УДК 004.2

А.А. Гриценко, С.Ю. Сероштан, Ю.Е. Зинченко
Донецкий национальный технический университет
anthony.grytsenko@gmail.com

Разработка аппаратного модуля сортировки с последовательным вводом данных и минимальным временем обработки

В предлагаемой статье рассматриваются вопросы, касающиеся построения модулей сортировки в базисе современных ПЛИС FPGA. Внимание уделяется разработке аппаратного модуля, который обеспечивает близкую к максимально возможной скорости с учетом использования ПЛИС FPGA, которые имеют низкую стоимость. В рассмотрение брались ПЛИС FPGA наиболее известных на данный момент производителей, а именно, ПЛИС фирм Altera и Xilinx.
***Ключевые слова**: аппаратная сортировка, ПЛИС FPGA, синтез.*

Введение

В данной статье рассматривается решение задачи сортировки, то есть задачи упорядочения множества чисел $\langle a_0, a_1, \ldots, a_n \rangle$ таким образом, чтобы следующее ограничение было корректным – $\forall a_i, a_{i+1} \in \langle a_0, a_1, \ldots, a_n \rangle, a_i \le a_{i+1}$ [1].

В этой статье рассматривается не какая-либо из программных реализаций алгоритма [2], а разработка аппаратного модуля, который будет решать поставленную задачу.

Использование специальных аппаратных модулей для решения тех задач, которые раньше, по большей части, решались программно, стало на данный момент актуальной задачей в связи с активным развитием, а также распространением, различных видов программируемых логических интегральных схем (ПЛИС).

Современные ПЛИС принято разделять на классы, основываясь, в частности, на способе их программирования [3]. В соответствии с данной классификацией можно выделить ПЛИС FPGA, которые пользуются все большей популярностью среди разработчиков [4]. В свою очередь, ПЛИС FPGA разделяют на ПЛИС с низкой и с высокой стоимостью. Ключевое различие заключается, в первую очередь, в сложности их архитектуры. В частности, это выражается в сложности базового элемента, то есть, генератора функции. В данной статье рассматриваются схемы, которые имеют низкую стоимость, то есть, это ПЛИС семейств Altera Cyclone [5-6] и Xilinx Spartan [7].

Причиной для разработки предлагаемого модуля сортировки стало исследование подходов к минимизации времени работы модуля. То есть, минимизации времени, которое необходимо для обработки одного элемента последовательности, которая упорядочивается.

Основной для разрабатываемого модуля стала схема, которая была описана в [8]. Следует отметить, что данное описание имеет достаточно низкий уровень детализации. Поэтому в данной статье рассматривается вариант его реализации, который может использоваться в базисе ПЛИС FPGA.

Концепция сортировки с использованием последовательного ввода данных

Аппаратные модули сортировки можно разделить на две группы, используя как критерий для классификации способ ввода данных (рис. 1).

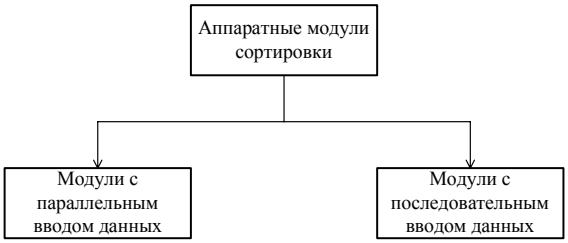

Рисунок 1 – Классификация аппаратных модулей сортировки по критерию способа ввода данных

К первой группе относятся аппаратные модули сортировки, позволяющие осуществлять параллельный ввод данных. Основным примером данного вида модулей являются различные сети сортировки, рассмотренные, например, в [2, 9]. В этом случае сортировка осуществляется по мере прохождения данных по сети. Этот вид модулей сортировки не будет рассматриваться в данной статье.

Ко второй группе относятся те модули, архитектура которых предполагает, что данные будут передаваться последовательно, то есть, не более чем по одному элементу за один рабочий цикл.

На рис. 2 показан вариант архитектуры, который может использоваться для построения модулей сортировки с последовательным вводом данных. Рассматриваемый вариант архитектуры

предполагает, что модуль сортировки состоит из набора одинаковых по своей функциональности блоков. Количество этих блоков, в общем случае, определяется исходя из максимального размера сортируемой последовательности. Учитывая это, можно определить, что необходимые аппаратные затраты можно определить исходя из количества блоков и затрат на отдельный блок. Необходимо также учитывать возможное снижение затрат при оптимизации блоков, находящихся на границах модуля. Как результат получим общую формулу, которую можно использовать для оценки затрат на развертывание аппаратных модулей, которые имеют архитектуру, аналогичную показанной на рис. 2:

$$C_{\mathrm{mod}ule} = C_{block} \times N_{blocks} - S_{borders}, \qquad (1)$$

где $C_{\mathrm{mod}ule}$ – это стоимость всего модуля, C_{block} – стоимость его отдельного блока, N_{blocks} – общее количество блоков, а $S_{borders}$ – экономия ресурсов в блоках, расположенных на границах модуля.

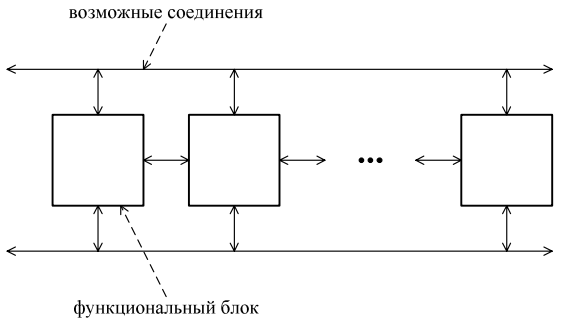

Рисунок 2 – Вариант структуры модулей сортировки с последовательным вводом данных

При использовании модулей сортировки с последовательным вводом данных их временная диаграмма работы должна в той или иной мере соответствовать показанной на рис. 3.

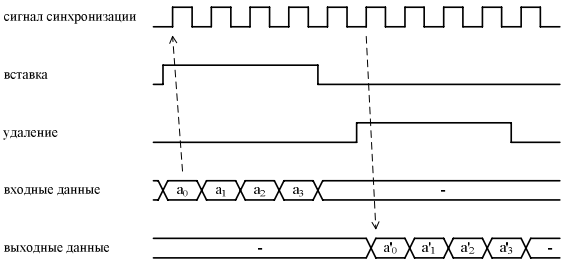

Рисунок 3 – Ожидаемая временная диаграмма роботы модуля сортировки с последовательным вводом данных

В соответствии данной диаграмме можно выделить два процесса, а именно, процесс ввода множества данных, подлежащего сортировке, и процесс получения упорядоченного множества.

Для различения режимов работы модуля, соответствующих показанным выше процессам, можно использовать два управляющих сигнала – сигнал вставки и сигнал удаления элемента.

Имена этих сигналов выбраны исходя из аналогии между рассматриваемым видом модуля сортировки и приоритетной очередью.

Управляющие сигналы вставки и удаления являются взаимоисключающими. То есть, если оба эти сигнала активны в одном рабочем цикле, в таком случае поведение модуля сортировки, в общем случае, непредсказуемо. Это ограничение является важным, во-первых, касательно синтеза модуля, во-вторых, касательно его верификации, что будет рассмотрено далее в этой статье.

Соответственно обобщенный интерфейс для модуля сортировки может выглядеть так, как показано на рис. 4. Этот интерфейс включает три группы сигналов: синхронизации, управления и данных.

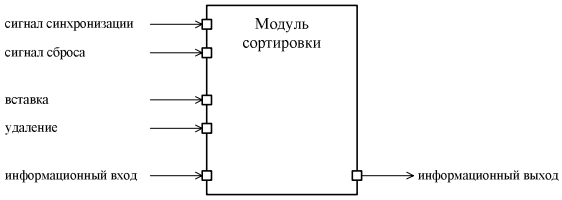

Рисунок 4 – Обобщенный интерфейс модуля сортировки с последовательным вводом данных

Показанный на рис. 4 вариант интерфейса предполагает разделение шин для ввода и вывода данных. Использование раздельных шин данных позволяет избежать сложностей проектирования общей шины, связанных с её работой в третьем состоянии. Относительным недостатком такого интерфейса является то, что в том случае, когда необходимо подключить такой модуль к некоей системной шине, например, в рамках системы на ПЛИС, потребуется разработка дополнительного модуля интеграции.

Структура и свойства модуля сортировки с минимальными аппаратными затратами

В [8] был рассмотрен модуль сортировки, который в данной статье будет рассматриваться как базовый. Следует отметить то, что в [8] дано только очень общее описание концепции, то есть внутренняя структура функционального блока не рассматривается. Также, согласно [8] сам модуль было предложено разделять на управляющую и операционную части.

Принцип работы данного модуля состоит в том, что элементы в каждом из блоков должны быть упорядочены по отношению друг к другу. При этом упорядочение происходит в каждом из

рабочих циклов, то есть, при каждой вставке или удалении элемента.

В этой статье предлагается использовать структуру функционального блока, показанную на рис. 5. В соответствии с показанной на рис. 5 структурой, функциональный блок предназначен для хранения двух элементов упорядочиваемого множества, а, следовательно, количество блоков можно определить по формуле:

$$N_{blocks} = \lfloor (M+1)/2 \rfloor, \qquad (2)$$

где M – максимальный размер входной последовательности. В формуле 2 учитывается, что M может быть нечетным числом. В таком случае, последний блок не будет использоваться эффективно.

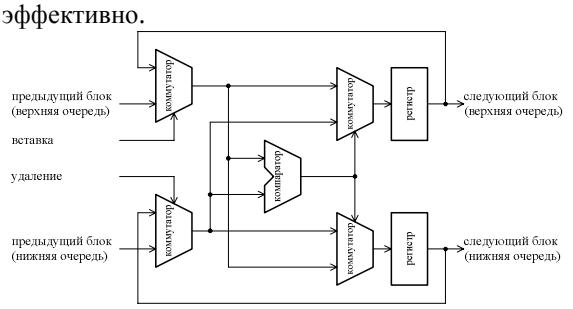

Рисунок 5 –Структура блока модуля сортировки с минимальными аппаратными затратами

Соответствующая структуре блока, общая структура модуля приведена на рис. 6:

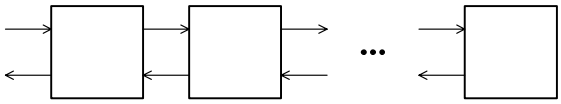

Рисунок 6 – Структура модуля сортировки с минимальными аппаратными затратами

Затраты на построение одного блока для комбинационной части составляют:

$$C_{block}^{comb} = 4 \times C_{mux}^{2x1} + C_{cmp}, \qquad (3)$$

где C_{mux}^{2x1} – затраты на размещение одного коммутатора, а C_{cmp} – соответственно, затраты на размещение компаратора.

Следовательно, затраты из расчета одного элемента упорядочиваемой последовательности составят:

$$C_{element}^{comb} = 2 \times C_{mux}^{2x1} + C_{cmp}/2. \qquad (4)$$

Соответственно, необходимое количество элементов памяти на блок определяется как:

$$C_{block}^{seq} = 2 \times N, \qquad (5)$$

где N – разрядность элемента множества.

Следовательно, затраты из расчета одного элемента составят:

$$C_{element}^{seq} = N, \qquad (6)$$

Время рабочего цикла для данного модуля можно определить на основе критического пути в блоке. В этом случае критический путь будет включать два коммутатора и компаратор (рис. 7).

Следовательно, задержка по критическому пути составит:

$$T_{c.path}^{area} = 2 \times T_{mux}^{p.d.} + T_{cmp}^{p.d.}, \qquad (7)$$

где $T_{mux}^{p.d.}$ и $T_{cmp}^{p.d.}$, соответственно, – время распространения сигнала для коммутатора и для компаратора.

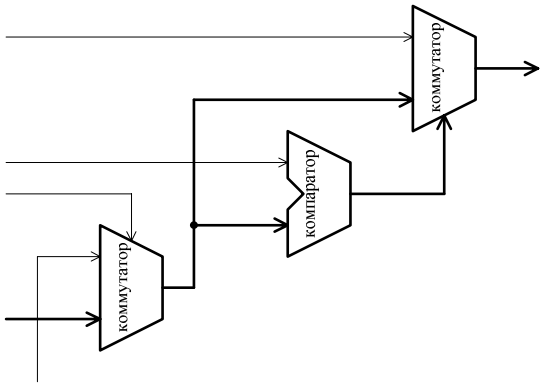

Рисунок 7 – Критический путь блока модуля сортировки с минимальными аппаратными затратами (один из альтернативных вариантов)

Исходя из того как формируется задержка для данного модуля можно предположить, что её уменьшение возможно в случае устранения хотя бы одного из коммутаторов, расположенного на критическом пути.

Структура и свойства модуля сортировки с минимальным временем обработки

Как отмечено выше, основной посылкой при разработке нового модуля сортировки стало желание сократить критический путь в блоке до одного компаратора и одного коммутатора. При этом, в случае использования в качестве базиса ПЛИС FPGA, которые имеют низкую стоимость, дальнейшее улучшение быстродействия модуля не представляется возможным.

В основу предлагаемого модуля положена структура блока, приведенная на рис. 8.

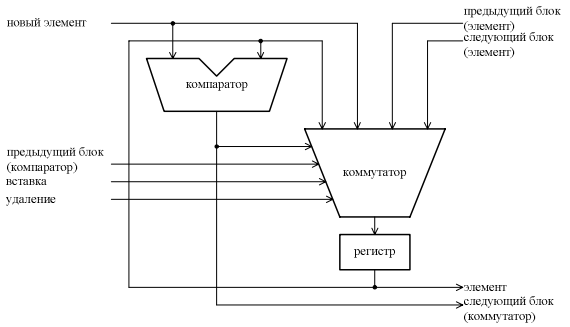

Рисунок 8 – Структура блока модуля сортировки с минимальным временем обработки

В этом случае, каждый блок предназначен для хранения одного элемента. Следовательно, общее количество блоков определяется как:

$$N_{blocks} = M \,. \qquad (8)$$

Также из этого следует то, что аппаратные затраты на блок эквивалентны затратам на один элемент. Для комбинационной части эти затраты будут составлять:

$$C_{block}^{comb} = C_{mux}^{4x1} + C_{cmp}\,, \qquad (9)$$

где C_{mux}^{4x1} – затраты на размещение одного коммутатора, а C_{cmp} – соответственно, затраты на размещение компаратора. Следует отметить то, что затраты на размещение компаратора для данного случая можно принять эквивалентными затратам для предыдущего случая (форм. 3). В то же время, затраты, необходимые для размещения коммутатора отличаются от предыдущего случая и будут рассматриваться более детально.

Для сравнительной оценки затрат нужно привести их к общему базису, то есть, требуется представить коммутатор, который имеет четыре информационных и четыре управляющих входа в форме дерева коммутаторов, которые имеют два информационных и один управляющий вход.

Без учета базиса, то есть, в общем случае, такая задача достаточно сложна, поэтому можно сделать ряд допущений, которые упростят ход ее решения, учитывая, что в данном случае базисом являются использоваться ПЛИС FPGA, которые имеют низкую стоимость. Для данного вида ИС базовой топологической единицей, используемой для построения комбинационных схем, является генератор функции, которая может принимать не более четырех аргументов. А основной единицей памяти является DE-триггер. Следует обратить внимание на то, что наличие входа разрешения у триггера является важным фактором при анализе структуры, показанной на рис. 8.

За счет использования входа разрешения триггера сократим количество информационных и управляющих входов коммутатора. Для этого определим, что состояние регистра, изменяется в том случае, если выполняется одно из условий: или активен сигнал удаления, или активен сигнал вставки и при этом сравнение нового элемента и текущего удовлетворяет условию сортировки. На рис. 9 приведен вариант такой схемы на уровне регистровых передач. При этом, если учитывать принятый базис, можно сделать вывод, что для размещения данной схемы понадобится не более одного генератора функции.

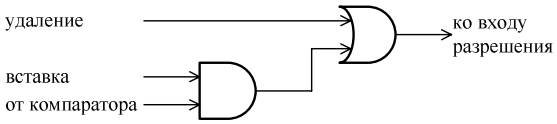

Рисунок 9 – Схема управления входом разрешения DE-триггера

После сокращения получим коммутатор, который имеет три информационных входа и два управляющих. Данный коммутатор может быть представлен в виде дерева из трех коммутаторов, каждый из которых имеет два информационных и один управляющий вход. Причем, как один из управляющих сигналов может быть использован либо сигнал вставки, либо сигнал удаления. На рис. 10 показан вариант реализации описанного дерева коммутаторов. Важным аспектом данного варианта является то, что сигнал, получаемый от компаратора, подается на компаратор, который находится в корне дерева.

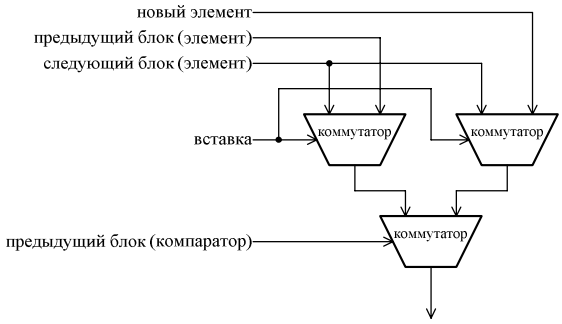

Рисунок 10 – Схема коммутатора 3x1 построенного на коммутаторах 2x1

Как результат имеем два альтернативных критических пути. Первый включает компаратор схему управления входом разрешения триггера (рис. 11). Задержка в данном случае составляет:

$$T_{c.path\ A}^{delay} = T_{cmp}^{p.d.} + T_{ena}^{p.d.}\,, \qquad (10)$$

где $T_{ena}^{p.d.}$ – это время, которое нужно для прохождения сигнала сквозь схему управления входом разрешения.

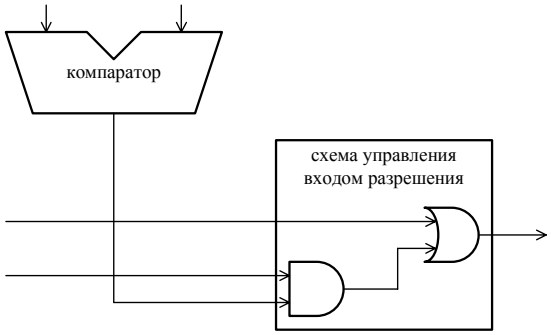

Рисунок 11 – Критический путь блока модуля сортировки с минимальным временем обработки (вариант А)

В свою очередь, второй вариант включает компаратор (только предыдущего, а не текущего блока) и один компаратор, являющийся корнем дерева на рис. 10.

В этом случае задержка составит:

$$T_{c.path\ B}^{delay} = T_{cmp}^{p.d.} + T_{mux}^{p.d.}\,. \qquad (11)$$

Предложенная схема управления входом разрешения работает как минимум не медленнее, чем схема коммутирования двух бит. А в случае использования рассматриваемых ПЛИС FPGA в обоих этих случаях задержка будет определяться задержкой генератора функции. Следовательно, учитывая, что $T_{ena}^{p.d.} \le T_{mux}^{p.d.}$, можно считать, что максимальное время обработки предложенного модуля может быть определено по формуле 11.

Структура модуля в данном случае будет соответствовать показанной на рис. 12. Учитывая сложность схем соединения на рис. 13 показана более подробная схема.

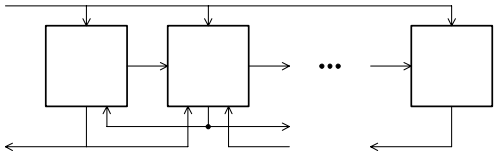

Рисунок 12 – Структура модуля сортировки с минимальным временем обработки

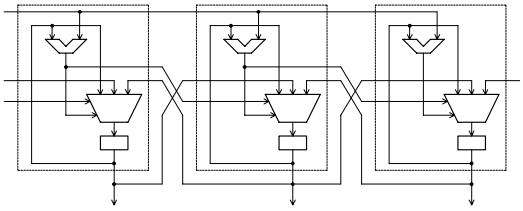

Рисунок 13 – Подробная структура модуля сортировки с минимальным временем обработки

Последним аспектом, который нуждается в рассмотрении, это расчет аппаратных затрат на размещение блока. В этом случае они составят:

$$C_{block}^{comb} = 3 \times C_{mux}^{2x1} + C_{cmp} + C_{ena}, \qquad (12)$$

Сравнив данные затраты с затратами для предыдущего модуля (формула 4) получим, что разница составит:

$$C_{diff} = C_{mux}^{2x1} + C_{cmp} / 2 + C_{ena}. \qquad (13)$$

Далее будут рассмотрены количественные показатели такого превышения расхода ресурсов.

Отметим также, что использование памяти в данном случае эквивалентно предыдущему, а потому может использоваться формула 6.

Описание и синтез модулей сортировки

На данный момент существует несколько языков, используемых для описания аппаратных модулей. В данной статье будет использоваться язык VHDL. Данный язык описания аппаратуры широко известен и описан в большом количестве различных источников, среди которых принято выделять [10], как наиболее фундаментальный. Также данный язык был стандартизирован, при этом, стандарт несколько раз пересматривался и на данный момент актуальна версия от 2008 года [11]. Но, на данный момент часто используется более старая версия стандарта от 1993 года.

В языке VHDL описание модуля включает две части: описание его интерфейса и описание его реализации.

Описание интерфейса рассматриваемых в данной статье модулей сортировки может быть выполнено на основе рис.4. При этом требуется также учитывать, что оба рассмотренных модуля обладают гибкостью в области размер множества и разрядности элементов. Эти параметры могут быть описаны в специальной секции [10]. Итак, учитывая обозначенные особенности интерфейс модуля сортировки можно описать следующим образом (рис. 14):

```
entity sorter is
  generic (
    n : positive := 8;
    m : positive := 16
  );
  port (
    reset  : in bit;
    clock  : in bit;

    insert : in bit;
    remove : in bit;

    din  : in bit_vector (n-1 downto 0);
    dout : out bit_vector (n-1 downto 0)
  );
end entity sorter;
```

Рисунок 14 – Описание интерфейса модуля сортировки на языке VHDL

Использование типа bit является спорным, что, в частности, рассматривается в [3], однако в этом случае принято считать его использование допустимым.

С другой стороны, описание реализации представляет собой значительно более сложную задачу. Причина заключается в том, что даже в случае использования ПЛИС FPGA это описание можно выполнить на нескольких разных уровнях абстракции. При этом возникает противоречие, которое рассматривается, например, в [12]. Оно заключено в том, что, с одной стороны, описание на высоком уровне абстракции легче переносить между разными средствами синтеза, а с другой стороны, результаты синтеза высокоуровневого описания могут сильно разниться. В частности, результаты синтеза такого описания для одного средства могут быть вполне предсказуемыми, в то же время, как другое средство синтеза может дать совершенно непредвиденные результаты.

Сначала, определим те основные уровни абстракции, которые могут быть применены при описании на языке VHDL рассмотренных ранее модулей. Наиболее высоким уровнем абстракции будет обладать то описание, которое не включает деталей реализации отдельных блоков, при этом описывая роботу модуля в целом. В этом случае появляется возможность использовать сложные средства поведенческого описания, кроме того, описание будет очень кратким и основываться на концепции работы модуля. Следующим уровнем абстракции является описание отдельных блоков

на основе их структурных схем, показанных на рис. 5 и рис. 8. В этом случае необходимо будет отдельно описать схему соединения блоков и их подключения к внешнему интерфейсу. Однако, может потребоваться описание на более низком уровне абстракции. В частности, оно может быть необходимо для описания схемы коммутации на рис. 10.

Учитывая, что сложность (как следствие, и объем) описания растет при снижении уровня абстракции, в данной статье приводятся примеры описаний на наиболее высоком уровне.

Например, на рис. 15 представлен вариант описания модуля с минимальными аппаратными затратами, а на рис. 16 – аналогичное описание модуля с минимальным временем обработки.

```
architecture min_area of sorter is
  subtype elem_t is bit_vector (n-1 downto 0);
  constant queue_len : positive := (m+1) / 2;
  type queue_t is array (0 to queue_len - 1) of elem_t;
  constant value_max : elem_t := (others => '1');
begin
  process (reset, clock) is
    variable upper, lower : queue_t;
    variable temp : elem_t;
  begin
    if reset = '1' then
      upper := (others => value_max);
      lower := (others => value_max);
    elsif rising_edge (clock) then
      if insert = '1' then
        upper := din & upper (0 to queue_len - 2);
      end if;
      if remove = '1' then
        lower := lower (1 to queue_len - 1) & value_max;
      end if;
      for i in 0 to queue_len - 1 loop
        if upper (i) < lower (i) then
          temp := lower (i);
          lower (i) := upper (i);
          upper (i) := temp;
        end if;
      end loop;
    end if;
    dout <= lower (0);
  end process;
end architecture min_area;
```

Рисунок 15 – Описание модуля сортировки с минимальными аппаратными затратами

```
architecture min_delay of sorter is
  subtype elem_t is unsigned (n-1 downto 0);
  constant value_max : elem_t := (elem_t'range => '1');
  type elems_t is array (0 to m-1) of elem_t;
  signal elems : elems_t;
begin
  process (reset, clock) is
  begin
    if reset = '1' then
      elems <= (others => value_max);
    elsif rising_edge (clock) then
      if remove = '1' then
        elems <= value_max & elems (0 to elems'right - 1);
      elsif insert = '1' then
        for i in 0 to elems'right - 1 loop
          if unsigned (din) < elems (i) then
            if unsigned (din) < elems (i + 1) then
              elems (i) <= elems (i + 1);
            else
              elems (i) <= unsigned (din);
            end if;
          end if;
        end loop;
        if unsigned (din) < elems (elems'right) then
          elems (elems'right) <= unsigned (din);
        end if;
      end if;
    end if;
  end process;
  dout <= bit_vector (elems (elems'right));
end architecture min_delay;
```

Рисунок 16 – Описание модуля сортировки с минимальным временем обработки

Следует отметить, что уменьшение уровня абстракции увеличивает сложность описания, но, кроме того, может приводить к зависимости от средства синтеза, что нежелательно.

Оба описания выполнены в соответствии со схемами, рассмотренными ранее, однако, как отмечалось, они большей частью описывают не структуру, а концепцию работы модуля.

В качестве средств синтеза были приняты распространенные САПР Altera Quartus II 10.1 и Xilinx ISE 12.2. ПЛИС, которые использовались в качестве примера, были выбраны по критерию распространенности. На данный момент такими можно считать ПЛИС Altera Cyclone III и Xilinx Spartan 3E.

Сначала рассмотрим результаты синтеза описания модуля с минимальными аппаратными затратами. Так как схема блока для этого модуля (рис. 5) достаточно проста, в том смысле, что она не требует специфичных результатов синтеза, то можно было ожидать, что описание на высоком уровне окажется переносимым.

Вначале, оценим ожидаемые результаты синтеза. Схема блока включает лишь компаратор и набор коммутаторов. Если учесть, что взятые в качестве базиса ПЛИС базируются на генераторе функции от четырех аргументов, можно ожидать, что для размещения коммутатора, который имеет два информационных и один управляющий вход, требуется не более N генераторов функции. В свою очередь аппаратные затраты на размещение компаратора сильно зависят от типа элементов. Например, если речь идет о двух целых числах, компаратор может быть модификацией модуля вычитания. В таком случае, для его размещения также потребуется не более чем N генераторов. В табл. 1 приведены все ожидаемые затраты на размещение модуля сортировки с минимальными аппаратными затратами и его компонент. Расчет осуществлялся на основе формул 1-3.

Таблица 1 – Ожидаемые затраты для модуля с минимальными аппаратными затратами (для комбинационной части)

Компонент	Затраты (генераторов функции от 4-х аргументов)
коммутатор	N
компаратор	N
блок	$5\times N$
модуль	$(5\times N)\times\lfloor (M+1)/2 \rfloor$

То есть, если нужно синтезировать модуль для сортировки 16-ти восьмиразрядных чисел, то в этом случае затраты составят 320 генераторов функции. Конечно, это будет верно только в том случае, когда схема, полученная после синтеза, соответствует показанным на рис. 5 и рис. 6.

При использовании САПР Altera Quartus II результаты синтеза описания, которое приведено на рис. 15, удовлетворяют ожиданиям. На рис. 17 приведена RTL-схема, полученная после синтеза модуля для 8-ми восьмиразрядных чисел. Если проанализировать показанную на рис. 17 схему, то можно сделать вывод, что она в полной мере соответствует ожиданиям.

Также, следует отметить то, что затраты, в данном случае, соответствуют табл. 1.

Однако, если данное описание перенести в САПР Xilinx ISE 12.2, то возникнет достаточно интересная проблема. Проанализировав затраты, можно увидеть, что превышение, относительно ожидаемых, в данном случае, затрат, составляет $M-1$ генераторов функций.

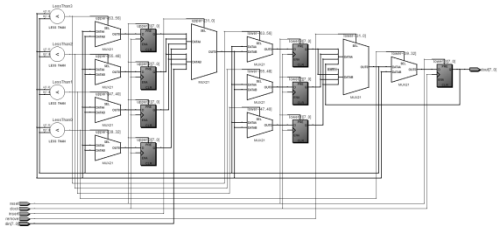

Рисунок 17 – Результаты синтеза модуля с минимальными аппаратными затратами в среде САПР Altera Quartus II 10.1

Проблема заключается в том, что данное средство синтеза дает неожиданный результат. В частности, данное средство добавляет ненужные схемы для управления входом разрешения DE-триггеров (рис. 18).

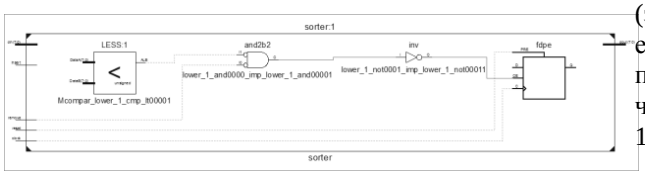

Рисунок 18 – Непредвиденные результаты синтеза модуля с минимальными аппаратными затратами в среде Xilinx ISE 12.2

Результатом появления этих схем является появление нового варианта критического пути, однако, задержка по данному пути не превышает рассмотренной ранее.

Следующий этап это анализ задержки по критическому пути анализируемого модуля. Как и в рассмотренном выше случае оценки затрат, вначале необходимо дать оценки для ожидаемой задержки. Задержка коммутатора составляет не более чем время задержки одного генератора, так как коммутирование всех разрядов выполняется параллельно. С другой стороны, компаратор это последовательная схема, время работы которой определяется разрядностью данных. Учитывая то, что рассматриваются целые числа, задержка будет равна задержке N генераторов функции. В табл. 2 показаны ожидаемые задержки модуля и компонентов, рассчитанные с учетом формулы 7:

Таблица 2 – Ожидаемые задержки для модуля с минимальными аппаратными затратами

Компонент	Задержка (множитель задержки одного генератора функции)
коммутатор	1
компаратор	N
модуль	$2+N$

Рассмотрим данные отчета, содержащего информацию о критическом пути, который был сформирован в среде Xilinx ISE 12.2 (рис. 19):

```
                                        Gate     Net
    Cell:in->out      fanout   Delay   Delay
    ------------------------------------------
     FDPE:C->Q             3   0.514   0.520
 1   LUT3:I1->O            2   0.612   0.532
 2   LUT2:I0->O            1   0.612   0.000
 3   MUXCY:S->O            1   0.404   0.000
 4   MUXCY:CI->O           1   0.052   0.000
 5   MUXCY:CI->O           1   0.052   0.000
 6   MUXCY:CI->O           1   0.052   0.000
 7   MUXCY:CI->O           1   0.052   0.000
 8   MUXCY:CI->O           1   0.052   0.000
 9   MUXCY:CI->O           1   0.052   0.000
10   MUXCY:CI->O          18   0.399   0.977
11   LUT2:I1->O            8   0.612   0.643
     FDPE:CE                   0.483
    ------------------------------------------
    Total                      6.618ns
```

Рисунок 19 – Часть отчета о задержке по критическому пути для модуля с минимальными аппаратными затратами

В соответствии с отчетом, показанным на рис. 19, критический путь включает: коммутатор (элемент 1), компаратор (элементы 2-10), а также еще один коммутатор (элемент 11). Эти данные полностью соответствуют ожиданиям. Отметим, что данные, полученные в САПР Altera Quartus II 10.1, аналогичны тем, что показаны выше.

При синтезе описания модуля сортировки с минимальным временем обработки проблемы возникают уже в САПР Altera Quartus 10.1. Но, сначала рассмотрим результаты синтеза данного модуля в среде Xilinx ISE 12.2.

Согласно табл. 1 и формул 8 и 12, а также с учетом того, что размещение схемы управления входом разрешения триггера потребует не более одного генератора функции, получим, исходную формулу расчета ожидаемых затрат для данного модуля:

$$C_{\mathrm{mod}\,ule}^{delay} = M \times (4 \times N + 1). \qquad (14)$$

Однако, согласно формулы 1 необходимо также учитывать экономию при синтезе блоков, которые находятся на границе модуля. В данном случае это актуально, поскольку, если выполнить анализ описания на рис. 16 и схем на рис. 8 и на рис. 10, то можно сделать вывод, что для блоков,

находящихся на границах модуля затраты будут составлять:

$$C_{block}^{comb} = C_{mux}^{2x1} + C_{cmp} + C_{ena}. \quad (15)$$

С учетом формулы 12 получим экономию:

$$S_{borders} = 2 \times \left(2 \times C_{mux}^{2x1}\right). \quad (16)$$

Тогда, подставив формулу 16 в формулу 1 получим конечную форму ожидаемых затрат для модуля с минимальным временем обработки:

$$C_{\mathrm{mod}\,ule}^{delay} = \left(M - 1\right) \times \left(4 \times N\right) + M\ , \quad (17)$$

то есть затраты на размещения модуля для 16-ти восьмиразрядных элементов должны быть равны 496 генераторам функции. Следовательно, перерасход аппаратных ресурсов, по отношению к модулю с минимальными затратами в данном случае составят 55 процентов.

Однако, такой перерасход оправдан, если учесть, что задержка критического пути в этом случае меньше, что можно видеть из частичного отчета, показанного на рис. 20.

```
                                    Gate     Net
    Cell:in->out      fanout        Delay    Delay
    ------------------------------------------
     FDPE:C->Q             4        0.514    0.568
 1   LUT2:I1->O            1        0.612    0.000
 2   MUXCY:S->O            1        0.404    0.000
 3   MUXCY:CI->O           1        0.052    0.000
 4   MUXCY:CI->O           1        0.052    0.000
 5   MUXCY:CI->O           1        0.052    0.000
 6   MUXCY:CI->O           1        0.052    0.000
 7   MUXCY:CI->O           1        0.052    0.000
 8   MUXCY:CI->O           1        0.052    0.000
 9   MUXCY:CI->O           9        0.399    0.727
10   LUT3:I2->O            8        0.612    0.643
     FDPE:CE                        0.483
    ------------------------------------------
    Total                           5.272ns
```

Рисунок 20 – Часть отчета о задержке по критическому пути для модуля с минимальным временем обработки

При использовании среды Altera Quartus II 10.1 возникает проблема, что результаты синтеза совершенно не удовлетворяют ожиданиям. При этом снижение уровня абстракции описания не дает результатов до тех пор, пока это описание не переходит на уровень примитивов ПЛИС. Эта проблема проявляется, в основном, при синтезе схемы коммутирования, показанной на рис. 10. Вместо нужной схемы, средство синтеза САПР формирует каскад, который содержит только два генератора функций (рис. 21).

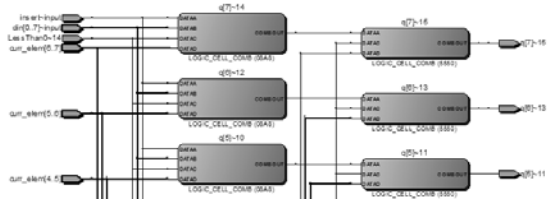

Рисунок 21 – Схема коммутатора 3x1, формируемого САПР Altera Quartus II 10.1 вместо схемы, показанной на рис. 10

В сгенерированной схеме сигнал, который формируется компаратором, подается на первый каскад мультиплексоров, что исключает выгоду в скорости работы.

Для того чтобы сформировать ожидаемую схему требуется воспользоваться примитивами, описанными в [13]. Такое описание может быть выполнено следующим образом (рис. 22).

```
entity routing_net is
  generic (
    n : positive := 8
  );
  port (
    dataa : in bit_vector (n - 1 downto 0);
    datab : in bit_vector (n - 1 downto 0);
    datac : in bit_vector (n - 1 downto 0);
    sela : in bit;
    selb : in bit;
    q : out bit_vector (n - 1 downto 0)
  );
end entity routing_net;

architecture low_level of routing_net is
 signal cascade1_in : bit_vector (n - 1 downto 0);
 signal cascade2_in : bit_vector (n - 1 downto 0);
 signal cascade3_in : bit_vector (n - 1 downto 0);
 signal cascade1_out : bit_vector (n - 1 downto 0);
 signal cascade2_out : bit_vector (n - 1 downto 0);

  component lcell
    port (a_in  : in bit; a_out : out bit);
  end component lcell;
begin
  cascade1 : for i in n - 1 downto 0 generate
    cascade1_in (i) <= (dataa (i) and sela) or datac (i);
    cascade1_lcell : lcell
      port map (cascade1_in (i), cascade1_out (i));
  end generate cascade1;
  cascade2 : for i in n - 1 downto 0 generate
    cascade2_in (i) <= (datab (i) and sela) or datac (i);
    cascade2_lcell : lcell
      port map (cascade2_in (i), cascade2_out (i));
  end generate cascade2;
  cascade3 : for i in n - 1 downto 0 generate
    cascade3_in (i) <= (cascade1_out (i) and selb)
      or cascade2_out (i);
    cascade3_lcell : lcell
      port map (cascade3_in (i), q (i));
  end generate cascade3;
end architecture low_level;
```

Рисунок 22 – Вариант описания схемы коммутирования, показанной на рис. 10, с использованием примитивов низкого уровня для САПР Altera Quartus II 10.1

В результате получим схему, которая была показана на рис. 10. Однако, останется еще одна проблема, которая заключается в том, что САПР не выполнит корректную оптимизацию блоков, которые находятся на границах модуля. Поэтому может необходимо описать эти блоки отдельно, что еще более увеличит сложность описания.

То есть, описание рассмотренных модулей сортировки может быть выполнено на высоком уровне абстракции, однако, при использовании конкретных средств нужно учитывать, что такое описание может некорректно синтезироваться.

Направления дальнейших исследований

Предложенная в этой статье схема модуля сортировки с минимальным временем обработки обладает преимуществом, которое не может быть реализовано в случае использования в качестве базиса ПЛИС FPGA с генераторами функции для трех или четырех аргументов. Это преимущество проявляется при использовании ПЛИС FPGA с более сложными генераторами функции.

В частности, в данном случае речь идет о ПЛИС FPGA, которые принадлежат, например,

семейству Altera Stratix III [14]. Отличием этих ПЛИС является то, что их генераторы функции поддерживают до шести различных аргументов. В этом случае наличие в схеме модуля сложного компаратора является важным преимуществом, так как это дает возможности для оптимизации схемы модуля в базисе данных ПЛИС.

Например, в табл. 3 приведены результаты синтеза рассмотренных модулей в базисе ПЛИС FPGA Altera Stratix III в САПР Altera Quartus II.

Таблица 3 – Результаты синтеза модулей сортировки в базисе ПЛИС FPGA Altera Stratix III в САПР Altera Quartus II

$M(N)$	Модуль с минимальными аппаратными затратами		Модуль с минимальным временем обработки	
	затраты	скорость	затраты	скорость
16(8)	304	360	259	383
16(16)	643	319	571	315
32(8)	610	342	521	387
32(16)	1287	304	1145	299

Отметим, что в табл. 3 затраты приводятся только для комбинационной части устройств, а скорость дается в МГц для худшего допустимого случая (медленная модель с учетом температуры рабочей среды в 85С).

Выполнив предварительный анализ табл. 3, можно сделать вывод о том, что в новом базисе характеристики модулей изменяются, а потому требуют более тщательного анализа.

Другим направлением для исследования является верификация модулей. Учитывая то, что модули сортировки имеют блочную структуру, их верификация может быть упрощена.

Выводы

В данной статье рассмотрены два варианта аппаратных модулей сортировки. Для каждого из этих вариантов даны точные оценки аппаратных затрат и временных характеристик. Однако, эти оценки даны для базиса ПЛИС FPGA, имеющих низкую стоимость.

Описывается концепция работы и предлагается структура модуля, который имеет минимальное время обработки. Как альтернатива этому модулю была предложена структура модуля, концепция которого описана в [8]. Вторая часть статьи посвящена вопросам описания этих модулей с использованием языка описания аппаратуры VHDL и последующего их синтеза с использованием современных САПР, дано полное обоснование всех оценок различных характеристик рассмотренных модулей, четко определены проблемы, которые могут проявиться в процессе описания и синтеза модулей.

Литература

1. Алгоритмы построение и анализ / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. – М. : Издательский дом «Вильямс», 2007. – 1296 с. : ил.
2. Кнут Дональд Искусство программирования: Т. 3. Сортировка и поиск; 2-е изд.: пер. с англ. / Дональд Кнут. – М. : ООО «И.Д. Вильямс», 2007. – 832с. : ил.
3. Chu Pong P. RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability / P. P. Chu. – New Jersey. : «John Wiley and Sons», 2006. – 669 pp.
4. Kilts Steve. Advanced FPGA Design Architecture, Implementation, and Optimization / S. Kilts. - New Jersey. : «John Wiley and Sons», 2007. – 355 pp.
5. Cyclone III Device Handbook [Electronic Resource] – Mode of access : http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf – Title from the screen.
6. Cyclone IV Device Handbook [Electronic Resource] – Mode of access : http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf – Title from the screen.
7. Spartan-3 Generation FPGA User Guide [Electronic Resource] – Mode of access : http://www.xilinx.com/support/documentation/user_guides/ug331.pdf – Title from the screen.
8. Палагин А.В. Реконфигурируемые вычислительные системы: Основы и приложения / А. В. Палагин, В. Н Опанасенко. – К.: Просвіта, 2006. – 280 с.: ил.
9. Batcher K. E. Sorting networks and their applications / K. E. Batcher. – «Goodyear Aerospace Corporation», 1968. – 8 pp.
10. Ashenden Peter J. The Designer’s Guide to VHDL / P. J. Ashenden. – SF. : «Morgan Kaufmann Publishers, Inc.», 1995. – 688 pp.
11. IEEE Standard VHDL Language Reference Manual: IEEE Std. 1076-1993 – [Чинний від 15.09.1993] – IEEE, 1994. – 246 pp. – International Standard.
12. Chu Pong P. FPGA Prototyping by VHDL Example Xilinx Spartan™-3 Version / P. P. Chu. – New Jersey. : «John Wiley and Sons», 2008. – 440 pp.
13. Designing with Low-Level Primitives [Electronic Resource] - Mode of access : http://www.altera.com/literature/ug/ug_low_level.pdf - Title from the screen.
14. Stratix III Device Handbook [Electronic Resource] - Mode of access : http://www.altera.com/literature/hb/stx3/stratix3_handbook.pdf - Title from the screen.

Надійшла до редакції 30.01.2011