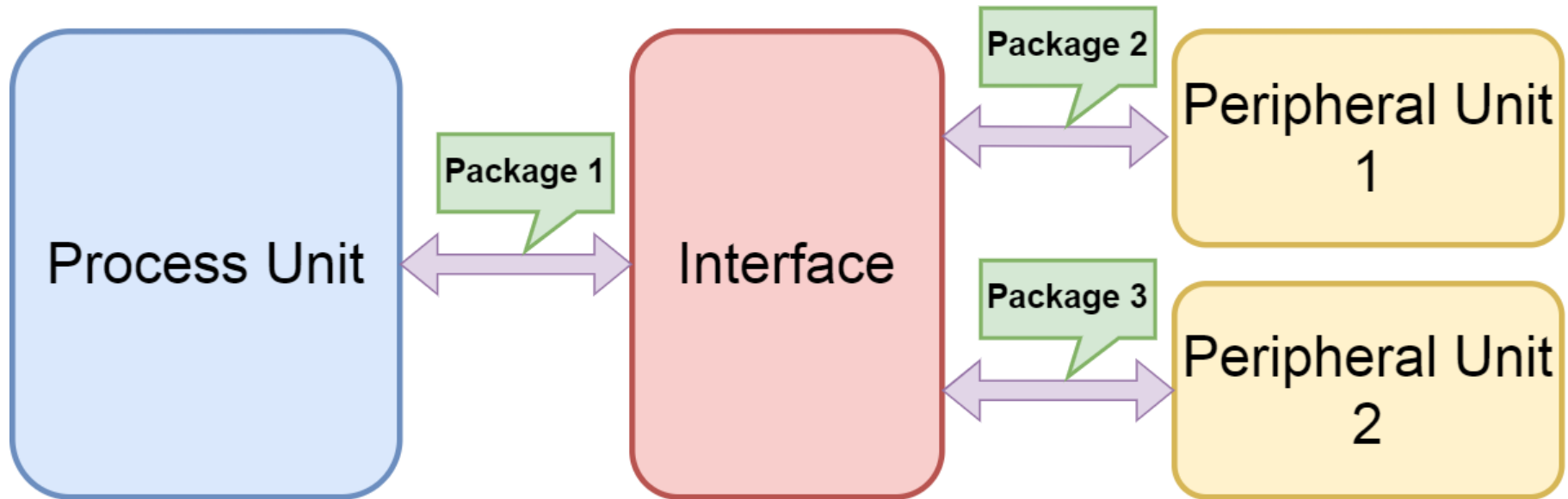


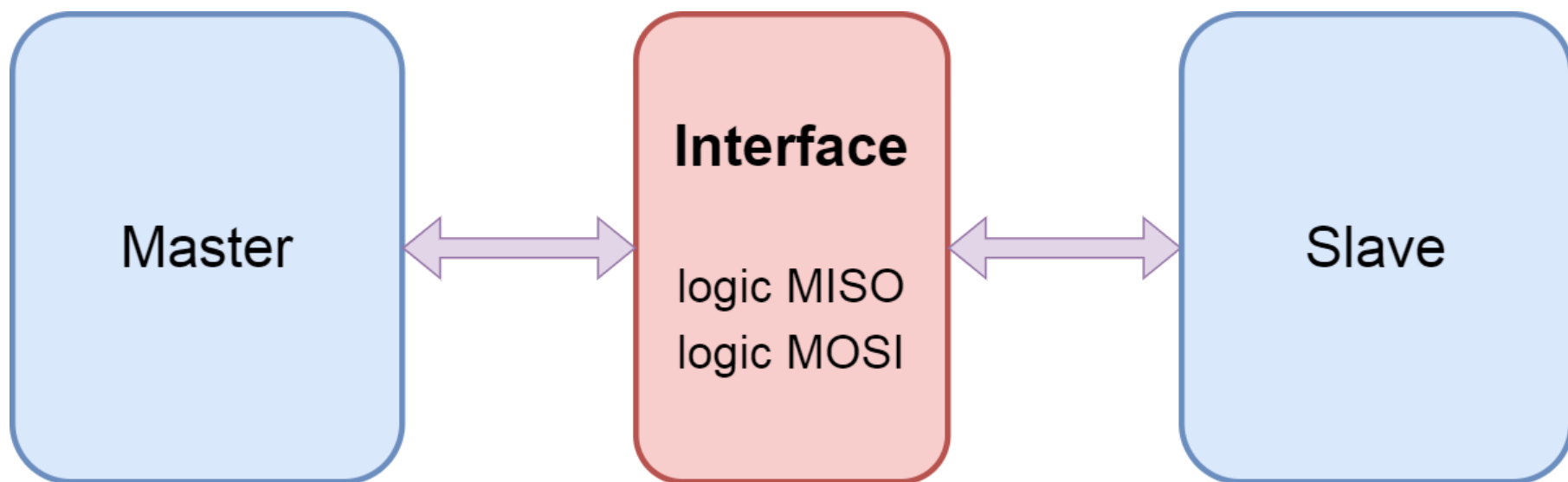
ПРОЕКТИРОВАНИЕ СНК С ПРОГРАММИРУЕМОЙ АРХИТЕКТУРОЙ

Лабораторная работа №2

Интерфейс



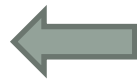
Конструкция интерфейса modport



Конструкция интерфейса modport

```
1 interface SPI(  
2     input clk,  
3     input rst  
4 );  
5  
6 logic MOSI;  
7 logic MISO;  
8 logic SS;  
9  
10 modport master(  
11     input  clk,  
12     input  rst,  
13     input  MISO,  
14     output SS,  
15     output MOSI  
16 );  
17  
18 modport slave(  
19     input  clk,  
20     input  rst,  
21     input  SS,  
22     input  MOSI,  
23     output MISO  
24 );  
25  
26 endinterface : SPI
```

Пример описания



Без определения
modport будут как
inout

```
1 modport [identifier] (  
2     input [port_list],  
3     output [port_list]  
4 );
```

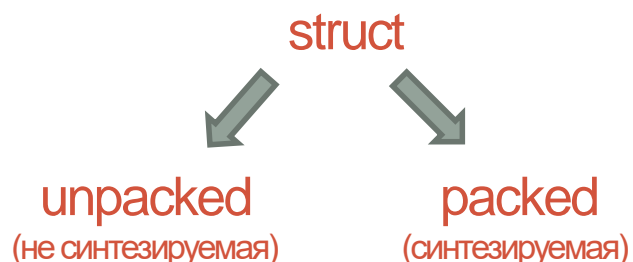
Синтаксис

```
1 module Slave(  
2     SPI.slave    spi_if  
3 );  
4  
5     /* some logic */  
6  
7 endmodule  
8  
9  
10 module TB;  
11  
12     /* some logic */  
13  
14     SPI my_spi(clk, rst);  
15     Slave udb_slave(spi_if.slave);  
16  
17 endmodule
```

Пример инстанцирования

Конструкция struct

Struct (структура) - позволяет нам создать группу из нескольких переменных (даже разного типа). На всю группу можно ссылаться как на одно целое, либо на отдельные её части можно ссылаться по имени.



```
1 struct [struct_type] {
2     [list_of_variables]
3 } struct_name;
```

Синтаксис конструкции

```
1 localparam YES = 1'b1,
2             NO  = 1'b0;
3
4 struct packed {
5     logic    [31:0] fruit_id;
6     bit      is_on_sale;
7     byte     expiry_date;
8 } apple, orange;
9
10 always_comb apple.expiry_date = 8'h7;
11 always_comb apple.is_on_sale  = YES;
12 always_comb apple.fruit_id    = 32'h0000_01FCD;
13
14 always_comb orange.expiry_date = 8'h9;
15 always_comb orange.is_on_sale  = NO;
16 always_comb orange.fruit_id    = 32'h0000_01FCE;
```

Пример использования

Конструкция typedef

В языке SystemVerilog можно создавать новые типы данных. Для этого используется конструкция typedef. В большинстве случаев мы просто используем typedef для присвоения имени объявлению типа, которое мы хотим использовать в нескольких местах вашего кода. Это полезно, так как мы можем создавать довольно сложные типы данных в SystemVerilog. Когда мы используем typedef вместо повторения сложного объявления типа, мы упрощаем наш код для понимания и поддержки.

```
1 | typedef data_type type_name [range];
```

Синтаксис конструкции

```
1 typedef struct packed {  
2     logic    [13:0]    calculation_result;  
3     logic    [1:0]     expression_type;  
4     logic    [7:0]     value_2;  
5     logic    [7:0]     value_1;  
6 } summ_register_t;
```

Пример использования

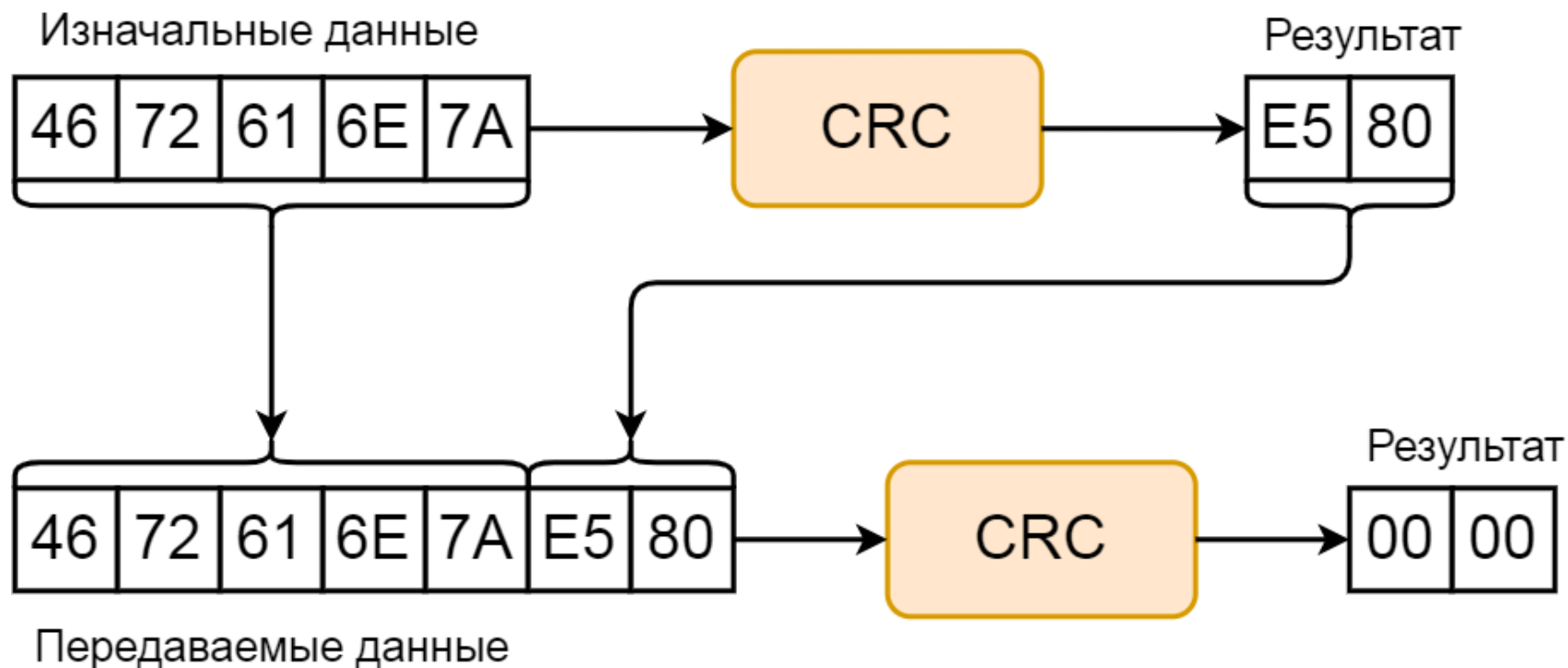
Конструкция package

Package реализует механизм хранения и передачи структур, методов, переменных, параметров и других конструкций HDL SystemVerilog между модулями для дальнейшего использования.

```
1 package registers_pkg;
2
3     typedef struct packed {
4         logic    [13:0] calculation_result;
5         logic    [1:0]  expression_type;
6         logic    [7:0]  value_2;
7         logic    [7:0]  value_1;
8     } summ_register_t;
9
10    typedef enum logic [1:0] {
11        IDLE,
12        PROCESSING,
13        FINISH
14    } fsm_calc_state_t;
15
16
17 endpackage : registers_pkg
```

```
1 module register_example(
2     APB3.Slave  APB3,
3 );
4
5     import registers_pkg::*;
6
7     summ_registers_t    REG1;
8     summ_registers_t    REG2;
9
10    fsm_calc_state_t     fsm_state_current;
11    fsm_calc_state_t     fsm_state_next;
12
13    /* some logic */
14
15 endmodule : register_example
```

Проверка целостности, CRC



Проверка целостности, CRC

Делимое
(Исходное сообщение)

Делитель
(Полином)

Частное
(Не используется)

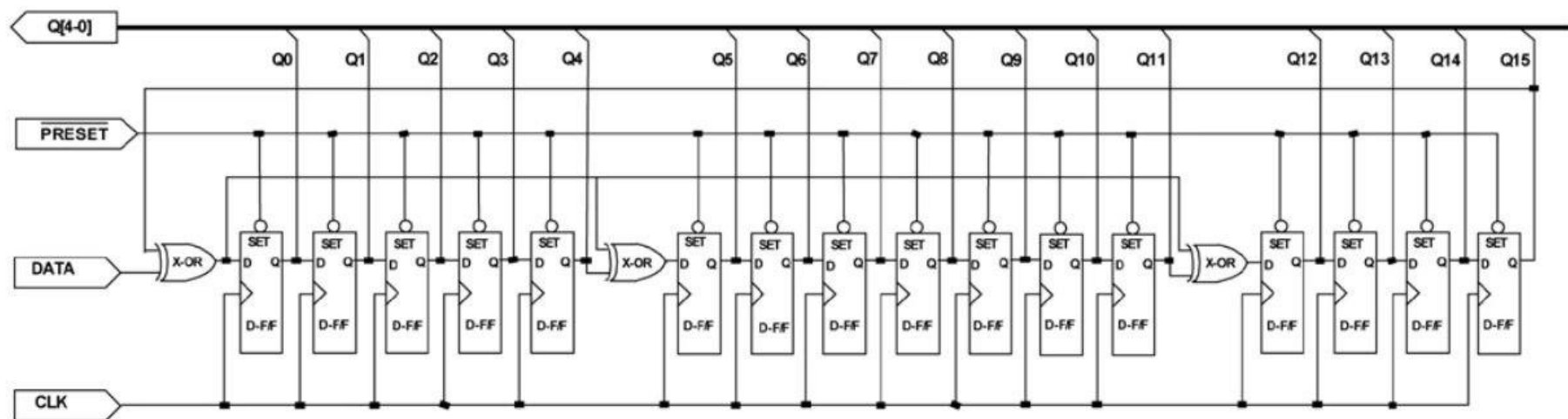
Остаток
(Результат CRC)

The diagram illustrates the CRC calculation process using polynomial division. The dividend (11000100110101) is divided by the divisor (10011). The quotient (110100011) is noted as not being used. The final remainder (000111) is the CRC result.

1 1 0 0 0 1 0 0 1 1 0 1 0 1														1 0 0 1 1					
⊕	1 0 0 1 1																		
	0 1 0 1 1 1																		
⊕	1 0 0 1 1																		
	0 0 1 0 0 0 0																		
⊕	1 0 0 1 1																		
	1 1 1 1 0																		
⊕	1 0 0 1 1																		
	1 1 0 1 1																		
⊕	1 0 0 1 1																		
	0 1 0 0 0 0																		
⊕	1 0 0 1 1																		
	0 0 0 1 1 1																		

CRC, реализация

Полином: $X^{16} + X^{12} + X^5 + 1$



CRC, реализация

Полином: $X^{16} + X^{12} + X^5 + 1$

```
1  always @ (posedge clk or negedge nrst)
2  if (!nrst)
3      crc      <= 16'h0000;
4  else begin
5      crc[0]    <= ~(crcbitin ^ ~crc[15]);
6      crc[1]    <= crc[0];
7      crc[2]    <= crc[1];
8      crc[3]    <= crc[2];
9      crc[4]    <= crc[3];
10     crc[5]    <= ~(~crc[4] ^ crcbitin ^ ~crc[15]);
11     crc[6]    <= crc[5];
12     crc[7]    <= crc[6];
13     crc[8]    <= crc[7];
14     crc[9]    <= crc[8];
15     crc[10]   <= crc[9];
16     crc[11]   <= crc[10];
17     crc[12]   <= ~(~crc[11] ^ crcbitin ^ ~crc[15]);
18     crc[13]   <= crc[12];
19     crc[14]   <= crc[13];
20     crc[15]   <= crc[14];
21  end
```

Цели лабораторной работы

- 1) Изучить вышеописанные конструкции, применить их в дальнейших задачах;
- 2) Дополнить модуль, написанный в л/р1 логикой адресов регистров, привязать к ним вычисление CRC-16 в соответствии с вариантом;
- 3) Сделать тестбенч для вашего модуля, продемонстрировать процессы записи и чтения, вычисления CRC, корректную работу флагов.

Проверить результаты вычисления CRC-16 можно тут: <https://crccalc.com>

Требования по адресной карте (адреса регистров и их содержание) смотрите в LabWork2.doc

Задание на Л/Р (Варианты CRC-16)

№	CRC-16 Тип	Полином	Начальное значение	RefIn	RefOut	XorOut
1	MAXIM	0x8005	0x0000	TRUE	TRUE	0xFFFF
2	CCITT-FALSE	0x1021	0xFFFF	FALSE	FALSE	0x0000
3	ARC	0x8005	0x0000	TRUE	TRUE	0x0000
4	AUG-CCITT	0x1021	0x1D0F	FALSE	FALSE	0x0000
5	BUYPASS	0x8005	0x0000	FALSE	FALSE	0x0000
6	CDMA2000	0xC867	0xFFFF	FALSE	FALSE	0x0000
7	DDS-110	0x8005	0x800D	FALSE	FALSE	0x0000
8	DECT-R	0x0589	0x0000	FALSE	FALSE	0x0001
9	DECT-X	0x0589	0x0000	FALSE	FALSE	0x0000
10	DNP	0x3D65	0x0000	TRUE	TRUE	0xFFFF
11	EN-13757	0x3D65	0x0000	FALSE	FALSE	0xFFFF