» (см. табл. 1). При этом время выполнения этой работы уменьшится с 80 до 50 единиц времени, но для выполнения этой работы потребуется виртуальный процессор со следующим значением доли физического процессора

$$V_{67}=\frac{t_{67}}{t_{67}-0{,}5\cdot R_{02}}=\frac{80}{80-0{,}5\cdot 60}=1{,}6.$$

Для решения задачи минимизации критического пути необходимо построить модель линейного программирования с целью определения значений T_{kr} для различных вариантов перераспределения ресурсов. Удобно это сделать в электронных таблицах, например, Excel. Воспользуемся для этого формализацией задачи поиска критического пути, предложенной в [6]. Исходные данные удобно представить в форме таблицы (табл. 2).

Таблица 2. Исходные данные для решения задачи поиска критического пути

	Переменные	x_{01}	x_{02}	x_{03}	x_{13}	x_{14}	x_{16}	x_{27}	x_{35}	x_{45}	x_{46}	x_{57}	x_{67}	Ограничив. функция	Ограничение
	Значения переменных	1	0	0	0	1	0	0	0	0	1	0	1	1	1
Номера узлов	0	1	1	1										0	0
	1	-1			1	1	1							0	0
	2		-1					1						0	0
	3			-1	-1				1					0	0
	4					-1				1	1			0	0
	5											1		0	0
	6						-1						1	0	0
	t_{ij}	30	70	50	30	60	120	110	100	40	70	50	80		

Задача поиска критического пути на основе табл. 2 заключается в определении значения функции

$$T_{kr} = \sum_{i=0}^{i=6} \sum_{j=1}^{j=7} t_{ij} \cdot x_{ij} \to \max \qquad (3)$$

при следующих ограничениях:

$$x_{01} + 1 \cdot x_{02} + 1 \cdot x_{03} = 1; \qquad (4)$$

$$-1 \cdot x_{01} + 1 \cdot x_{13} + 1 \cdot x_{14} + 1 \cdot x_{16} = 0; \qquad (5)$$

$$-1 \cdot x_{02} + 1 \cdot x_{27} = 0; \qquad (6)$$

$$-1 \cdot x_{03} - 1 \cdot x_{13} + 1 \cdot x_{35} = 0; \qquad (7)$$

$$-1 \cdot x_{14} + 1 \cdot x_{45} + 1 \cdot x_{46} = 0; \qquad (8)$$

$$-1 \cdot x_{35} - 1 \cdot x_{45} + 1 \cdot x_{57} = 0; \qquad (9)$$

$$-1 \cdot x_{16} - 1 \cdot x_{46} + 1 \cdot x_{67} = 0; \qquad (10)$$

$$\forall x_{ij} \in \{0, 1\} | i = 0, 1, \ldots 6; \; j = 1, 2, \ldots 7. \qquad (11)$$

Решение задачи (3) – (11) для исходного графа сетевой модели показано на рис. 2.

Определим теперь, как изменится величина критического пути, если 40 единиц резерва изъять у работы «4, 5» и передать их для выполнения работе «6, 7». При этом задача «4, 5» будет выполняться 80 единиц времени, а задача «6, 7» – 40 единиц времени. Решение задачи в этом случае показано на рис. 3 и дает значение критического пути, равное 220. Если рассчитанное значение критического пути окажется больше директивного, можно рассмотреть другие варианты использования резервов других работ, пока не будет получено требуемое значение критического пути. Если этого не удается сделать за счет резервов в выполнении работ, необходимо увеличить количество ресурсов для их выполнения. В данном случае – перейти к более производительным вычислителям или переписать программы для возможности организации параллельного вычислительного процесса.

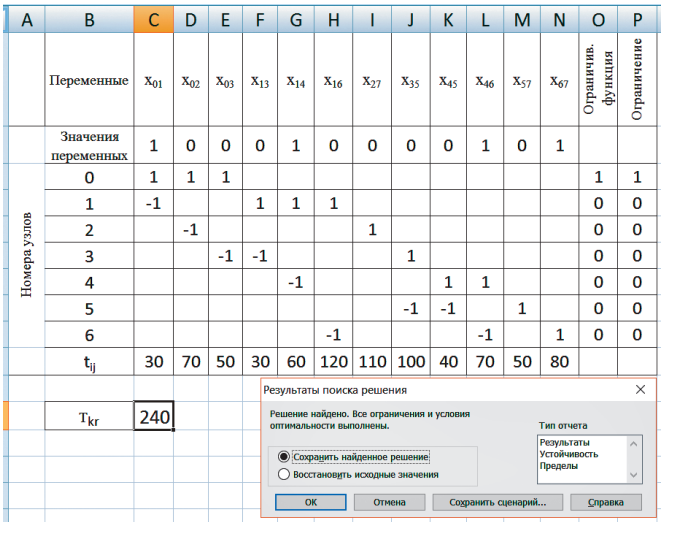
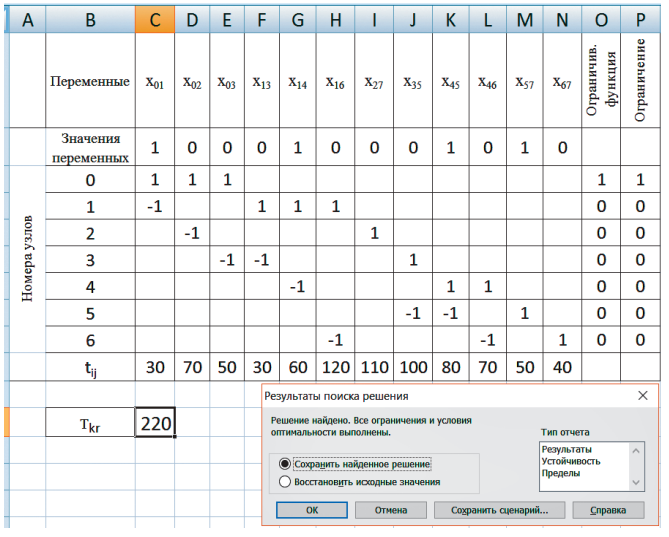
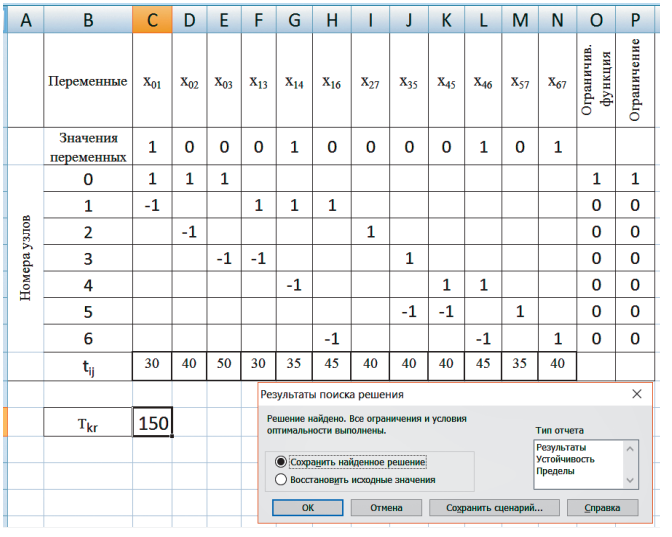


Рис. 2. Критический путь исходного графа сетевой модели



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
		Переменные	x_{01}	x_{02}	x_{03}	x_{13}	x_{14}	x_{16}	x_{27}	x_{35}	x_{45}	x_{46}	x_{57}	x_{67}	Ограничив. функция	Ограничение
		Значения переменных	1	0	0	0	1	0	0	0	1	0	1	0		
Номера узлов		0	1	1	1										1	1
		1	-1			1	1	1							0	0
		2		-1					1						0	0
		3			-1	-1				1					0	0
		4					-1				1	1			0	0
		5								-1	-1		1		0	0
		6						-1				-1		1	0	0
		t_{ij}	30	70	50	30	60	120	110	100	80	70	50	40		
		T_{kr}	220													

Рис. 3. Вариант расчета критического пути сетевой модели выполнения графа задач при использовании резерва задачи «4, 5»



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
		Переменные	x_{01}	x_{02}	x_{03}	x_{13}	x_{14}	x_{16}	x_{27}	x_{35}	x_{45}	x_{46}	x_{57}	x_{67}	Ограничив. функция	Ограничение
		Значения переменных	1	0	0	0	1	0	0	0	0	1	0	1		
Номера узлов		0	1	1	1										1	1
		1	-1			1	1	1							0	0
		2		-1					1						0	0
		3			-1	-1				1					0	0
		4					-1				1	1			0	0
		5								-1	-1		1		0	0
		6						-1				-1		1	0	0
		t_{ij}	30	40	50	30	35	45	40	40	40	45	35	40		
		T_{kr}	150													

Рис. 4. Расчет критического пути выполнения графа задач при использовании максимального распараллеливания задач ЗНЗ

Использование потенциальной параллельности задач ЗНЗ для минимизации критического пути

Как отмечено в постановке задачи данной статьи, каждая задача ЗНЗ обладает внутренним параллелизмом. Время выполнения задачи сокращается пропорционально числу выделенных задаче вычислителей. Пусть известен возможный уровень параллелизма по каждой задаче и, соответственно, время выполнения каждой задачи ЗНЗ при максимально возможном распараллеливании (табл. 3).

Определим длину критического пути для случая достаточного количества вычислителей в БВС. В данном случае распараллеленное выполнение задач «0, 2», «1, 4», «1, 6», «2, 7», «3, 5», «4, 5», «4, 6» и «5, 7» потребует кроме основных 12 вычислителей (по одному на каждую задачу) дополнительно 11 вычислителей. Для учета этого факта достаточно в разработанной электронной таблице по рис. 3 изменить содержимое ячеек, содержащих значения t_{ij}. Решение в этом случае показано на рис. 4. Таким образом, t_{rk}=150. Можно ли еще сократить длину критического пути? За счет распараллеливания – нет. Здесь все возможности использованы в полной мере. Осталась только одна задача критического пути «0, 1», но ее распараллелить невозможно, разве что написать заново.

Обновленные данные по параметрам сетевого графика приведены в табл. 4. Критический путь длиной 150 условных единиц образует цепочка вершин 0–1–4–5–7. Суммарная трудоемкость критического пути определяется суммой времени выполнения задач «0, 1», «1, 4», «4, 6», «6, 7» и составляет 150 условных единиц. Имеющийся общий резерв, как видно из табл. 4, составляет 240 условных единиц. Этот факт свидетельствует о возможности уменьшения числа вычислителей, выделяемых на задачи некритического пути.

Во второй части статьи будут рассмотрены вопросы минимизации количества ресурсов для выполнения пакета ЗНЗ без увеличения длины критического пути, возможности организации мультипроцессорного выполнения пакета задач, представленного сетевой моделью, вопросы минимизации загрузки БВС и предложены соответствующие математические модели, методы и алгоритмы решения этих задач.

Таблица 3. Время выполнения задач ЗНЗ при максимальном распараллеливании

Название задачи	«0, 1»	«0, 2»	«0, 3»	«1, 3»	«1, 4»	«1, 6»	«2, 7»	«3, 5»	«4, 5»	«4, 6»	«5, 7»	«6, 7»
Время выполнения на одном вычислителе	30	70	50	30	60	120	110	100	70	70	50	40
Возможный уровень параллелизма	1	1–2	1	1	1–2	1–3	1–3	1–3	1–2	1–2	1–2	1
Время выполнения при максимальном распараллеливании	**30**	**40**	**50**	**30**	**35**	**45**	**40**	**40**	**40**	**45**	**35**	**40**

Таблица 4. Расчет параметров сетевого графика с параллельным выполнением ЗНЗ

Работа «i, j»	Продолжительность работы t_{ij}	Ранннее начало работы $t_{ij}^{р.н.}$	Раннее окончание работы $t_{ij}^{р.о.}$	Позднее начало работы $t_{ij}^{п.н.}$	Позднее окончание работы $t_{ij}^{п.о.}$	Общий резерв времени работы R_{ij}	Частный резерв времени работы r_{ij}
«0, 1»	**30**	**0**	**30**	**0**	**30**	**0**	**0**
«0, 2»	40	0	40	70	110	70	0
«0, 3»	50	0	50	25	75	25	0
«1, 3»	30	30	60	45	75	15	0
«1, 4»	**35**	**30**	**65**	**30**	**65**	**0**	**0**
«1, 6»	45	30	75	65	110	35	0
«2, 7»	40	40	80	110	150	70	0
«3, 5»	40	60	100	75	115	15	0
«4, 5»	40	65	105	75	115	10	0
«4, 6»	**45**	**65**	**110**	**65**	**110**	**0**	**0**
«5, 7»	35	115	115	115	115	0	0
«6, 7»	**40**	**110**	**150**	**110**	**150**	**0**	**0**

ЛИТЕРАТУРА

1. **Гохрингер Д., Хюбнер М., Бекер Ю.** Архитектура адаптивных многопроцессорных систем на кристалле: новая степень свободы при проектировании систем и в поддержке при выполнении // Мир радиоэлектроники. – М.: ТЕХНОСФЕРА, 2012. С. 146–173.
2. **Хитт Д.** Стратегия Xilinx – быстрее двигаться к адаптируемому, интеллектуальному миру // ЭЛЕКТРОНИКА: Наука, Технология, Бизнес. 2018. № 4. С. 84–87.
3. Xilinx Ultrascale + Обзор. [Электронный ресурс]. URL: https://developer.ridgerun.com/wiki/index.php?title=Xilinx_-Ultrascale%2B_Overview
4. **Кофман А. В.** Сетевые методы планирования: Применение системы ПЕРТ и ее разновидностей при управлении производственными и научно-исследовательскими проектами / Пер. с фр. – М.: Прогресс, 1968. 181 с.
5. Сетевой график. [Электронный ресурс]. URL: http://www.stroitelstvonew.ru/1/sete-voy_grafik. shtml.
6. **Вагнер Г.** Основы исследования операций. Том 1 / Пер. с англ. – М.: МИР, 1972. 336 с.
7. **Назаров С. В.** Операционные системы специализированных вычислительных комплексов: теория построения и системного проектирования. – М.: Машиностроение, 1989. 400 с.