

# Модель узла управления динамическим 7-сегментным индикатором с подавлением дребезга контактов кнопок в объеме ПЛИС Xilinx Artix-7 для отладочной платы Digilent Nexys 4

Николай БОРИСЕНКО  
fpga-mechanic@rambler.ru

В статье детально описан логический проект ПЛИС Xilinx XC7A100T, установленной на отладочной плате Nexys 4 фирмы Digilent. В состав проекта входят следующие функциональные узлы, описанные в виде синтезируемых моделей на языке Verilog: схема синтеза тактовой частоты, генератор сигнала начальной установки, фильтр подавления дребезга контактов кнопок, реверсивный счетчик с загрузкой и узел управления динамическим семисегментным индикатором. Рассмотренный в статье проект может быть полезен для быстрого освоения отладочной платы Nexys 4, а также изучения практических приемов описания синтезируемых моделей цифровых устройств и их реализации в ПЛИС.

Среди отладочных плат на основе ПЛИС фирмы Xilinx большое распространение получили изделия компании Digilent, поддерживаемые программным пакетом Xilinx Design Suite iMPACT. Компания Digilent давно выпускает ассорти-

мент отладочных средств для различных серий ПЛИС, в него входят как простые платы на основе ПЛИС CPLD, разъемы, переключатели и светодиоды, так и сложные изделия на ПЛИС FPGA большой логической емкости с развитой периферией. В статье опи-

сана работа с демонстрационной отладочной платой Nexys 4, основанной на ПЛИС FPGA XC7A100T современной серии Artix-7 фирмы Xilinx (рис. 1).

Отладочная плата Nexys 4 оснащена разъемами и блоками физического уровня для реализации таких интерфейсов, как Ethernet, SPI, Audio, VGA, SRAM, PS/2 (работает через микроконтроллер с USB-интерфейсом), UART и GPIO.

Средства ручного ввода дискретных логических сигналов в ПЛИС включают пять тактовых кнопок (Button Up/Center/Down/Left/Right) и 16 статических переключателей (switch sw0–sw15).

К средствам визуального отображения состояний ПЛИС платы Nexys 4 относятся: 16 дискретных зеленых светодиодов (led ld0–ld15), два дискретных трехцветных RGB-светодиода (rgb-led ld16, ld17) и два четырехразрядных семисегментных светодиодных дисплея с общим анодом (display disp-1, disp-2). Кроме перечисленных индикаторов, на плате установлен красный светодиод, отражающий наличие питания, а также зеленый светодиод состояния загрузки ПЛИС (включается при успешной конфигурации FPGA на основе сигнала DONE).

Предлагается построить проект для ПЛИС в составе отладочной платы Nexys 4, имеющий следующую функциональность. Центральным элементом является 16-разрядный реверсивный двоичный счетчик с воз-

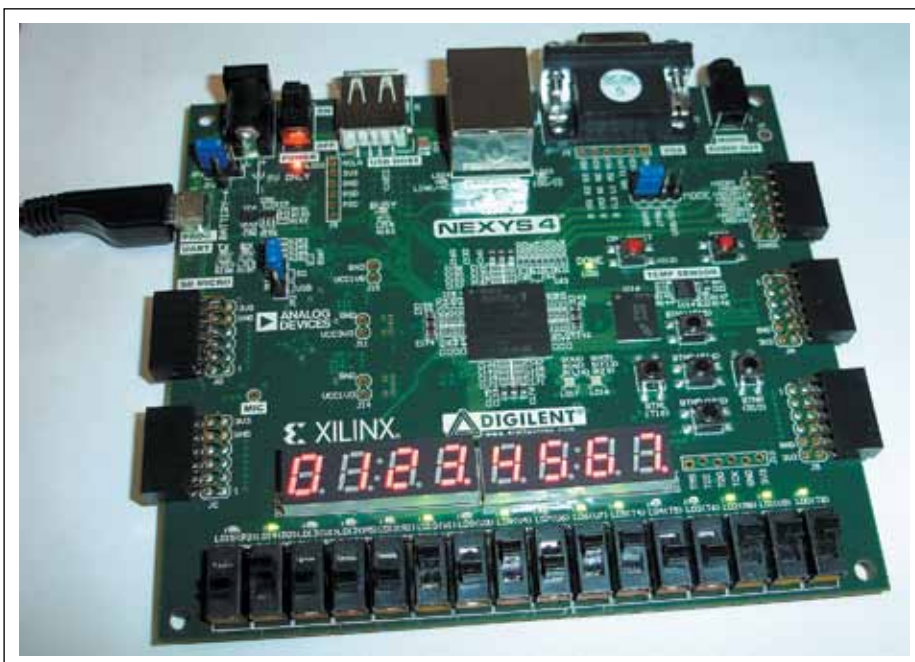


Рис. 1. Внешний вид отладочной платы Digilent Nexys 4

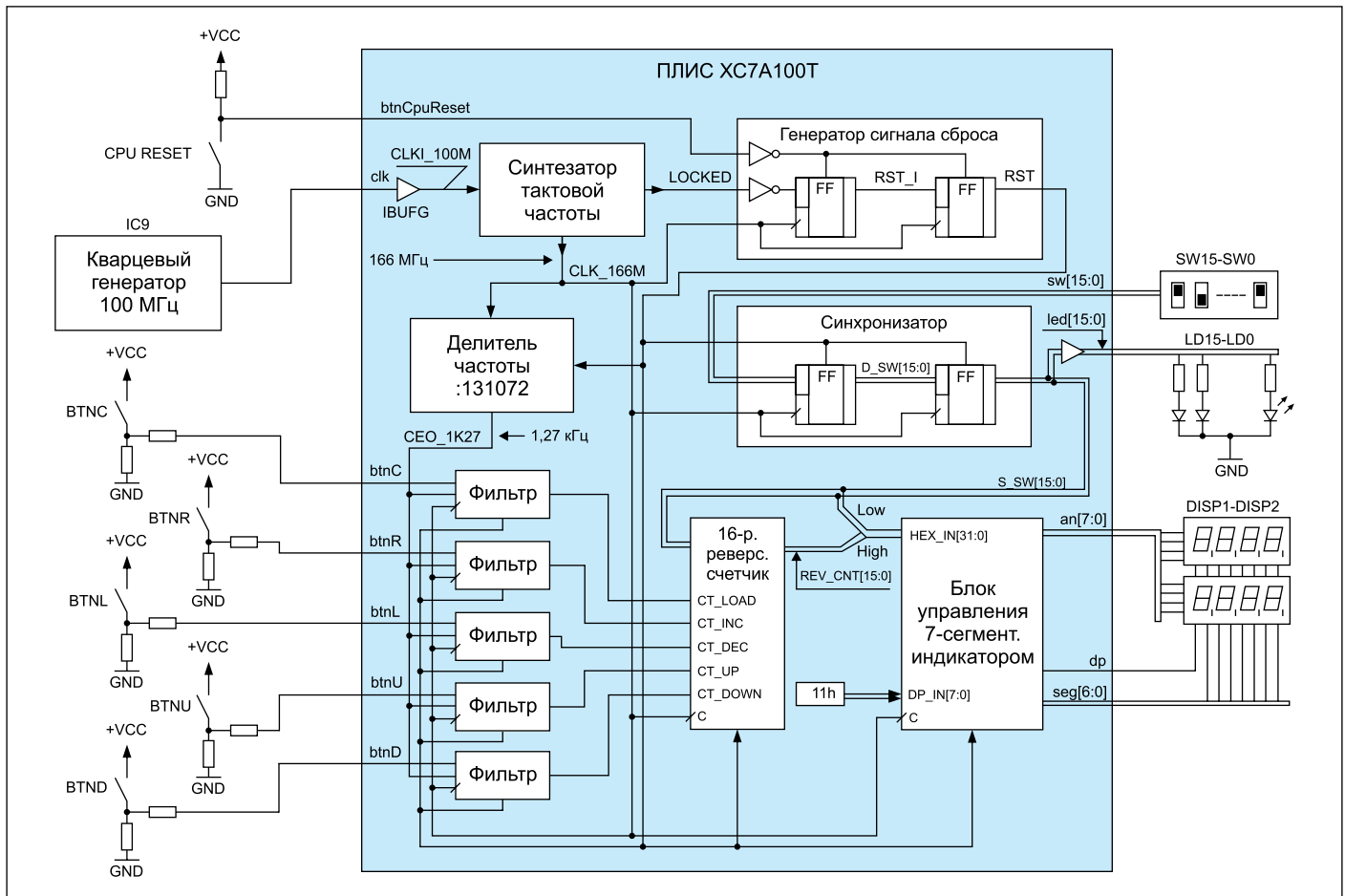


Рис. 2. Структурная схема проекта ПЛИС

возможностью загрузки, установки в максимальное состояние и синхронным сбросом в ноль. Сигналы управления счетчиком формируются путем нажатия на пять кнопок, причем центральная кнопка контролирует загрузку данных, левая кнопка — уменьшение на единицу (декремент), правая кнопка — увеличение на единицу (инкремент), верхняя кнопка управляет установкой максимального состояния, а нажатие на нижнюю кнопку приводит к синхронному сбросу счетчика.

Комбинация, загружаемая в счетчик при нажатии на центральную кнопку, задается переключателями sw0–sw15 и отображается на светодиодах led0–led15. Причем логической единице соответствует верхнее согласно рис. 1 положение переключателя и свечение соответствующего светодиода. Разряд-0 контролируется положением переключателя sw0 и отображается на светодиоде led0, разряд-1 контролируется положением переключателя sw1 и отображается на светодиоде led1. (Далее — аналогично.)

Текущее состояние счетчика и поступающей с переключателей sw0–sw15 комбинации отображается на динамических семисегментных дисплеях в шестнадцатеричной системе счисления. Причем положение переключателей отображается на левом дисплее DISP1, а состояние счетчика — на правом дисплее

DISP2. Крайние правые точки обоих дисплеев засвечены.

Структурная схема проекта ПЛИС показана на рис. 2.

Все внутренние функциональные узлы в проекте ПЛИС синхронизируются тактовым сигналом с частотой 166 МГц (6 нс). Источником опорного синхросигнала на плате Nexys 4 служит кварцевый генератор, формирующий частоту 100 МГц. Для ввода синхросигнала в кристалле ПЛИС задействуется специальный

входной буфер типа IBUFG, которому соответствует одноименный примитив в библиотеке элементов Xilinx. Прошедший через входной буфер IBUFG синхросигнал поступает на узел синтезатора тактовой частоты. Формирование внутренних синхросигналов в седьмой серии FPGA фирмы Xilinx реализуется на ресурсах кристалла CMT (Clock Management Tiles), содержащих узлы синтеза тактовых частот MMCM (mixed-mode clock manager) и узлы ВАПЧ PLL (phase-locked loop) [1].

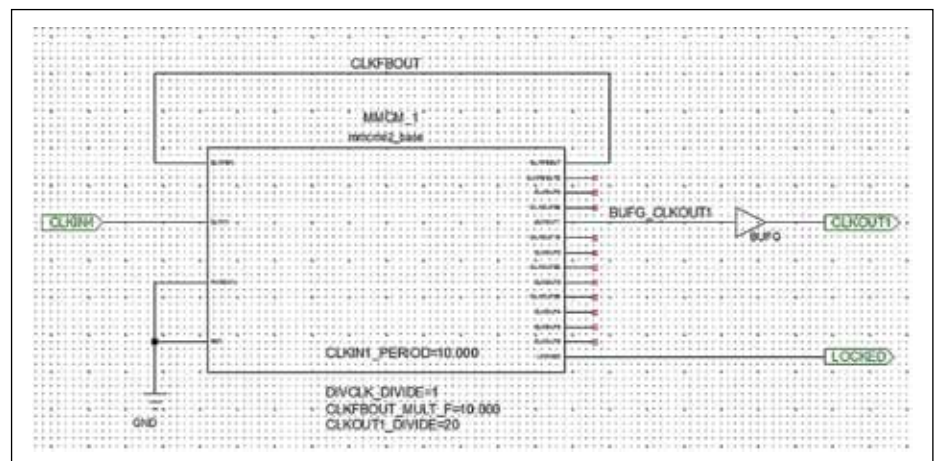


Рис. 3. Пример схематического описания синтезатора тактовой частоты

Рассмотрим в качестве примера схематехнический модуль проекта ПЛИС, созданный в САПР Xilinx ISE DS и предназначенный для синтеза внутреннего синхросигнала с частотой 50 МГц из входного опорного сигнала 100 МГц с использованием библиотечного примитива `mcm2_base` (рис. 3).

Примитив `mcm2_base` соответствует упрощенному представлению узла MMCM кристалла FPGA седьмой серии компании Xilinx. Узел синтеза частот MMCM принимает на вход `CLKIN1` опорную частоту и формирует на выходах до семи синхросигналов `CLKOUT0`–`CLKOUT6`, часть из которых имеют парафазный выход. Вход `CLKFBIN` предназначен для цепи обратной связи с выхода `CLKFBOUT`. Появление единицы на выходе `LOCKED` отражает установку стабильных выходных синхросигналов и может быть использовано при формировании сброса.

Опорный синхросигнал внутри узла MMCM поступает с входа `CLKIN1` на делитель частоты, коэффициент деления для которого задается целочисленным атрибутом `DIVCLK_DIVIDE` из диапазона от 1 до 106 (по умолчанию 1). С выхода делителя частоты сформированный промежуточный сигнал подается на умножитель частоты, коэффициент умножения которого определяется дробным атрибутом `CLKFBOUT_MULT_F` из диапазона от 2000 до 64000, дробная часть которого округляется до 1/8 (0,125). По умолчанию атрибут `CLKFBOUT_MULT_F` установлен в значение 5. При задании атрибутов следует учитывать, что полученный на выходе умножителя частоты сигнал должен иметь частоту от 600 до 1200 МГц.

Атрибут `CLKIN1_PERIOD` задает период входной опорной частоты в наносекундах.

Выходные сигналы `CLKOUT0`–`CLKOUT6` формируются путем деления выходной частоты умножителя. Для выхода `CLKOUT0` допустимо задавать дробный коэффициент деления, определяемый атрибутом `CLKOUT[0]_DIVIDE_F` с точностью до 1/8. Для остальных выходов коэффициенты деления задаются целочисленными атрибутами `CLKOUT[1:6]_DIVIDE`. Максимальный коэффициент деления для всех выходов ограничен числом 128, а минимальный — единицей. По умолчанию все коэффициенты деления установлены в единицу.

В рассмотренном на рис. 3 примере с выхода `CLKOUT1` снимается синхросигнал с частотой 50 МГц. Для синтеза такого сигнала при опорной частоте 100 МГц примитив `mcm2_base` может иметь следующие значения атрибутов: `CLKIN1_PERIOD` = 10000 (10 нс — период 100 МГц), `DIVCLK_DIVIDE` = 1 (на вход умножителя поступает сигнал 100 МГц), `CLKFBOUT_MULT_F` = 10000 (на выходе умножителя частота 1 ГГц), `CLKOUT1_DIVIDE` = 20 (на выход `CLKOUT1` выдается сигнал, полученный делением на 20 частоты 1 ГГц).

В результате синтеза схематехнического модуля проекта САПР Xilinx ISE DS формируется одноименный файл с расширением `*.vf` (Preferred Language — Verilog) или `*.vhf` (Preferred Language — VHDL). Это текстовый файл, сгенерированный автоматически в процессе синтеза схемы. Синтаксис файла соответствует выбранному в настройках проекта языку описания аппаратуры (HDL). В рассматриваемом проекте для построения модели синтезатора тактовой частоты 166 МГц был отредактирован файл `*.vf`, созданный в результате синтеза схемы, показанной на рис. 3. Ниже приведена модель синтезатора частоты, описанная на языке Verilog в файле `Kit_Nexys4_Clock.v`:

```

`timescale 1ns / 1ps

module Kit_Nexys4_Clock(CLKIN1, CLKOUT1, LOCKED);

    input CLKIN1;
    output CLKOUT1;
    output LOCKED;

    wire BUFG_CLKOUT1;
    wire CLKFBOUT;
    wire G_N_D;

    MCM2_BASE #(
        .BANDWIDTH("OPTIMIZED"),
        .CLKFBOUT_MULT_F(10.000), // 1GHz Internal
        .CLKFBOUT_PHASE(0.000),
        .CLKIN1_PERIOD(10.000), // 10ns @ 100MHz input
        // CLKOUT0_DIVIDE - CLKOUT6_DIVIDE:
        // Divide amount for each CLKOUT (1-128)
        .CLKOUT1_DIVIDE(6), // 6ns @ 166MHz output
        .CLKOUT2_DIVIDE(1),
        .CLKOUT3_DIVIDE(1),
        .CLKOUT4_DIVIDE(1),
        .CLKOUT5_DIVIDE(1),
        .CLKOUT6_DIVIDE(1),
        .CLKOUT0_DIVIDE_F(1.000), // Divide amount for CLKOUT0
        // (1.000-128.000).
        // CLKOUT0_DUTY_CYCLE - CLKOUT6_DUTY_CYCLE:
        // Duty cycle for each CLKOUT (0.01-0.99).
        .CLKOUT0_DUTY_CYCLE(0.500),
        .CLKOUT1_DUTY_CYCLE(0.500),
        .CLKOUT2_DUTY_CYCLE(0.500),
        .CLKOUT3_DUTY_CYCLE(0.500),
        .CLKOUT4_DUTY_CYCLE(0.500),
        .CLKOUT5_DUTY_CYCLE(0.500),
        .CLKOUT6_DUTY_CYCLE(0.500),
        // CLKOUT0_PHASE - CLKOUT6_PHASE:
        // Phase offset for each CLKOUT (-360.000-360.000).
        .CLKOUT0_PHASE(0.000),
        .CLKOUT1_PHASE(0.000),
        .CLKOUT2_PHASE(0.000),
        .CLKOUT3_PHASE(0.000),
        .CLKOUT4_PHASE(0.000),
        .CLKOUT5_PHASE(0.000),
        .CLKOUT6_PHASE(0.000),
        .CLKOUT4_CASCADE("FALSE"), // Cascade CLKOUT4 counter
        // with CLKOUT6 (FALSE, TRUE)
        .DIVCLK_DIVIDE(1), // Master division
        // value (1-106)
        .REF_JITTER1(0.010), // Reference input jitter
        // in UI (0.000-0.999).
        .STARTUP_WAIT("FALSE") // Delays DONE until MMCM
        // is locked (FALSE, TRUE)
    )
    MCM2_1 (CLKFBIN(CLKFBOUT),
        .CLKIN1(CLKIN1),
        .PWRDWN(G_N_D),
        .RST(G_N_D),
        .CLKFBOUT(CLKFBOUT),
        .CLKFBOUTB(),
        .CLKOUT0(),
        .CLKOUT0B(),
        .CLKOUT1(BUFG_CLKOUT1),
        .CLKOUT1B(),
        .CLKOUT2(),
        .CLKOUT2B(),
        .CLKOUT3(),
        .CLKOUT3B(),
        .CLKOUT4(),
        .CLKOUT5(),
        .CLKOUT6(),
        .LOCKED(LOCKED));
    GND_XLX1_3 (.G(G_N_D));
    BUFG_XLX1_4 (.I(BUFG_CLKOUT1), .O(CLKOUT1));
endmodule

```

Генератор внутреннего сигнала сброса RST построен на двухтактном синхронизаторе, устанавливаемом в «1» (активный уровень RST) при подаче асинхронного сброса с кнопки CPU RESET платы Nexys 4. Сигнал сброса RST будет установлен в «0» только после установки асинхронного входного сигнала в единицу (кнопка отпущена) и при единице на выходе `LOCKED` синтезатора тактовой частоты.

Генератор сигнала сброса RST описан в верхнем модуле иерархии проекта `Kit_Nexys4_Top_A.v`.

Для обеспечения работы фильтров подавления дребезга контактов кнопок на пониженной частоте построен делитель частоты синхронизации 166 МГц на фиксированный коэффициент 131072. Делитель частоты формирует на выходе синхронный по отношению к сигналу `CLK_166M` выходной сигнал разрешения синхронизации `CEO_1K27` (Clock Enable Output) с частотой 1,27 кГц. Сигнал `CEO_1K27` устанавливается в единицу в течение одного такта частоты 166 МГц, после чего 131071 такт находится в нуле. Таким образом, единичные однократные импульсы высокого уровня разрешают переключение внутренних счетчиков фильтров с частотой 1,27 кГц.

Делитель частоты построен на основе 17-разрядного двоичного счетчика, описанного переменной `CLK_DIV_FLTR[16:0]` в верхнем модуле иерархии проекта `Kit_Nexys4_Top_A.v`. Выходной сигнал `CEO_1K27` устанавливается в единицу на один такт по достижении счетчиком `CLK_DIV_FLTR` максимального значения: `&(CLK_DIV_FLTR)`.

Положения переключателей `sw15`–`sw0` задают входные сигналы ПЛИС `sw[15:0]`, поступающие на двухтактный синхронизатор с промежуточным регистром `D_SW[15:0]` и выходным регистром `S_SW[15:0]`. Выходные сигналы синхронизатора транслируются на светодиоды `ld15`–`ld0`, на вход параллельной загрузки счетчика и младшую половину входной шины блока управления семисегментным дисплеем. Синхронизатор входных асинхронных сигналов необходим для обеспечения устойчивой работы внутренних синхронных узлов проекта ПЛИС. Описание синхронизатора включено в верхний модуль иерархии проекта `Kit_Nexys4_Top_A.v`.

При нажатии на кнопки `VTNC`, `VTNR`, `VTNL`, `VTNU` и `VTND` реакция счетчика должна происходить в виде одиночного действия на каждое однократное нажатие. Используя синхронизатор, аналогичный установленному на вводе сигналов с переключателей, достичь четкого одиночного срабатывания счетчика по отдельному нажатию кнопки невозможно. Это обусловлено явлением дребезга контактов, возникающим при механической коммутации любой электрической цепи. В силу неровностей на поверхности соединяемых или размыкаемых проводников на микронном уровне возникают многократные замыкания и разьедине-

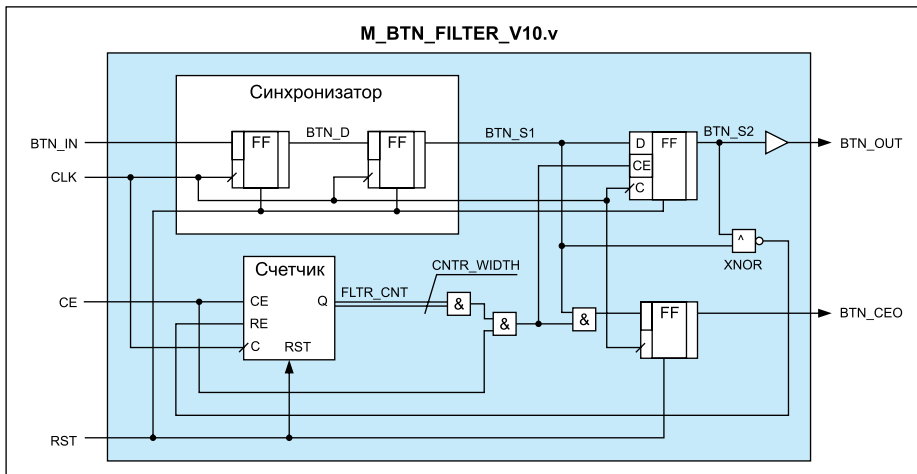


Рис. 4. Функциональная схема цифрового фильтра дребезга контактов

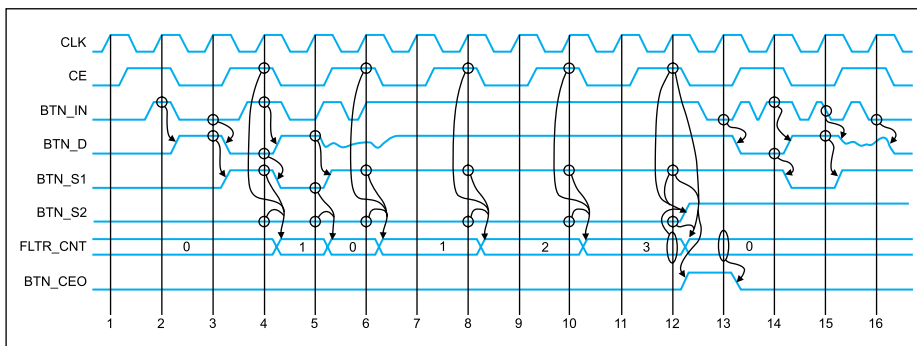


Рис. 5. Диаграмма работы цифрового фильтра дребезга контактов

цесса на входе  $BTN\_IN$  с рабочим положительным фронтом синхросигнала  $CLK$  либо при неудовлетворении требований по задержкам предустановки и удержания входного сигнала первого триггера в синхронизаторе.

Ниже приведена синтезируемая модель цифрового фильтра, подавляющего дребезг контактов, описанная на языке Verilog:

```

`timescale 1ns / 1ps

module M_BTN_FILTER_V10(
    input CLK,
    input CE,
    input BTN_IN,
    input RST,
    output BTN_OUT,
    output reg BTN_CEO
);

parameter [3:0] CNTR_WIDTH = 4; // Internal Counter Width

// Internal signals declaration:
reg [CNTR_WIDTH - 1:0] FLTR_CNT;
reg BTN_D, BTN_S1, BTN_S2;

// Main Counter:
always @ (posedge CLK, posedge RST)
    if (RST) FLTR_CNT <= {CNTR_WIDTH{1'b0}};
    else
        if (! (BTN_S1 ^ BTN_S2)) // if BTN_S1 = BTN_S2
            FLTR_CNT <= {CNTR_WIDTH{1'b0}}; // Return to Zero
        else if (CE) // else if Clock Enable
            FLTR_CNT <= FLTR_CNT + 1; // Increment

// Input Synchronizer:
always @ (posedge CLK, posedge RST)
    if (RST)
        begin
            BTN_D <= 1'b0;
            BTN_S1 <= 1'b0;
        end
    else
        begin
            BTN_D <= BTN_IN;
            BTN_S1 <= BTN_D;
        end

// Output Register:
always @ (posedge CLK, posedge RST)
    if (RST) BTN_S2 <= 1'b0;
    else if (&(FLTR_CNT) & CE) BTN_S2 <= BTN_S1;

// Output Front Detector Clock Enable:
always @ (posedge CLK, posedge RST)
    if (RST) BTN_CEO <= 1'b0;
    else BTN_CEO <= &(FLTR_CNT) & CE & BTN_S1;

assign BTN_OUT = BTN_S2;

endmodule

```

Центральным функциональным элементом рассматриваемого логического проекта ПЛИС является реверсивный счетчик с загрузкой. Это двоичный 16-разрядный счетчик, описанный поведенческим способом в верхнем модуле проекта *Kit\_Nexys4\_Top\_A.v*. Переключения счетчика управляются одноктактовыми положительными сигналами

Таблица приоритетов

RST	Входы					Действие
	CT_LOAD	CT_DEC	CT_INC	CT_UP	CT_DOWN	
1	X	X	X	X	X	Асинхронный сброс
0	1	X	X	X	X	Загрузка
0	0	1	X	X	X	Уменьшение на 1
0	0	0	1	X	X	Увеличение на 1
0	0	0	0	1	X	Установка FFFFh
0	0	0	0	0	1	Синхронный сброс
0	0	0	0	0	0	Хранение

ния в коммутируемой цепи при каждом нажатии или отпускании кнопки, что приводит к многократной смене логического уровня на входах ПЛИС. Длительности интервалов замкнутого и разомкнутого состояния в течение дребезга контактов достаточно велики, что позволяет подобным помехам проходить через высокочастотные синхронизаторы.

Для подавления дребезга контактов кнопок в рассматриваемом проекте ПЛИС построен цифровой фильтр, описанный в виде синтезируемой модели на языке Verilog в модуле *M\_BTN\_FILTER\_V10.v*. Функциональная схема, поясняющая работу цифрового фильтра, показана на рис. 4. Помимо цифровых фильтров, возможны реализации аналоговых фильтров в виде RC-цепочек, поглощающих дребезг и формирующих на выходе пологие фронты, обусловленные перезарядом больших емкостей.

Принцип работы цифрового фильтра основан на измерении длительности несовпадения текущего состояния на выходе фильтра с поступающим входным сигналом. В основе фильтра лежит двоичный счетчик, разрядность которого задается через параметр  $CNTR\_WIDTH$  и по умолчанию равна 4 (16 состояний). Этот счетчик измеряет длительность несовпадения входного синхронизированного сигнала  $BTN\_S1$  и выходного сигнала  $BTN\_S2$  ( $BTN\_OUT$ ). Всякий раз,

когда два указанных сигнала совпадают, элемент  $XNOR2$  генерирует на выходе единицу, что приводит к синхронному сбросу счетчика в ноль. Увеличение счетчика происходит при условии несовпадения входного и выходного сигналов и установке единицы на выходе разрешения синхронизации  $CE$  (Clock Enable).

В том случае, если несовпадение входного и выходного сигналов будет столь длительным, что счетчик достигнет максимального состояния, при котором на его выходе  $Q$  будут все единицы, произойдет запись принятого сигнала в выходной триггер. Факт записи единицы в выходной триггер регистрируется нижним триггером, формирующим одноктактовый импульс высокого уровня на выходе  $BTN\_CEO$ . Именно последний выходной сигнал используется для управления основным счетчиком в рассматриваемом проекте ПЛИС.

Наглядно работу цифрового фильтра дребезга контактов иллюстрирует временная диаграмма, представленная на рис. 5. Здесь представлена работа фильтра с трехразрядным счетчиком (параметр  $CNTR\_WIDTH = 3$ ) при частоте следования разрешающих сигналов  $Clock\ Enable$  ( $CE$ ) один раз в два такта.

На пятом, шестом и пятнадцатом тактах временной диаграммы показаны возможные искажения входного сигнала на синхронизаторе  $BTN\_D$ . Подобные явления могут возникнуть при совпадении во времени переходного про-

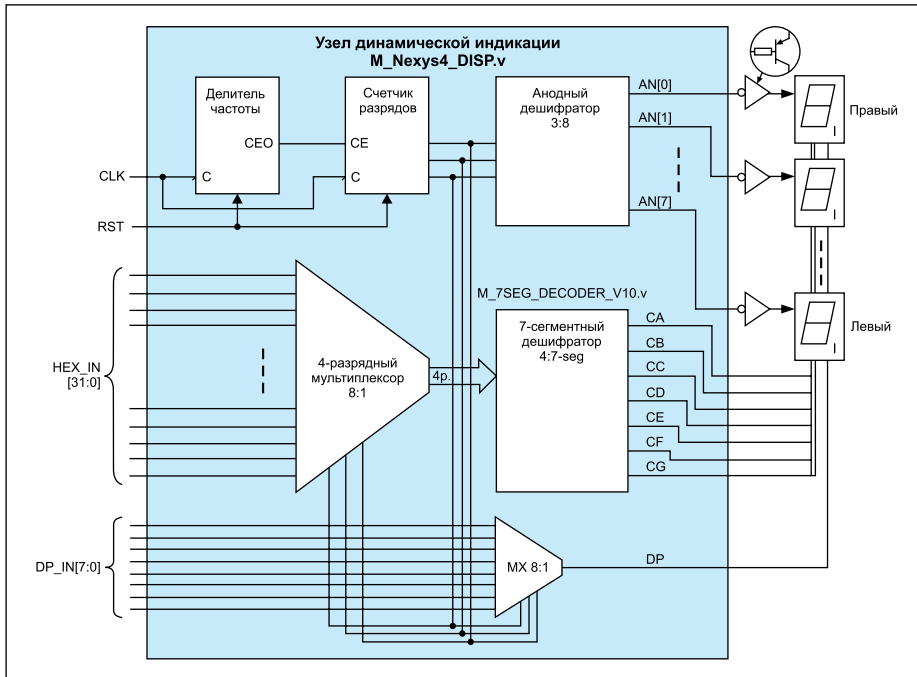


Рис. 6. Функциональная схема узла динамической индикации



Рис. 7. Отображение шестнадцатеричных цифр на семисегментном дисплее

лами, формируемыми на выходах фильтров `BTN_CEO`. В коде описания верхнего модуля проекта счетчику соответствует переменная `REV_CNT[15:0]`. Приоритетность управляющих сигналов счетчика задана специальной таблицей.

Блок управления семисегментными индикаторами `DISP1` и `DISP2` ориентирован на подключение восьми цифровых разрядов с точками, соединенных по схеме с общим анодом. Иными словами, в каждом разряде индикатора светодиоды имеют общий анод и независимые катоды. На анод подается положительное напряжение через коммутирующий транзистор с *p-n-p*-проводимостью,

а на катоды светодиодов, требующих засвечивания, — низкий уровень (0 В) через токоограничивающий резистор.

На демонстрационной плате `Nexys 4` с ПЛИС выходят сигналы `an[7:0]`, открывающие низким уровнем транзисторы в цепях анодов. Два четырехразрядных дисплея `DISP1` и `DISP2` имеют в совокупности восемь анодных цепей. Анод левого разряда управляется выходным сигналом ПЛИС `an[7]`, а анод правого — сигналом `an[0]`. Катодные цепи одноименных сегментов всех разрядов соединены параллельно: катод сегмента-а первого разряда соединен с катодами сегментов-а остальных разрядов и управляется выходным сигналом

ПЛИС “ca” (`seg[0]` в `Nexys4_Master.ucf`), катод сегмента-f первого разряда соединен с катодами сегментов-f остальных разрядов и управляется выходным сигналом ПЛИС “cf” (`seg[6]` в `Nexys4_Master.ucf`), катод точки первого разряда соединен с катодами точек остальных разрядов и управляется выходным сигналом ПЛИС “dp”.

Чтобы вывести на разряды индикатора различные символы, необходимо организовать динамическую индикацию, при которой в любой момент времени засвечиваются сегменты только одного разряда. При чередовании подсвечиваемых разрядов с частотой от сотен герц до десятков килогерц изображение на индикаторе воспринимается человеческим глазом как стабильное. Таким образом, для построения динамической индикации необходимо построить знакосинтезирующую схему-дешифратор, на которую будут подаваться через мультиплексор входные данные, отображаемые на активном разряде индикатора (рис. 6). Подробности схемотехнической реализации подключения индикатора к ПЛИС можно найти в документации [3].

Дешифратор четырехразрядного двоичного кода в семисегментный код отображает на выходе шестнадцатеричные цифры от 0 до F, как показано на рис. 7. Дешифратор является комбинационной логической схемой, описанной на поведенческом уровне на языке Verilog в файле `M_7SEG_DECODER_V10.v`.

Модуль дешифратора для семисегментного индикатора в старой элементной базе представлен микросхемами `KP514 ИД1`/`KP514 ИД2`, использовавшимися в 1970–80-е годы для вывода десятичных цифр на индикаторы с общим катодом (`ИД1`) и общим анодом (`ИД2`). Внешний вид микросхем `KP514 ИД2` в коммерческом исполнении показан на рис. 8.

В современных ПЛИС архитектуры `FPGA` такой дешифратор занимает семь табличных преобразователей `LUT`, что менее 0,1% от общего количества `LUT` в кристалле ПЛИС. Ниже приведена синтезируемая модель дешифратора семисегментного индикатора на языке Verilog:

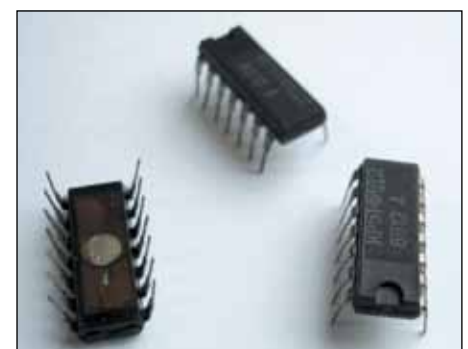


Рис. 8. Микросхемы дешифратора сегментного индикатора

```

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Engineer: FPGA-Mechanic
//
// Design Name: 7-segment Display
// Module Name: M_7SEG_DECODER_V10
// Project Name: MCH Project
// Target Devices: FPGA & CPLD
// Description: I_CODE - One Hex Digit 1-2-4-8 Coded Input
// O_SEG_x - Segment-x Active High Output
// Revision: 1.0
//////////////////////////////////////////////////////////////////
module M_7SEG_DECODER_V10(
input [3:0] I_CODE, // Input HEX-Digit
output reg O_SEG_A, // Segment-A Active High
output reg O_SEG_B, // Segment-B Active High
output reg O_SEG_C, // Segment-C Active High
output reg O_SEG_D, // Segment-D Active High
output reg O_SEG_E, // Segment-E Active High
output reg O_SEG_F, // Segment-F Active High
output reg O_SEG_G // Segment-G Active High
);

//-----
// Decoder Comb. Logic:
always @ (I_CODE)
case(I_CODE)
4'h0:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b0;
end

4'h1:
begin
O_SEG_A <= 1'b0;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b0;
O_SEG_E <= 1'b0;
O_SEG_F <= 1'b0;
O_SEG_G <= 1'b0;
end

4'h2:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b0;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b0;
O_SEG_G <= 1'b1;
end

4'h3:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b0;
O_SEG_F <= 1'b0;
O_SEG_G <= 1'b1;
end

4'h4:
begin
O_SEG_A <= 1'b0;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b0;
O_SEG_E <= 1'b0;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

4'h5:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b0;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b0;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

4'h6:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b0;

```

```

O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

4'h7:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b0;
O_SEG_E <= 1'b0;
O_SEG_F <= 1'b0;
O_SEG_G <= 1'b0;
end

4'h8:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

4'h9:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b0;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

4'hA:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b0;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

4'hB:
begin
O_SEG_A <= 1'b0;
O_SEG_B <= 1'b0;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

4'hC:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b0;
O_SEG_C <= 1'b0;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b0;
end

4'hD:
begin
O_SEG_A <= 1'b0;
O_SEG_B <= 1'b1;
O_SEG_C <= 1'b1;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b0;
O_SEG_G <= 1'b1;
end

4'hE:
begin
O_SEG_A <= 1'b1;
O_SEG_B <= 1'b0;
O_SEG_C <= 1'b0;
O_SEG_D <= 1'b1;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

default:
begin

```

```

O_SEG_A <= 1'b1;
O_SEG_B <= 1'b0;
O_SEG_C <= 1'b0;
O_SEG_D <= 1'b0;
O_SEG_E <= 1'b1;
O_SEG_F <= 1'b1;
O_SEG_G <= 1'b1;
end

endcase
//-----
endmodule

```

Все остальные узлы функциональной схемы блока управления дисплеем (рис. 6) описаны непосредственно в файле *M\_Nexys4\_DISP.v*. Делитель частоты состоит из двух ступеней: делителя с настраиваемым коэффициентом деления до 1024, описанного сигналом CLK\_DIV\_H[9:0], и фиксированного делителя на 1024, описанного сигналом CLK\_DIV\_L[9:0]. Перенос с первой ступени на вторую осуществляется сигналом CEO\_DIV\_H, а управление счетчиком — сигналом CEO\_DIV\_L. Счетчику разрядов соответствует переменная DIGIT\_CNT[2:0].

Максимальный коэффициент деления составляет 1 048 576, что при тактовой частоте 166 МГц формирует частоту сканирования разрядов индикатора 159 Гц. Если учесть, что в дисплее восемь разрядов, частота обновления всего дисплея составит приблизительно 20 Гц, что создаст видимый человеком эффект мерцания.

С другой стороны, при установке высоких частот, измеряемых в мегагерцах, выходные каскады ПЛИС и ключевые транзисторы не будут справляться с перезарядом паразитных емкостей сигнальных линий и внутренних емкостей светодиодов, вследствие чего схема окажется неработоспособна.

Оптимальные значения частоты работы динамического индикатора устанавливаются опытным путем. В рассматриваемом проекте ПЛИС был выбран коэффициент деления 102 400 (100×1024), при котором частота обновления дисплея составляет 203,5 Гц и не вызывает ощущения мерцания.

Для вывода на семисегментный восьмиразрядный дисплей для ПЛИС XC7A100T в составе платы Nexys 4 была написана на языке Verilog следующая синтезируемая модель блока управления динамическим индикатором:

```

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Engineer: FPGA-Mechanic
//
// Design Name: Prototype Project
// Module Name: M_Nexys4_DISP
// Project Name: MCH_Nexys4_Test
// Target Devices: XC7A100 FPGA / Nexys4 Digilent Board
// Tool versions: Xilinx DS 14.4
// Description: 7-Segment Display Signal Generator
//
//////////////////////////////////////////////////////////////////
module M_Nexys4_DISP(
input CLK,
input RST,
input [3:0] HEX_IN,
output CA,
output CB,
output CC,
output CD,

```

```

output CE,
output CF,
output CG,
output reg DP,
output [7:0] AN,
input [7:0] DP_IN
);

// Internal signals declaration:
//-----
reg [9:0] CLK_DIV_H, CLK_DIV_L;
reg CEO_DIV_H, CEO_DIV_L;
reg [2:0] DIGIT_CNT;
reg [3:0] I_CODE;
wire O_SEG_A, O_SEG_B, O_SEG_C, O_SEG_D, O_SEG_E, O_SEG_F, O_SEG_G;
reg [7:0] ANODE_DC;
//-----
// 1048576 Clock Divider:
always @ (posedge CLK, posedge RST)
if (RST)
begin
CLK_DIV_H <= 10'h000;
CLK_DIV_L <= 10'h000;
CEO_DIV_H <= 1'b0;
CEO_DIV_L <= 1'b0;
end
else
begin
if (CLK_DIV_H == 10'h063) // 3FF - :1024 , 063 - :100
begin
CLK_DIV_H <= 10'h000;
CEO_DIV_H <= 1'b1;
end
else
begin
CLK_DIV_H <= CLK_DIV_H + 1;
CEO_DIV_H <= 1'b0;
end
end
if (CEO_DIV_H)
begin
if (&(CLK_DIV_L))
CLK_DIV_L <= 10'h000;
else
CLK_DIV_L <= CLK_DIV_L + 1;
end
if (&(CLK_DIV_L) & CEO_DIV_H)
CEO_DIV_L <= 1'b1;
else
CEO_DIV_L <= 1'b0;
end
//-----
// Display Digit Counter:
always @ (posedge CLK, posedge RST)
if (RST) DIGIT_CNT <= 3'd0;
else if (CEO_DIV_L) DIGIT_CNT <= DIGIT_CNT + 1;
//-----
// Display Digit Multiplexer:
always @ (DIGIT_CNT, HEX_IN, DP_IN)
case (DIGIT_CNT)
3'd0:
begin
I_CODE <= HEX_IN[3:0];
DP <= ~DP_IN[0];
ANODE_DC <= 8'd1;
end
3'd1:
begin
I_CODE <= HEX_IN[7:4];
DP <= ~DP_IN[1];
ANODE_DC <= 8'd2;
end
3'd2:
begin
I_CODE <= HEX_IN[11:8];
DP <= ~DP_IN[2];
ANODE_DC <= 8'd4;
end
3'd3:
begin
I_CODE <= HEX_IN[15:12];
DP <= ~DP_IN[3];
ANODE_DC <= 8'd8;
end
3'd4:
begin
I_CODE <= HEX_IN[19:16];
DP <= ~DP_IN[4];
ANODE_DC <= 8'd16;
end
3'd5:
begin

```

```

I_CODE <= HEX_IN[23:20];
DP <= ~DP_IN[5];
ANODE_DC <= 8'd32;
end
3'd6:
begin
I_CODE <= HEX_IN[27:24];
DP <= ~DP_IN[6];
ANODE_DC <= 8'd64;
end
end
default:
begin
I_CODE <= HEX_IN[31:28];
DP <= ~DP_IN[7];
ANODE_DC <= 8'd128;
end
endcase
//-----
// 7-Segment Decoder:
M_7SEG_DECODER_V10 DISP_DEC (.I_CODE(I_CODE), .O_SEG_A(O_SEG_A), .O_SEG_B(O_SEG_B), .O_SEG_C(O_SEG_C), .O_SEG_D(O_SEG_D), .O_SEG_E(O_SEG_E), .O_SEG_F(O_SEG_F), .O_SEG_G(O_SEG_G));
//-----
assign AN = ~ANODE_DC | {8{RST}};
assign CA = ~O_SEG_A;
assign CB = ~O_SEG_B;
assign CC = ~O_SEG_C;
assign CD = ~O_SEG_D;
assign CE = ~O_SEG_E;
assign CF = ~O_SEG_F;
assign CG = ~O_SEG_G;
//-----
endmodule

```

После описания всех основных функциональных узлов проекта ПЛИС рассмотрим организацию на верхнем уровне иерархии. Иерархическая структура проекта показана на рис. 9. Верхний модуль *Kit\_Nexys4\_Top\_A* описан на языке Verilog в файле *Kit\_Nexys4\_Top\_A.v*. В нем реализованы некоторые из рассмотренных выше функциональных узлов, а также внутренние соединения и сигналы, связанные с выводами ПЛИС.

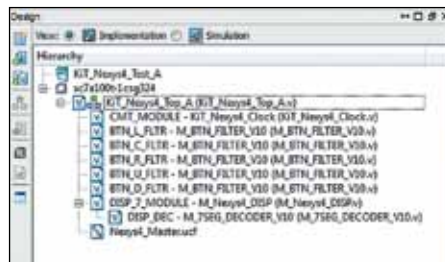


Рис. 9. Иерархия проекта ПЛИС

Код модуля верхнего уровня:

```

`timescale 1ns / 1ps
module Kit_Nexys4_Top_A(
input clk, // 100 MHz Clock Generator
input btnCpuReset, // Manual Reset Button

input [15:0] sw, // Switches - Couther Load Value
output [15:0] led, // Leds - Switches State

input btnL, // Button-L "1"
input btnC, // Button-C "Load"
input btnR, // Button-R "+1"
input btnU, // Button-U "Set Max"
input btnD, // Button-D "Set Min"

output [6:0] seg, // DISP Segments
output dp, // DISP Dots
output [7:0] an[], // DISP Anodes

//output RGB1_Red,
//output RGB1_Green
);

```

```

// Internal signals declaration:
wire CLK_100M;
wire CLK_166M, LOCKED;
reg RST_I, RST;
reg [15:0] D_SW, S_SW;
reg [16:0] CLK_DIV_FLTR;
reg CEO_1K27;
wire CT_INC, CT_DEC, CT_LOAD, CT_UP, CT_DOWN;
reg [15:0] REV_CNT;
//-----
// Xilinx IBUFG Primitive:
(* IOSTANDARD = "DEFAULT" *) (* IBUF_DELAY_VALUE = "0" *)
IBUFG XLXI_IBUFG (.I(clk), .O(CLK_100M));
//-----
// Xilinx MMCM Module:
Kit_Nexys4_Clock CMT_MODULE (.CLKIN1(CLK_100M), .CLKOUT1(CLK_166M), .LOCKED(LOCKED));
//-----
// Asynch. Reset Input Trigger:
always @ (posedge CLK_166M, negedge btnCpuReset)
if (btnCpuReset)
begin
RST_I <= 1'b1;
RST <= 1'b1;
end
else
begin
RST_I <= ~LOCKED;
RST <= RST_I;
end
//-----
// Switches Input Synchronizer:
always @ (posedge CLK_166M, posedge RST)
if (RST)
begin
D_SW <= 16'h0000;
S_SW <= 16'h0000;
end
else
begin
D_SW <= sw;
S_SW <= D_SW;
end
//-----
// Filter Clock Enable 1.27kHz:
always @ (posedge CLK_166M, posedge RST)
if (RST)
begin
CLK_DIV_FLTR <= 17'h00000;
CEO_1K27 <= 1'b0;
end
else
begin
if (&(CLK_DIV_FLTR)) // CLK_DIV_FLTR = Max: 17'h1FF
begin
CLK_DIV_FLTR <= 17'h00000;
CEO_1K27 <= 1'b1;
end
else
begin
CLK_DIV_FLTR <= CLK_DIV_FLTR + 1;
CEO_1K27 <= 1'b0;
end
end
//-----
// Filter for Button-L:
M_BTN_FILTER_V10 #(CNTR_WIDTH(3))
BTN_L_FLTR (.CLK(CLK_166M), .CE(CEO_1K27), .BTN_IN(btnL), .RST(RST), .BTN_OUT(), .BTN_CEO(CT_DEC));
// Filter for Button-C:
M_BTN_FILTER_V10 #(CNTR_WIDTH(3))
BTN_C_FLTR (.CLK(CLK_166M), .CE(CEO_1K27), .BTN_IN(btnC), .RST(RST), .BTN_OUT(), .BTN_CEO(CT_LOAD));
// Filter for Button-R:
M_BTN_FILTER_V10 #(CNTR_WIDTH(3))
BTN_R_FLTR (.CLK(CLK_166M), .CE(CEO_1K27), .BTN_IN(btnR), .RST(RST), .BTN_OUT(), .BTN_CEO(CT_INC));
// Filter for Button-U:
M_BTN_FILTER_V10 #(CNTR_WIDTH(3))
BTN_U_FLTR (.CLK(CLK_166M), .CE(CEO_1K27), .BTN_IN(btnU), .RST(RST), .BTN_OUT(), .BTN_CEO(CT_UP));
// Filter for Button-D:
M_BTN_FILTER_V10 #(CNTR_WIDTH(3))
BTN_D_FLTR (.CLK(CLK_166M), .CE(CEO_1K27), .BTN_IN(btnD), .RST(RST), .BTN_OUT(), .BTN_CEO(CT_DOWN));
//-----
// Reversible Counter:
always @ (posedge CLK_166M, posedge RST)
if (RST)
REV_CNT <= 16'h0000;
else if (CT_LOAD)
REV_CNT <= S_SW;
else if (CT_DEC)
REV_CNT <= REV_CNT - 1;
else if (CT_INC)
REV_CNT <= REV_CNT + 1;

```

```

else if(CT_UP)
  REV_CNT <= 16'hFFFF;
else if(CT_DOWN)
  REV_CNT <= 16'h0000;
//-----
// Nexys4 7-Segment Display Driver:
M_Nexys4_DISP_DISP_7_MODULE (CLK(CLK_166M), RST(RST),
.HEX_IN({REV_CNT, S_SW}), .DP_IN(8'h11), .CA(seg[0]),
.CB(seg[1]), .CC(seg[2]), .CD(seg[3]), .CE(seg[4]), .CF(seg[5]),
.CG(seg[6]), .DP(dp), .AN(an));
//-----
assign led[15:0] = S_SW[15:0];
//-----
endmodule

```

Названия сигналов интерфейса верхнего модуля иерархии заимствованы из готового файла пользовательских ограничений *Nexys4\_Master.ucf*, который можно загрузить с официального сайта Digilent [2]. В исходном виде этот файл представляет собой шаблон распределения сигналов по выводам ПЛИС, в котором все строки закрыты как комментарии. При удалении символа # для соответствующих строк пользователь платы Nexys 4 может задействовать нужные в проекте линии ввода/вывода.

В рассматриваемом проекте файл пользовательских ограничений описывает подключение следующих сигналов:

```

## This file is a general .ucf for the Nexys4 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used signals according to the project

# Clock signal
NET "clk" LOC = "E3" | IOSTANDARD = "LVCMOS33"; #Bank =
35, Pin name = IO_L12P_T1_MRCC_35, Sch name = CLK100MHZ
NET "clk" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH
50%;

# Switches
NET "sw<0>" LOC = "U9" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L21P_T3_DQS_34, Sch name = SW0
NET "sw<1>" LOC = "U8" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_25_34, Sch name = SW1
NET "sw<2>" LOC = "R7" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L23P_T3_34, Sch name = SW2
NET "sw<3>" LOC = "R6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L19P_T3_34, Sch name = SW3
NET "sw<4>" LOC = "R5" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L19N_T3_VREF_34, Sch name = SW4
NET "sw<5>" LOC = "V7" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L20P_T3_34, Sch name = SW5
NET "sw<6>" LOC = "V6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L20N_T3_34, Sch name = SW6
NET "sw<7>" LOC = "V5" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L10P_T1_34, Sch name = SW7
NET "sw<8>" LOC = "U4" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L8P_T1_34, Sch name = SW8
NET "sw<9>" LOC = "V2" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L9N_T1_DQS_34, Sch name = SW9
NET "sw<10>" LOC = "U2" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L9P_T1_DQS_34, Sch name = SW10

```

```

NET "sw<11>" LOC = "T3" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L11N_T1_MRCC_34, Sch name = SW11
NET "sw<12>" LOC = "T1" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L17N_T2_34, Sch name = SW12
NET "sw<13>" LOC = "R3" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L11P_T1_SRCC_34, Sch name = SW13
NET "sw<14>" LOC = "P3" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L14N_T2_SRCC_34, Sch name = SW14
NET "sw<15>" LOC = "P4" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L14P_T2_SRCC_34, Sch name = SW15

```

```

# LEDs
NET "led<0>" LOC = "T8" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L24N_T3_34, Sch name = LED0
NET "led<1>" LOC = "V9" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L21N_T3_DQS_34, Sch name = LED1
NET "led<2>" LOC = "R8" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L24P_T3_34, Sch name = LED2
NET "led<3>" LOC = "T6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L23N_T3_34, Sch name = LED3
NET "led<4>" LOC = "T5" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L12P_T1_MRCC_34, Sch name = LED4
NET "led<5>" LOC = "T4" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L12N_T1_MRCC_34, Sch name = LED5
NET "led<6>" LOC = "U7" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L22P_T3_34, Sch name = LED6
NET "led<7>" LOC = "U6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L22N_T3_34, Sch name = LED7
NET "led<8>" LOC = "V4" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L10N_T1_34, Sch name = LED8
NET "led<9>" LOC = "U3" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L8N_T1_34, Sch name = LED9
NET "led<10>" LOC = "V1" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L7N_T1_34, Sch name = LED10
NET "led<11>" LOC = "R1" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L17P_T2_34, Sch name = LED11
NET "led<12>" LOC = "P5" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L13N_T2_MRCC_34, Sch name = LED12
NET "led<13>" LOC = "U1" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L7P_T1_34, Sch name = LED13
NET "led<14>" LOC = "R2" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L15N_T2_DQS_34, Sch name = LED14
NET "led<15>" LOC = "P2" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L15P_T2_DQS_34, Sch name = LED15

```

```

#NET "RGB1_Red" LOC = "K5" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L5P_T0_34, Sch name = LED16_R
#NET "RGB1_Green" LOC = "F13" | IOSTANDARD = "LVCMOS33";
#Bank = 15, Pin name = IO_L5P_T0_AD9P_15, Sch name = LED16_G
#NET "RGB1_Blue" LOC = "P6" | IOSTANDARD = "LVCMOS33";
#Bank = 35, Pin name = IO_L19N_T3_VREF_35, Sch name =
LED16_B
#NET "RGB2_Red" LOC = "K6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_0_34, Sch name = LED17_R
#NET "RGB2_Green" LOC = "H6" | IOSTANDARD = "LVCMOS33";
#Bank = 35, Pin name = IO_24P_T0_AD9P_15, Sch name = LED17_G
#NET "RGB2_Blue" LOC = "L16" | IOSTANDARD = "LVCMOS33";
#Bank = CONFIG, Pin name = IO_L3N_T0_DQS_EMCLK_14, Sch
name = LED17_B

```

```

# 7 segment display
NET "seg<0>" LOC = "L3" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L2N_T0_34, Sch name = CA
NET "seg<1>" LOC = "N1" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L3N_T0_DQS_34, Sch name = CB
NET "seg<2>" LOC = "L5" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L6N_T0_VREF_34, Sch name = CC
NET "seg<3>" LOC = "L4" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L5N_T0_34, Sch name = CD
NET "seg<4>" LOC = "K3" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L2P_T0_34, Sch name = CE
NET "seg<5>" LOC = "M2" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L4N_T0_34, Sch name = CF
NET "seg<6>" LOC = "L6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L6P_T0_34, Sch name = CG

```

```

NET "dp" LOC = "M4" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L16P_T2_34, Sch name = DP

```

```

NET "an<0>" LOC = "N6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L18N_T2_34, Sch name = AN0
NET "an<1>" LOC = "M6" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L18P_T2_34, Sch name = AN1
NET "an<2>" LOC = "M3" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L4P_T0_34, Sch name = AN2
NET "an<3>" LOC = "N5" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L13_T2_MRCC_34, Sch name = AN3
NET "an<4>" LOC = "N2" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L3P_T0_DQS_34, Sch name = AN4
NET "an<5>" LOC = "M4" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L16N_T2_34, Sch name = AN5
NET "an<6>" LOC = "L1" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L1P_T0_34, Sch name = AN6
NET "an<7>" LOC = "M1" | IOSTANDARD = "LVCMOS33";
#Bank = 34, Pin name = IO_L1N_T034, Sch name = AN7

```

```

# Buttons
NET "btnCpuReset" LOC = "C12" | IOSTANDARD = "LVCMOS33";
#Bank = 15, Pin name = IO_L3P_T0_DQS_ADIP_15, Sch name =
CPU_RESET
NET "btnC" LOC = "E16" | IOSTANDARD = "LVCMOS33";
#Bank = 15, Pin name = IO_L11N_T1_SRCC_15, Sch name = BTNC

```

```

NET "btnU" LOC = "F15" | IOSTANDARD = "LVCMOS33";
#Bank = 15, Pin name = IO_L14P_T2_SRCC_15, Sch name = BTNU
NET "btnL" LOC = "T16" | IOSTANDARD = "LVCMOS33";
#Bank = CONFIG, Pin name = IO_L15N_T2_DQS_DOUT_
CSO_B_14, Sch name = BTNL
NET "btnR" LOC = "R10" | IOSTANDARD = "LVCMOS33";
#Bank = 14, Pin name = IO_25_14, Sch name = BTNR
NET "btnD" LOC = "V10" | IOSTANDARD = "LVCMOS33";
#Bank = 14, Pin name = IO_L21P_T3_DQS_14, Sch name = BTND

```

После добавления в проект САПР Xilinx ISE Design Suite рассмотренных в статье моделей и выполнения операций синтеза и имплементации проекта в кристалл генерируется файл *kit\_nexys4\_top.a.bit*, позволяющий сконфигурировать ПЛИС через отладочный разъем micro-USB, распаянный на плате Nexys 4. Программирование ПЛИС фирмы Xilinx средствами пакета iMPACT, входящего в дистрибутивы САПР и Xilinx LabTools, рассматривалось в статьях [4, 5].

После загрузки в ПЛИС конфигурационного файла *kit\_nexys4\_top.a.bit* можно, нажимая на кнопки, управлять внутренним счетчиком и наблюдать за изменениями на семи-сегментном дисплее.

Описанный в статье проект занял 147 LUT преобразователей и 129 триггеров, что составляет 0,25% от общего объема ресурсов кристалла XC7A100T.

## Выводы

Отладочная плата Nexys 4 компании Digilent является удобным инструментом для быстрого освоения работы с современными семействами ПЛИС фирмы Xilinx. Пользователям платы доступны широкие возможности ввода и вывода логических сигналов, включающие ручной ввод с кнопок и переключателей и визуальный вывод на светодиодные индикаторы. Доступна подробная документация по этой плате и шаблоны проектов для САПР ISE и Vivado.

Предложенный в статье проект позволяет быстро изучить и воспроизвести базовые технические решения по вводу в ПЛИС асинхронных сигналов, синтезу внутренних тактовых частот, фильтрации дребезга контактов кнопок и выводу данных на динамический светодиодный дисплей.

## Литература

- 7 Series FPGAs Clocking Resources. UG472 (v1.8) Xilinx, Inc. Aug. 7, 2013.
- <http://digilentinc.com/Products/Detail.cfm?NavPath=2,400,1184&Prod=NEXYS4>
- Nexys 4 FPGA Board Reference Manual. Digilent, Inc. Nov. 19, 2013.
- Борисенко Н. В. Программирование ПЛИС фирмы Xilinx в составе смешанной JTAG-цепочки средствами САПР Xilinx ISE Design Suite 14.4 при помощи кабеля Parallel Download Cable III // Компоненты и технологии. 2013. № 7.
- Ермошин Н. Самостоятельная реализация недорогого программатора для ПЛИС Xilinx // Компоненты и технологии. 2013. № 4.