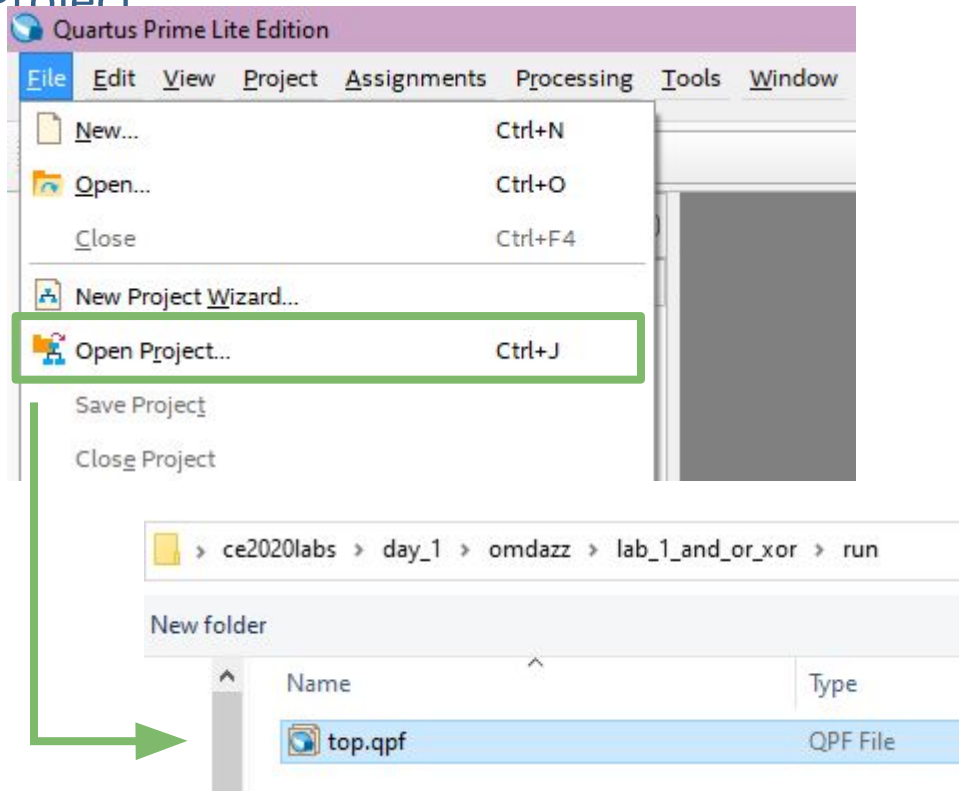


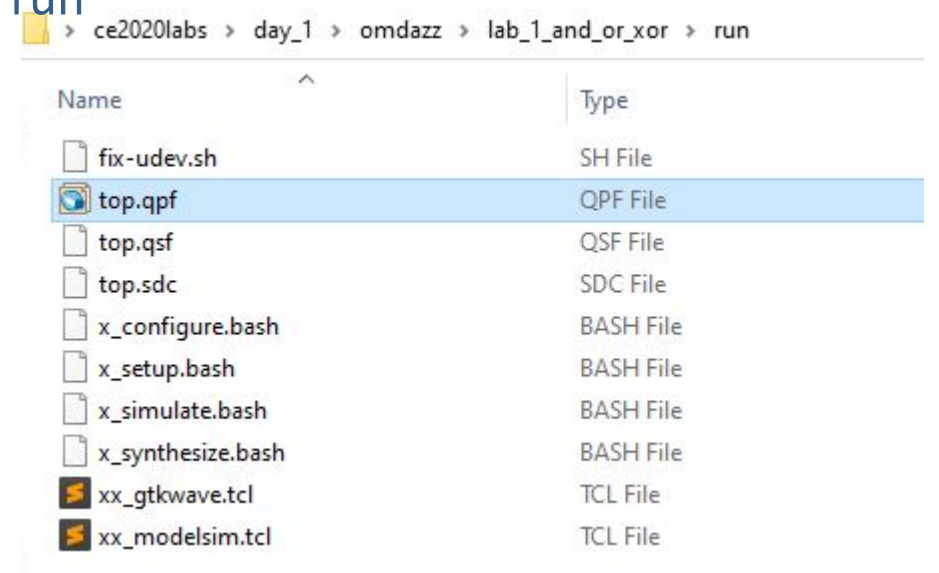
Комбинационная логика на ПЛИС. Последовательностная логика на ПЛИС: схемы с тактовым сигналом и состоянием.

Открытие проекта в Quartus

В Quartus откройте в меню “File” → “Open Project...”

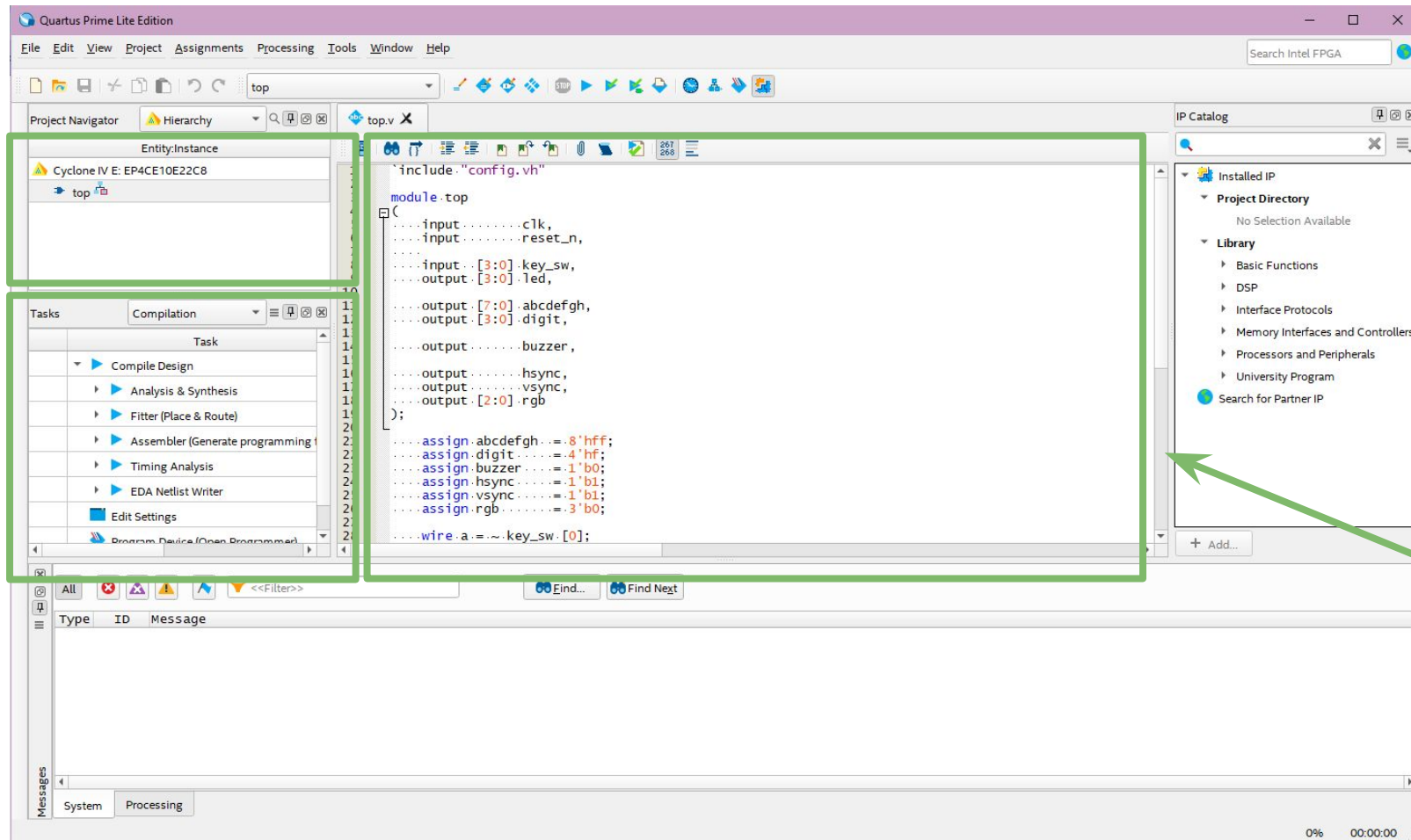


Запустите файл `top.qpf` из директории `run`



ИЛИ

Работа с проектом Quartus



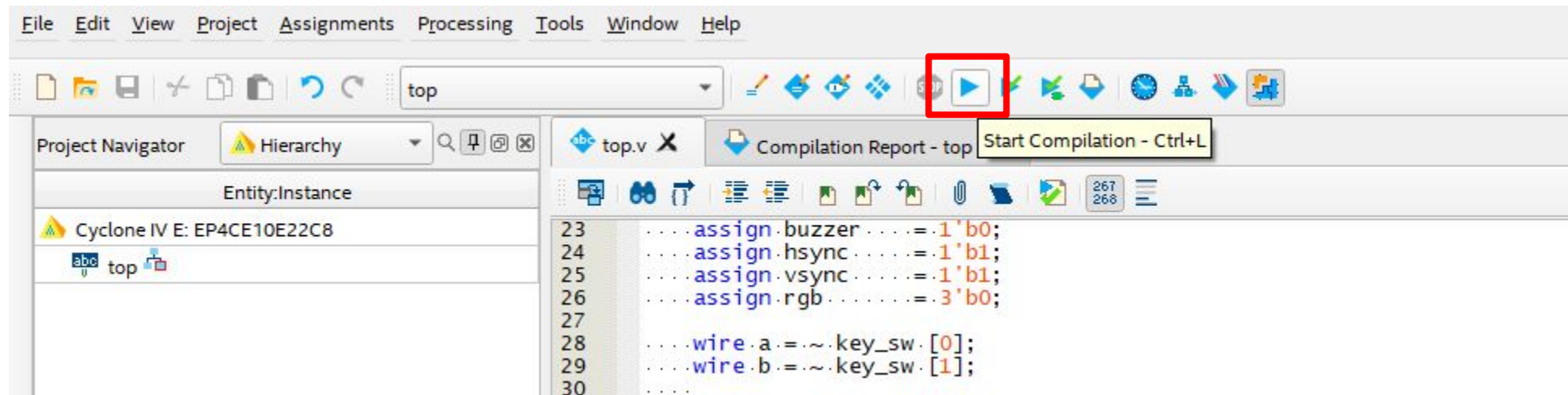
Иерархия
проекта

Окно
задач

Редактора
кода

Генерация прошивки для ПЛИС

1. Запустите компиляцию проекта через кнопку “Start Compilation”



Генерация прошивки для ПЛИС

2. Проверьте наличие ошибок и исправьте их

The screenshot displays the Quartus Prime IDE interface during a compilation process. The 'Task' pane on the left shows the 'Compile Design' task with sub-tasks like 'Analysis & Synthesis' and 'Fitter (Place & Route)'. The 'Flow Messages' pane shows 'Flow Suppressed Messages'. The 'Resource Usage' pane lists various resources, all marked as 'N/A until Partition Merge'. The 'Messages' pane at the bottom shows the following error messages:

```
Running Quartus Prime Analysis & synthesis
Command: quartus_map --read_settings_files=on --write_settings_files=off top -c top
20032 Parallel compilation is enabled and will use up to 4 processors
10170 Verilog HDL syntax error at top.v(40) near text: "assign"; expecting ";". Check for and fix any syntax errors that appear immediately before or at the specified
10112 Ignored design unit "top" at top.v(3) due to previous errors
12021 Found 0 design units, including 0 entities, in source file /users/frar/desktop/ce2020labs/day_1/omdazz/lab_1_and_or_xor/top.v
Quartus Prime Analysis & Synthesis was unsuccessful. 2 errors, 0 warnings
293001 Quartus Prime Full Compilation was unsuccessful. 4 errors, 0 warnings
```

Генерация прошивки для ПЛИС

2. Проверьте наличие ошибок и исправьте их

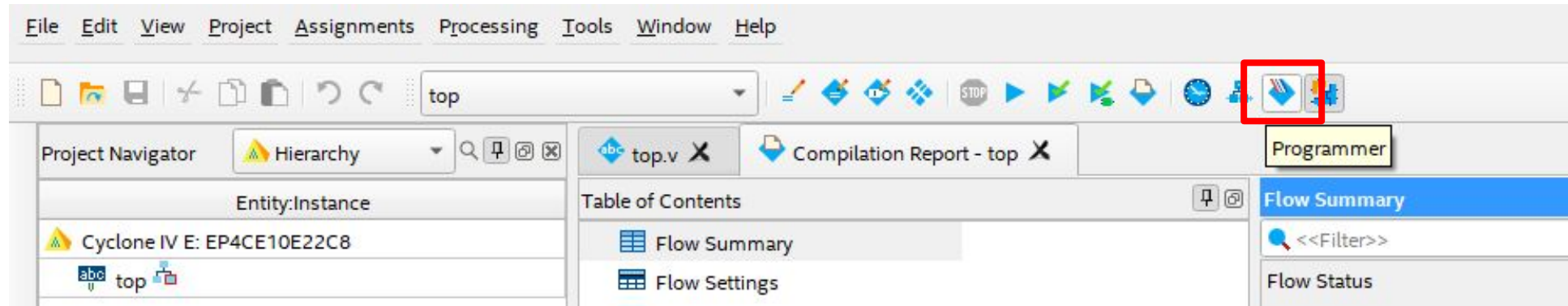
Нажмите на “Show Error Messages” чтобы отобразить только сообщения об ошибках.

The screenshot shows the Quartus Prime IDE interface. On the left, the 'Task' pane lists various compilation steps, with 'Compile Design' and 'Analysis & Synthesis' marked with red 'X' icons. The main window displays the 'Messages' pane, which is currently showing a list of messages. A red box highlights the 'Show Error Messages' button (a red 'X' icon) in the toolbar. A green box highlights the text 'Нажмите на “Show Error Messages” чтобы отобразить только сообщения об ошибках.' with a green arrow pointing to the button. The messages list includes:

Type	ID	Message
Information		Running Quartus Prime Analysis & synthesis
Information		Command: quartus_map --read_settings_files=on --write_settings_files=off top -c top
Information	20032	Parallel compilation is enabled and will use up to 4 processors
Error	10170	Verilog HDL syntax error at top.v(40) near text: "assign"; expecting ";" . Check for and fix any syntax errors that appear immediately before or at the specified
Error	10112	Ignored design unit "top" at top.v(3) due to previous errors
Information	12021	Found 0 design units, including 0 entities, in source file /users/frar/desktop/ce2020labs/day_1/omdazz/lab_1_and_or_xor/top.v
Error		Quartus Prime Analysis & Synthesis was unsuccessful. 2 errors, 0 warnings
Error	293001	Quartus Prime Full Compilation was unsuccessful. 4 errors, 0 warnings

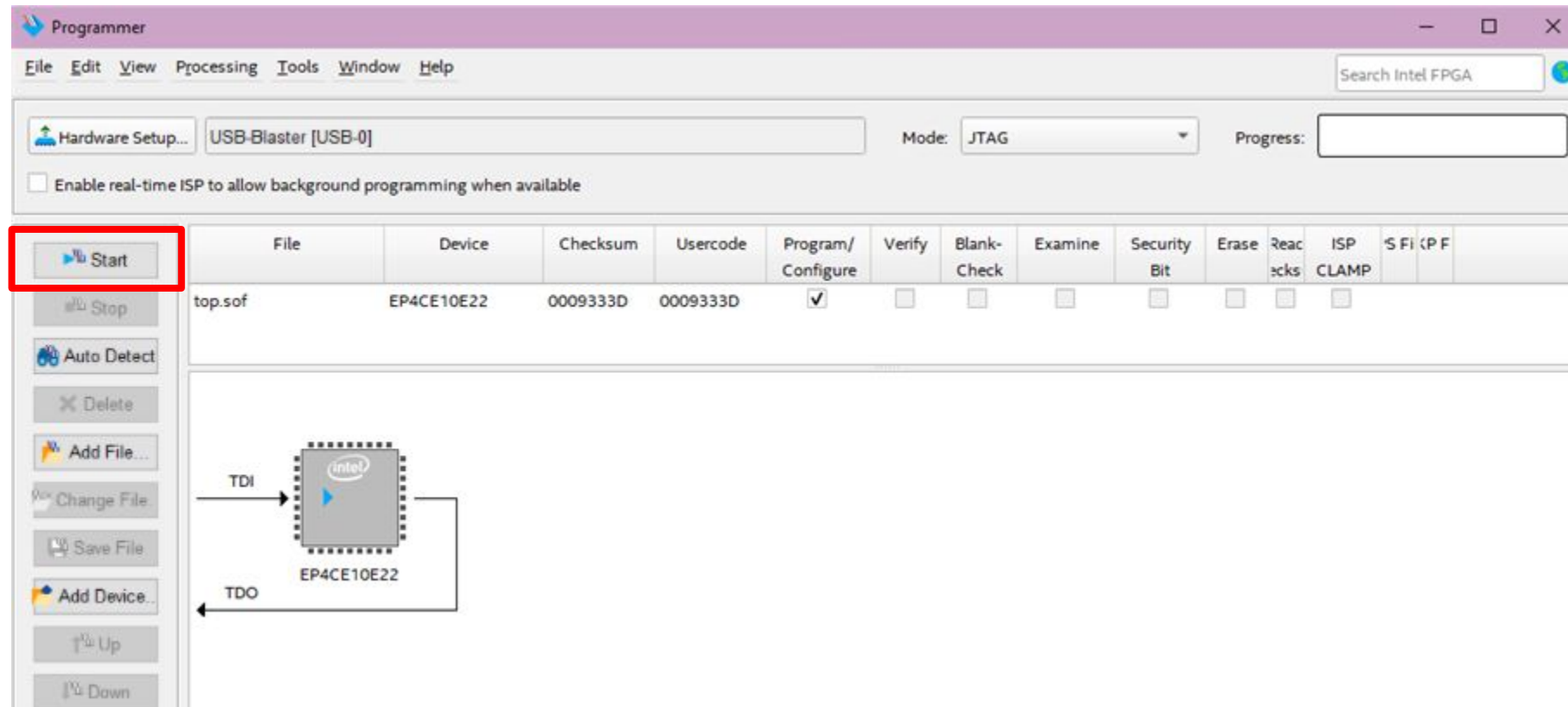
Запись прошивки в ПЛИС

3. Нажмите на кнопку “Programmer”

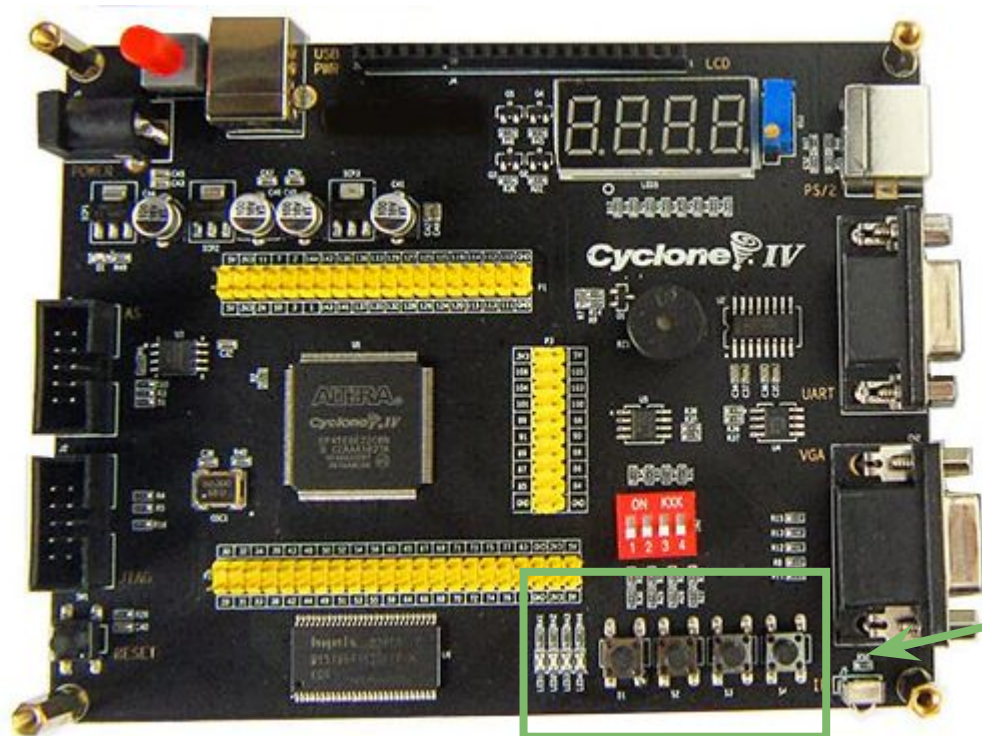


Запись прошивки в ПЛИС

4. В открывшемся окне нажмите на “Start”



Упражнение с логическими элементами



Altera Cyclone IV EP4CE6 FPGA

Вывод результатов логических операций над входными воздействиями с кнопок на светодиоды.

Упражнение с логическими элементами

```
module top
(
  ...
  input [3:0] key_sw,
  output [3:0] led,
  ...
);
...

// Exercise 1: Change the code below.
// Assign to led [2] the result of AND operation

assign led [2] = 1'b0;

endmodule
```

```
module top
(
  ...
  input [3:0] key_sw,
  output [3:0] led,
  ...
);
...

// Exercise 2: Change the code below.
// Assign to led [3] the result of XOR operation
// without using "^" operation.
// Use only operations "&", "|", "~" and parenthesis

assign led [3] = 1'b0;

endmodule
```

Основные операции Verilog HDL

Символ	Назначение
{}	Конкатенация (concatenation)
+ - * /	Арифметические (arithmetic)
%	Модуль (modulus)
> >= < <=	Отношения (relational)
!	Логическое отрицание (logical NOT)
&&	Логическое И (logical AND)
	Логическое ИЛИ (logical OR)

Основные операции Verilog HDL

Символ	Назначение
==	Логическое равенство (logical equality)
!=	Логическое неравенство (logical inequality)
===	Идентичность (case equality)
!==	Не идентичность (case inequality)
~	Побитовая инверсия (bit-wise NOT)
&	Побитовое И (bit-wise AND)
	Побитовое ИЛИ (bit-wise OR)

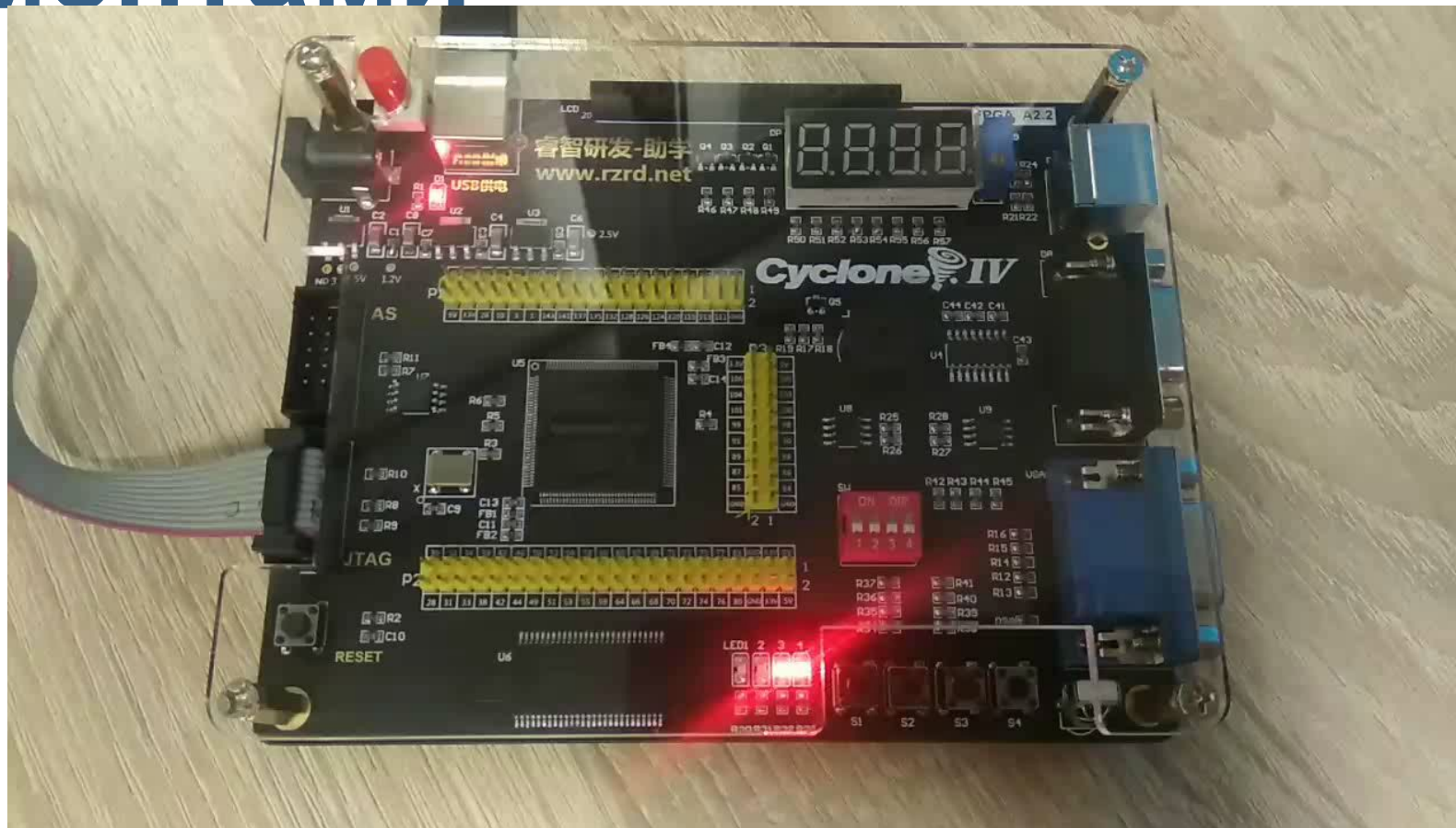
Основные операции Verilog HDL

Символ	Назначение
<<	Сдвиг влево (left shift)
>>	Сдвиг вправо (right shift)
<<<	Циклический сдвиг влево (arithm. left shift)
>>>	Циклический сдвиг вправо (arithm. right shift)
?:	Тернарный оператор (ternary)

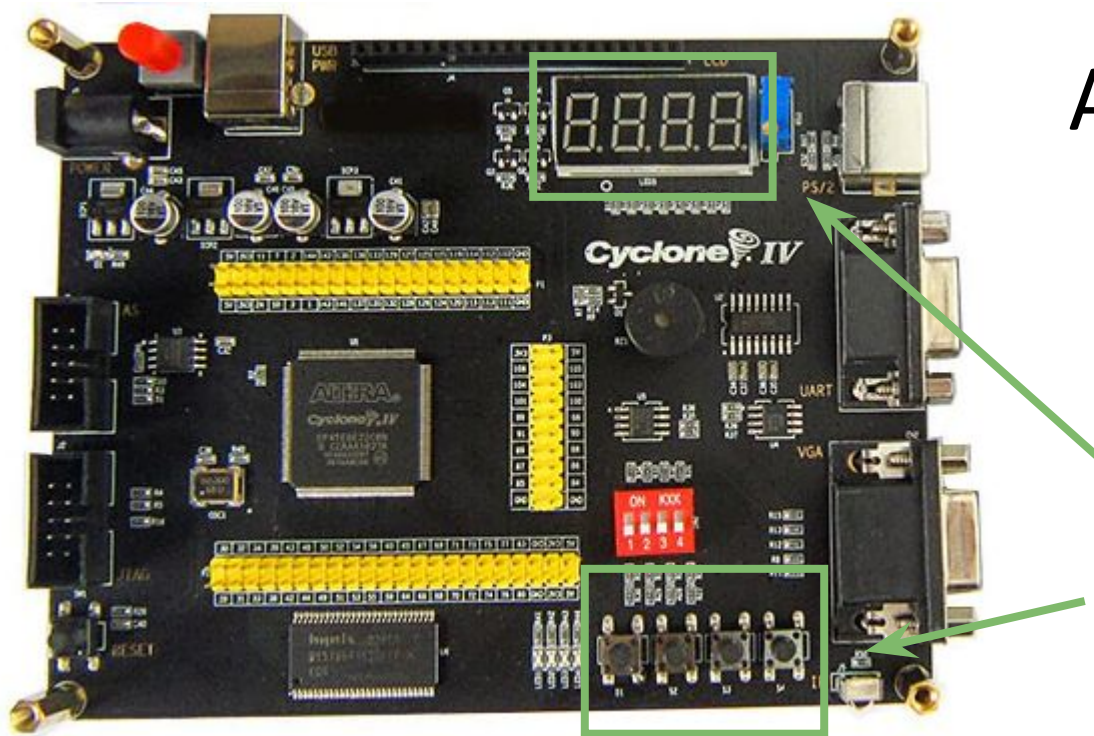
Упражнение с логическими элементами

1. Присвойте `led[2]` результат операции И (AND).
2. Присвойте `led[3]` результат операции исключающего ИЛИ (XOR) **без ИСПОЛЬЗОВАНИЯ** ее оператора.

Упражнение с логическими элементами



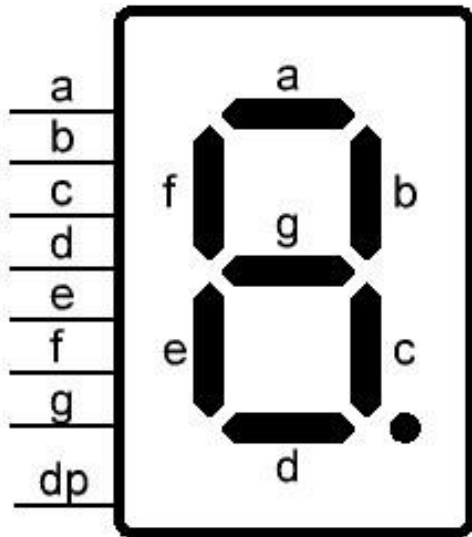
Упражнение с выводом буквы на семисегментный индикатор



Altera Cyclone IV EP4CE6 FPGA

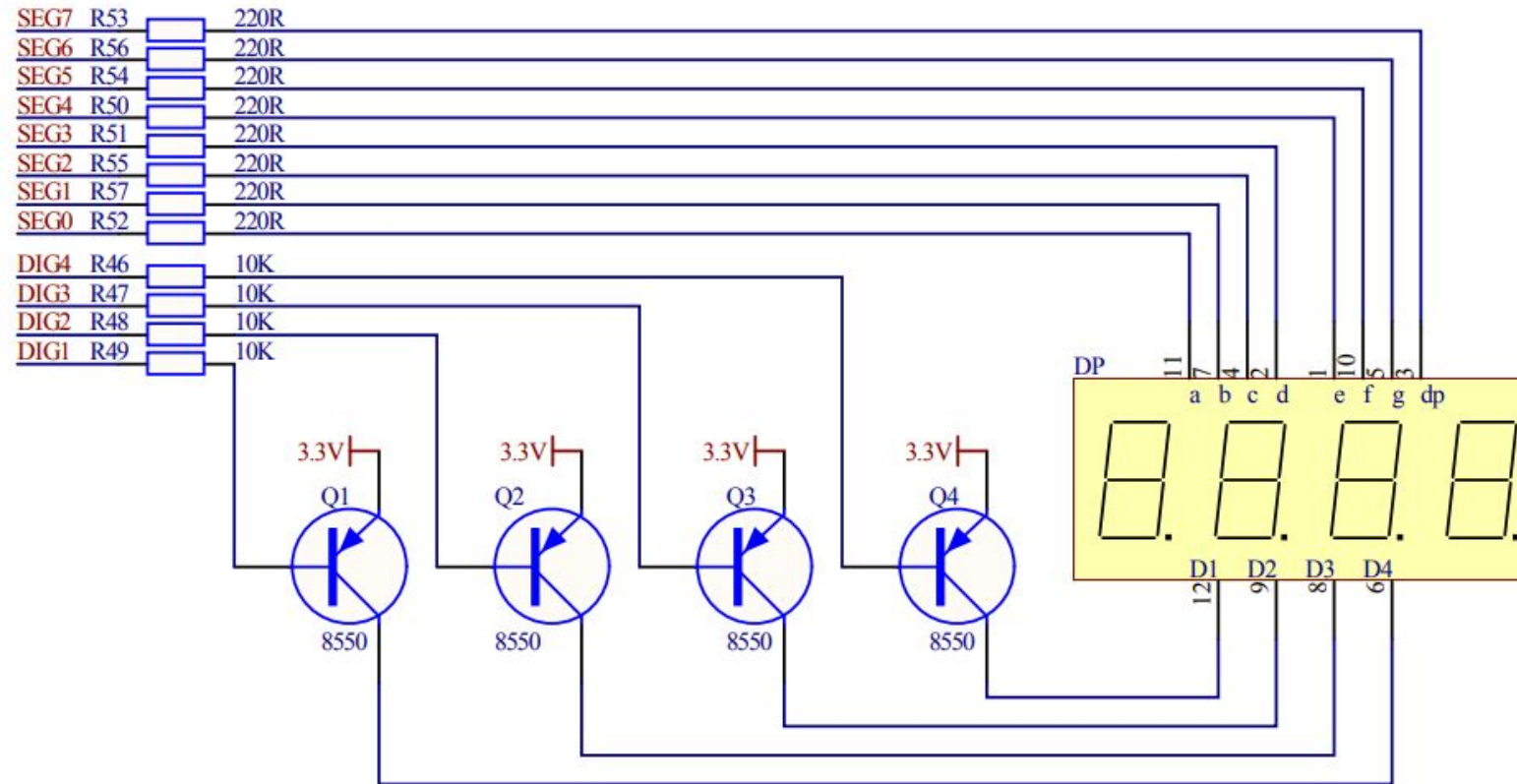
Вывод статичных значений на семисегментный индикатор.
Взаимодействие с семисегментым индикатором через кнопки.

Семисегментный индикатор

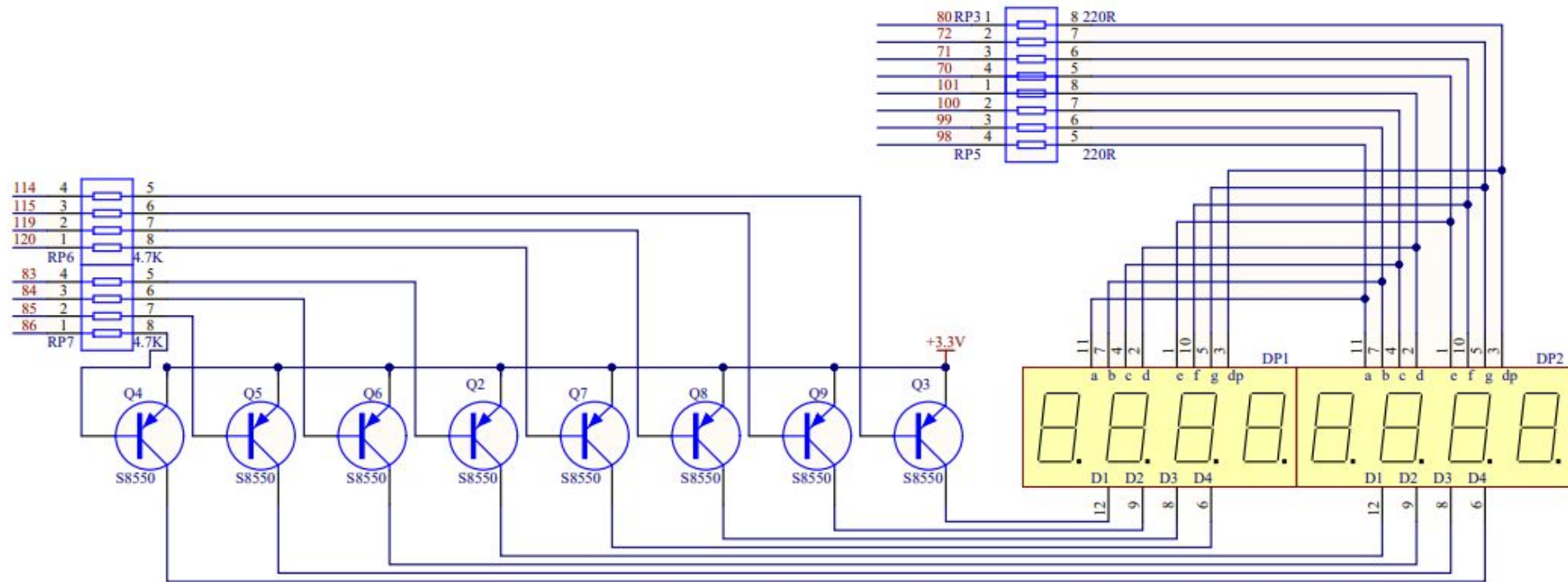


Dec	4-bit шина				7-сегментный индикатор						
	3	2	1	0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Семисегментный индикатор



Семисегментный индикатор



Упражнение с выводом буквы на семисегментный индикатор

```
module top
(
  ...
  input [3:0] key_sw,
  output [3:0] led,
  output [7:0] abcdefgh,
  output [3:0] digit,
  ...
);
...
// Exercise 1: Display the first letters
// of your first name and last name instead.
assign abcdefgh =
assign digit =
endmodule
```

```
module top
(
  ...
);
...
// Exercise 2: Display letters of a 4-character word
// using this code to display letter of ChIP as an example
reg [7:0] letter;
always @*
  case (key_sw)
    4'b0111: letter = C;
    ...
  endcase

assign abcdefgh = letter;

endmodule
```

Verilog HDL. Case

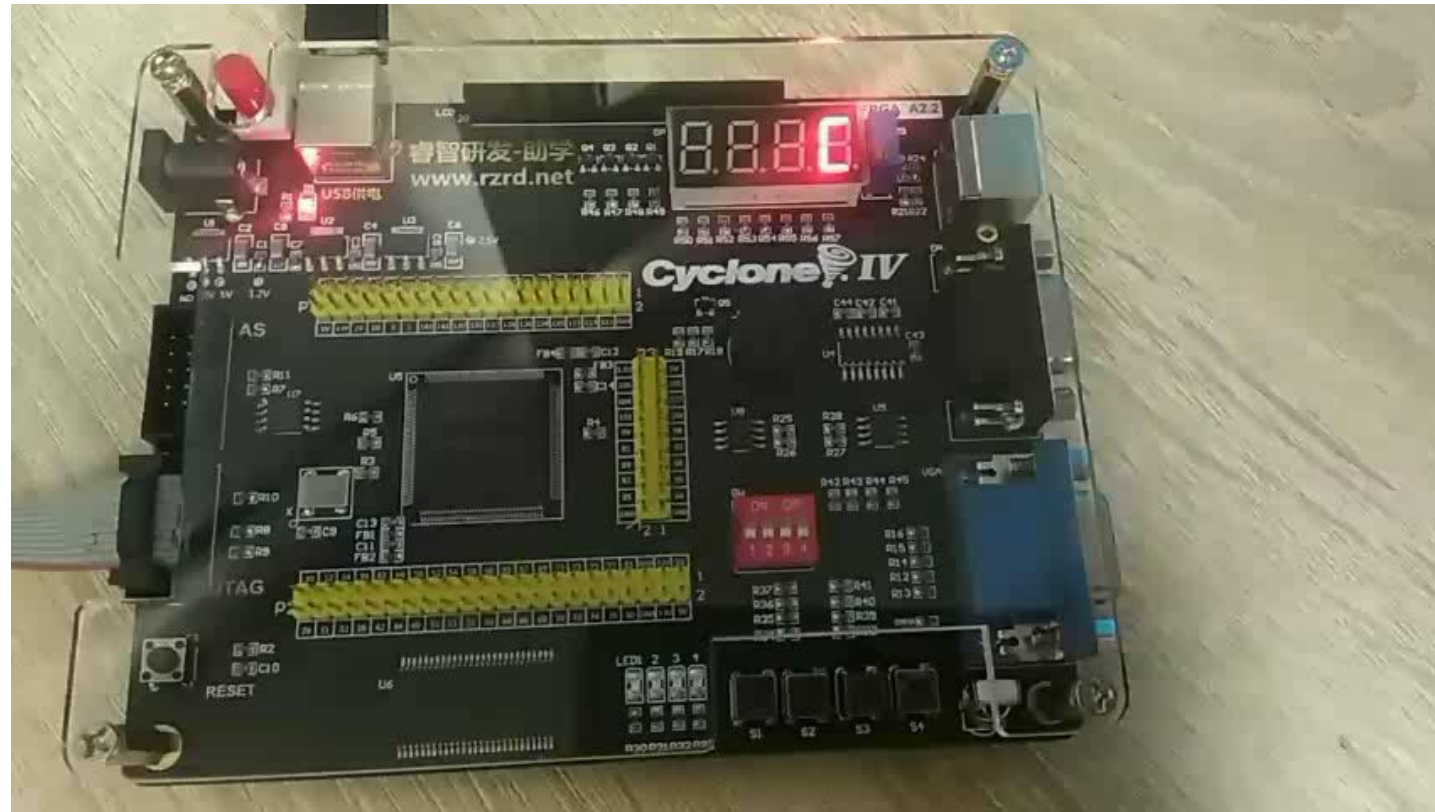
//многовходовые мультиплексоры и дешифраторы можно описывать через case

```
reg [3:0] c;  
wire [1:0] option;  
wire [7:0] a, b, c, d;  
reg [7:0] e;  
always @(a or b or c or d or option) begin  
case (option)  
  0: e = a;  
  1: e = b;  
  2: e = c;  
  3: e = d;  
endcase  
end
```

Упражнение с выводом буквы на семисегментный индикатор

1. Выведите первые буквы своего имени и фамилии на семисегментный индикатор.
2. Выведите слово CHIP на семисегментный индикатор.

Упражнение с выводом буквы на семисегментный индикатор



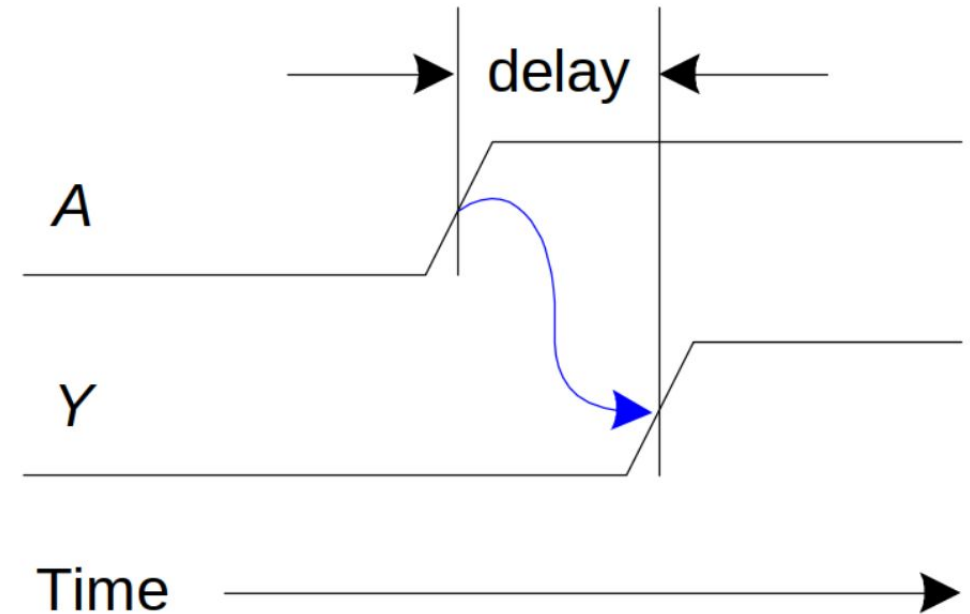
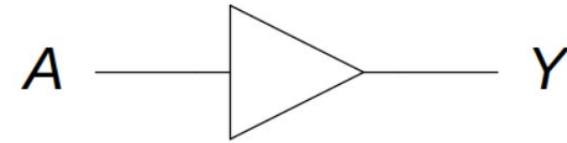
Последовательностная логика



Проблема вычислений в комбинационной логике.

Вычисления начинаются при изменении входов логики.

Как понять когда результат на выходе комбинационной логики будет готов для дальнейших вычислений?



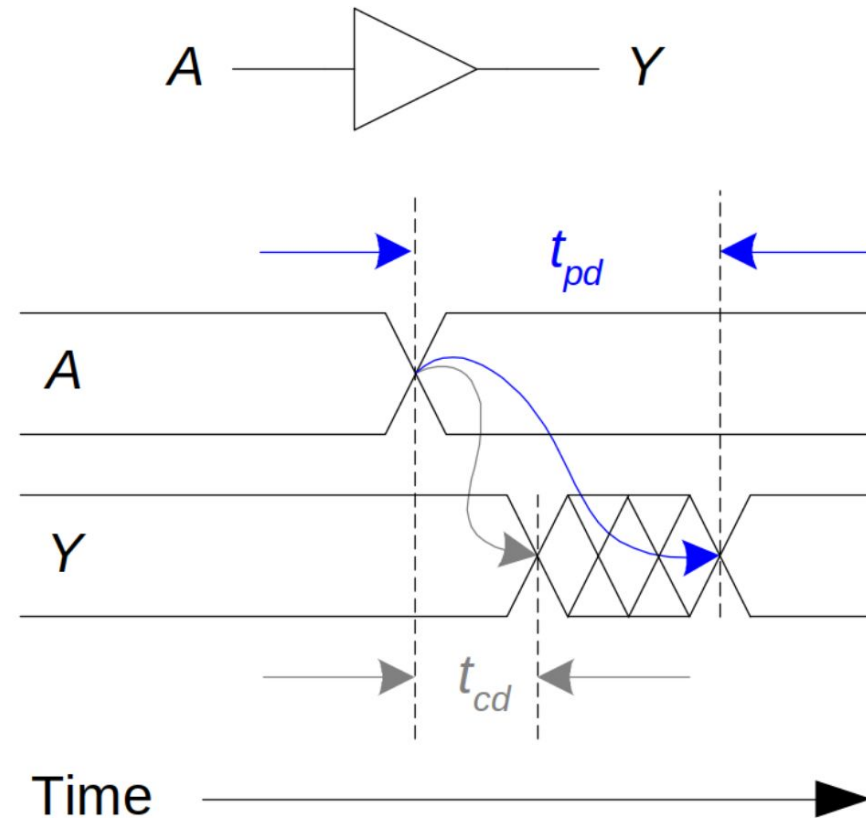
Contamination and propagation delays.

Contamination delay

Входы изменились, но на выходе результат пока что нестабильный.

Propagation delay

Стабильный результат на выходе комбинационной схемы.

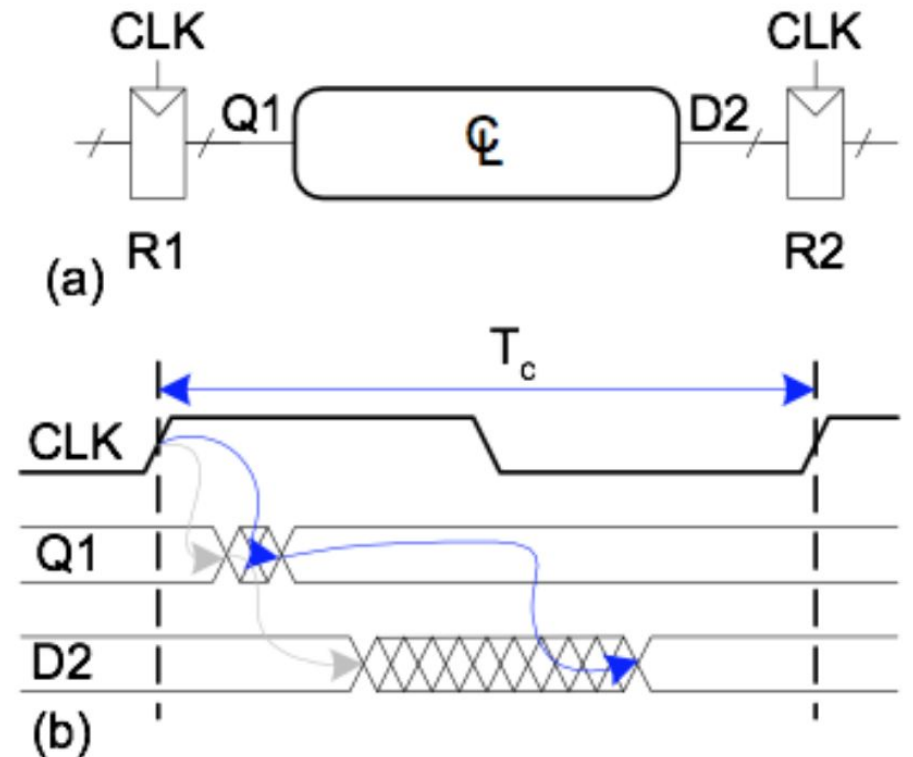


Использование тактового сигнала.

Перед завершением вычислений выходные данные могут содержать случайные значения.

Как логике узнать когда результаты готовы и могут использоваться на следующем этапе вычислений?

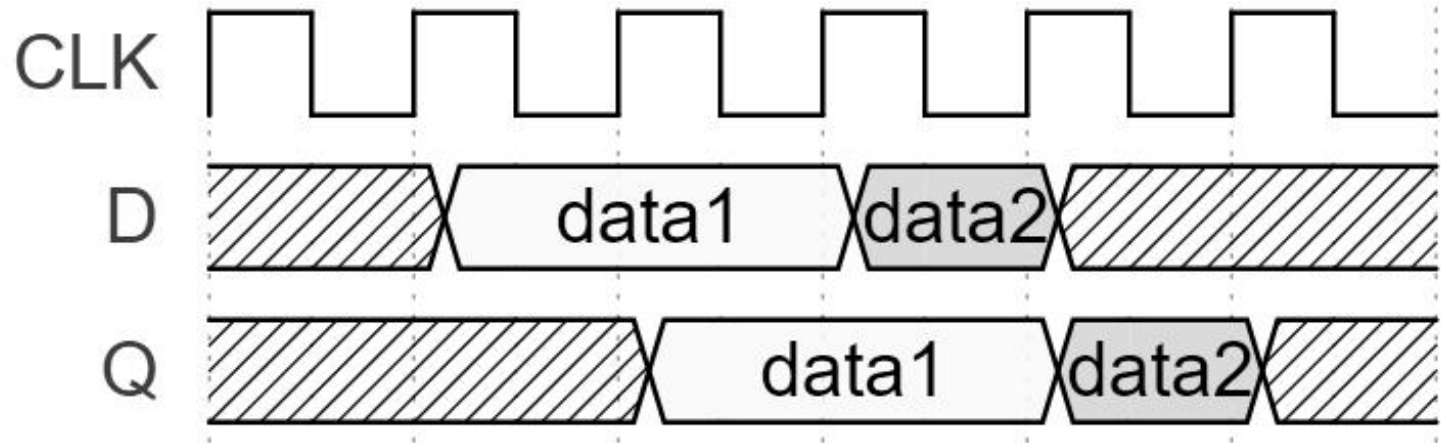
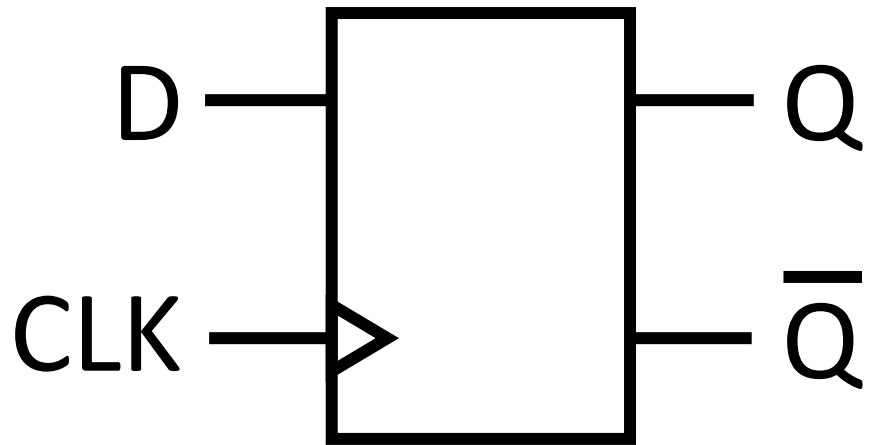
Вычисления можно синхронизировать с помощью специального сигнала – сигнала тактирования.



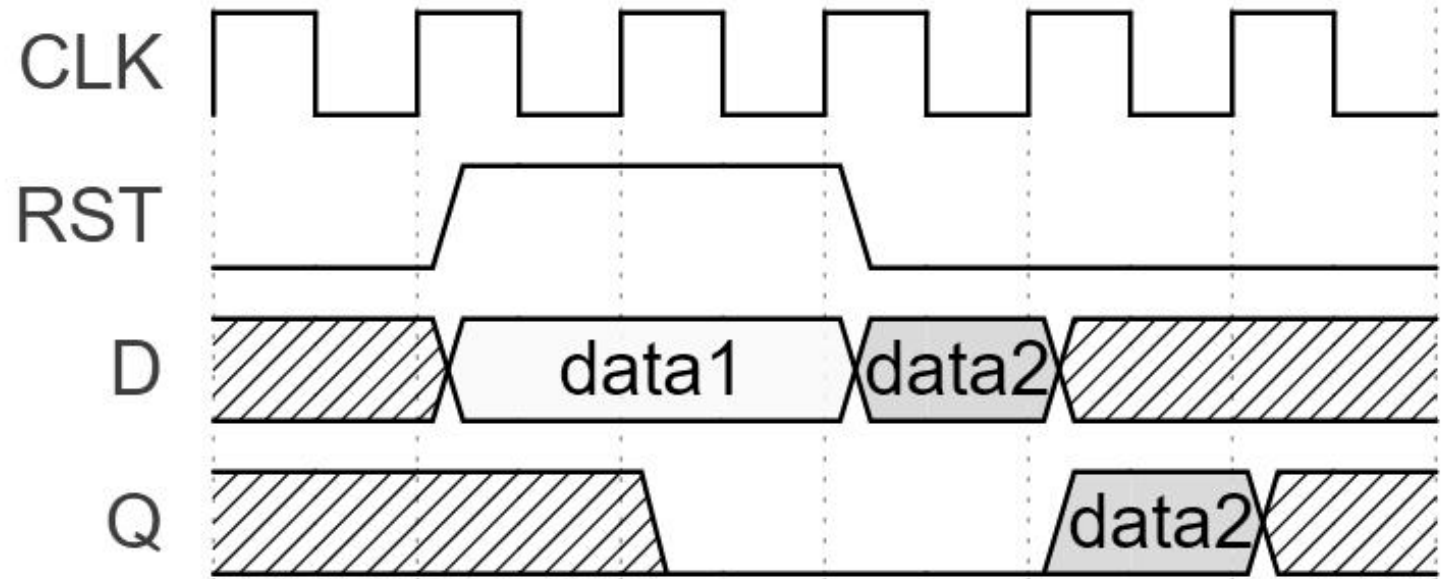
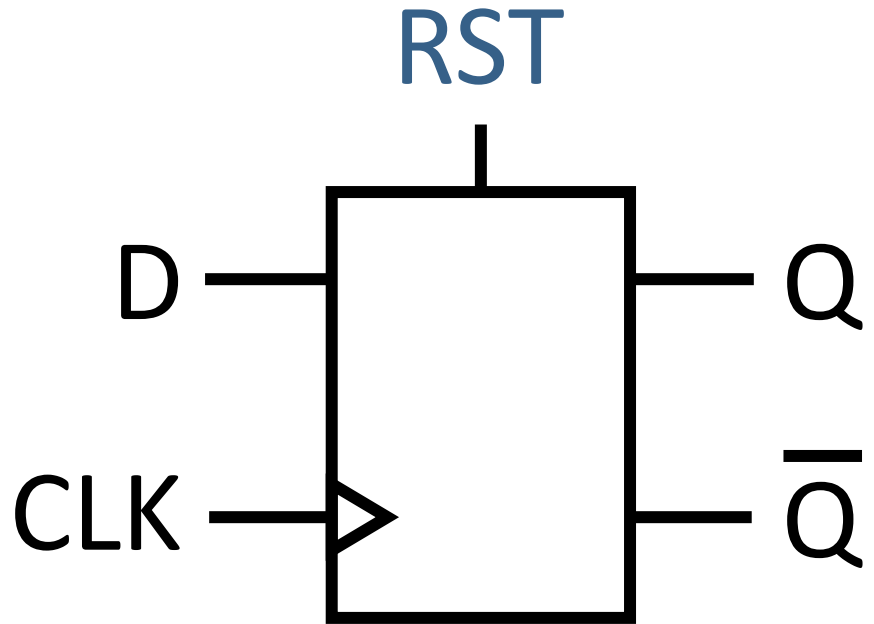
Последовательная логика



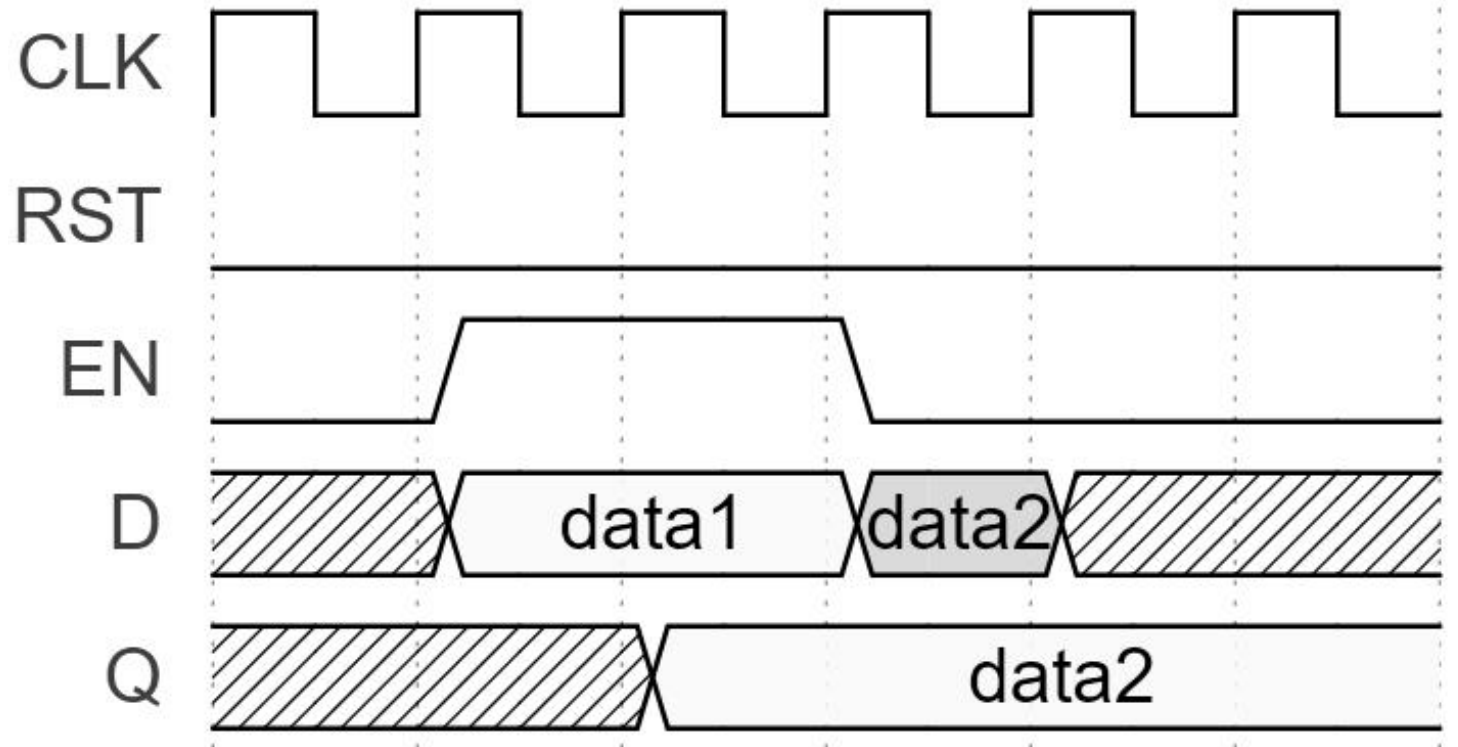
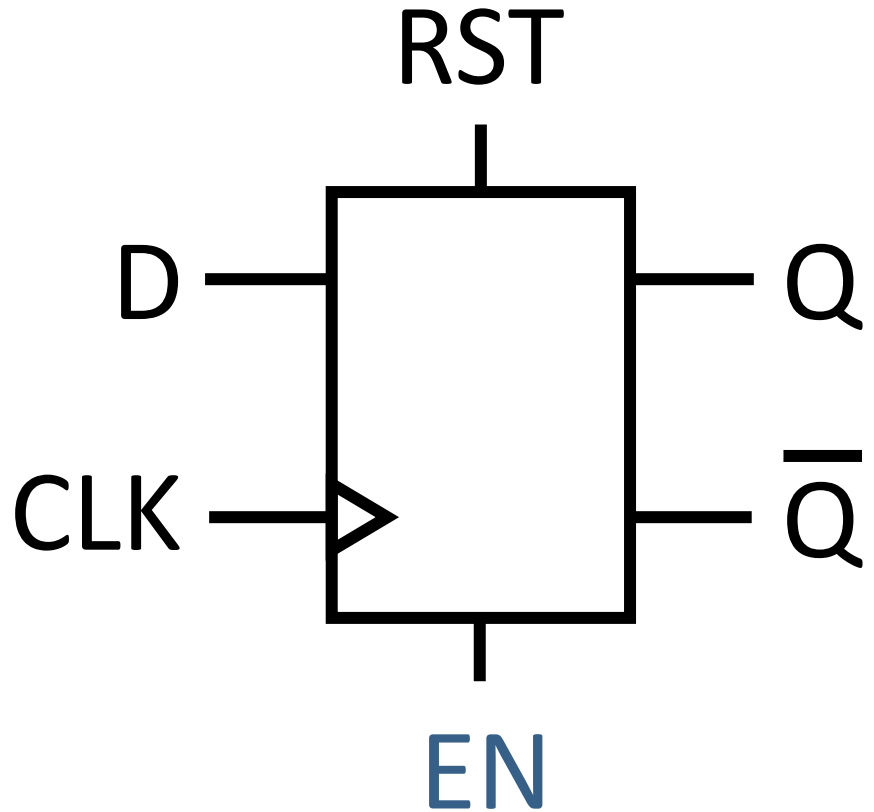
D-триггер



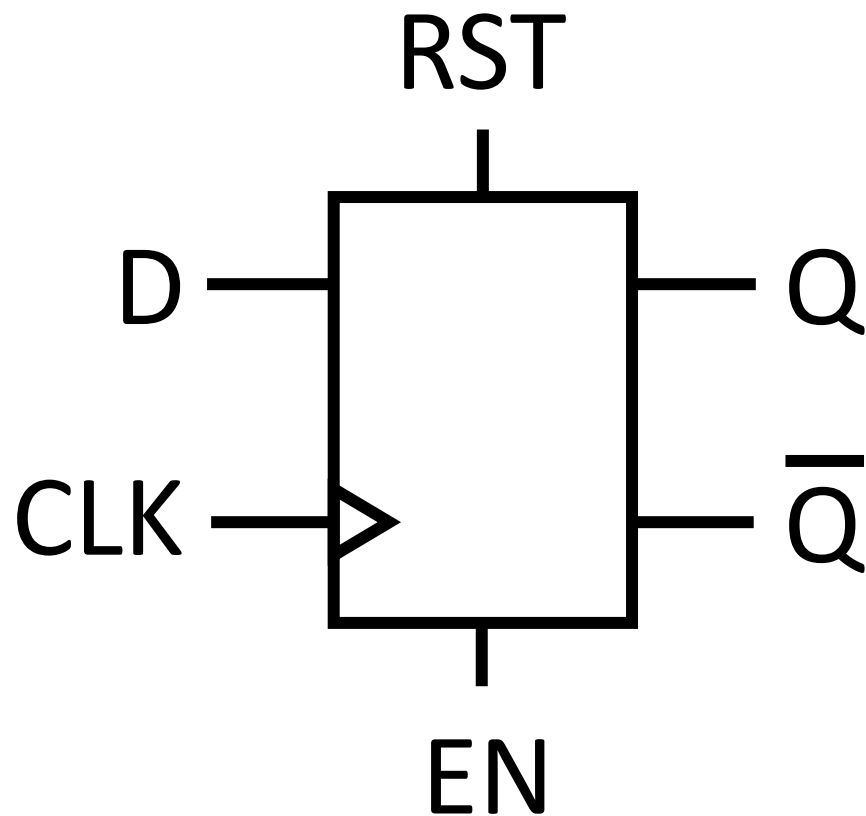
D-триггер. Сигнал сброса



D-триггер. Сигнал разрешения



D-триггер. Verilog HDL

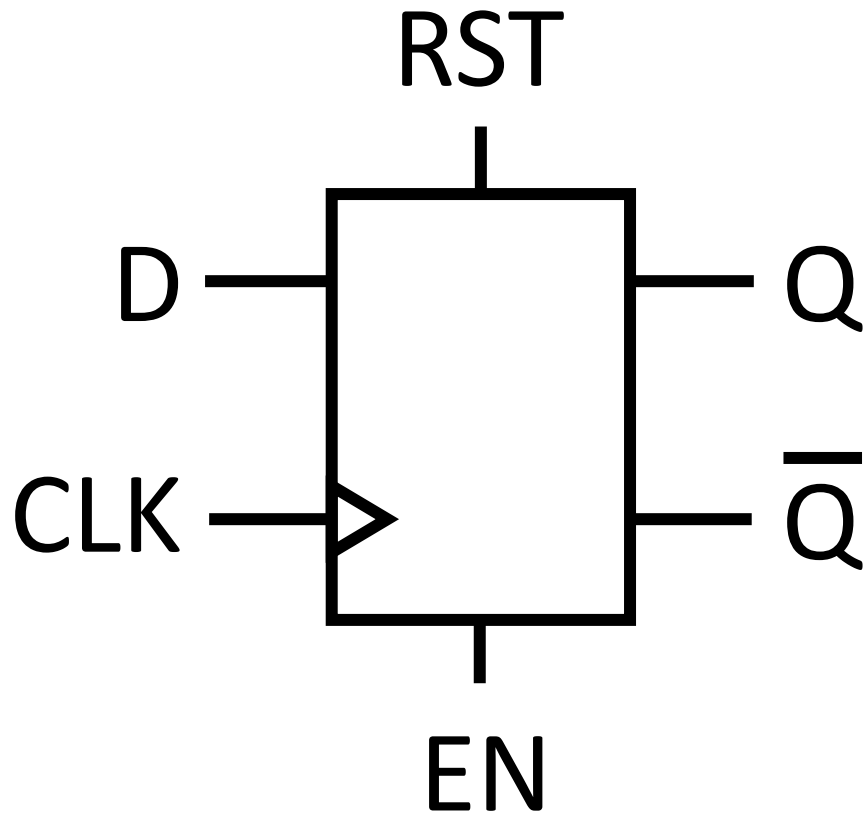


```
module my_reg
(
    input CLK,
    input RST,
    input EN,
    input D,
    output Q
);

reg OUT;
always @(posedge CLK) begin
    if(RST) OUT <= 1'b0; else OUT <= D;
end
assign Q = OUT;

endmodule
```


D-триггер. Новый синтаксис



```
module my_reg  
(
```

```
    input CLK,  
    input RST,  
    input EN,  
    input D,  
    output Q
```

```
);
```

```
    reg OUT;
```

```
    always @(posedge CLK) begin
```

```
        if(RST) OUT <= 1'b0; else OUT <= D;
```

```
    end
```

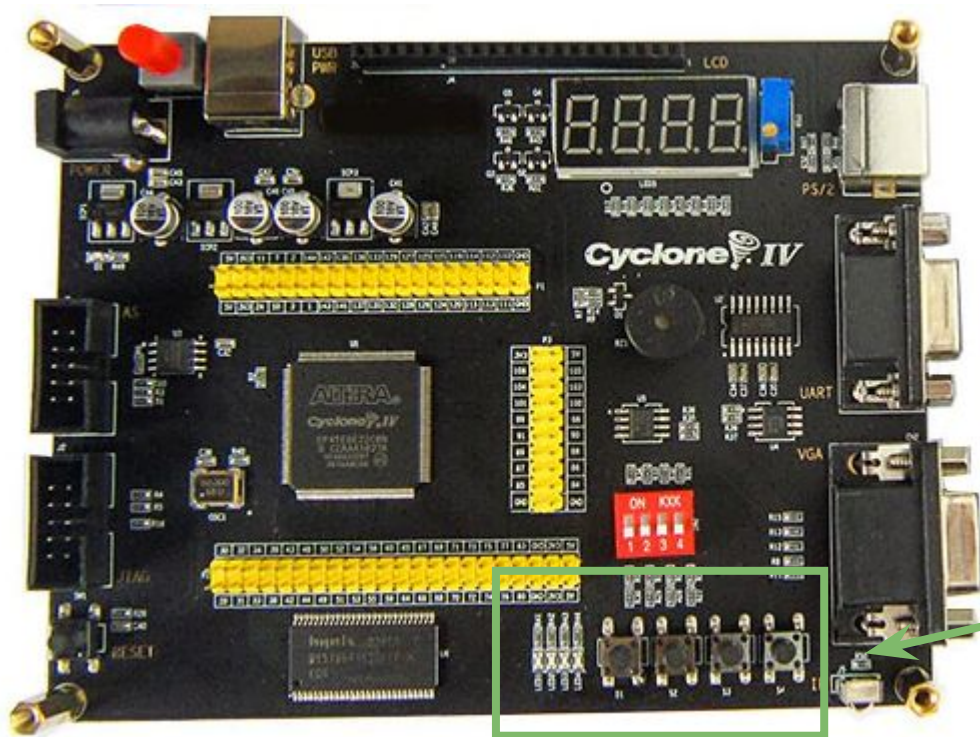
```
    assign Q = OUT;
```

```
endmodule
```

Использование
ключевого
слова reg

Использование
конструкции
if-else

Упражнение со счетчиком



Altera Cyclone IV EP4CE6 FPGA

Вывод значений счетчика на светодиоды.
Использование
кнопок для изменения направления счета.

Упражнение со счетчиком

```
module top
(
  ...
);
...
reg [31:0] cnt;
always @ (posedge clk or posedge reset)
  if (reset)
    cnt <= 32'b0;
  else
    cnt <= cnt + 32'b1;

assign led = ~ cnt [27:24];

endmodule
```

```
module top
(
  ...
);
...
wire key = key_sw [0];

reg key_r;
always @ (posedge clk or posedge reset)
  if (reset)
    key_r <= 1'b0;
  else
    key_r <= key;

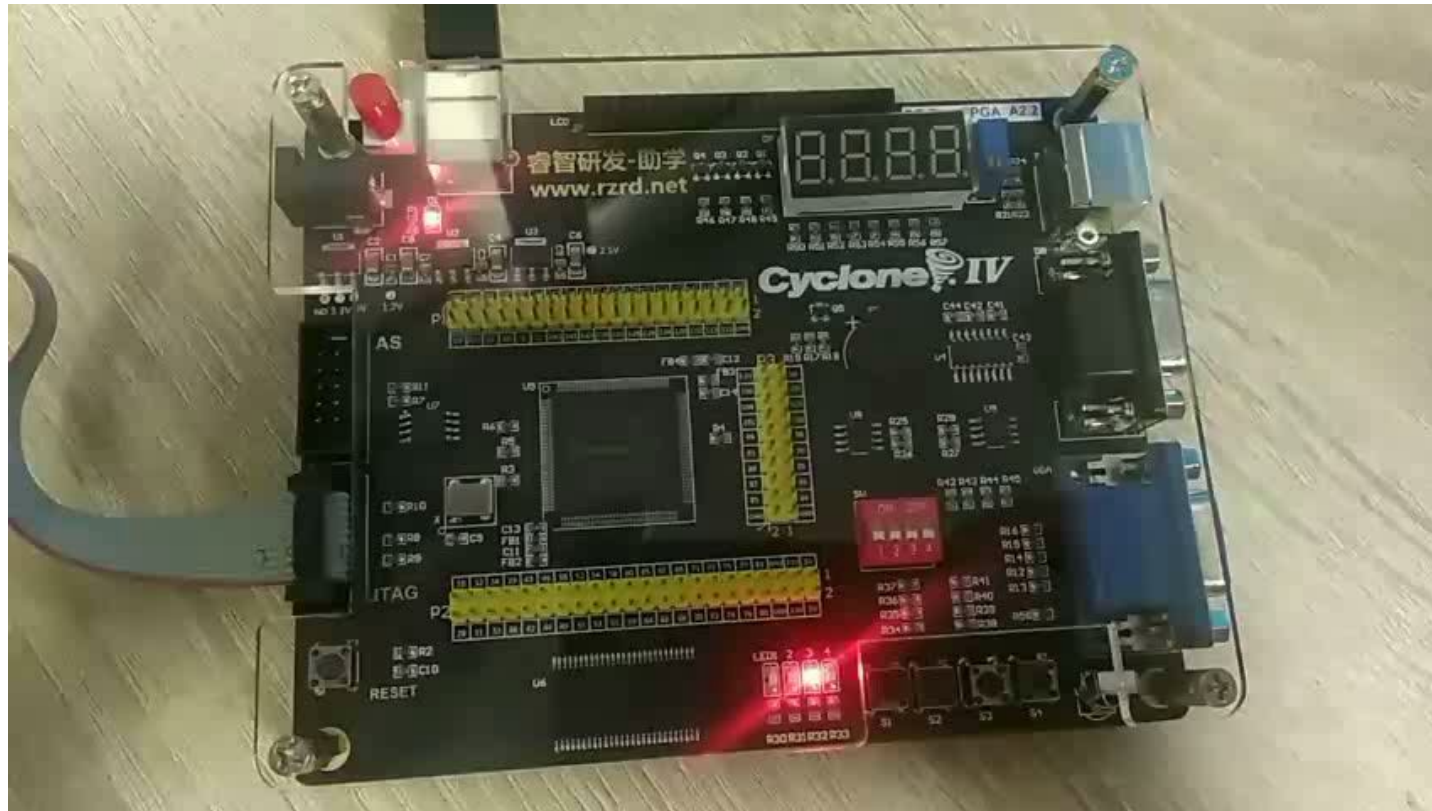
wire key_pressed = ~ key & key_r;

endmodule
```

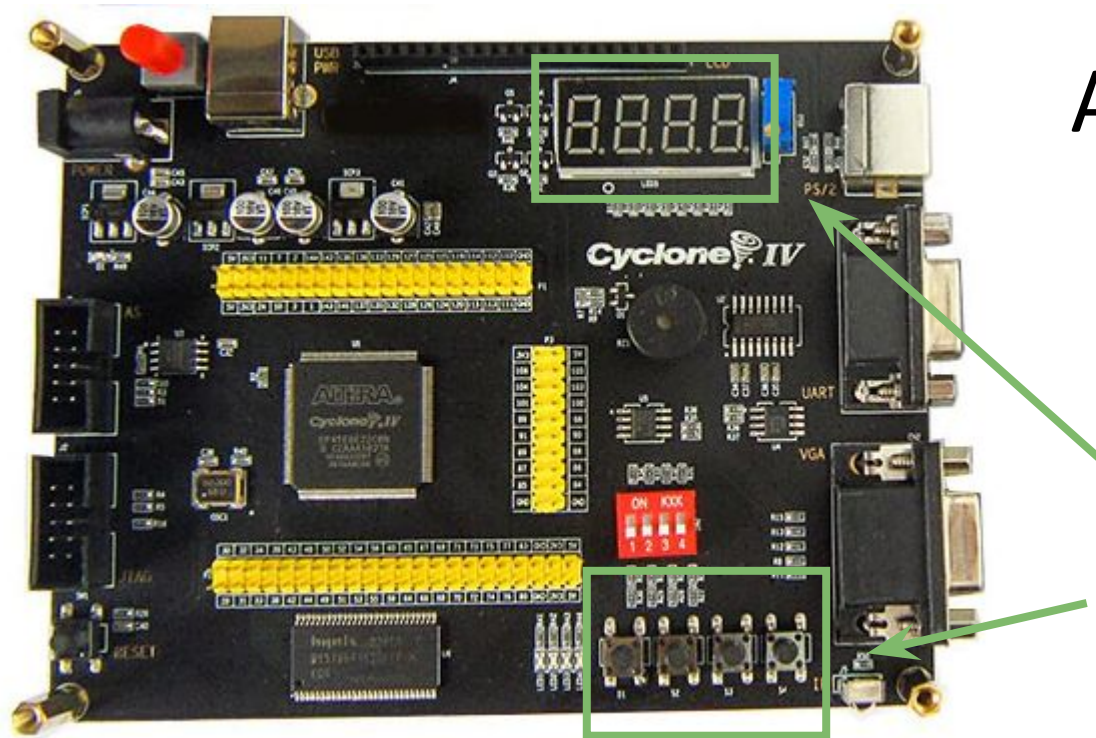
Упражнение со счетчиком

1. Свободно запустите счетчик. Как изменить скорость мигания светодиодов?
2. Измените дизайн. Например, добавьте управление направлением счета кнопками.

Упражнение со счетчиком



Упражнение со сдвиговым регистром



Altera Cyclone IV EP4CE6 FPGA

Управление мерцанием светодиодов и семисегментного индикатора при помощи последовательной логики и воздействий на кнопки управления на плате.

Упражнение со сдвиговым регистром

```
module top
(
  ...
);
...
reg [31:0] cnt;
always @ (posedge clk or posedge reset)
  if (reset)
    cnt <= 32'b0;
  else
    cnt <= cnt + 32'b1;
wire enable = (cnt [22:0] == 23'b0);

endmodule
```

```
module top
(
  ...
);
...
wire button_on = ~ key_sw [0];
reg [3:0] shift_reg;
always @ (posedge clk or posedge reset)
  if (reset)
    shift_reg <= 4'b0;
  else if (enable)
    shift_reg <= { button_on, shift_reg [3:1] };

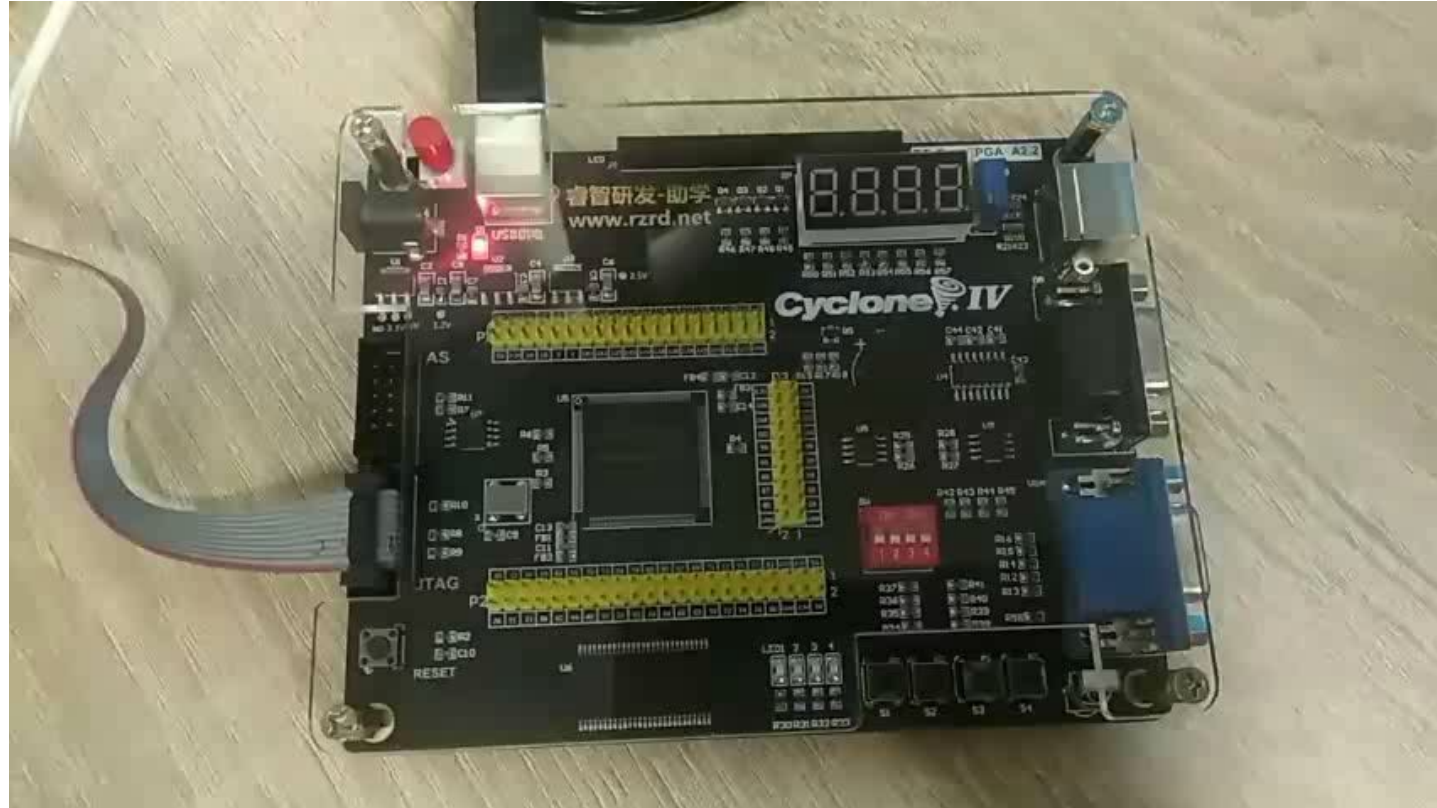
assign led = ~ shift_reg;

endmodule
```

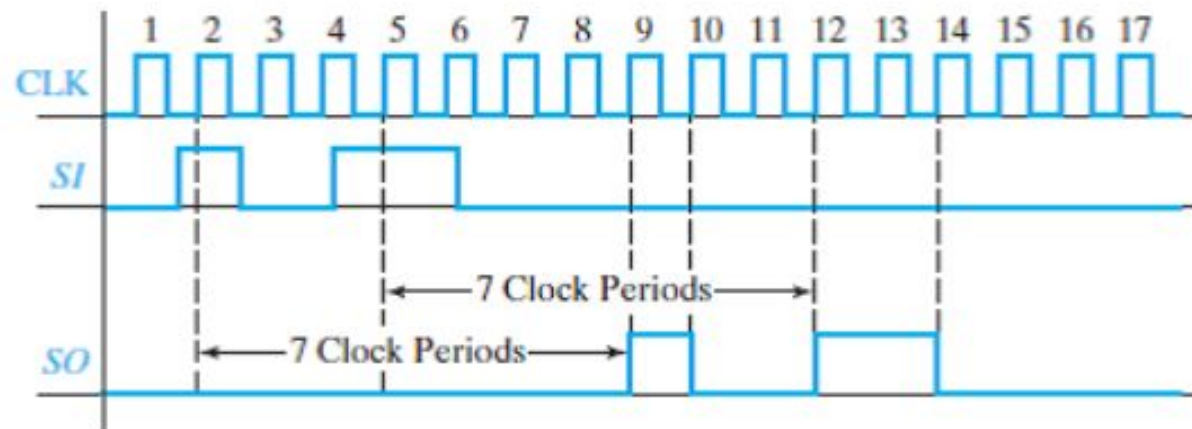
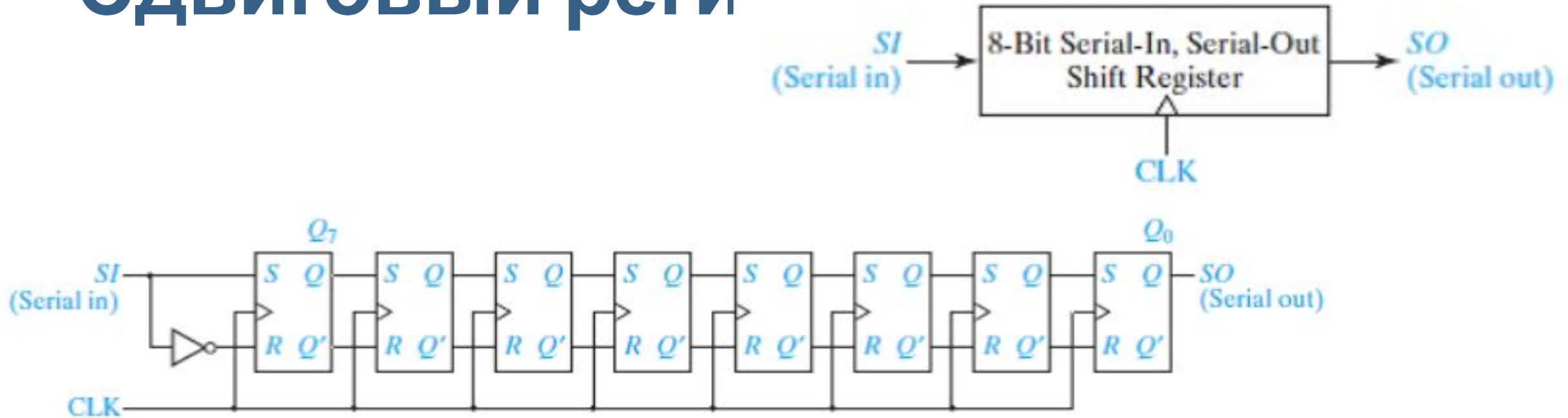
Упражнение со сдвиговым регистром

1. “Заставьте” светодиоды изменить направление мерцания.
2. Зациклите мерцание светодиодов.
3. Измените состояние семисегментного индикатора, “заставив” его поочередно зажигать светодиоды по кругу.

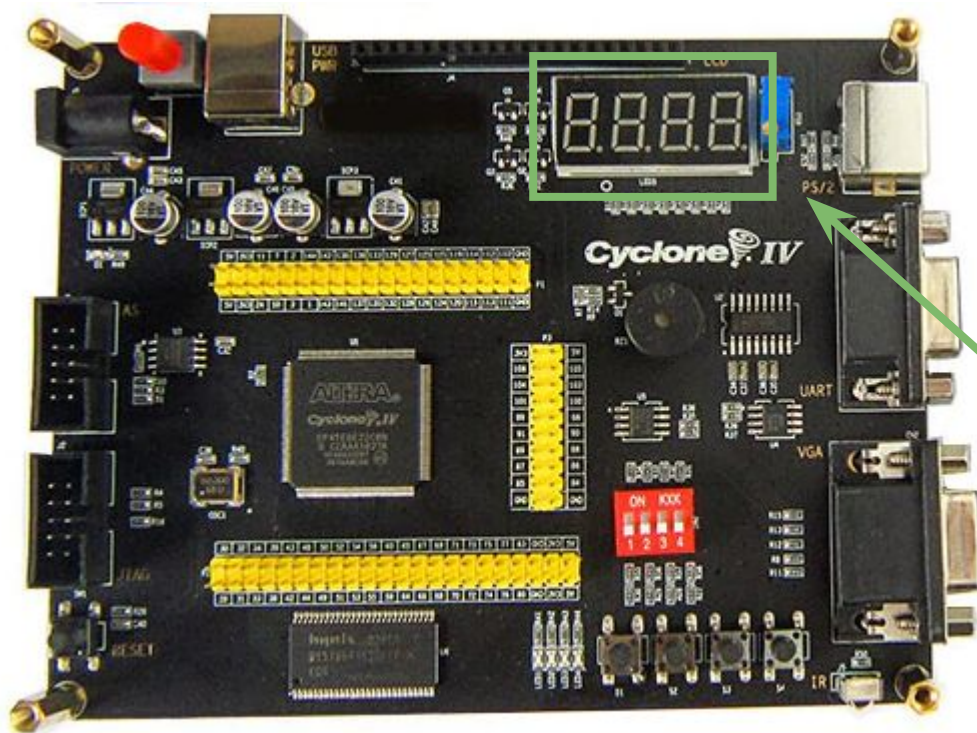
Упражнение со сдвиговым регистром



Сдвиговой реги



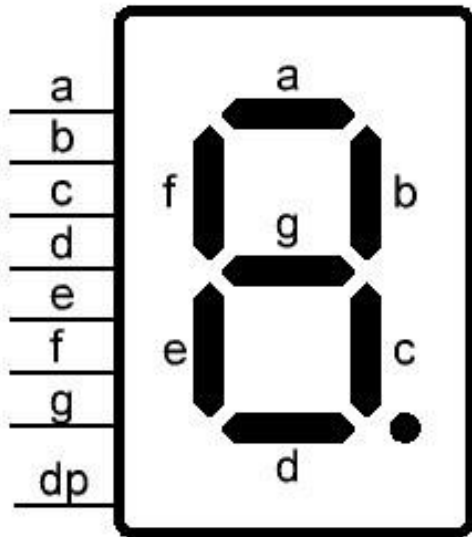
Упражнение: вывод слова на семисегментный индикатор



Altera Cyclone IV EP4CE6 FPGA

Вывод слова на семисегментный индикатор при помощи последовательной логики.

Семисегментный индикатор



Dec	4-bit шина				7-сегментный индикатор						
	3	2	1	0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Упражнение: вывод слова на семисегментный индикатор

```
module top
(
  ...
);
...
reg [7:0] letter;
always @*
  case (shift_reg)
    4'b1000: letter = C;
    ...
  default: letter = E;
  endcase

  assign abcdefgh = letter;
endmodule
```

```
module top
(
  ...
);
...
reg [31:0] cnt;

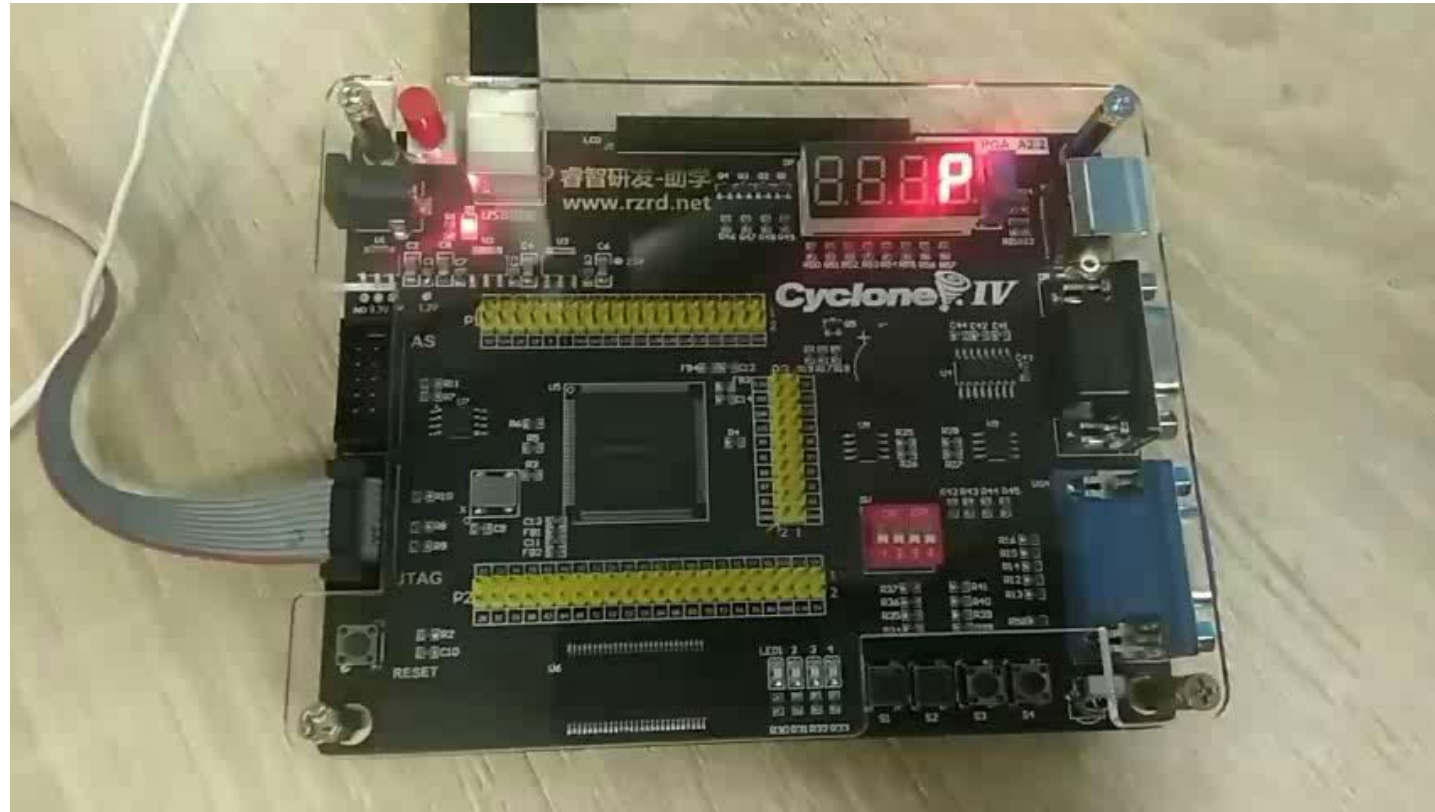
always @ (posedge clk or posedge reset)
  if (reset)
    cnt <= 32'b0;
  else
    cnt <= cnt + 32'b1;

  wire enable = (cnt [22:0] == 23'b0);
endmodule
```

Упражнение со сдвиговым регистром

1. Увеличьте частоту сигнала `enable`, чтобы текст на семисегментном индикаторе был легко читаем и не мерцал.
2. Выведите на семисегментном индикаторе любое другое слово.
3. Закомментируйте строчку со словом `default` в конструкции `case-endcase` и попробуйте заново синтезировать дизайн; постарайтесь объяснить появившееся сообщения САПРа.

Упражнение: вывод слова на семисегментный индикатор



Спасибо за внимание.