Цифровые устройства и микропроцессоры I часть

Лекция 16

Лектор: Богаченков Алексей Николаевич

e-mail: microproc@mail.ru

Темы лекции:

Моделирование на языке VHDL

Конечный автомат

Генератор IP ядер

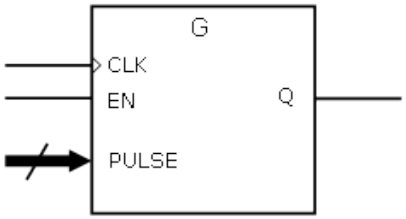
Пример синтеза

Из предыдущей лекции

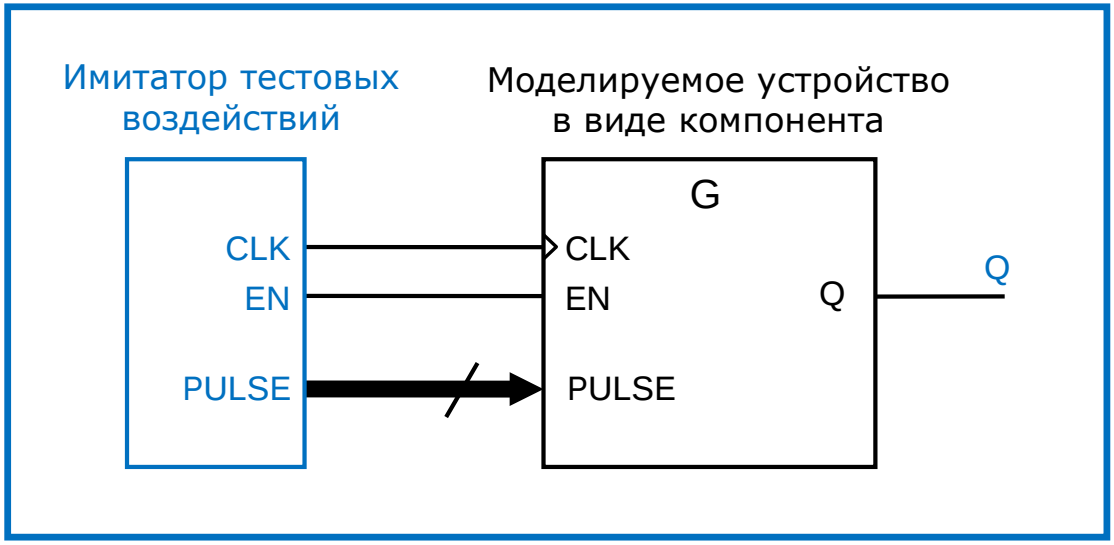
Генератор ШИМ сигнала

Функциональное моделирование (симуляция)

Структура тестового модуля



Test Bench

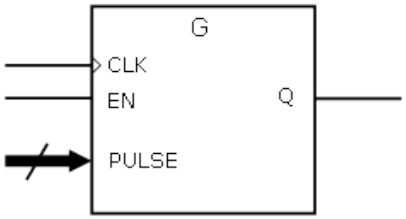


Из предыдущей лекции

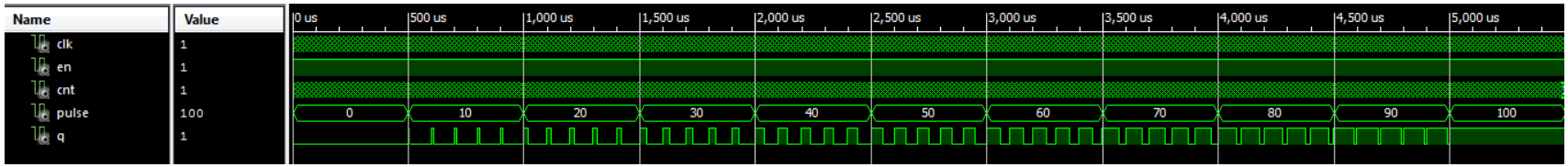
Генератор ШИМ сигнала

Функциональное моделирование (симуляция)

Фрагменты описания тестового модуля




```
   -- Процесс симуляции тактового сигнала
   clk_process: process
   begin
      clk <= '1';                -- Лог. 1
      wait for clk_period/2;     -- Задержка на полпериода
      clk <= '0';                -- Лог. 0
      wait for clk_period/2;     -- Задержка на полпериода
   end process;


   -- Процесс симуляции других входных воздействий
   stim_proc: process
   begin

      pulse <= 0;                -- Начальное значение коэффициента

      for i in 1 to 10 loop      -- Цикл по изменению коэффициента
         wait for clk_period*500;
         pulse <= pulse + 10;
      end loop;

      wait;                      -- Остановка
   end process;

END;
```



Некоторые операторы моделирования

```
wait on  CLK, RST;
```

Продолжение выполнения процесса начнется по событию любого изменения сигналов CLK или RST.

```
wait until  CLK='1';
```

Продолжение начнется в момент изменения состояния CLK из '0' в '1', т.е. по фронту сигнала.

```
wait for  CLK_PERIOD;
```

Остановка процесса на время, заданное переменной (константой) CLK_PERIOD типа **time**.

```
after  DELAY;
```

Задержка присвоения нового значения сигналу на время, заданное переменной (константой) DELAY типа **time**, например:

```
y <= x1 and x2  after 3 ns; -- Моделирование задержки в вентиле И
```

```
inertial     transport
```

Вид задержки (инерционная, транспортная).

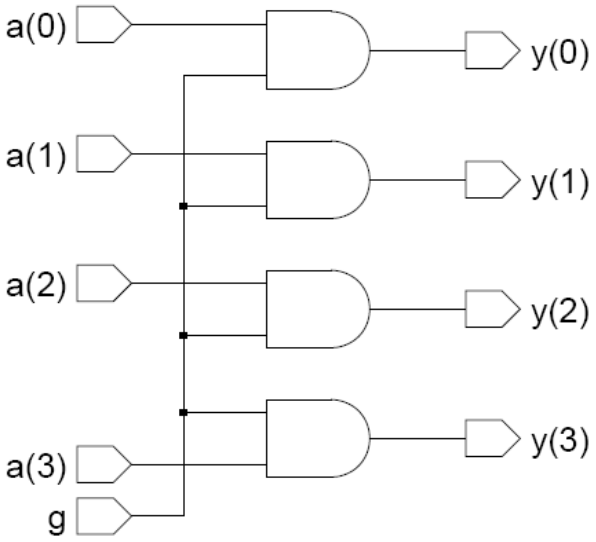
Все указанные операторы используются только в процессе моделирования и **игнорируются** при синтезе схемы.

Операторы циклов

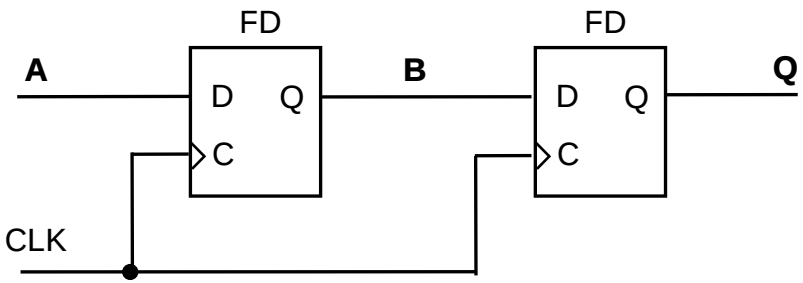
```
loop...
while...loop...
for...in...loop...
```

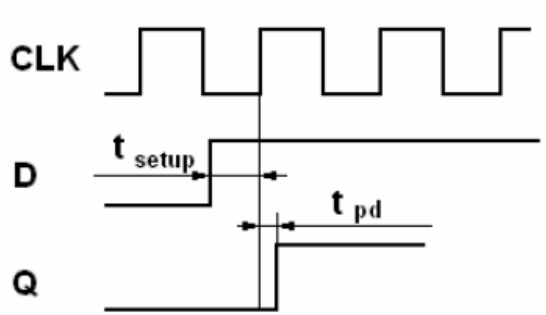
При моделировании возможна организация последовательных повторяющихся действий.
При синтезе — это обычно **размещение** множественных экземпляров фрагментов схемы, например:

```
for i in 0 to 3 generate
   y(i)<= a(i) and g;
end generate;
```



Порядок присваивания сигналов внутри процесса






```
process (clk)
begin
   if rising_edge(clk) then
      B <= '0';       -- ??
      B <= A;         -- На самом деле B еще не получило нового значения A
      Q <= B;         -- Для Q используется старое значение B (не 0 и не A)
   end if;
end process;
```

Если внутри процесса сигнал "изменяется" неоднократно, он получает только последнее значение и только по **завершении** процесса.

Конечный автомат

Используется для описания системы с несколькими состояниями и последовательными переходами между этими состояниями по определенным правилам.

Рассмотрим простейший пример — светофор.

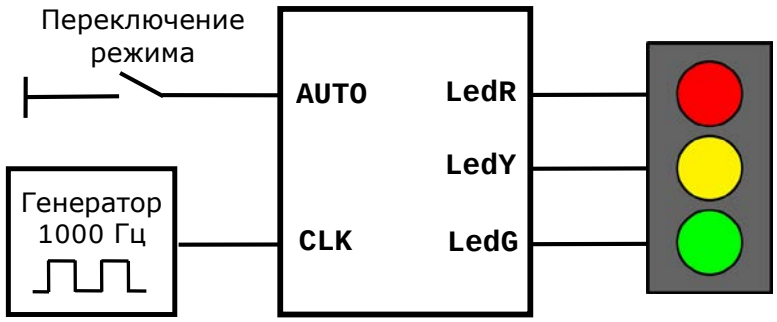
Задание:

Последовательность работы в режиме "автоматический":

Красный в течение 15 с
Зеленый в течение 20 с
Желтый в течение 5 с
... далее повторение

В режиме "нерегулируемый":

мигающий желтый с периодом 1 с



Конечный автомат (пример: светофор)

Исходный текст на VHDL (стр. 1 из 3)

```
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Lights is
   port
   (  clk   : in  std_logic;              -- Вход тактирования 1000 Гц
      auto  : in  std_logic;              -- Режим: автомат / нерегулир.
      LedR, LedG, LedY : out std_logic    -- Выходы на светоизлучатели
   );
end entity;

architecture Behavioral of Lights is

-- Константы для временных интервалов (значения в мс)
constant clock_period : integer := 1;
constant red_delay    : integer := 15000;
constant green_delay  : integer := 20000;
constant yellow_delay : integer := 5000;
constant blink_delay  : integer := 500;

-- Счетчик для временных задержек
signal cnt : integer range 0 to green_delay/clock_period := 0;

-- Сигнал управления миганием
signal blink : std_logic := '0';

-- Объявление перечислимого типа данных для номера состояния
type state_type is (stBegin, stRed, stGreen, stYellow, stBlink);
signal state : state_type := stBegin;
```

Конечный автомат (пример: светофор)

Исходный текст на VHDL (стр. 2 из 3)

```
-- Описание функционирования
begin

   process (clk)
   begin
      if rising_edge(clk) then          -- Все действия - по фронту
                                        -- тактового сигнала
         -- Обслуживание счетчика задержки
         if cnt > 0 then
            cnt <= cnt - 1;
         end if;

         -- Работа конечного автомата, переход в состояние stBlink
         -- только из состояний stBegin и stYellow
         case state is
            when stBegin =>             -- Исходное состояние
               if auto = '0' then
                  state <= stBlink;
               else
                  cnt <= red_delay/clock_period;
                  state <= stRed;
               end if;

            when stRed =>               -- Стадия "красный"
               if cnt = 0 then
                  cnt <= green_delay/clock_period;
                  state <= stGreen;
               end if;

            when stGreen =>             -- Стадия "зеленый"
               if cnt = 0 then
                  cnt <= yellow_delay/clock_period;
                  state <= stYellow;
               end if;
```

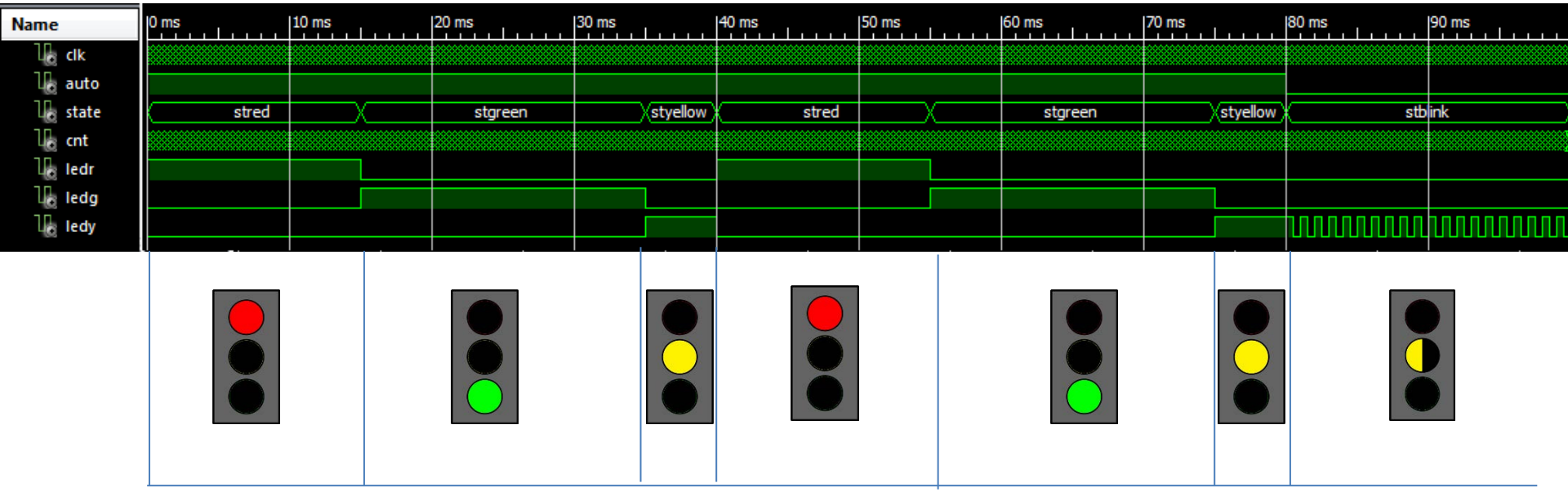
Конечный автомат (пример: светофор)

Исходный текст на VHDL (стр. 3 из 3)

```
            when stYellow =>           -- Стадия "желтый"
               if auto = '0' then
                  state <= stBlink;
               elsif cnt = 0 then
                  cnt <= red_delay/clock_period;
                  state <= stRed;
               end if;

            when stBlink =>            -- Стадия "нерегулируемый"
               if auto = '1' then
                 state <= stBegin;
               elsif cnt = 0 then
                 cnt <= blink_delay/clock_period;
                 blink <= not blink;
               end if;

            when others =>             -- Резервное состояние

         end case;
      end if;
   end process;

   -- Управление светоизлучателями
   LedR <= '1'   when state = stRed    else '0';
   LedG <= '1'   when state = stGreen  else '0';
   LedY <= '1'   when state = stYellow else
           blink when state = stBlink  else '0';

end Behavioral;
```

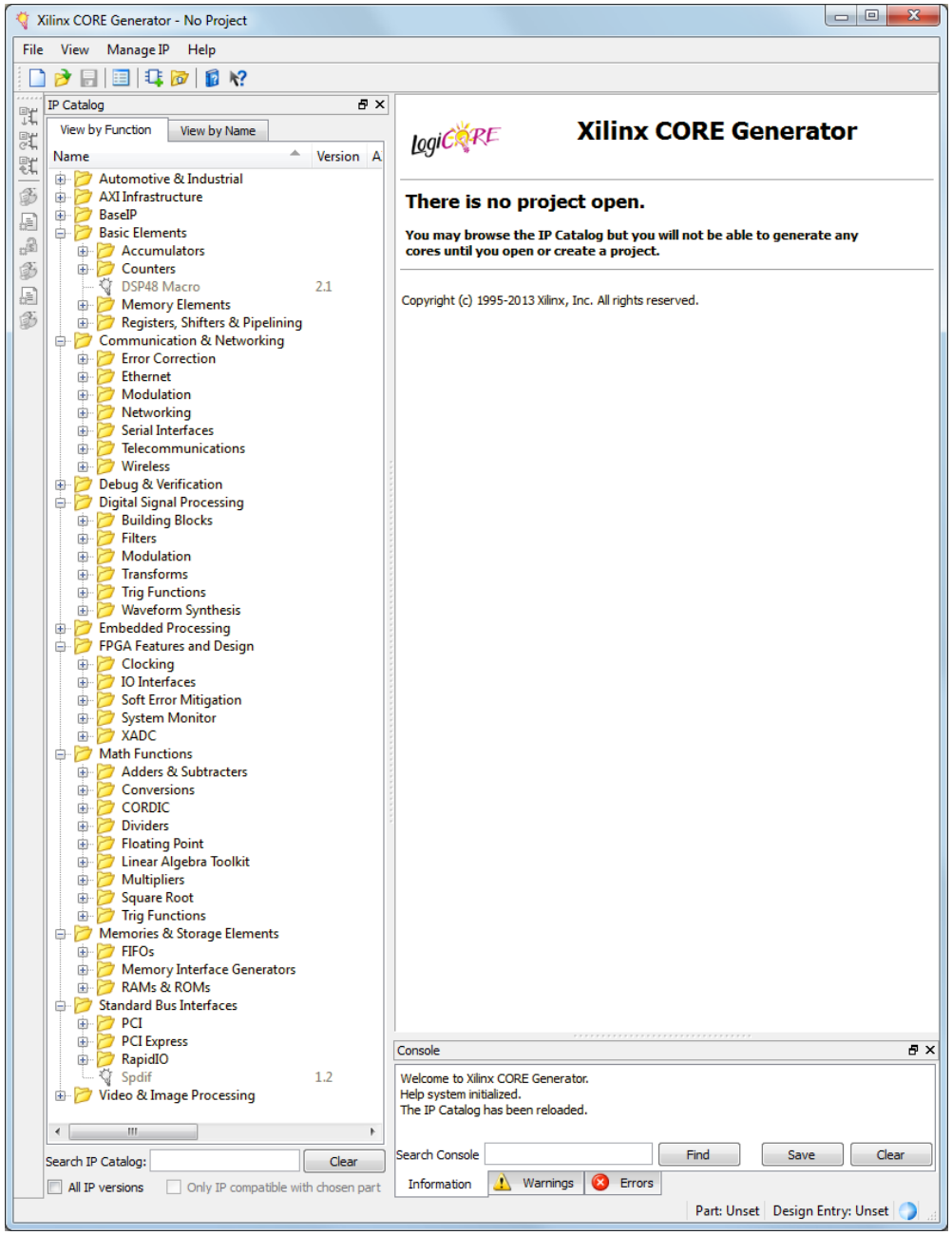
Конечный автомат (пример: светофор)

Симуляция

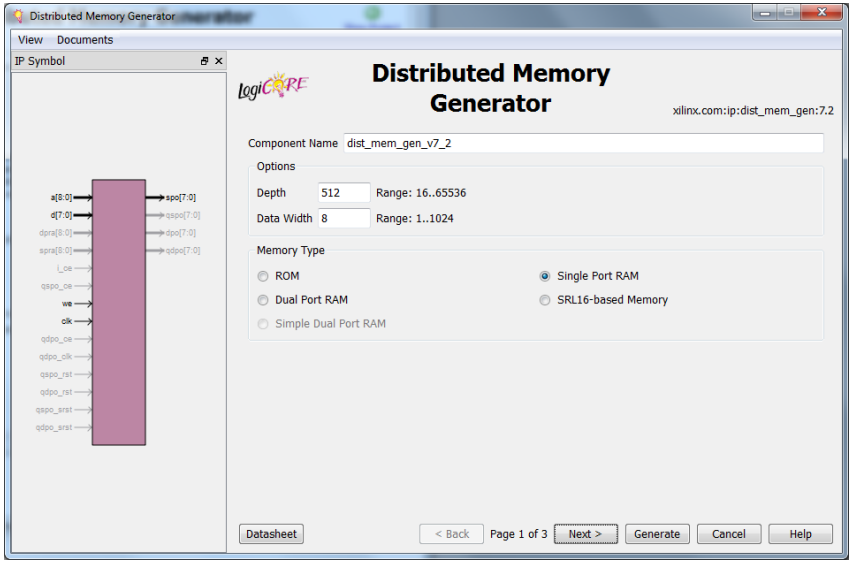


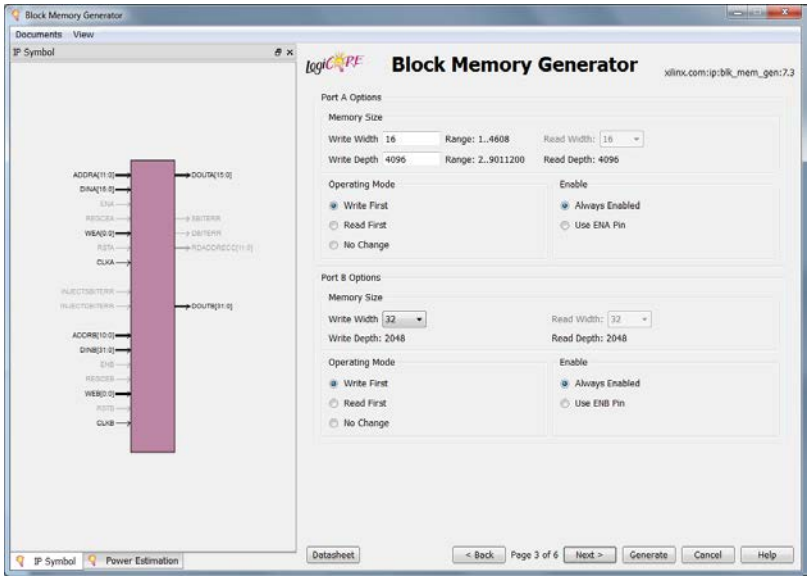
Примечание. Период тактовой частоты уменьшен в 1000 раз, миллисекунды во временных метках следует интерпретировать как секунды.

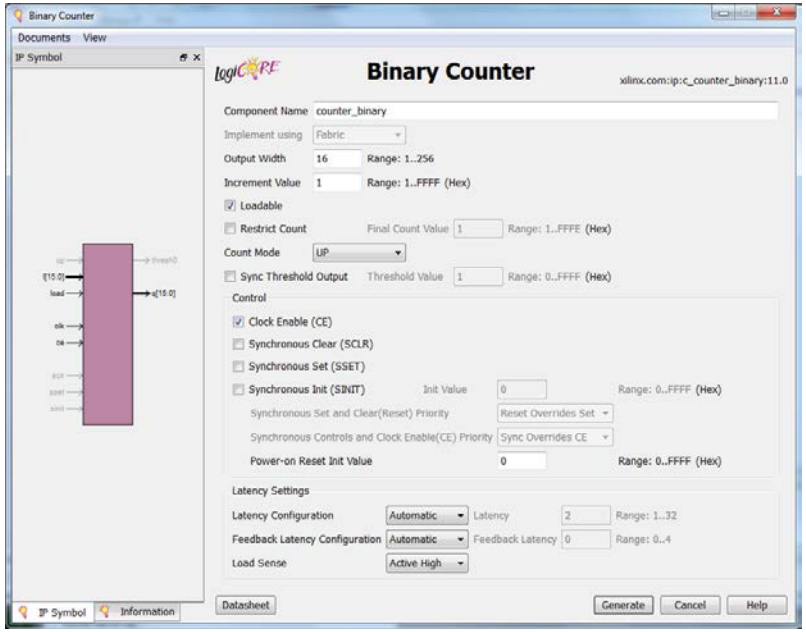
Генератор IP ядер (CORE Generator)

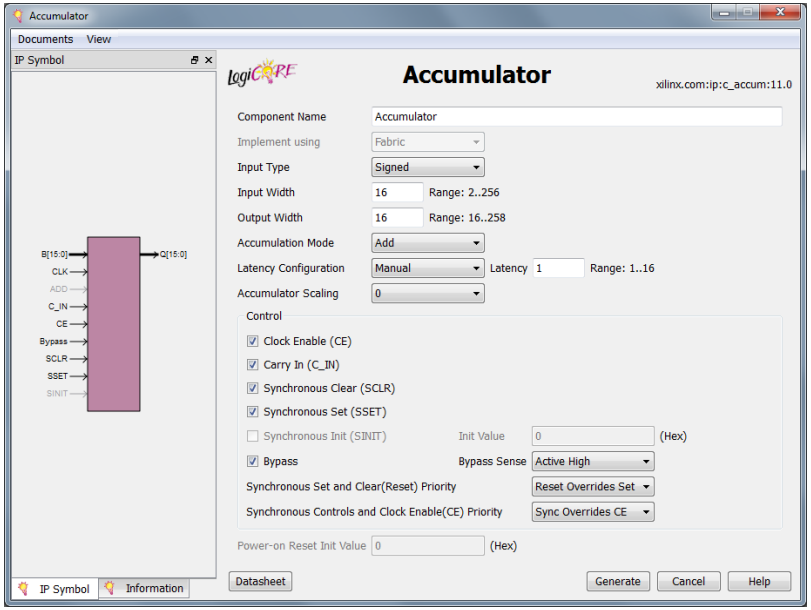


Генератор IP ядер (CORE Generator)

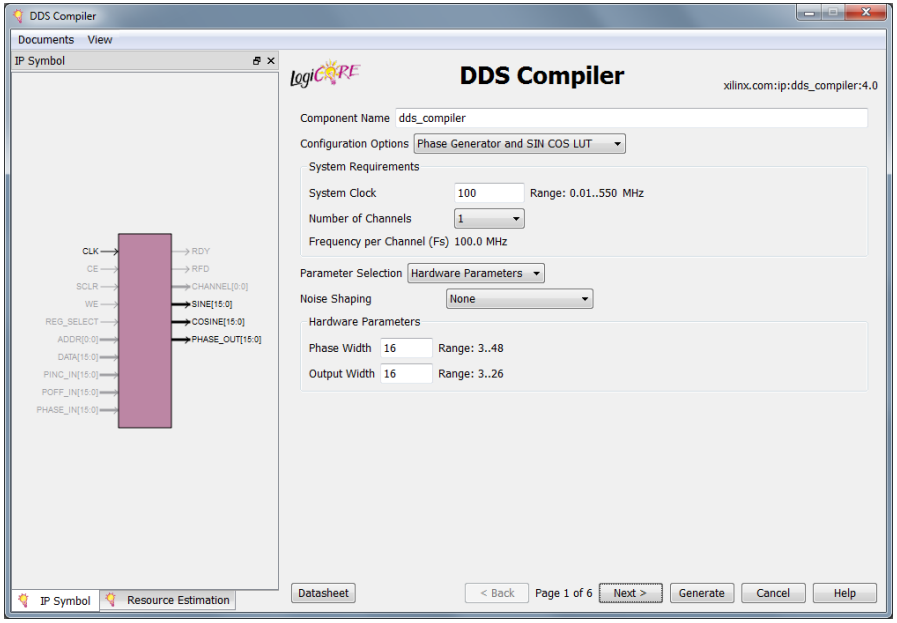


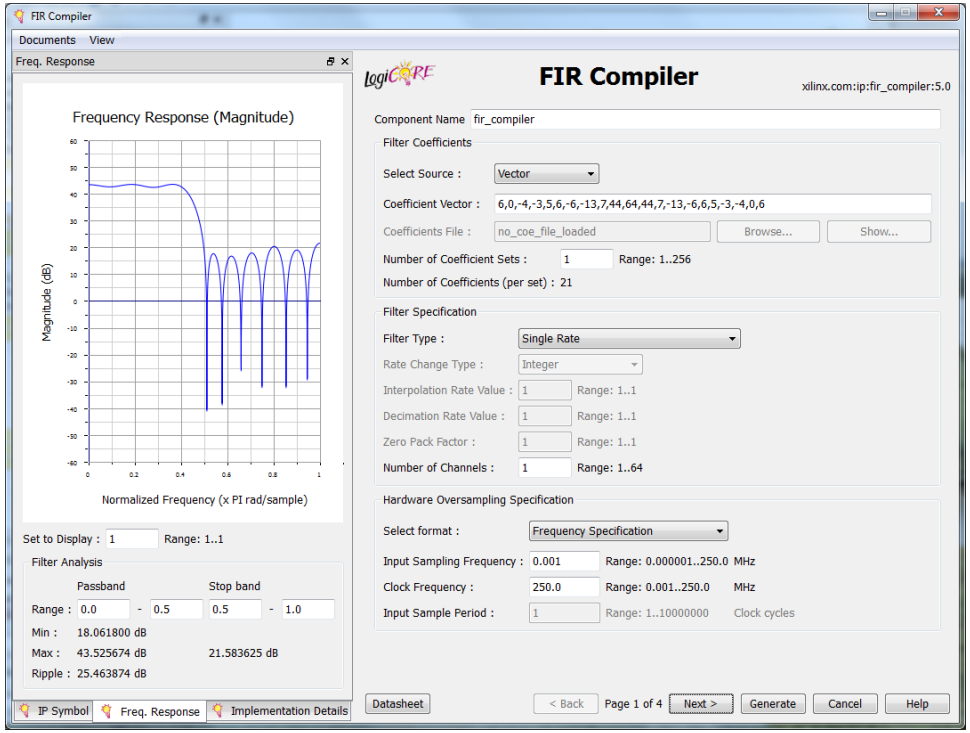






Генератор IP ядер (CORE Generator)

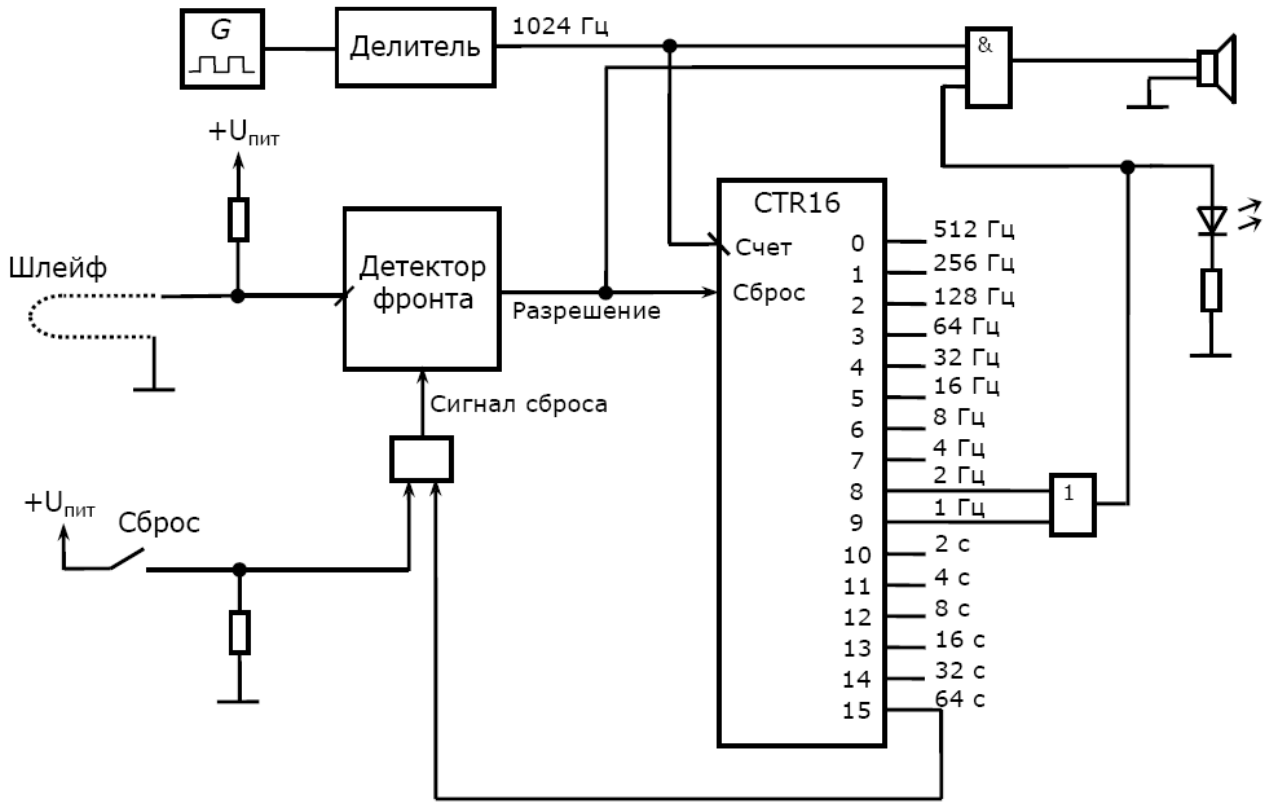




Пример синтеза

Устройство охранной сигнализации. При разрыве шлейфа в течение 32 с формировать звуковой и световой сигналы с периодичностью 1 с и коэффициентом длительности 75% (0.75 с – сигнал, 0.25 с – пауза). Частота звукового сигнала — 1024 Гц. При восстановлении целостности шлейфа и последующем нарушении процесс повторять.

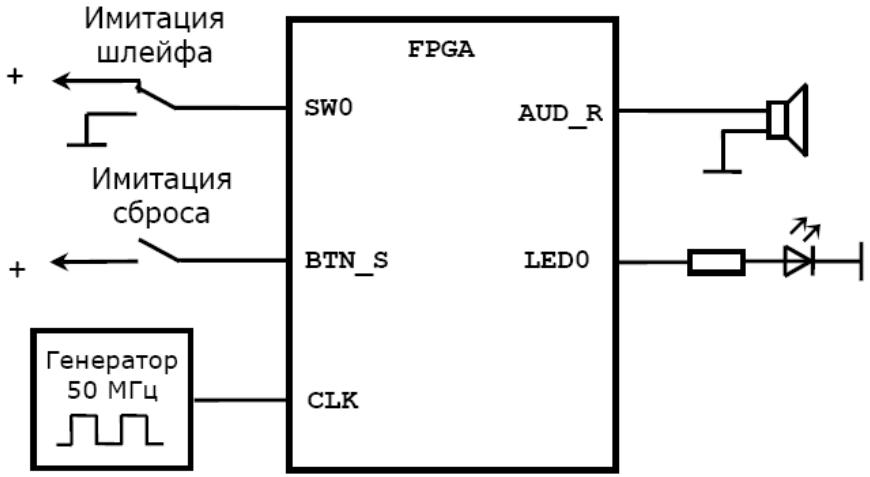
Функциональная схема устройства на дискретной логике

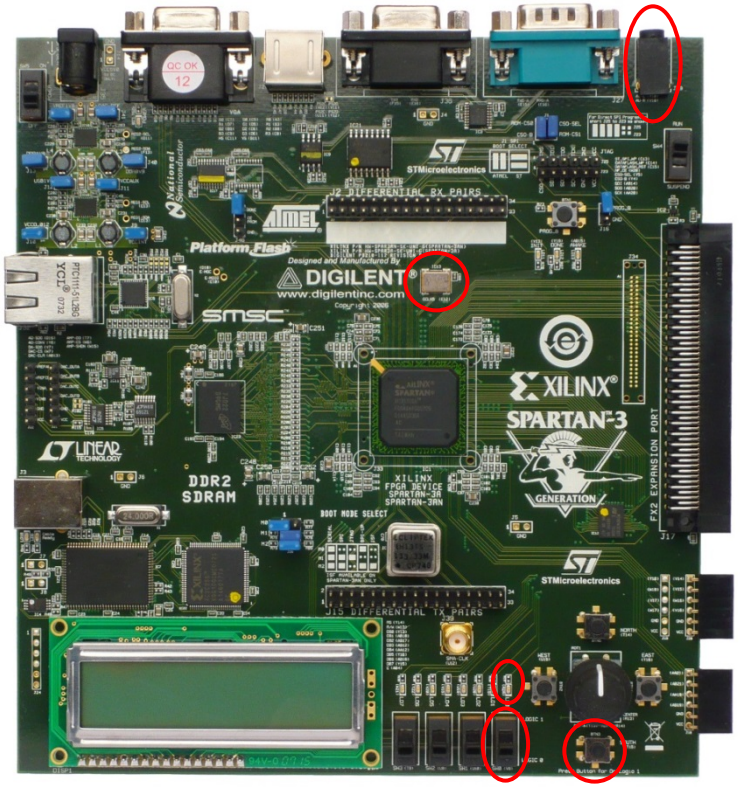


Пример синтеза

Устройство охранной сигнализации. При разрыве шлейфа в течение 32 с формировать звуковой и световой сигналы с периодичностью 1 с и коэффициентом длительности 75% (0.75 с – сигнал, 0.25 с – пауза). Частота звукового сигнала — 1024 Гц. При восстановлении целостности шлейфа и последующем нарушении процесс повторять.

Интерфейс устройства на базе отладочной платы





Пример синтеза

Файл проектных ограничений

```
##############################################################################
#                  ФАЙЛ ПРОЕКТНЫХ ОГРАНИЧЕНИЙ
#         для отладочного модуля Spartan-3A/3AN FPGA Starter Kit
#                    Тип FPGA: XC3S700A-FG484
##############################################################################

#=============================================================================
# Вход тактовой частоты 50 МГц (CLK)
#=============================================================================

NET "CLK"     LOC = "E12"  | IOSTANDARD = LVCMOS33 | PERIOD = 20.000 ;

#=============================================================================
# Дискретные светодиодные индикаторы (LED)
#=============================================================================

NET "LED0"    LOC = "R20"  | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

#=============================================================================
# Кнопки общего назначения - при нажатии формируют на входе FPGA лог. '1'.
# На входе обязательны подтягивающие резисторы к лог. '0'.
#=============================================================================

NET "BTN_S"   LOC = "T15"  | IOSTANDARD = LVCMOS33 | PULLDOWN ;

#=============================================================================
# Механические переключатели (SW) - формируют лог. '0' и '1'.
# Подтягивающий резистор не обязателен
#=============================================================================

NET "SW0"     LOC = "V8"   | IOSTANDARD = LVCMOS33 | PULLUP ;

#=============================================================================
# Аудиовыход
#=============================================================================

NET "AUD_R"   LOC = "V10"  | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
```

Пример синтеза

Исходный текст головного модуля (стр. 1 из 3)

```
library IEEE;                                   -- Объявление библиотек
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--------------------------------------------------------------------------------
-- Описание входных и выходных сигналов устройства
--------------------------------------------------------------------------------
entity Alarm is
   port (   CLK      : in  std_logic;         -- Вход тактовой частоты 50 МГц
            BTN_S    : in  std_logic;         -- От кнопки сброса
            SW0      : in  std_logic;         -- От переключателя имитации шлейфа
            AUD_R    : out std_logic;         -- Выход звукового сигнала
            LED0     : out std_logic          -- Выход на индикатор
        );
end entity;

--------------------------------------------------------------------------------
-- Начало описания архитектуры устройства
--------------------------------------------------------------------------------
architecture arch of Alarm is

------- Объявление внутренних сигналов -----------------------------------------
-- Счетчик-делитель входной тактовой частоты 50 МГц до базовой 1024 Гц
signal div_clk : integer range 0 to 50000 := 0;

-- Сигнал базовой частоты
signal clk_ms : std_logic := '0';

-- Признак разрешения генерации сигналов оповещения
signal alarm_enable : std_logic := '0';

-- Двоичный счетчик-делитель базовой частоты 1024 Гц (период около 1 мс)
signal counter : std_logic_vector(15 downto 0) := (others => '0');

-- Предыдущие состояния сигнала с датчика - используются для фильтрации помех
signal sw0_p1, sw0_p2 : std_logic := '0';
```

Пример синтеза

Исходный текст головного модуля (стр. 2 из 3)

```
----------------------------------------------------------------------------
-- Описание функционирования устройства
----------------------------------------------------------------------------
begin

-- Генерация частоты 1024 Гц делением на 50 МГц / 2 / 1024 Гц = 24414
-- (кэффициент 2 - для отсчета полупериодов генерируемой частоты)
process (clk)
begin
   if rising_edge(clk) then
      if div_clk = 0 then
         div_clk <= 24414-1; clk_ms <= not clk_ms;
      else
         div_clk <= div_clk - 1;
      end if;
   end if;
end process;

-- Идентификация срабатывания датчика (шлейфа) путем проверки текущего и
-- предыдущих состояний.
-- Установка признака разрешения оповещения, если сработал датчик.
-- Сброс разрешения при нажатии кнопки сброса или окончании временного интервала
process (clk_ms)
begin
   if rising_edge(clk_ms) then
      if SW0 = '1' and sw0_p1 = '1' and sw0_p2 = '0' and alarm_enable = '0' then
         alarm_enable <= '1';
      end if;
      sw0_p2 <= sw0_p1; sw0_p1 <= SW0;
      if BTN_S = '1' or counter(15) = '1' then
         alarm_enable <= '0';
      end if;
   end if;
end process;
```

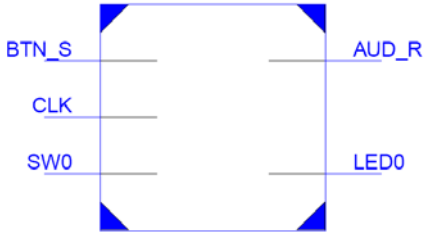
Пример синтеза

Исходный текст головного модуля (стр. 3 из 3)

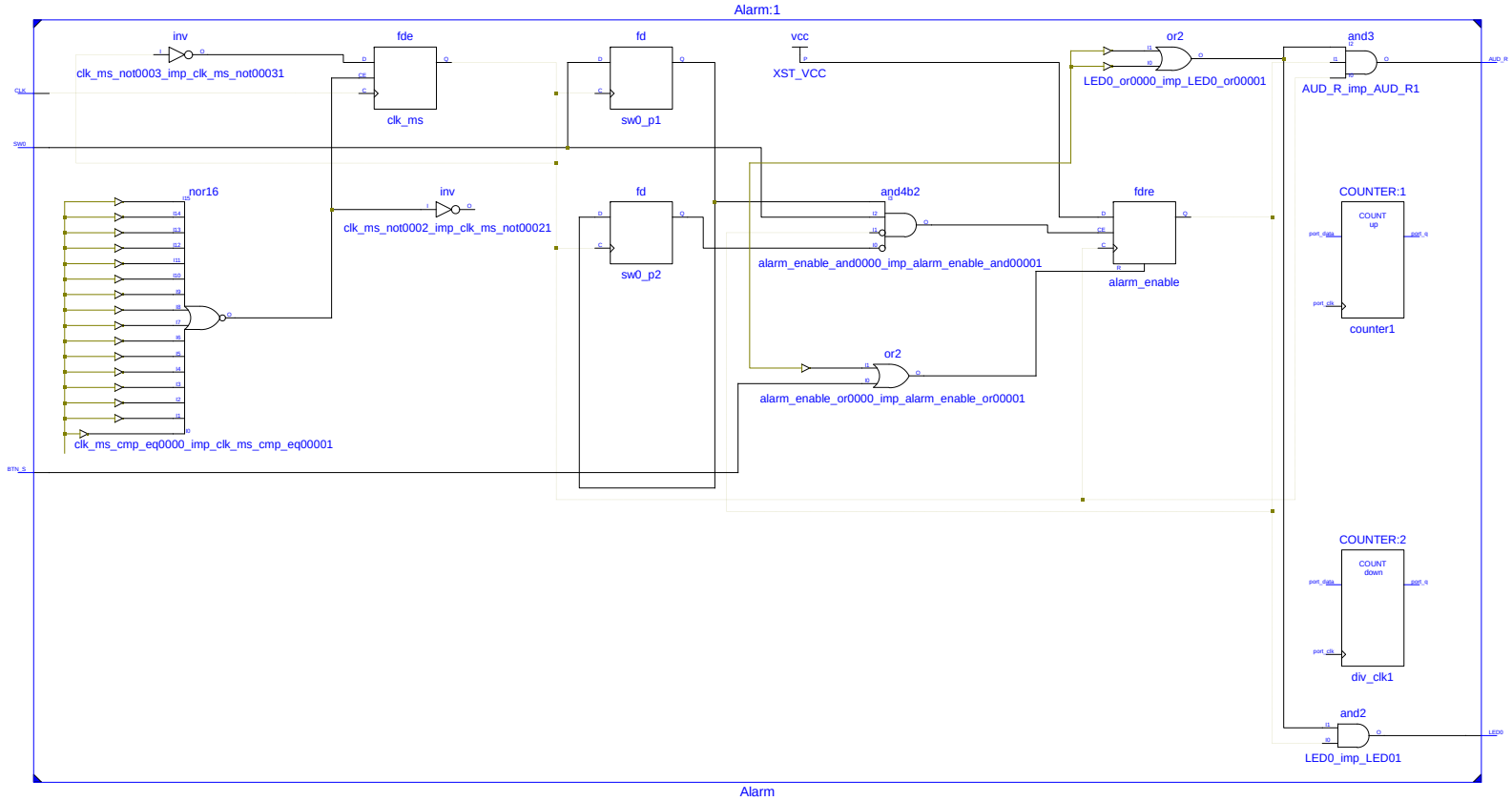
```
-- Обслуживание счетчика формирования временных интервалов
process (clk_ms, alarm_enable)
begin
   if alarm_enable = '0' then
      counter <= (others => '0');
   elsif rising_edge(clk_ms) then
      counter <= counter + 1;
   end if;
end process;

-- Выдача звукового и светового сигналов оповещения
AUD_R <= clk_ms and alarm_enable and (counter(8) or counter(9));
LED0 <= alarm_enable and (counter(8) or counter(9));

------------------------------------------------------------------------------
end arch;       -- КОНЕЦ описания архитектуры и функционирования
------------------------------------------------------------------------------
```

Пример синтеза

Результаты синтеза



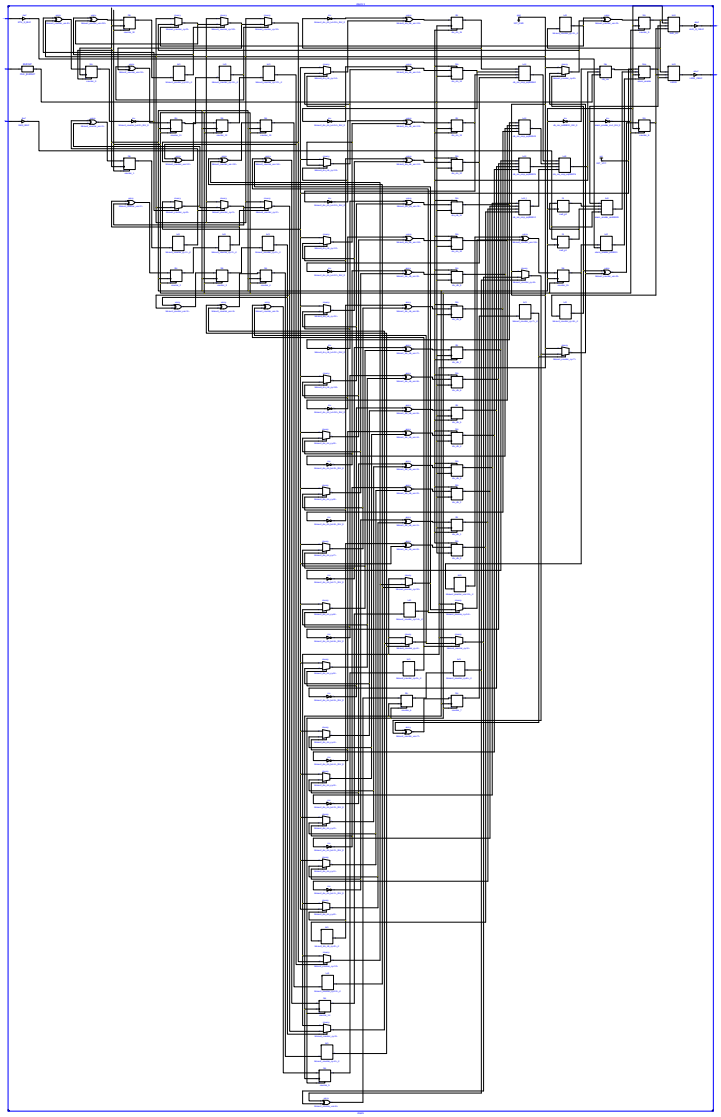
RTL схема (на уровне регистровых передач)



Пример синтеза

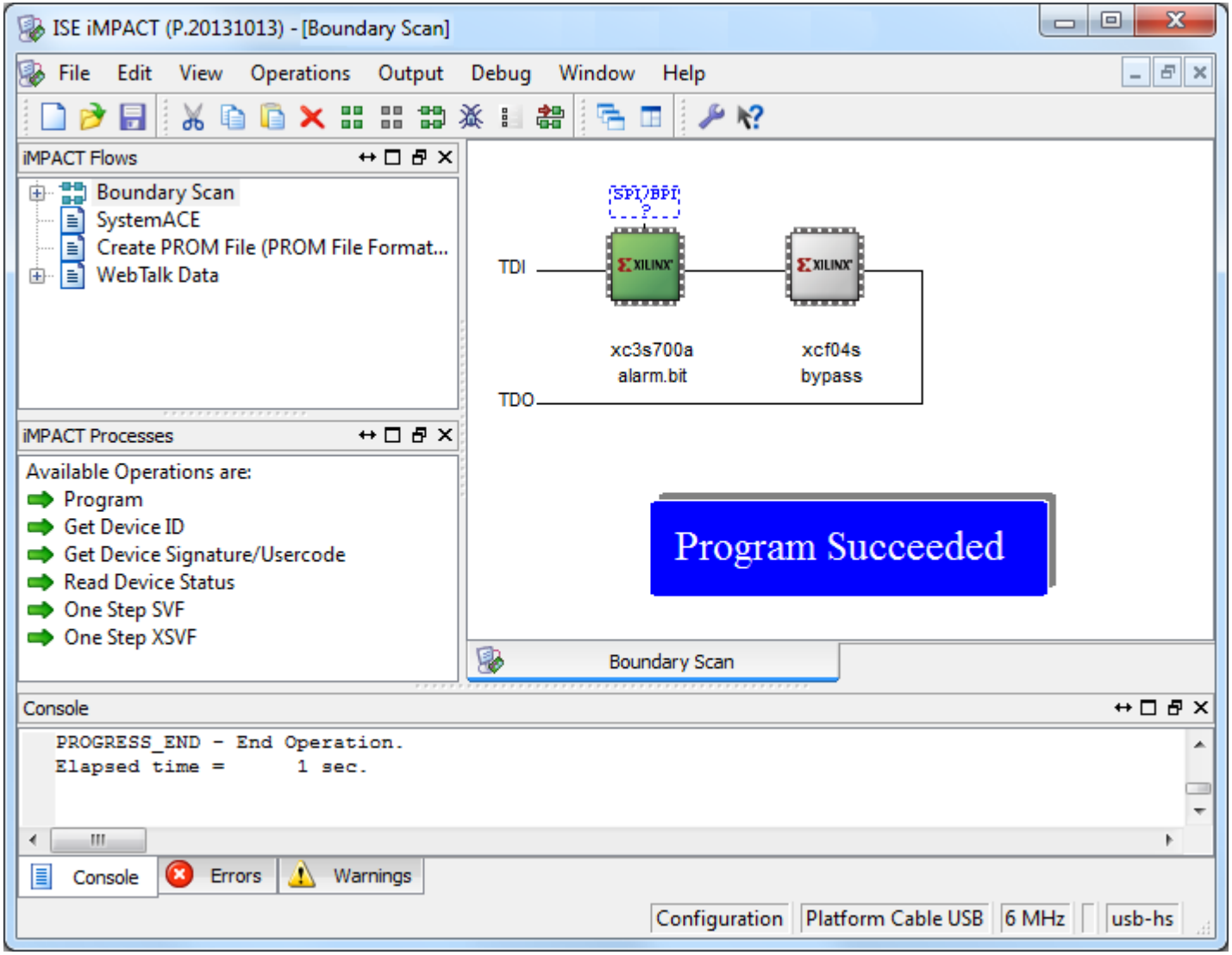
Результаты синтеза

Технологическая схема (на уровне ячеек)



Пример синтеза

Утилита загрузки конфигурации в ПЛИС



Пример синтеза

Отчет об использовании ресурсов ПЛИС

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	36	11,776	1%	
Number of 4 input LUTs	25	11,776	1%	
Number of occupied Slices	26	5,888	1%	
Number of Slices containing only related logic	26	26	100%	
Number of Slices containing unrelated logic	0	26	0%	
Total Number of 4 input LUTs	41	11,776	1%	
Number used as logic	25			
Number used as a route-thru	16			
Number of bonded IOBs	6	372	1%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	1.97			

Пример синтеза

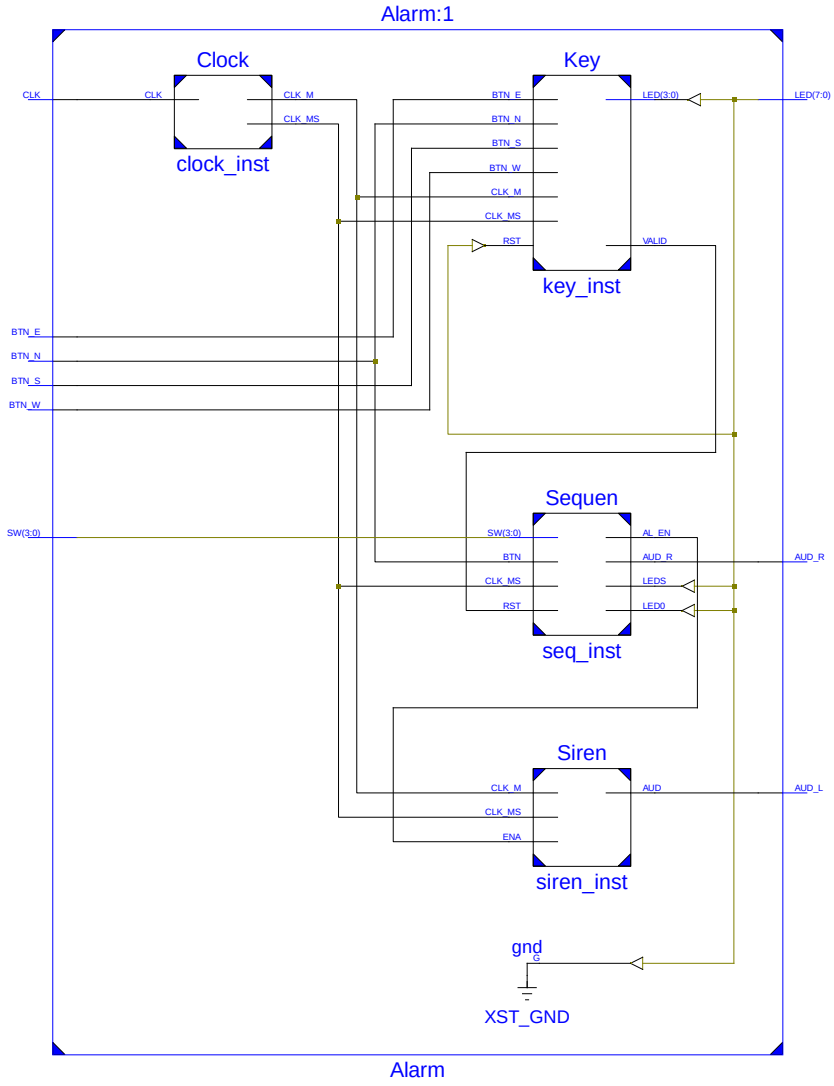
Модификация проекта

Рекомендации по модернизации устройства
(предлагается для самостоятельной работы):

- добавить режим постановки на охрану отдельной кнопкой;
- использовать несколько датчиков (шлейфов);
- ввести индикацию текущего состояния шлейфа;
- ввести функцию ввода пароля (ключа) для снятия с охраны, предусмотреть возможность модификации ключа (по содержанию и размеру), для целей отладки ввести индикацию правильности каждого нажатия;
- при срабатывании датчиков ввести задержку формирования сигналов оповещения (для отключения сигнализации);
- ввести индикацию всех режимов — сигнализация отключена, сигнализация включена, задержка после срабатывания датчика, формирование сигнала оповещения — можно реализовать на одном индикаторе, меняя характер индикации;
- добавить звуковой сигнал типа сирены.

Пример синтеза

Пример RTL схемы головного модуля модифицированного проекта



Конец