УДК 621.3; 004.3 DOI: 10.31114/2078-7707-2021-2-9-16

Методика ASMD-FSMD проектирования цифровых устройств

В.В. Соловьев

Белостокский технологический университет (Польша), valsol@mail.ru

***Аннотация* — Рассмотрена методика ASMD-FSMD проектирования цифровых устройств на основе конечных автоматов с трактом обработки данных (finite state machine with datapath – FSMD), когда функционирование устройства представляется в виде блок-схемы автомата с трактом обработки данных (algorithmic state machine with datapath – ASMD) и описывается на языке Verilog. Приведено сравнение традиционного подхода и методики ASMD-FSMD при проектировании синхронных умножителей и процессоров PIC на программируемых логических интегральных схемах (ПЛИС – field programmable gate array – FPGA). Показано, что методика ASMD-FSMD позволяет в большинстве случаев уменьшить стоимость реализации (для отдельных примеров на 47%) и заметно увеличить быстродействие (для отдельных примеров в 2.96 раза), а также значительно сократить время проектирования (приблизительно в 5-7 раз). Приведены рекомендации по использованию методики ASMD-FSMD и указаны возможные направления ее дальнейшего развития.**

***Ключевые слова* — методика проектирования цифровых устройств, конечный автомат с трактом обработки данных, блок-схема автомата с трактом обработки данных, язык Verilog, программируемая логическая интегральная схема (ПЛИС).**

I. Введение

В последнее время наблюдается возрастание сложности цифровых систем с одновременным ужесточением требований к срокам разработки и повышению быстродействия проектов. Одним из направлений решения указанной проблемы является внедрение в практику инженерного проектирования новых методик проектирования цифровых устройств. К таким методикам относятся методы проектирования быстрых конечных автоматов [1] и быстрых компараторов большой разрядности [2]. В настоящей работе рассматривается более общая методика проектирования цифровых устройств, которая позволяет уменьшить стоимость реализации, повысить быстродействие и сократить время разработки для большинства цифровых устройств.

Традиционно проектируемое цифровое устройство принято представлять в виде операционного устройства (datapath) и устройства управления (control unit), которые обычно проектируются отдельно: операционное устройство – в виде совокупности стандартных функциональных узлов (регистров, шин, мультиплексоров и др.), а устройство управления – в виде конечного автомата (finite state machine – FSM).

В [3] предложено операционное и управляющее устройства объединить вместе и представить как конечный автомат с трактом обработки данных (finite state machine with datapath – FSMD). Модель FSMD быстро стала популярной, в [4] приведены FSMD для синхронных и асинхронных проектов. В [5] предложено цифровую систему представлять в виде сети FSMD, которая приводит к реализации аппаратуры свободной от гонок.

Общая модель FSMD не всегда удобна при проектировании конкретных приложений. Поэтому в ряде работ предлагаются расширения FSMD: в [6] – для представления архитектуры процессора и ASIC (application-specific integrated circuit); в [7] – для синхронного доступа к памяти. Сравнение эффективности FSM и FSMD приводится в [8].

Благодаря своей наглядности *блок-схемы автоматов* (algorithmic state machine – ASM) получили широкое распространение для представления алгоритмов функционирования конечных автоматов. ASM впервые были предложены в [9], как альтернатива графам автоматов. Традиционно ASM используются для представления алгоритма функционирования устройства управления. В [10] предложено ASM использовать как для описания поведения устройства управления, так и для описания регистровых операций, выполняемых в операционном устройстве. Такая ASM получила название блок-схема автомата с трактом обработки данных (algorithmic state machine with datapath – ASMD). Диаграммы ASMD в последнее время все чаще применяются в проектах на FPGA (field programmable gate array – программируемая логическая интегральная схема – ПЛИС): при реализации промышленных систем управления [11], для реализации функции asin с помощью алгоритма CORDIC [12], при аппаратной реализации криптографического алгоритма AES [13], при проектировании универсального асинхронного приема-передатчика UART [14] и др.

В настоящей работе рассматривается методика проектирования цифровых устройств ASMD-FSMD, когда функционирование устройства представляется в виде диаграммы ASMD, которая на языке Verilog описывается в виде FSMD, на примере реализации на FPGA синхронных умножителей и процессоров PIC.

II. Традиционный подход к проектированию цифровых устройств

Рассмотрим простейший школьный алгоритм умножения, выполняющий арифметическую операцию умножения P = A*B двух двоичных чисел шириной N битов.

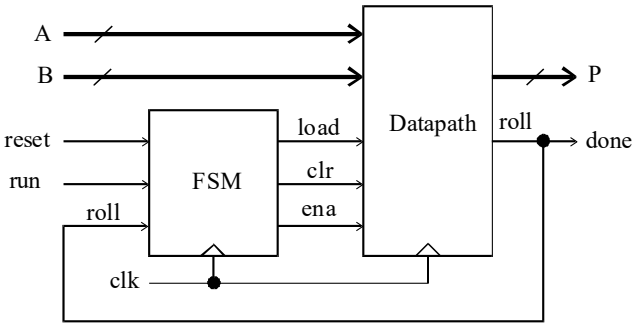

Рис. 1. Структурная схема синхронного умножителя в виде операционного устройства и устройства управления

На рис. 1 представлена структурная схема синхронного умножителя в виде *операционного устройства* (Datapath) и *устройства управления* (FSM). Значения чисел A и B поступают на вход операционного устройства. На выходах операционного устройства формируется произведение P и сигнал done, указывающий на окончание процесса умножения. Кроме того, операционное устройство формирует внутренний сигнал roll, совпадающий с сигналом done. Устройство управления FSM формирует следующие управляющие сигналы: load – для загрузки в регистры операционного устройства значений умножаемых слов A и B; clr – для сброса регистров операционного устройства; ena – для разрешения операции сдвига содержимого сдвиговых регистров.

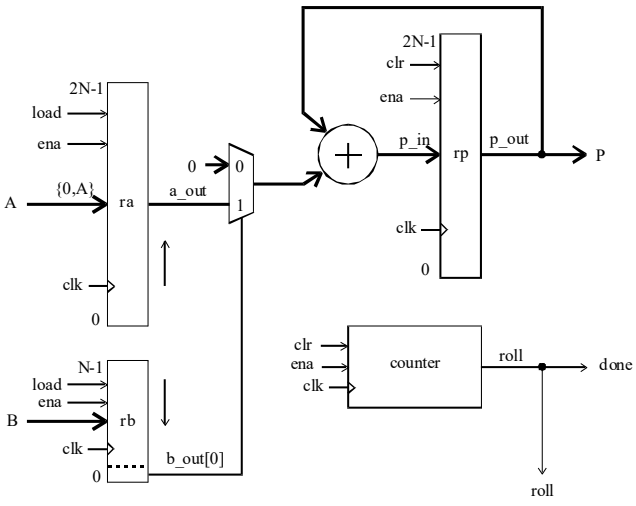

Рис. 2. Схема операционного устройства синхронного умножителя

Схема операционного устройства, реализующего алгоритм умножения, показана на рис. 2. Операционное устройство включает сдвиговый регистр влево ra, сдвиговый регистр вправо rb и регистр rp для хранения слов A, B и P соответственно, шинный мультиплексор 2-1 и сумматор на 2N разрядов, а также счетчик по модулю counter, который формирует флаг окончания операции умножения roll.

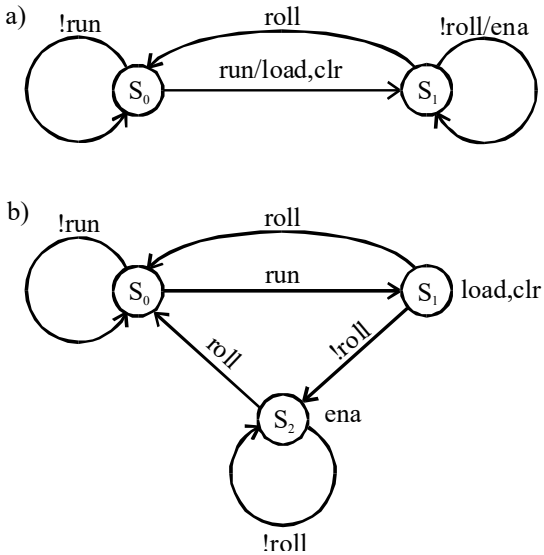

Рис. 3. Представление устройства управления синхронного умножителя в виде графа конечного автомата: a – типа Мили, b – типа Мура

Функционирование устройства управления синхронного умножителя можно представить в виде автомата типа Мили (рис. 3,a) или автомата типа Мура (рис. 3,b). Детали описания компонентов операционного устройства и конечных автоматов Мили и Мура на языке Verilog при реализации синхронного умножителя в случае традиционного подхода приведены в [15].

III. Блок-схемы автоматов ASM

Блок-схема автомата (ASM) предназначена для наглядного описания алгоритма функционирования конечного автомата и представляет собой ориентированный связный граф [9], содержащий вершины трех типов: прямоугольники – *вершины состояний* (state box); ромбы – *условные вершины* (decision box); овалы – *вершины выходов по условию* (conditional output box).

Вершина состояния ASM (прямоугольник) определяет состояние автомата. Вблизи вершины состояния может записываться имя состояния (например, S0, START, INITIAL и др.), а также двоичный код состояния. В случае автомата Мура внутри вершины состояния записываются выходные сигналы, принимающие единичное значение в данном состоянии. По умолчанию полагается, что все остальные выходные сигналы в данном состоянии имеют нулевое значение.

В условных вершинах ASM (ромбах) записываются проверяемые условия. Условная вершина ASM представляет собой точку ветвления алгоритма. Выходы условной вершины обозначаются значениями

0 и 1, которые соответствуют переходам в случае нулевого (ложного) или единичного (истинного) значения результата проверки условия. В качестве условия может выступать входная переменная конечного автомата, логическое выражение, единичный разряд битового вектора и др.

В вершинах выходов по условию (овалах) записываются выходные сигналы автомата Мили, принимающие единичное значение на определенном переходе. В ASM для автоматов Мура отсутствуют вершины выходов по условию (овалы), а в ASM для автоматов Мили в вершинах состояний ничего не записывается.

Главным строительным элементом блок-схемы ASM является *блок ASM* (рис. 4). Блок ASM описывает поведение автомата в одном состоянии в течение одного такта синхронизации. Поэтому блок ASM включает только одну вершину состояния (прямоугольник) и может иметь несколько условных вершин (ромбов) и вершин выходов по условию (овалов), причем ромбы могут как предшествовать овалам, так и следовать после овалов. Входы и выходы вершин соединяются с помощью дуг. Блок ASM имеет только один вход, который является входом в вершину состояния, и может иметь один или несколько выходов. Внутри блока ASM запрещены обратные связи. Циклы алгоритма и ждущие вершины в ASM реализуются с помощью внешних обратных связей.

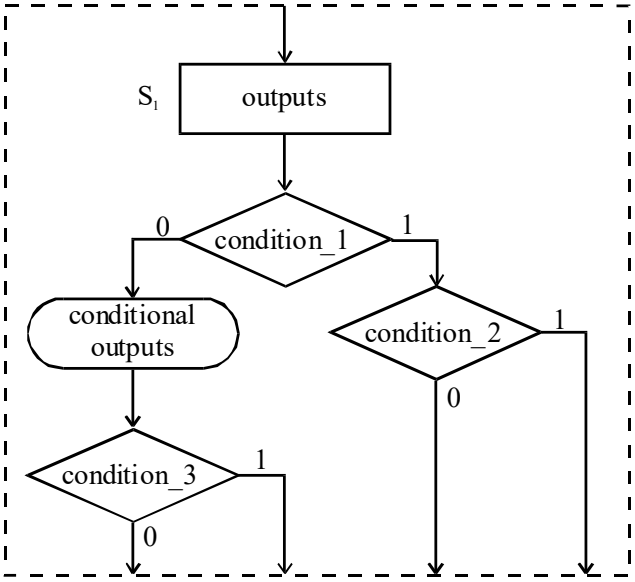

Рис. 4. Блок ASM

Блок-схема или диаграмма ASM представляет собой композицию соединенных между собой блоков ASM. При этом каждый выход любой вершины ASM может быть соединен только с одним входом другой вершины, т.е. ветвление алгоритма возможно только в условных вершинах.

IV. Методика ASMD-FSMD

Блок-схема ASM с трактом обработки данных (ASM with datapath – ASMD) представляет собой ASM, в которой в прямоугольниках и овалах можно записывать любые операции над регистрами, которые допустимы в языке Verilog, а в условных вершинах можно проверять любые логические выражения языка Verilog. Схема ASMD, также как ASM, состоит из блоков. Действия, описанные внутри блока ASMD, выполняются в течение одного такта синхронизации. Реализуемый по ASMD конечный автомат называется *конечным автоматом с трактом обработки данных* (FSM with datapath – FSMD). Методику ASMD-FSMD представим в виде следующего алгоритма.

Алгоритм. Методика ASMD-FSMD проектирования цифровых устройств.

1. Определяются состояния FSMD.
2. Для каждого состояния строится блок ASMD.
   2.1. В условных вершинах ASMD записываются логические функции, значения которых проверяются в данном состоянии.
   2.2. Для FSMD Мура в вершинах состояний (прямоугольниках) записываются операции, выполняемые с содержимым регистров в данном состоянии.
   2.3. Для FSMD Мили в вершинах выходов по условию (овалах) записываются операции, выполняемые с содержимым регистров на данном переходе.
3. Блоки ASMD соединяются между собой в соответствии с алгоритмом работы устройства. При этом каждый выход блока ASMD может быть соединен только с одним входом данного или другого блока ASMD.
4. При необходимости выполняется оптимизация ASMD.
5. Непосредственно по ASMD строится код FSMD на языке Verilog. Переменным ASMD в коде соответствуют регистры или триггеры (для однобитных переменных). Логическим функциям, проверяемым в условных вершинах ASMD, соответствуют логические выражения в операторах **if**. Действия, выполняемые в блоках ASMD, описываются в виде процедурных блоков **begin…end**. Операции, выполняемые в прямоугольниках ASMD (для автоматов Мура) описываются вначале блока **begin…end**, а операции, выполняемые в ромбах (для автоматов Мили) описываются в соответствующих местах операторов **if** (возможно с использованием операторных скобок **begin…end**).
6. Выполняется реализация FSMD с помощью соответствующего средства проектирования.
7. Конец.

Главным этапом методики ASMD-FSMD является построение ASMD на основе алгоритма функционирования устройства. Представление функционирования цифрового устройства в виде ASMD больше напоминает алгоритмическое описание проекта [15]. Однако в отличие от алгоритмического описания, в ASMD явно определены состояния FSMD,

которые могут соответствовать состояниям устройства управления. В отличие от традиционного подхода, в ASMD отсутствует строгое разделение на операционное устройство и устройство управления, а также в ASMD явно не определяется структура операционного устройства. Кроме того, представленная методика ASMD-FSMD позволяет реализовывать как автоматы типа Мили, так и автоматы типа Мура, а также совмещенную модель автоматов Мили и Мура.

Отметим также, что один и тот же алгоритм функционирования устройства может быть описан различными ASMD, что влияет на быстродействие и стоимость реализации проекта. Для увеличения быстродействия циклы алгоритма следует описывать с минимальным числом состояний, чтобы минимизировать число состояний в пути цикла. Для этого лучше подходят автоматы типа Мили.

На рис. 5 представлена ASMD, которая соответствует FSMD типа Мили для реализации рассматриваемого синхронного умножителя. Особенностью схемы ASMD на рис. 5 является то, что здесь имеется только два блока ASMD: в состоянии S_0 выполняется инициализация регистров, а состояние S_1 соответствует одному циклу умножения.

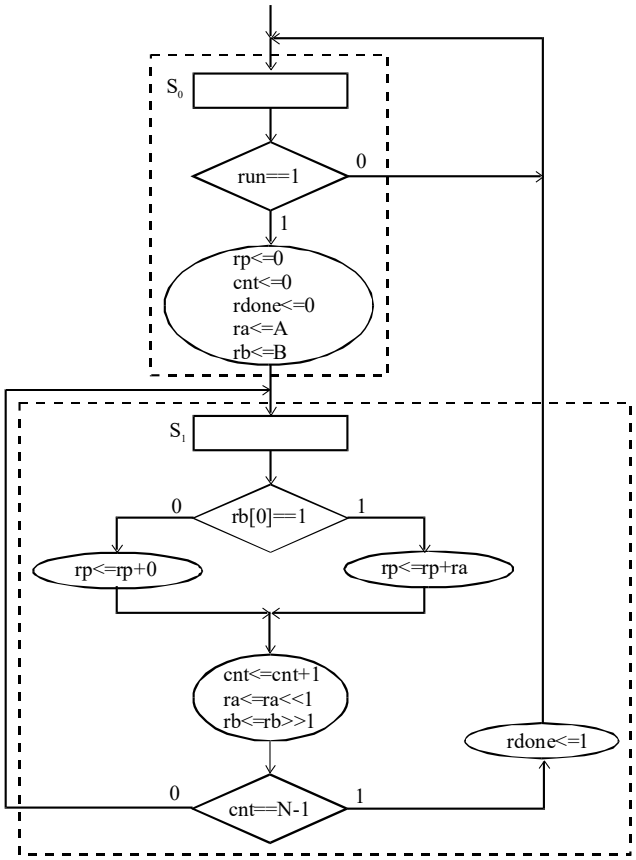

Рис. 5. Схема ASMD для автомата Мили, обеспечивающая наибольшее быстродействие синхронного умножителя а

Из рис. 5 видно, что один цикл умножения выполняется за один такт синхронизации, поэтому умножение N-разрядных двоичных чисел по ASMD на рис. 5 осуществляется за n тактов синхронизации, где $n = N + 1$.

Описание FSMD на языке Verilog по ASMD на рис. 5 имеет следующий вид:

```
module mult_FSMD_Mealy #(parameter N=4)
    (input clk, reset, run,     // управляющие сигналы
    input [N-1:0] a,b,          // входные слова
    output [2*N-1:0] p,         // произведение
    output done);               // флаг окончания умножения
    reg [2*N-1:0] ra,rp;        // объявление регистров
    reg [N-1:0] rb;
    reg rdone;
    reg [N-1:0] cnt;            // счетчик
    localparam [0:0] s0=0,s1=1;         // состояния FSMD
    reg [0:0] state;            // переменная состояний
    always @(posedge clk)  // начало цикла умножения
        if(reset)   state <= s0;
        else
            case (state)
                s0: if(run)
                    begin
                        rp <= 0; cnt <= 0; rdone <= 0;
                        ra <= {{N{1'b0}},a};
                        rb <= b;
                        state <= s1;
                    end
                    else   state <= s0;
                s1: begin
                        if(rb[0])   rp <= rp + ra;
                        else        rp <= rp + {2*N{1'b0}};
                        cnt <= cnt + 1'b1;
                        rb <= rb >> 1;
                        ra <= ra << 1;
                        if (cnt == N-1)
                        begin
                            rdone <= 1'b1;
                            state <= s0;
                        end
                        else        state <= s1;
                    end
                default: state <= s0;
            endcase
        assign p = rp;          // формирование результата
        assign done = rdone;
endmodule
```

Отметим отличия методики ASMD-FSMD от известных подходов к проектированию цифровых устройств. Методика ASMD-FSMD позволяет:

- описывать как автоматы типа Мили, так и автоматы типа Мура, а также совмещенные модели автоматов типа Мили и Мура;
- объединять в одной схеме ASMD описание операционного устройства и устройства управления;
- описывать цифровое устройство на языке Verilog с помощью одного модуля, что значительно сокращает время проектирования и способствует повышению надежности проекта;
- привлекать к разработке цифровых устройств программистов и математиков-алгоритмистов, которые не знакомы с тонкостями проектирования электронных устройств.

V. Экспериментальные исследования

Исследование эффективности методики ASMD-FSMD проводилась при реализации на FPGA семейства Cyclone IV E с помощью системы Quartus Prime версии 18.1 известных методов умножения [16]. Были исследованы следующие классические методы умножения: a, b, c и d. Кроме того, исследовался *алгоритм умножения Бута* и *модифицированный алгоритм умножения Бута* (всего 6 методов умножения).

Каждый метод умножения был реализован с помощью традиционного подхода и с помощью методики ASMD-FSMD (всего 12 проектов). Проекты исследовались с шириной входных слов 4, 8, 16, 32, 64 и 128 битов (72 примера).

Результаты экспериментальных исследований приведены в табл. 1, где mult_a_N, mult_b_N, mult_c_N и mult_d_N – проекты, реализующие алгоритмы умножения a, b, c и d соответственно; booth_N и mbooth_N – проекты, реализующие алгоритм умножения Бута и модифицированный алгоритм умножения Бута; N – ширина входных слов умножителей в битах; L_T и L_A – число используемых логических элементов FPGA (стоимость реализации) в случае традиционного подхода и при использовании методики ASMD-FSMD соответственно; t_T и t_A – время выполнения операции умножения в наносекундах в случае традиционного подхода и при использовании методики ASMD-FSMD; L_T/L_A и t_T/t_A – отношения соответствующих параметров.

Анализ табл. 1 показывает, что использование методики ASMD-FSMD для большинства примеров позволяет уменьшить стоимость реализации и увеличить быстродействие. При этом в отдельных случаях стоимость реализации уменьшается в 1.47 раза, т.е. на 47% (пример mult_a_128), а быстродействие увеличивается в 2.96 раза (пример mult_d_16).

Для исследования методики ASMD-FSMD при проектировании более сложных проектов были созданы следующие проекты процессора PIC [15]:

- PIC_1_c_N – одно-тактовый процессор PIC;
- PIC_m_N – много-тактовый процессор PIC;
- PIC_2_c_N – двух-тактовый процессор PIC;
- PIC_4_c_N – четырех-тактовый процессор PIC,

где число N обозначает ширину шины данных в битах.

Каждый проект был реализован с помощью традиционного подхода и с помощью методики ASMD-FSMD (всего 8 проектов). Проекты процессора PIC исследовались при ширине шины данных N равной 4, 8, 16, 32, 64 и 128 битов (всего 48 примеров). Результаты исследования проектов процессоров PIC приведены в табл. 2, где F_T и F_A – максимальная частота синхронизации процессора в мегагерцах при традиционном подходе и при использовании методики ASMD-FSMD; F_T/F_A – отношение соответствующих параметров; L_T, L_A и L_T/L_A имеют прежнее значение.

Таблица 1

Результаты исследования реализации алгоритмов умножения

Пример	**L_T**	**L_A**	**L_T/L_A**	**t_T**	**t_A**	**t_T/t_A**
mult_a_4	36	28	1.29	24.46	17.49	1.40
mult_a_8	66	49	1.35	46.96	31.15	1.51
mult_a_16	123	90	1.37	80.08	84.08	0.95
mult_a_32	236	172	1.37	219.03	191.79	1.14
mult_a_64	464	335	1.38	598.42	582.23	1.03
mult_a_128	919	627	**1.47**	4136.21	3969.23	1.04
mult_b_4	36	26	1.38	23.42	16.39	1.43
mult_b_8	64	49	1.30	50.47	47.81	1.06
mult_b_16	121	89	1.36	89.32	106.86	0.84
mult_b_32	234	170	1.38	218.65	178.43	1.23
mult_b_64	461	332	1.39	603.31	580.67	1.04
mult_b_128	913	653	1.40	4093.18	3916.21	1.05
mult_c_4	35	35	1.00	40.25	20.22	1.99
mult_c_8	60	65	0.92	93.11	47.82	1.95
mult_c_16	112	138	0.81	183.55	97.72	1.88
mult_c_32	208	270	0.77	452.46	238.75	1.98
mult_c_64	405	540	0.75	1267.69	639.39	1.98
mult_c_128	793	1006	0.79	8216.11	4200.59	1.96
mult_d_4	32	29	1.10	41.77	20.67	2.02
mult_d_8	58	48	1.21	82.48	36.41	2.27
mult_d_16	98	75	1.31	159.26	53.86	**2.96**
mult_d_32	176	141	1.25	410.87	158.33	2.59
mult_d_64	338	271	1.25	940.51	355.62	2.64
mult_d_128	663	528	1.26	4851.80	2178.69	2.23
booth_4	44	38	1.16	37.11	30.37	1.22
booth_8	82	69	1.19	80.73	52.54	1.54
booth_16	152	125	1.21	111.02	111.21	1.00
booth_32	301	238	1.26	263.55	228.73	1.15
booth_64	590	464	1.27	794.75	654.96	1.21
booth_128	1168	917	1.27	4474.04	4245.59	1.05
mbooth_4	65	68	0.96	25.72	18.25	1.41
mbooth_8	122	138	0.88	36.89	31.59	1.17
mbooth_16	236	278	0.85	68.81	62.48	1.10
mbooth_32	467	492	0.95	150.53	124.27	1.21
mbooth_64	925	973	0.95	413.81	404.31	1.02
mbooth_128	1842	1935	0.95	2396.53	2236.75	1.07

Анализ табл. 2 показывает, что при проектировании процессоров PIC методика ASMD-FSMD уступает традиционному подходу по стоимости реализации, однако в большинстве случаев выигрывает по быстродействию, для отдельных примеров в 1.46 раза (проект PIC_2_c_128).

Среднеарифметические значения отношений L_T/L_A и t_T/t_A для каждого метода умножения представлены в табл. 3, где mid среднее значение параметра.

Из табл. 3 следует, что методика ASMD-FSMD позволяет в среднем уменьшить стоимость реализации для большинства методов умножения (исключение составляют методы mult_c и mbooth), а также для всех методов увеличить быстродействие. Особенно заметно увеличение быстродействия для методов mult_c и mult_d.

Таблица 2

Результаты исследования реализации процессоров PIC

Пример	**L_T**	**L_A**	**L_T/L_A**	**F_T**	**F_A**	**F_A/F_T**
PIC_1_c_4	1002	1047	0.96	70.67	75.60	1.07
PIC_1_c_8	1653	1712	0.97	66.80	72.45	1.08
PIC_1_c_16	2978	3012	0.99	67.85	68.02	1.00
PIC_1_c_32	5561	5645	0.99	53.58	66.86	1.25
PIC_1_c_64	10804	10925	0.99	52.59	56.16	1.07
PIC_1_c_128	21290	21543	0.99	39.02	42.29	1.08
PIC_m_c_4	920	1359	0.68	73.17	76.09	1.04
PIC_m_c_8	1585	2160	0.73	70.64	74.43	1.05
PIC_m_c_16	2913	3502	0.83	62.79	71.41	1.14
PIC_m_c_32	5506	6157	0.89	62.41	69.31	1.11
PIC_m_c_64	10719	11304	0.95	55.22	68.29	1.24
PIC_m_c_128	21204	21849	0.97	41.97	59.41	1.42
PIC_2_c_4	1009	1344	0.75	70.53	79.47	1.13
PIC_2_c_8	1654	2143	0.77	68.25	73.03	1.07
PIC_2_c_16	2980	3497	0.85	58.67	75.34	1.28
PIC_2_c_32	5596	6104	0.92	52.72	69.59	1.32
PIC_2_c_64	10792	11300	0.96	53.26	68.61	1.29
PIC_2_c_128	21287	21834	0.97	41.20	59.96	**1.46**
PIC_4_c_4	998	1325	0.75	95.22	70.49	0.74
PIC_4_c_8	1670	2198	0.76	90.16	69.52	0.77
PIC_4_c_16	2984	3477	0.86	85.42	67.82	0.79
PIC_4_c_32	5592	6111	0.92	77.91	67.34	0.86
PIC_4_c_64	10829	11260	0.96	62.70	63.71	1.02
PIC_4_c_128	21308	21550	0.99	55.66	55.63	1.00

Таблица 3

Среднеарифметические значения отношений L_T/L_A и t_T/t_A для разных методов умножения

Метод	**mid(L_T/L_A)**	**mid(t_T/t_A)**
mult_a	1.37	1.18
mult_b	1.37	1.11
mult_c	0.84	**1.96**
mult_d	1.23	**2.45**
booth	1.23	1.20
mbooth	0.92	1.16
mid	1.16	1.51

Таблица 4

Среднеарифметические значения отношений L_T/L_A и t_T/t_A для разных методов умножения

Метод	**mid(L_T/L_A)**	**mid(F_A/F_T)**
PIC_1_cycle	0.98	1.09
PIC_1_multi	0.84	1.17
PIC_2_cycle	0.87	**1.26**
PIC_4_cycle	0.87	0.86
mid	0.89	1.10

Аналогичные средние значения отношений L_T/L_A и F_T/F_A для проектов процессоров PIC представлены в табл. 4. Анализ табл. 4 показывает, что использование методики ASMD-FSMD позволяет увеличить быстродействие процессоров PIC в среднем в 1.1 раза (т.е. на 10%), а для двух-тактовых процессоров PIC – в 1.26 раза (т.е. на 26%). В то же время методика ASMD-FSMD уступает традиционному подходу по стоимости реализации в среднем от 2 до 16%.

Чтобы оценить время проектирования при традиционном подходе и с использованием методики ASMD-FSMD, все проекты создавались одним разработчиком. Время в минутах, затраченное на разработку каждого проекта, приведено в табл. 5, где DT_T – время разработки в случае использования традиционного подхода; DT_A – время разработки в случае использования методики ASMD-FSMD; DT_T/DT_A – отношение соответствующих параметров; mid – среднее значение.

Таблица 5

Время разработки проектов

Метод	**DT_T**	**DT_A**	**DT_T/DT_A**
mult_a	188	33	5.70
mult_b	176	29	6.07
mult_c	172	28	6.14
mult_d	237	43	5.51
booth	167	27	6.19
mbooth	195	35	5.57
PIC_1_cycle	10560	1440	7.33
PIC_1_multi	6240	960	6.50
PIC_2_cycle	840	120	7.00
PIC_4_cycle	980	150	6.53
mid			6.25

Анализ табл. 5 показывает, что использование методики ASMD-FSMD позволяет значительно (в среднем в 6.25 раза) уменьшить время разработки проектов. Это связано с тем, что в случае применения методики ASMD-FSMD отпадает необходимость в разработке (написании кода и отладке) каждого компонента операционного устройства, самого операционного устройства, устройства управления, а также главного модуля проекта.

Проведенные экспериментальные исследования позволили сделать следующие выводы:

- при проектировании цифровых устройств, поведение которых хорошо представляется в виде алгоритма функционирования, методика ASMD-FSMD по стоимости реализации и быстродействию имеет преимущество по сравнению с традиционным подходом;
- при проектировании сложных функциональных блоков (например, процессоров, цифровых фильтров и др.) традиционный подход и специальные методы проектирования могут иметь преимущество над методикой ASMD-FSMD;

- во всех случаях методика ASMD-FSMD имеет преимущество над традиционным подходом по времени проектирования.

VI. Заключение

На примере проектирования синхронных умножителей и процессоров PIC рассмотрены две технологии проектирования цифровых устройств: традиционный подход и методика ASMD-FSMD.

Методика ASMD-FSMD может использоваться при проектировании цифровых устройств на любой элементной базе (не обязательно FPGA), например, на специализированных интегральных схемах ASIC. В качестве языка проектирования может использоваться любой язык описания аппаратуры (не обязательно Verilog), например, VHDL или SystemVerilog.

Методика ASMD-FSMD требует дальнейшего совершенствования путем:

- использования особенностей языков проектирования, например, применение в одном блоке ASMD нескольких операторов блокирующего назначения к одному и тому же регистру [16];
- разработки методов повышения быстродействия ASMD, например, путем уменьшения числа состояний в циклах алгоритма;
- разработки методов оптимизации ASMD с целью повышения производительности и надежности, а также уменьшения стоимости реализации и др.

Перспективным направлением также видится использование методики ASMD-FSMD для высокоуровневого проектирования сложных проектов.

Поддержка

Работа выполнена при частичной финансовой поддержке Белостокского технологического университета (Белосток, Польша, грант WZ/WI-IIT/4/2020).

Литература

[1] Соловьев В.В. Проектирование на программируемых логических интегральных схемах быстрых конечных автоматов // Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС). 2016. Вып.1. С. 24-31.

[2] Соловьев В.В. Проектирование на программируемых логических интегральных схемах быстрых компараторов большой разрядности // Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС). 2016. Вып. 3. С. 198-205.

[3] Gajski D.D. et al. High—Level Synthesis: Introduction to Chip and System Design. Boston: Kluwer, 1992. 374 p.

[4] Auletta R., Reese B., Traver C. A comparison of synchronous and asynchronous FSMD designs // Proceedings of 1993 IEEE International Conference on Computer Design ICCD'93. IEEE, 1993. С. 178-182.

[5] Schaumont P., Shukla S., Verbauwhede I. Design with race-free hardware semantics // Proceedings of the Design Automation & Test in Europe Conference. IEEE, 2006. Т. 1. С. 6 pp.

[6] Zhu J., Gajski D.D. A unified formal model of ISA and FSMD // Proceedings of the Seventh International Workshop on Hardware/Software Codesign (CODES'99) (IEEE Cat. No. 99TH8450). IEEE, 1999. С. 121-125.

[7] Kavvadias N., Masselos K. Automated synthesis of FSMD-based accelerators for hardware compilation // 2012 IEEE 23rd International Conference on Application-Specific Systems, Architectures and Processors. IEEE, 2012. С. 157-160.

[8] Babakov R., Barkalov A., Titarenko L. Research of efficiency of microprogram final-state machine with datapath of transitions // 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM). IEEE, 2017. С. 203-206.

[9] Clare C.R. Designing logic systems using state machines. New York: McGraw-Hill Book Company, 1973. 116 p.

[10] Ciletti M.D. Advanced digital design with the Verilog HDL. New Delhi: Prentice Hall of India, 2005. 984 p.

[11] Martín P. et al. A methodology for optimizing the FPGA implementation of industrial control systems // 2009 35th Annual Conference of IEEE Industrial Electronics. IEEE, 2009. С. 2811-2816.

[12] Saha A., Ghosh A., Kumar K.G. FPGA Implementation of Arcsine Function Using CORDIC Algorithm // Proceedings of International Conference on Advances in Science and Technology. 2017. С. 138-140.

[13] Burciu P. An efficient (low resources) modular hardware implementation of the AES algorithm // Journal of Electrical Engineering, Electronics, Control and Computer Science. – 2019. Т. 5. №. 3. С. 1-10.

[14] Sowmya K.B., Gomes S., Tadiparthi V.R. Design of UART module using ASMD technique //2020 5th International Conference on Communication and Electronics Systems (ICCES). IEEE, 2020. С. 176-181.

[15] Соловьев В.В. Проектирование функциональных блоков встраиваемых систем на FPGA. М.: Горячая линия – Телеком, 2020. 348 с.

[16] Соловьев В.В. Язык Verilog в проектировании встраиваемых систем на FPGA. М.: Горячая линия – Телеком, 2020. 440 с.

ASMD-FSMD Technique for Digital Device Design

V. Salauyou

Institute Bialystok University of Technology, Poland

***Abstract* — Recently, there has been an increase in the complexity of digital device designs and an increase in the requirements for the design time and the design reliability. One of the directions for solving this problem is developing new techniques for designing digital devices. The paper proposes a technique for designing digital devices based on finite state machines with datapath (FSMD), when the functioning of the device is described in the form of an algorithm state machine with datapath (ASMD) chart and is described in Verilog HDL. The ASMD-FSMD technique is compared with the traditional approach for designing synchronous multipliers and PIC processors on the field programmable gate array (FPGA). Methods for increasing the performance of digital devices using the ASMD-FSMD technique are shown. The ASMD-FSMD technique, compared to the traditional approach, in most cases allows us to reduce the area (for some examples, by 47%) and increase speed (for some examples by a factor of 2.96). In addition, using the ASMD-FSMD technique allows reducing the design time by a factor of 5-7. In conclusion, recommendations on the use of the ASMD-FSMD technique and possible directions for the further development of the ASMD-FSMD technique are presented.**

***Keywords* — digital device design technique, finite state machines with datapath (FSMD), algorithm state machine with datapath (ASMD) chart, Verilog HDL, field programmable gate array (FPGA).**

REFERENCES

[1] Salauyou V.V. Designing on FPGA of high-speed finite state machines // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2016. Part 1. P. 24-31.

[2] Salauyou V.V. Designing on FPGA and SoC high-performance binary comparators of a big dimensionality // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2016. Part 3. P. 198-205.

[3] Gajski D.D. et al. High—Level Synthesis: Introduction to Chip and System Design. Boston: Kluwer, 1992. 374 p.

[4] Auletta R., Reese B., Traver C. A comparison of synchronous and asynchronous FSMD designs // Proceedings of 1993 IEEE International Conference on Computer Design ICCD'93. IEEE, 1993. pp. 178-182.

[5] Schaumont P., Shukla S., Verbauwhede I. Design with race-free hardware semantics // Proceedings of the Design Automation & Test in Europe Conference. IEEE, 2006. V. 1. 6 pp.

[6] Zhu J., Gajski D.D. A unified formal model of ISA and FSMD // Proceedings of the Seventh International Workshop on Hardware/Software Codesign (CODES'99) (IEEE Cat. No. 99TH8450). IEEE, 1999. pp. 121-125.

[7] Kavvadias N., Masselos K. Automated synthesis of FSMD-based accelerators for hardware compilation // 2012 IEEE 23rd International Conference on Application-Specific Systems, Architectures and Processors. IEEE, 2012. pp. 157-160.

[8] Babakov R., Barkalov A., Titarenko L. Research of efficiency of microprogram final-state machine with datapath of transitions // 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM). IEEE, 2017. pp. 203-206.

[9] Clare C.R. Designing logic systems using state machines. New York: McGraw-Hill Book Company, 1973. 116 p.

[10] Ciletti M.D. Advanced digital design with the Verilog HDL. New Delhi: Prentice Hall of India, 2005. 984 p.

[11] Martín P. et al. A methodology for optimizing the FPGA implementation of industrial control systems // 2009 35th Annual Conference of IEEE Industrial Electronics. IEEE, 2009. pp. 2811-2816.

[12] Saha A., Ghosh A., Kumar K.G. FPGA Implementation of Arcsine Function Using CORDIC Algorithm // Proceedings of International Conference on Advances in Science and Technology. 2017. pp. 138-140.

[13] Burciu P. An efficient (low resources) modular hardware implementation of the AES algorithm // Journal of Electrical Engineering, Electronics, Control and Computer Science. – 2019. V. 5. №. 3. pp. 1-10.

[14] Sowmya K.B., Gomes S., Tadiparthi V.R. Design of UART module using ASMD technique //2020 5th International Conference on Communication and Electronics Systems (ICCES). IEEE, 2020. pp. 176-181.

[15] Salauyou V. Projektirovanie funkcionalnych blokov vstraivaemych sistem na FPGA (Designing functional blocks of embedded systems on FPGA). Moskow: Hot Line – Telecom , 2020. 348 p.

[16] Salauyou V. Jazyk Verilog v projektirovanii vstraivajemych sistem na FPGA (Velilog HDL in designing embedded systems on FPGA). Moskow: Hot Line – Telecom , 2020. 440 p.