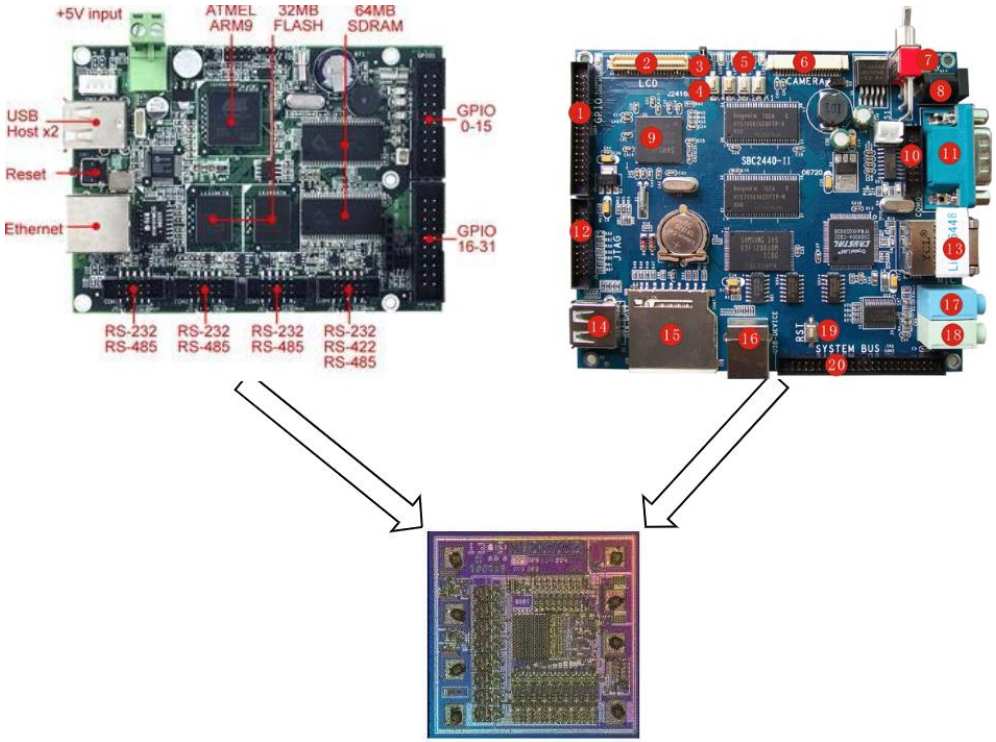
ЛАБОРАТОРНАЯ РАБОТА №1 ОСНОВЫ ПРОЕКТИРОВАНИЯ ВЕДОМЫХ МОДУЛЕЙ, ОБЛАДАЮЩИХ СИСТЕМНЫМ ИНТЕРФЕЙСОМ

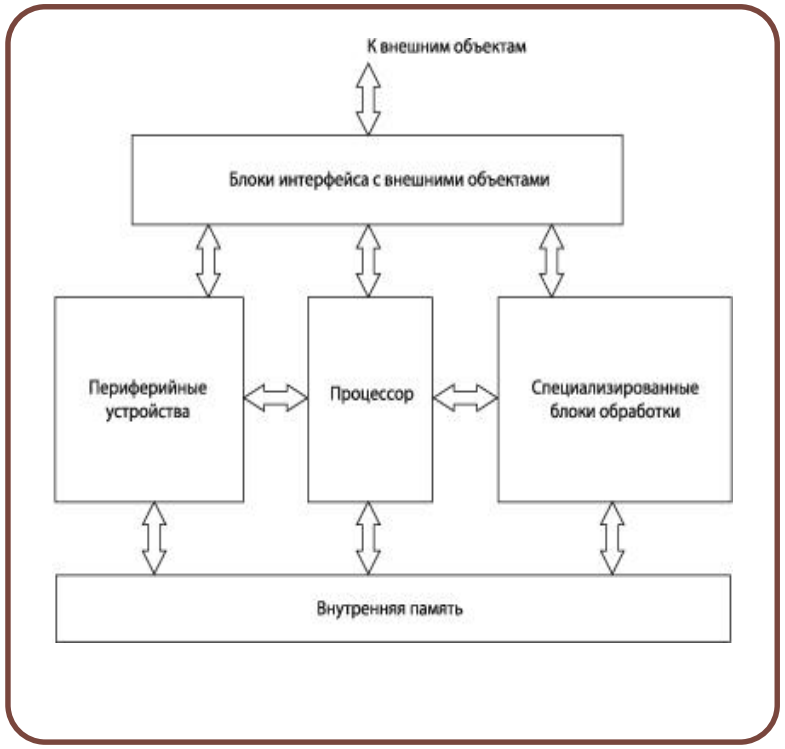
Курс «Проектирование СнК с программируемой архитектурой»

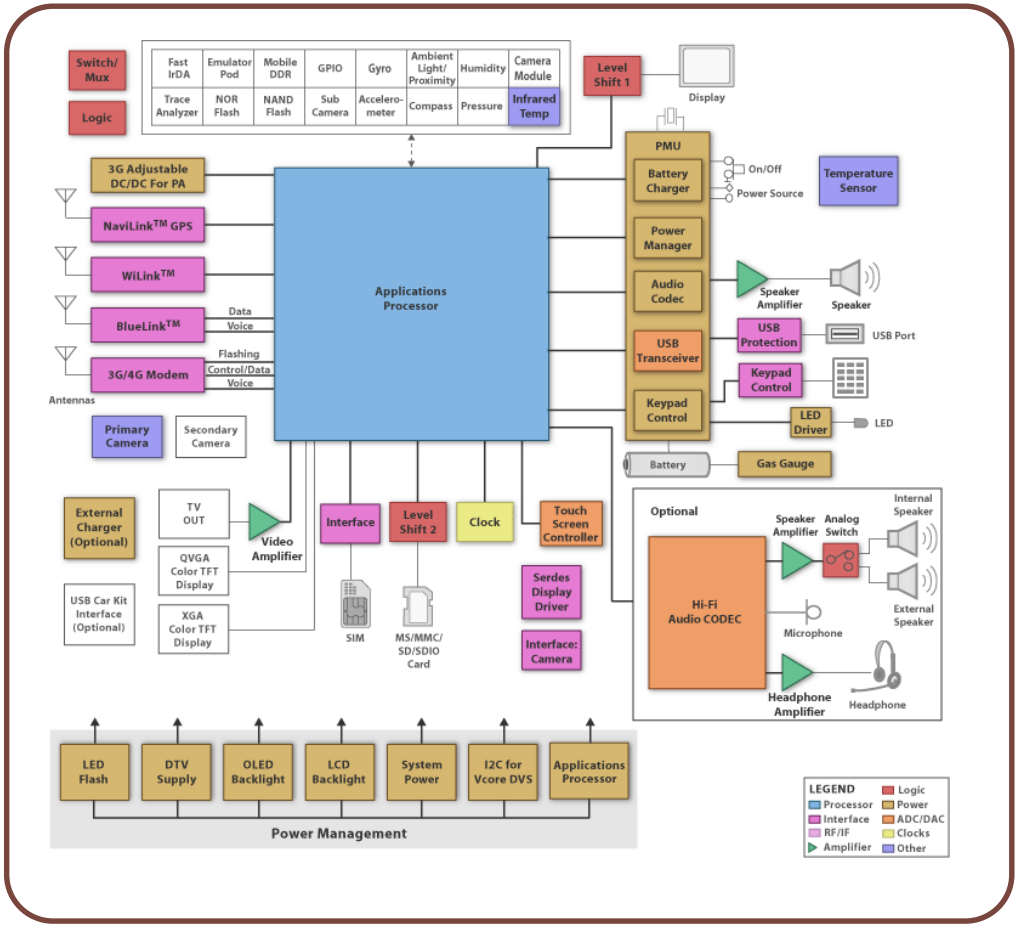
Цель разработки СнК



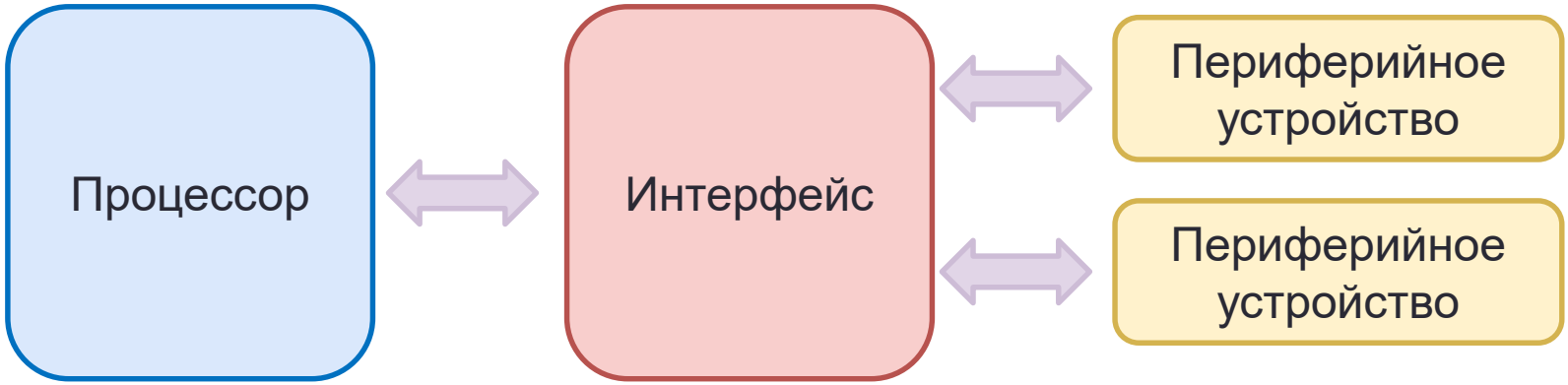
Системы на Кристалле (СнК, SoC) – являются одними из основных методик проектирования устройств в современной микроэлектронике, позволяющей разрабатывать гибкие, энергоэффективные и производительные решения различного назначения

Пример структуры типовой СнК





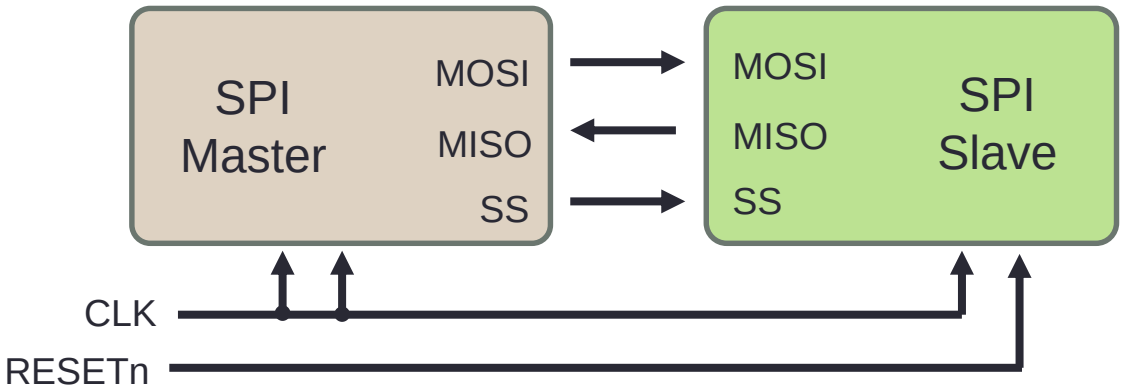
Структурное представление интерфейса



Interface (в SystemVerilog) – метод инкапсуляции сигналов в логическую группу с целью упрощения взаимодействия с необходимой группой сигналов.

Свойства:
- Параметризация;
- Наследуемость параметров;
- Могут быть внешние сигналы, как и module;
- Могут иметь явно заданное направление сигналов для разных устройств.

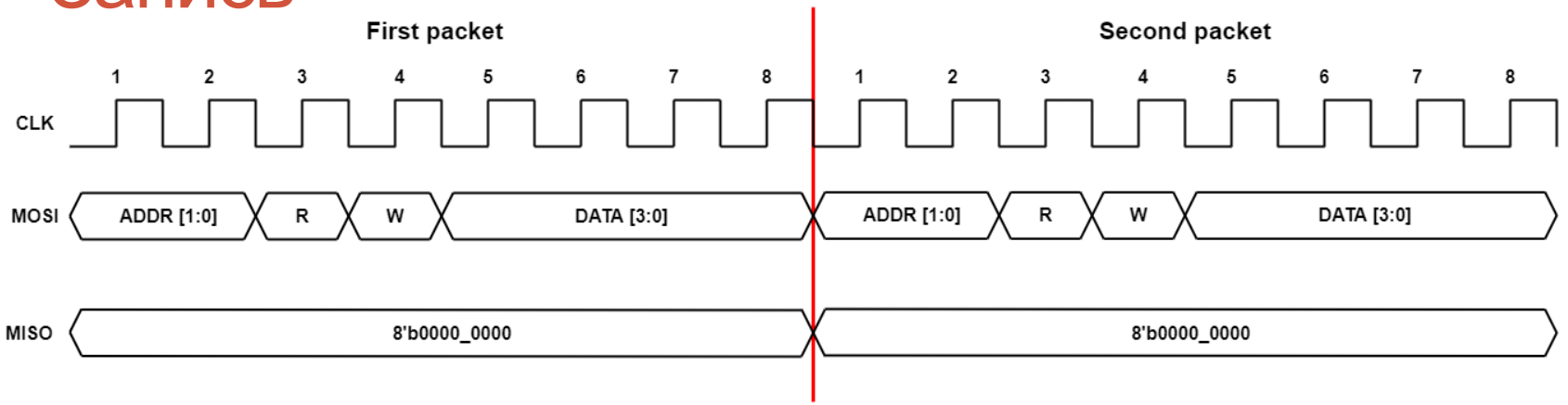
Интерфейс SPI



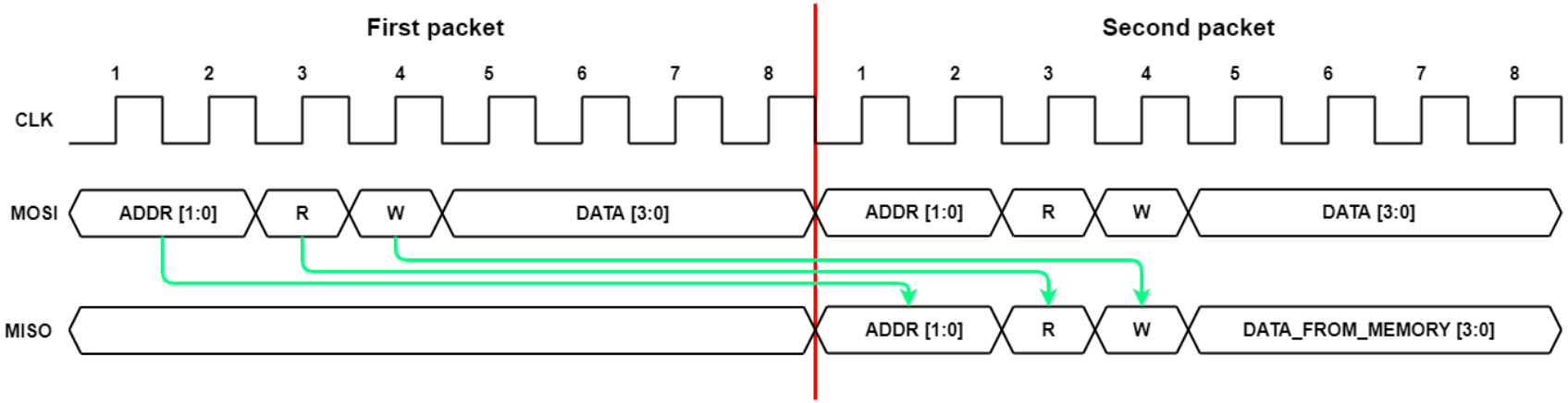
Название сигнала	Ведущий (Master)	Ведомый (Slave)	Описание
CLK	input	input	Служит для передачи тактового сигнала для ведомых устройств.
RESETn	input	input	Сброс, низкий активный уровень
MOSI	output	input	Служит для передачи данных от ведущего устройства ведомому.
MISO	input	output	Служит для передачи данных от ведомого устройства ведущему
SS	output	input	выбор ведомого

Протоколы интерфейса SPI

Запись



Чтение



Описание интерфейса SPI на SystemVerilog

Пример реализации интерфейса (без modport)

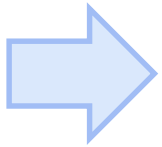
```
interface SPI_if# (
    parameter SLAVES_NUM = 2

)(
    input   CLK,
    input   RESETn
);
    /* --- Interface's signals --- */
    logic                       MOSI;
    logic                       MISO;
    logic   [SLAVES_NUM-1:0]    SS;

endinterface : SPI_if
```

Конструкция интерфейса синтаксически похожа на конструкцию module в HDL Verilog/SystemVerilog, обладает аналогичным механизмом параметризации.

```
interface SPI_if# (
    parameter SLAVES_NUM = 2

)(
    input   CLK,
    input   RESETn
);
    /* --- Interface's signals --- */
    logic                       MOSI;
    logic                       MISO;
    logic   [SLAVES_NUM-1:0]    SS;


    /* --- Master modport --- */
    modport Master(
        input CLK,
        input RESETn,
        output MOSI,
        output SS,
        input MISO
    );

    /* --- Slave modport --- */
    modport Slave(
        input CLK,
        input RESETn,
        input MOSI,
        input SS,
        output MISO
    );

endinterface : SPI_if
```

Использование interface в модулях

```
1  module Slave (
2      input CLK,
3      input RESETn,
4      input MOSI,
5      output MISO,
6      input SS
7  );
8
9      /* -- Slave logic --*/
10
11 endmodule : Slave
```



```
1  module Slave (
2      SPI_if SPI
3  );
4
5      /* -- Slave logic --*/
6
7  endmodule : Slave
```

Выше показано, как изменяется описание входных и выходных сигналов в модуле при использовании интерфейса.

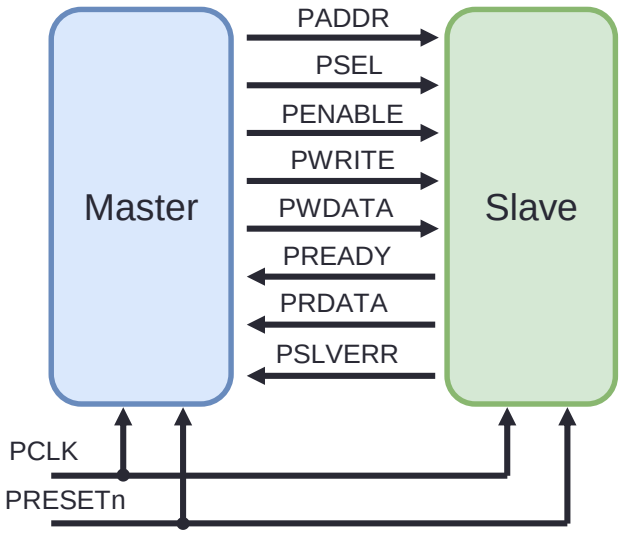
Использование interface в модулях

```
module Slave (
    SPI_if          SPI
);

logic [3:0]         cnt;
logic [7:0]         mosi_reg;
logic [7:0]         miso_reg;
logic [3:0] [3:0]   memory;


 /* --- Bit Counter --- */
always_ff@(negedge SPI.CLK or negedge SPI.RSTN)
if (!SPI.RSTN)
    cnt <= 4'hF;
else if (cnt <= 4'h7)
    cnt <= 4'h0;
else
    cnt <= cnt + 4'b1;

always_ff@(posedge SPI.CLK or negedge SPI.RSTN)
if (!SPI.RSTN)
    mosi_reg <= 8'h00;
else
    mosi_reg <= {mosi_reg[7:0], SPI.MOSI};

 /* --- Write Transaction --- */
always_ff@(posedge SPI.CLK or negedge SPI.RSTN)
if (!SPI.RSTN)
    memory                <= '0;
else if (cnt == 4'h7 && mosi_reg[4])
    memory[mosi_reg[7:6]] <= mosi_reg[3:0];

 /* --- Read Transaction --- */
always_ff@(posedge SPI.CLK or negedge SPI.RSTN)
if (!SPI.RSTN)
    miso_reg <= 8'h0;
else if (cnt == 4'h7 && mosi_reg[5])
    miso_reg <= {mosi_reg[7:4], memory[mosi_reg[7:6]]};
else
    miso_reg <= miso_reg << 1;

always_comb SPI.MISO = miso_reg[7];

endmodule : Slave
```

Использование interface в testbench

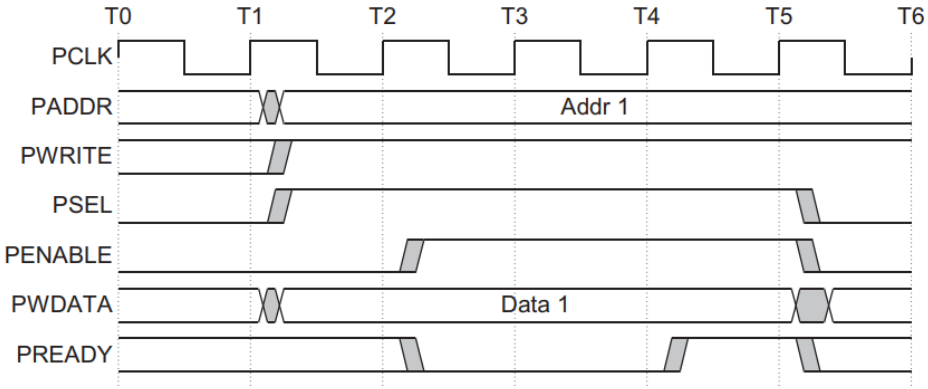
```
1   module tb;
2
3   logic           clk = '0;
4   logic           rstn = '0;
5   logic [7:0]     mosi_pkg;
6   logic [7:0]     miso_pkg
7   logic [7:0]     temp_reg;
8
9   SPI_if my_spi( .CLK(clk), .RSTN(rstn) );
10
11  Slave udb_slave( my_spi );
12
13  task spi_do (
14      input [1:0] addr,
15      input       r,
16      input       w,
17      input [3:0] data
18  );
19
20      my_spi.SS   = 1;
21      temp_reg    = {addr, r, w, data };
22
23      repeat(8) @(negedge clk) begin
24          my_spi.MOSI = temp_reg[7];
25          temp_reg    = temp_reg << 1;
26      end
27
28  endtask
29
30  initial begin
31      #5ns;
32      rstn = 1;
33      spi_do( 2'b01, 1'b0, 1'b1, 4'b1011 );
34      #15ns $stop;
35  end
36
37  always #10ns clk = ~clk;
38
39  endmodule : tb
40
```

Системный интерфейс AMBA APB3

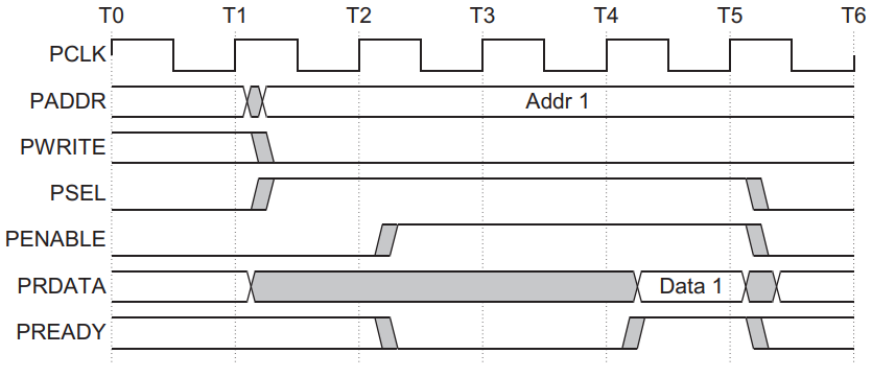


Название сигнала	Ведущий (Master)	Ведомый (Slave)	Описание
PCLK	input	input	Тактовый сигнал
PRESETn	input	input	Сброс, низкий активный уровень
PSEL	output	input	Выбор определенного slave устройства
PENABLE	output	input	Инициализация транзакции
PWDATA	output	input	Данные для записи
PWRITE	output	input	Строб записи (1 –запись, 0 -чтение)
PREADY	input	output	Сигнал окончания транзакции
PRDATA	input	output	Данные для чтения
PSLVERR	input	output	Сигнал наличия ошибки
PADDR	output	input	Адрес

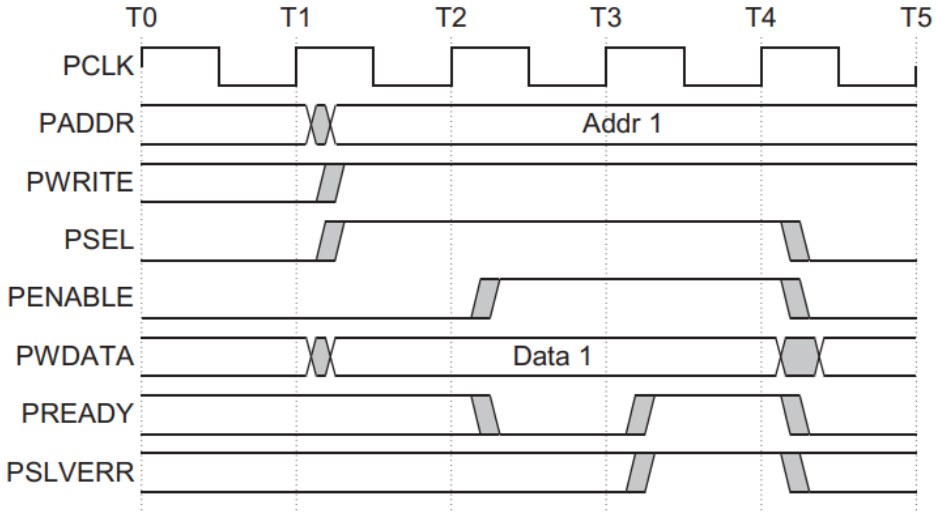
Протоколы интерфейса AMBA APB3



Транзакция записи APB3



Транзакция чтения APB3



Пример транзакции записи с ошибочным ответом

Лабораторное задание

1. Изучить применение конструкции interface в HDL SystemVerilog;
2. Изучить перечень и назначение сигналов интерфейса AMBA APB3 и его протокол для взаимодействия с модулями;
3. Самостоятельно описать interface для AMBA APB3;
4. Разработать APB3-Slave модуль, который будет выполнять следующие задачи через интерфейс APB3 в соответствии с протоколом:
    a. Записывать данные в регистр памяти, основываясь на адрес регистра;
    b. Считывать данные из регистра памяти, основываясь на адрес регистра.
5. Разработать тестовое окружение;
6. Провести моделирование в САПР Synopsys VCS.