Инженерный вестник Дона, №5 (2021)
ivdon.ru/ru/magazine/archive/n5y2021/6997

Преимущества специализированных ресурсов программируемых логических интегральных схем для гибких сложно-функциональных блоков

С. В. Гаврилов, В. М. Хватов

Национальный исследовательский университет «МИЭТ»

Аннотация: Гибкие сложно-функциональные блоки (СФ-блоки) на базе программируемой логической интегральной схемы (ПЛИС) – это блоки, не имеющие конкретного размещения и разводки межсоединений на кристалле. Данные блоки позволяют ускорить процесс проектирования цифровых пользовательских схем на базе ПЛИС при этом использование специализированных ресурсов архитектуры ПЛИС способно повысить производительность разрабатываемых устройств, включающих эти блоки. Для достижения наибольшей эффективности эти ресурсы необходимо учесть, как при проектировании самих гибких СФ-блоков, так и при разработке методов и алгоритмов систем автоматизированного проектирования.

В данной работе спроектированы две реализации гибких СФ-блоков и выполнен сравнительный анализ полученного объема блоков и используемых трассировочных ресурсов. В первой реализации при логическом и физическом синтезе учитываются структурные особенности базового кристалла ПЛИС и его специализированные ресурсы. Во второй – используются только стандартные библиотечные элементы и общее трассировочное дерево. На основе результатов анализа показаны преимущества специализированных ресурсов архитектуры ПЛИС при проектировании гибких СФ-блоков. Также в статье рассматривается специфика ресурсов, заложенных разработчиками ПЛИС, и особенности реализации следующих гибких СФ-блоков: n-разрядные сумматор / вычитатель, счетчик до n и n-разрядный сдвиговый регистр.

Ключевые слова: сложно-функциональный блок, система автоматизированного проектирования, программируемая логическая интегральная схема, маршрут проектирования, трассировка.

Введение

Гибкие сложно-функциональные блоки (СФ-блоки) — это готовые блоки, представленные для практического использования на языке описания аппаратуры, использующиеся как для ускорения процесса проектирования интегральных схем на программируемых логических интегральных схемах (ПЛИС), так и для повышения производительности разрабатываемого устройства [1]. Они формируются из логических элементов (ЛЭ) ПЛИС на этапе логического синтеза [2] и требуют выполнения этапов физического синтеза. Гибкие СФ-блоки не привязаны к конкретным группам логических

блоков (ГЛБ), т.е для них необходимо выполнить кластеризацию и размещение [3]. Также они не имеют заранее трассированные межсоединения и необходимо выполнить трассировку их межсоединений [4].

Для более эффективных результатов логического и физического синтеза в архитектуру Программируемых Логических Интегральных Схем (ПЛИС), добавлены специализированные ресурсы, которые требуется учесть при разработке методов и алгоритмов для системы автоматизированного проектирования (САПР) [5-6]. ПЛИС, представленная в этой статье, разработана компанией АО НИИМЭ с технологическим уровнем 90 нм. Её ближайшим зарубежным аналогом со схожей структурой ЛЭ и ГЛБ является ПЛИС MAX II компании Altera, изготовленная по технологии 180 нм.

Чтобы показать потенциальные преимущества имеющихся ресурсов для проектирования СФ-блоков, были разработаны две реализации арифметических блоков. Блоки первого типа состоят из элементов специальной библиотеки СФ-блоков и используют архитектурные особенности ПЛИС. Блоки второго типа получены с помощью автоматического логического синтеза на основе стандартных библиотечных элементов. Рассмотрены следующие арифметические блоки: полный сумматор двух чисел шириной от 2 до 20 бит, сдвиговый регистр и асинхронный счетчик, содержащие от 2 до 20 DFF триггеров. Логический синтез проводился в Yosys Open SYnthesis Suite [7]. Весь маршрут проектирования рассмотренных СФ-блоков был выполнен в САПР, разработанном в ИППМ РАН, для программирования гетерогенных ПЛИС и Реконфигурируемых Систем на Кристалле (РСнК) [8].

Особенности проектирования СФ арифметических блоков

N-разрядный сумматор / вычитатель

Стандартное представление полного сумматора включает в себя схему формирования сигнала суммы и схему формирования сигнала переноса.

Представление сумматора на вентильном уровне обычно состоит из нескольких стандартных элементов. Архитектура рассматриваемой ПЛИС позволяет реализовать полный сумматор, используя только один ЛЭ. Это возможно благодаря LUT с 4 входами [9-10] в совокупности с цепочкой быстрого переноса [11-13]. LUT программируется в специальном режиме, при котором он делится на две части. С помощью двух входов выполняется операция сложения, с помощью двух других формируется перенос. Ко входам, отвечающим за перенос, коммутируются цепи ci0, ci1. Сигнал LAB_carry_in осуществляет выбор между этими двумя цепями. Структурная схема полного 10-ти разрядного сумматора представлена на рисунке 1.

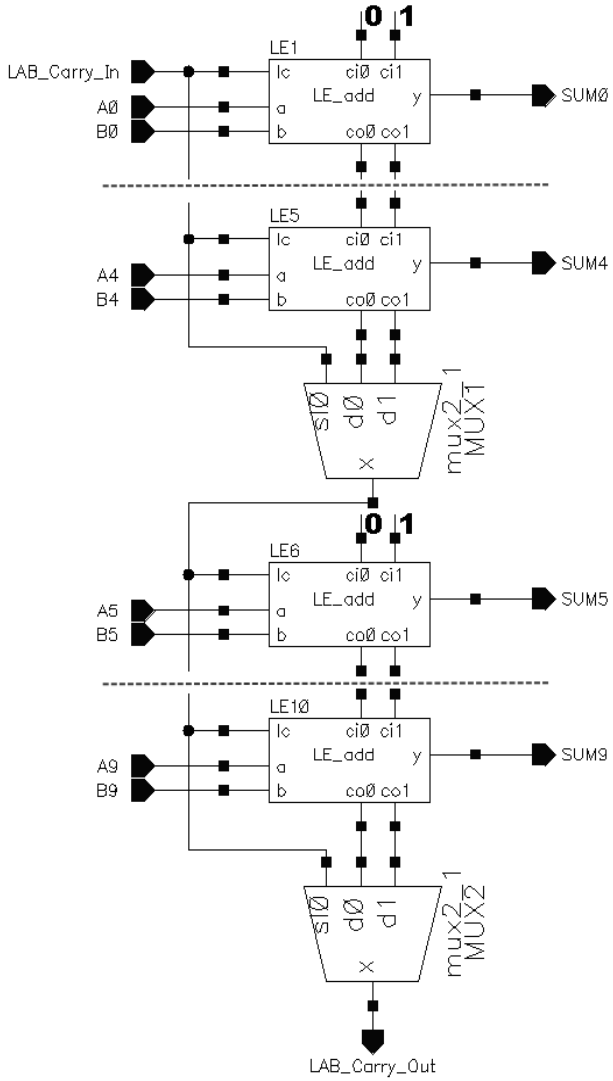

Рис. 1 – Структурная схема 10-разрядного сумматор

При добавлении возможности использования блока сумматора с ускоренным переносом в САПР учитываются несколько особенностей. Во-первых, накладывается ограничение на кластеризацию. Цепи переноса могут быть продлены до 10 ЛЭ внутри ГЛБ. При этом, если СФ-блоку требуется больше 10 ЛЭ, цепь переноса продлевается до следующей ГЛБ. Таким образом, можно объединить строку, состоящую из нескольких ГЛБ. Цепь переноса обрывается в конце этой строки. Во-вторых, структура схемы ускоренного переноса предполагает наличие мультиплексора после каждого пятого ЛЭ. Следовательно, в алгоритм трассировки добавляется специальный элемент. В-третьих, для формирования последнего бита переноса добавляется дополнительный ЛЭ, буферизирующий сигнал с выходов co0 и ci1.

При выборе режима вычитателя входные пины данных, а также ограничения, имеющиеся у режима сумматора, не изменяются. Изменяется лишь значение конфигурационного бита, отвечающего за переключения между режимами, при программировании ЛЭ.

N-разрядный сдвиговый регистр

Архитектура рассматриваемой ПЛИС также предполагает возможность прямого соединения выхода триггера со входом данных соседнего триггера, так называемый register chain. Это позволяет объединить триггеры в цепочку, т.е. в сдвиговый регистр, без использования дополнительных трассировочных элементов (ТЭ), и не занимая локальные шины данных. Структурная схема 10-ти разрядного сдвигового регистра показана на рисунке 2. Также, как и в сумматоре, цепь сдвига данных может быть продлена до 10 ЛЭ в ГЛБ и до конца каждой строки в общей матрице ГЛБ. Использование специального входа reg_chain вместо стандартного входа данных учитывается при программировании ЛЭ.

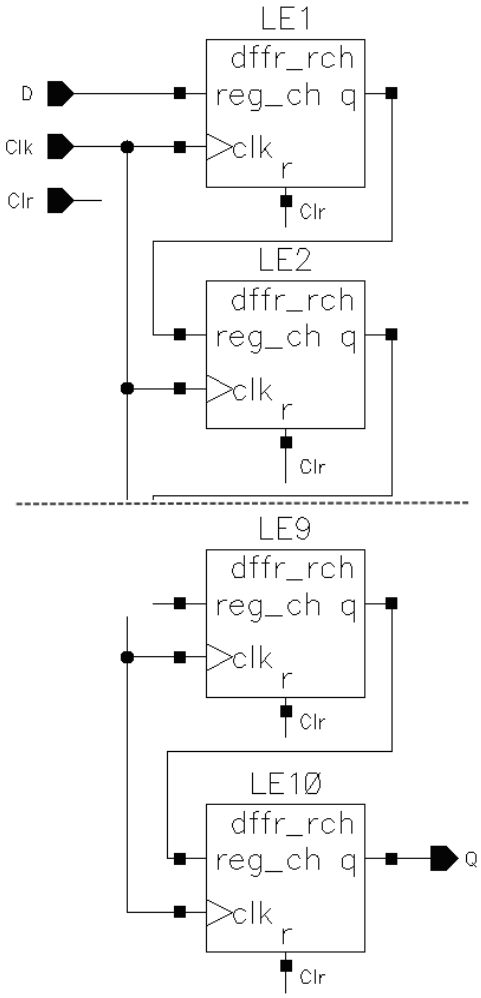

Рис. 2 – Структурная схема 10-разрядного сдвигового регистра

N-разрядный счетчик

Для проектирования асинхронных N-разрядных счетчиков в структуре ЛЭ присутствует несколько архитектурных особенностей. Одной их них является использование контакта register chain, в качестве входа для тактового сигнала. Выбор между тактовым сигналом и данными, приходящими на него, осуществляется на этапе программирования ЛЭ. Вторая особенность - это наличие обратной связи от выхода триггера к его входу данных. Масштабирование тактовой цепи выполняется аналогично цепи сдвига в сдвиговом регистре.

На рисунке 3б продемонстрирована структурная схема 10-ти разрядного счетчика, разработанного с учетом структуры ЛЭ и архитектуры

трассировочного дерева. На рисунке 3а продемонстрирован счетчик такой же разрядности, но в базисе стандартных DFF триггеров с типичным набором входов - clk, data, r, q. Для чистоты сравнения двух реализаций этот счетчик разработан на вентильном уровне, а размещение его элементов выполнено в тех же самых ГЛБ. В сравнении с рис.3а на рисунке 3б отсутствует обратная связь со входом данных, что обусловлено наличием этой связи внутри ЛЭ. Инверсия сигнала данных триггера, необходимая для изменения выхода из 0 в 1, и инверсия тактового сигнала на входе триггера для прямого счета также находятся внутри ЛЭ и учтены при программировании.

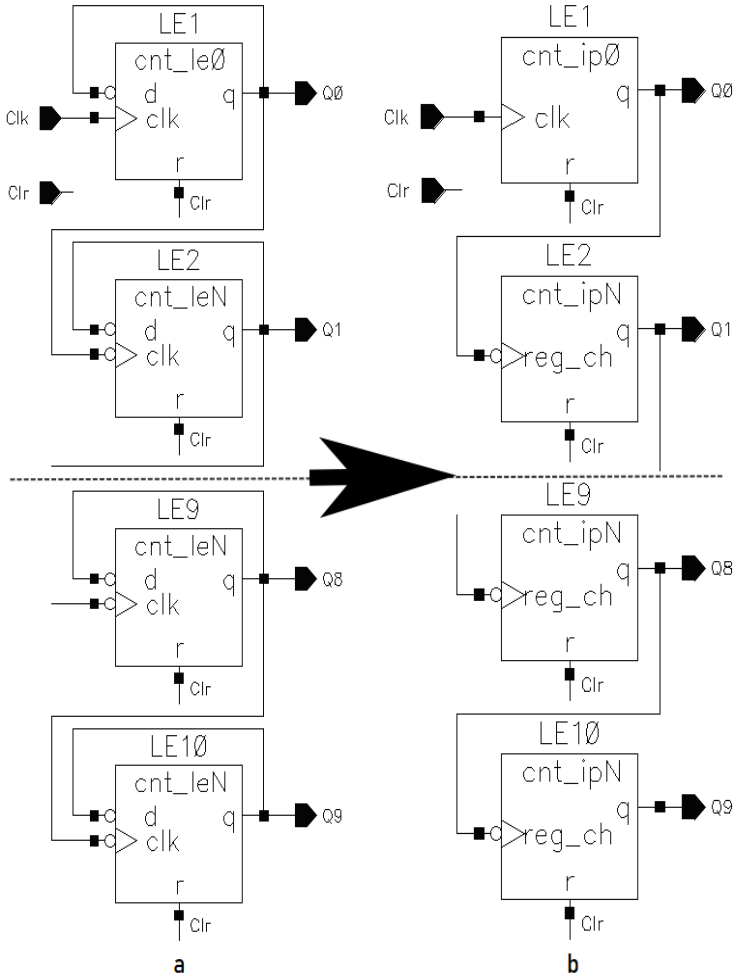

Рис. 3 – Структурная схема 10-разрядного счетчика на основе различных DFF а) из библиотеки стандартных ячеек б) из специализированной библиотеки арифметических СФ-блоков

Анализ результатов реализаций СФ-блоков

Для сравнения двух реализаций сумматоров были выбраны следующие параметры: количество ЛЭ, полученное в результате логического синтеза, количество трассировочных элементов, занятых в ходе трассировки, и общее число цепей, которые требуется трассировать. Результаты сравнения показаны в таблице 1.

Общее количество задействованных ЛЭ непосредственно определяет площадь, занимаемую СФ-блоком на ПЛИС, и косвенно влияет на длину его межсоединений. Количество ЛЭ у сумматора как софт СФ-блока равно разрядности сумматора плюс один ЛЭ для формирования старшего бита (бита переноса). В то же время количество ЛЭ “стандартного” сумматора увеличивается от 4 ЛЭ для 2 разрядов до 67 ЛЭ для 20 разрядов. При этом каждые два разряда сумматора на стандартных элементах добавляют в 3-4 раза больше ЛЭ, чем два разряда сумматора, разработанного с использованием специализированных ресурсов.

Стоит заметить, что под цепями в данном анализе имеются в виду межсоединения от выхода одного элемента до входа другого. Причем рассматриваются только межсоединения, участвующие в трассировке. Цепи, соединяющие вход и выход напрямую, без использования ТЭ, в статистику не попадают.

Количество цепей в схеме сумматора увеличивается с разрядностью слагаемых. Причем каждые два разряда сумматора, использующего специальные ресурсы, добавляют 6 цепей, а два разряда “стандартного” сумматора, в среднем, 22 цепи. С увеличением количества цепей повышается количество используемых трассировочных ресурсов. Чем больше ТЭ занято гибким СФ-блоком, тем меньше их остается для трассировки схемы, частью которой он является. Размер синтезированного блока и затраченные ресурсы также влияют на производительность разработанной схемы. Spice

моделирование двух реализаций 10 битного сумматора показало, что каждый выход схемы из библиотеки СФ-блоков переключается быстрее выходов схемы на стандартных элементах. Разница между задержками в двух реализациях находится на интервале от 1 нс до 4.2 нс.

Таблица № 1

Сравнение ресурсов, требующихся для имплементаций сумматора на ПЛИС

	Number of LE			***Number of RE***			***Number of nets***		
W	***LE***	***IP***	***RAT***	***LE***	***IP***	***RAT***	***LE***	***IP***	***RAT***
2	4	3	**1,33**	140	90	**1,56**	16	7	**2,29**
4	10	5	**2,00**	421	158	**2,66**	28	13	**2,15**
6	16	7	**2,29**	636	306	**2,08**	60	19	**3,16**
8	23	9	**2,56**	860	464	**1,85**	83	25	**3,32**
10	30	11	**2,73**	1124	552	**2,04**	105	31	**3,39**
12	36	13	**2,77**	1320	640	**2,06**	127	37	**3,43**
14	46	15	**3,07**	1668	723	**2,31**	152	43	**3,54**
16	55	17	**3,24**	1921	821	**2,34**	175	49	**3,57**
18	63	19	**3,32**	2210	899	**2,46**	197	55	**3,58**
20	67	21	**3,19**	2722	1002	**2,72**	219	61	**3,50**

Метод проектирования сдвигового регистра не влияет на число занятых ЛЭ. Исходя из этого, для сравнения были выбраны только два параметра: количество используемых ТЭ и количество цепей в схеме. Трассировка цепей тактового сигнала и сброса выполняется по выделенному тактовому дереву, поэтому её результат одинаков для обеих реализаций. Исходя из этого, сравнение проводилось только для цепей, проводящих сигналы данных. Статистические данные, полученные для проведения сравнительного анализа, представлены в таблице 2.

Таблица № 2

Сравнение ресурсов, требующихся для имплементаций сдвигового регистра на ПЛИС

		Number of RE			***Number of nets***		
WIDTH	***Number of LE***	***LE***	***IP***	***DIFF***	***LE***	***IP***	***DIFF***
2	2	36	33	**1,09**	3	2	**1,5**
4	4	42	33	**1,27**	5	2	**2,5**
6	6	48	33	**1,46**	7	2	**3,5**
8	8	54	33	**1,64**	9	2	**4,5**
10	10	60	33	**1,82**	11	2	**5,5**
12	12	70	34	**2,06**	13	3	**4,33**
14	14	76	34	**2,23**	15	3	**5**
16	16	82	34	**2,41**	17	3	**5,67**
18	18	88	34	**2,59**	19	3	**6,33**
20	20	94	34	**2,77**	21	3	**7**

В схеме сдвигового регистра с использованием специальных ресурсов трассировка необходима только для входа данных в первый DFF триггер, выхода последнего триггера в цепочке и цепей, соединяющих между собой ГЛБ. В схеме сдвигового регистра на стандартных DFF в трассировке участвуют все цепи. Таким образом межсоединения, объединяющие пару соседних триггеров занимают локальные шины в ГЛБ.

Spice моделирование двух реализаций 10 разрядного сдвиг. регистра показало, что специальные соединения позволяют сократить задержку выходного сигнала на выходе конечного триггера на 13,85 пс. При этом эти соединения позволяют значительно уменьшить количество используемых

трассировочных ресурсов. Например, стандартный 20 разрядный сдвиговой регистр имеет в 7 раз больше цепей и в 2.77 раз больше трассировочный ресурсов, чем сдвиг. регистр из библиотеки гибких СФ-блоков.

Также, как и сдвиговый регистр, схема асинхронного счетчика состоит только из DFF триггеров. Соответственно, метод реализации этой схемы не влияет на число занятых ЛЭ. Исходя из этого, сравнивались два параметра: количество ТЭ и число цепей. Результаты сравнения приведены в таблице 3. Аналогично анализу ресурсов, занимаемых сдвиговым регистром, в статистике не учитывались межсоединения, которые трассируются по выделенному тактовому дереву. Это цепи, соединяющие входную ИО ячейку и пины сброса, а также цепь, соединяющая входную тактовую ИО ячейку и пин CLK первого триггера в цепочке.

Количество цепей в схеме асинхронного счетчика для обеих реализаций увеличивается пропорционально увеличению количества триггеров. У счетчика из элементов, учитывающих специальные ресурсы, каждые два триггера в схеме добавляют 2 цепи, которые требуется трассировать. Это цепи, соединяющие выходы триггеров с выходными И/О ячейками. У счетчика на стандартных библиотечных элементах такое же количество триггеров добавляет в 3 раза больше цепей - 6. Это число складывается из трех составляющих. Цепи, соединяющие выходы и входы данных соседних триггеров. Цепи, соединяющие выходы и тактовые входы соседних триггеров. А также цепи от выходов триггеров до выходных И/О ячеек.

Число используемых трассировочных ресурсов будет отличаться при изменении ГЛБ, в которой размещен блок. Но независимо от размещения это число всегда будет больше в стандартной реализации счетчика. Это связано с тем, что стандартный сумматор всегда имеет обратную связь ко входу данных и соединение выходов и тактовых входов соседних триггеров, в

отличие от специализированного асинхронного счетчика. Например, 20 разрядный счетчик на стандартных элементах занимает в 2 раза больше трассировочных ресурсов, чем счетчик такой же разрядности из специализированных элементов.

Таблица № 3

Сравнение ресурсов, требующихся для имплементаций счетчика на ПЛИС

		Number of RE			***Number of nets***		
WIDTH	***Number of LE***	***LE***	***IP***	***DIFF***	***LE***	***IP***	***DIFF***
2	2	57	34	**1,68**	5	2	**2,50**
4	4	119	68	**1,75**	11	4	**2,75**
6	6	197	102	**1,93**	17	6	**2,83**
8	8	246	141	**1,74**	23	8	**2,88**
10	10	317	175	**1,81**	29	10	**2,90**
12	12	392	200	**1,96**	35	13	**2,69**
14	14	470	224	**2,10**	41	15	**2,73**
16	16	558	253	**2,21**	47	17	**2,76**
18	18	626	292	**2,14**	53	19	**2,79**
20	20	679	336	**2,02**	59	21	**2,81**

Для оценки быстродействия было выполнено Spice-моделирование двух вариантов реализации схемы 10-ти разрядного счетчика. В асинхронном счетчике время переключения сигнала каждого следующего разряда зависит от времени переключения предыдущего. Моделирование показало, что младший (0) разряд стандартного асинхронного счетчика переключается на 0.77 нс быстрее. При этом использование структурных особенностей ЛЭ позволяет уменьшить задержку выходного сигнала на 4,29 нс на первом

триггере и на 24,42 нс на девятом триггере, формирующем старший бит результата.

Заключение

В данной работе рассмотрено влияние специализированных ресурсов ПЛИС на повышение эффективности проектирования гибких СФ-блоков. Представлены арифметические блоки, которые можно разработать, используя эти ресурсы. Описаны особенности реализации таких блоков, как N-битный сумматор/вычитатель, сдвиговый регистр на N-бит и счетчик до N, а также показан учет особенностей при добавлении этих блоков в САПР для программирования ПЛИС.

Кроме этого, в статье проведен сравнительный анализ двух реализаций описанных выше блоков. Сравнивались четыре параметра: площадь, количество цепей, требующих трассировки, количество занятых трассировочных ресурсов и быстродействие схемы. По результатам анализа продемонстрированы преимущества реализации гибких СФ-блоков с использованием архитектурных особенностей ПЛИС

Литература

1. Stempkovsky A. L., Telpukhov D. V., Solovyev R. A., Rukhlov V. S. and Mikhmel A. S. Hardware and Software Solutions to Increase the Reliability of Combinational Logic in the FPGA Basis // 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2020, pp. 1860-1863.
2. Wei X., Diao Y., Lam Tak-Key, Wu Yu-Liang, A universal macro block mapping scheme for arithmetic circuits. 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, 2015, pp. 1629-1634.
3. Фролова П.И., Чочаев Р.Ж., Иванова Г.А., Гаврилов С.В. Алгоритм размещения с оптимизацией быстродействия на основе матриц задержек

для реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем. 2020. Выпуск 1. С. 2-7.

4. Заплетина М.А., Железников Д.А., Гаврилов С.В. Иерархический подход к трассировке реконфигурируемой системы на кристалле островного типа // Проблемы разработки перспективных микро- и наноэлектронных систем. 2020. Выпуск 3. С. 16-21.
5. Spartan-6 FPGA Configurable Logic Block. User Guide. URL: xilinx.com/support/documentation/user_guides/ug384.pdf (дата обращения: 10.05.2021)
6. MAX II Device Handbook. URL: intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max2/max2_mii5v1.pdf (дата обращения: 10.05.2021)
7. Yosys Open SYnthesis Suite. URL: clifford.at/yosys/about.html (дата обращения: 08.05.2021)
8. Gavrilov S.V, Zheleznikov D.A., Zapletina M.V., Khvatov V.M., Chochaev R. Zh., Enns V.I. Layout Synthesis Design Flow for Special-Purpose Reconfigurable Systems-on-a-Chip. Russian Microelectronics, 2019, Vol. 48, No. 3, pp. 176–186
9. Тельпухов Д.В., Рухлов В.С., Рухлов И.С. Исследование и разработка методов оценки сбоеустойчивости комбинационных схем, реализованных в базисе ПЛИС // Инженерный вестник Дона, 2016, №1. URL: ivdon.ru/uploads/article/pdf/IVD_18_Telpukhov.pdf_c9f05ee7b1.pdf (дата обращения: 11.05.2021).
10. Тельпухов Д.В., Рухлов В.С., Иванова Г. А., Рыжова Д.И. и др. Исследование вариантов частичного резервирования при проектировании сбоеустойчивых логических блоков ПЛИС// Инженерный вестник Дона, 2018, №1. URL:ivdon.ru/uploads/article/pdf/IVD_51_Telpuhov_Ruhlov.pdf_594469bce2.pdf (дата обращения: 11.05.2021).

11. Tyagi A. A reduced-area scheme for carry-select adders. IEEE Transactions on Computers, oct 1993, vol.42, no. 10, pp. 1163 - 1170.
12. Hauck S., Hosler M.M., Fry T.W. High-performance carry chains for FPGA's. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, 2000, vol. 8, no. 2, pp. 138-147.
13. Parmar S., Singh P.S., Design of high-speed hybrid carry select adder, 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, 2013, pp. 1656-1663.

References

1. Stempkovsky A. L., Telpukhov D. V., Solovyev R. A., Rukhlov V. S. and Mikhmel A. S. Hardware and Software Solutions to Increase the Reliability of Combinational Logic in the FPGA Basis, 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2020, pp. 1860-1863.
2. Wei X., Diao Y., Lam Tak-Key, Wu Yu-Liang, A universal macro block mapping scheme for arithmetic circuits. 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, 2015, pp. 1629-1634.
3. Frolova P.I., Chochaev R. Zh., Ivanova G. A., Gavrilov S.V. Problemy razrabotki perspektivnyh mikro- i nanoelektronnyh sistem (MES), 2020, № 1, pp. 2-7.
4. Zapletina M. A., Zheleznikov D. A., Gavrilov S.V., Problemy razrabotki perspektivnyh mikro- i nanoelektronnyh sistem (MES), 2020. №3. P. 16-21.
5. Spartan-6 FPGA Configurable Logic Block. User Guide. URL: xilinx.com/support/documentation/user_guides/ug384.pdf (accessed 10 May 2021)
6. MAX II Device Handbook. URL: intel.com/content/dam/www/programmable /us/en/pdfs/literature/hb/max2/max2_mii5v1.pdf (accessed 10 May 2021)

7. Yosys Open SYnthesis Suite: URL: clifford.at/yosys/about.html (accessed 8 May 2021)
8. Gavrilov S.V, Zheleznikov D.A., Zapletina M.V., Khvatov V.M., Chochaev R. Zh., Enns V.I. Russian. Microelectronics, 2019, Vol. 48, No. 3, pp. 176–186.
9. Tel'pukhov D.V., Rukhlov V.S., Rukhlov I.S. Inzhenernyj vestnik Dona, 2016, №1. URL: ivdon.ru/ru/magazine/archive/n1y2016/3504 (accessed 11 May 2021).
10. Tel'pukhov D.V., Rukhlov V.S., Ivanova G. A., Ryzhova D.I. et al. Inzhenernyj vestnik Dona, 2018, №1. URL: ivdon.ru/ru/magazine/archive/n1y2018/4681 (accessed 11 May 2021).
11. Tyagi A. IEEE Transactions on Computers, oct 1993, vol.42, no. 10, pp. 1163 - 1170.
12. Hauck S., Hosler M.M., Fry T.W. High-performance carry chains for FPGA's. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, 2000, vol. 8, no. 2, pp. 138-147.
13. Parmar S., Singh P.S., Design of high-speed hybrid carry select adder, 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, 2013, pp. 1656-1663.