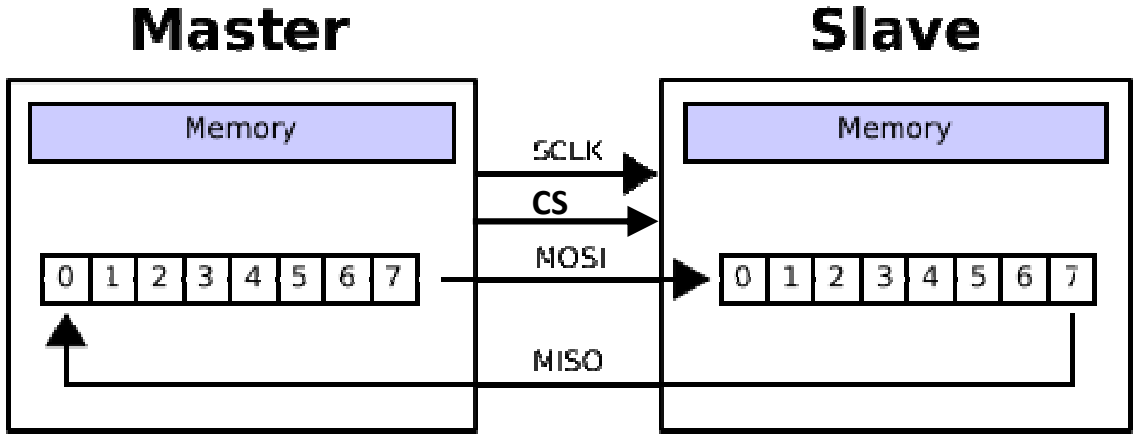
Serial Peripheral Interface

(przykład uogólniony)

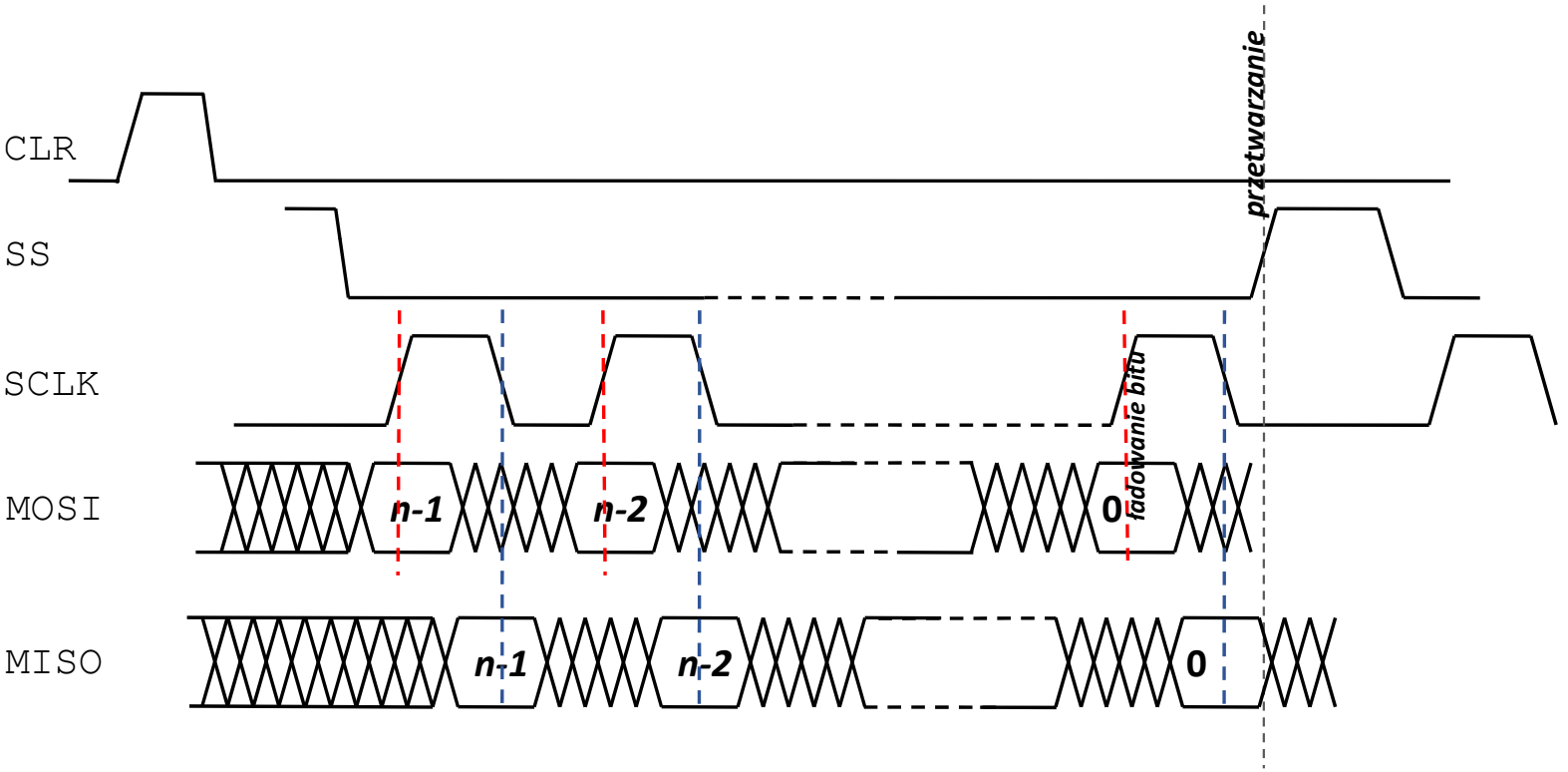
Brak standardu.
Inne stosowane nazwy: Synchronous Serial Port (SSP), 4-wire SSI (Synchronous Serial Interface, *Texas Instrument*), Microwire (*National Semiconductor*).

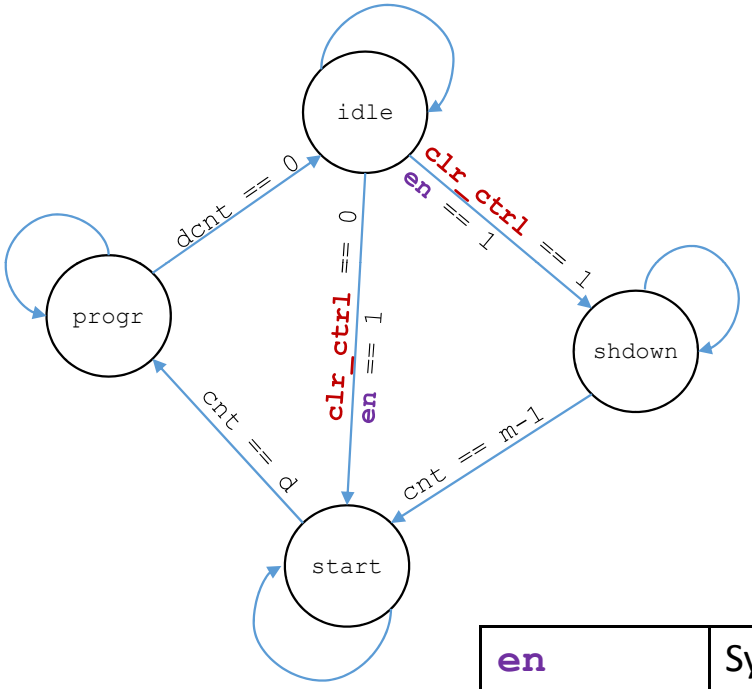
Poniżej opisano podstawowe cechy i przykładową implementację synchronicznego, szeregowego interfejsu z adresowaniem sprzętowym, przeznaczonego do komunikacji między układami scalonymi na płytkach drukowanych.

Laboratorium Języków Opisu Sprzętu AGH WFiIS

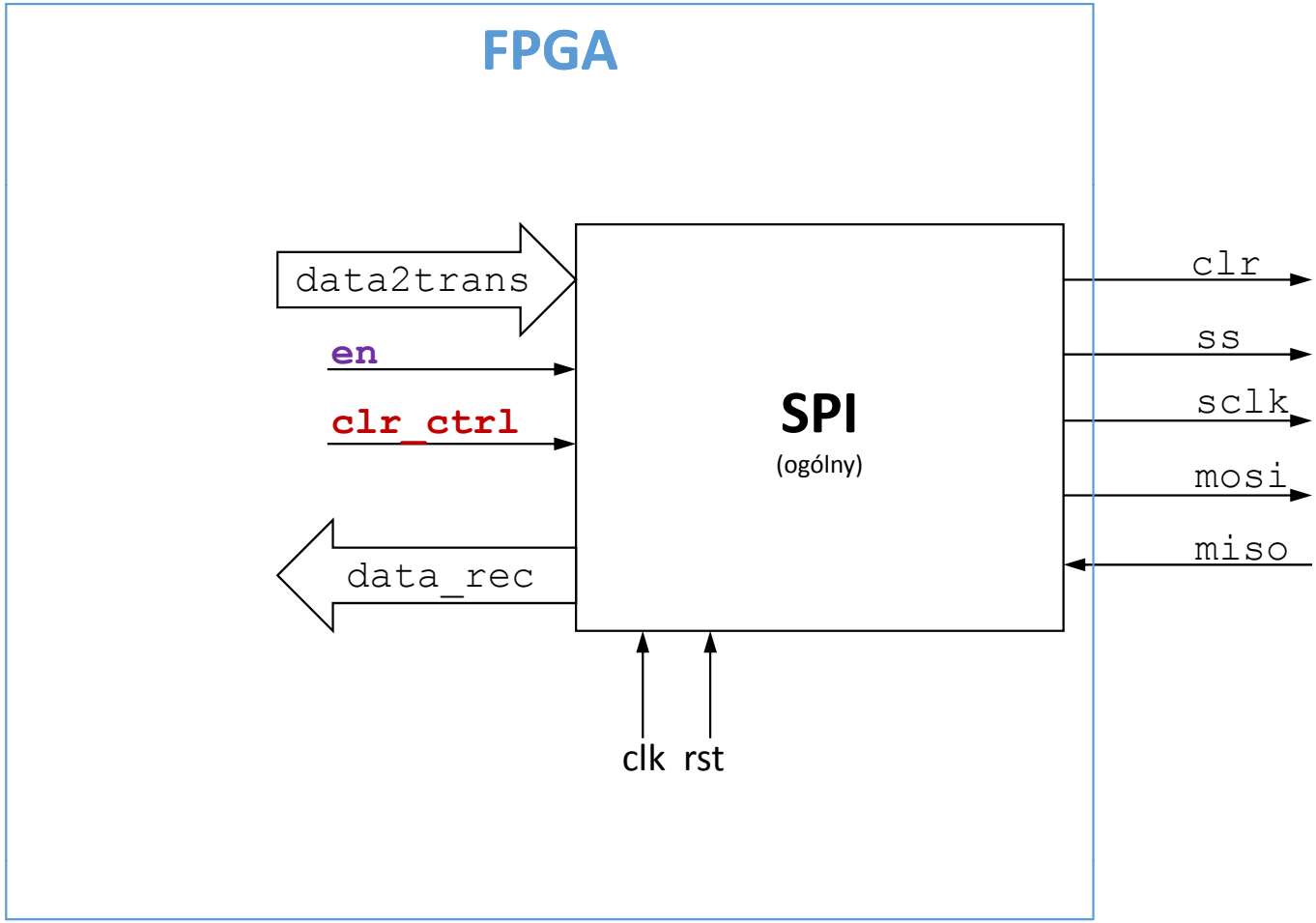


Sygnał	FPGA	Opis
SCLK	Serial Clock	Zegar; Na zboczu narastającym (lub opadającym) do układu Slave wpisywana jest zawartość linii danych MOSI, a na przeciwnym zboczu wysyłana na linię MISO jest zawartość rejestru wyjściowego układu Slave
SS	Slave Select	Wybór układu (chip select) aktywny stanem niskim.
MOSI	Master Output, Slave Input	Dane szeregowe; *n* bity muszą mieścić się pomiędzy zboczem opadającym, a narastającym sygnału SS zaczynając od najbardziej znaczącego.
MISO	Master Input, Slave Output	Dane szeregowe; *n* bitów, które wędrują od układu Slave do Układu Master.
Czasami także:		
CLR	Clear	Asynchroniczny reset.


CLR
SS
SCLK
MOSI
MISO
przetwarzanie
ładowanie bitu
n-1
n-2
0
n-1
n-2
0



en	Sygnał zapisu danej do rejestru `shr`
clr_ctrl	Sygnał zezwolenia kasowania
`cnt`	Licznik czasu trwania stanów
`dcnt`	Licznik bitów


FPGA
data2trans
en
clr_ctrl
data_rec
SPI
(ogólny)
clr
ss
sclk
mosi
miso
clk
rst

```
module spi #(parameter bits = 8) (input clk, rst, en, miso, clr_ctrl, input [bits-1:0] data2trans,
                                  output clr, ss, sclk, mosi, output reg [bits-1:0] data_rec);

`include "func.v"   //funkcja logarytmująca

//Parametry czasu trwania:
//m - czas jednego bitu (w połowie zbocze opadające sclk)
//d - opóźnienia na początku
//Parametry obliczane:
//bm - rozmiar licznika czasu
//bdcnt - rozmiar licznika bitów
localparam m = 5, d = 2, bm = clogb2(m), bdcnt = clogb2(bits);

//kodowanie stanów
localparam idle = 2'b00, shdown = 2'b01, progr = 2'b10, start = 2'b11;

/*fsm_encoding = "user"*/
reg [1:0] st, nst;

reg [bits-1:0] shr; //rejestr przesuwny
reg [bm-1:0] cnt;   //licznik czasu trwania stanów
reg [bdcnt:0] dcnt; //licznik bitów transmitowanych
reg tmp, tm, cnten;

//rejestr stanu
//logika automatu
//licznik czasu trwania stanów i poziomów zegara transmisji
//logika sygnałów wyjściowych
//generator zezwolenie dla rejestru przesuwnego
//licznik bitów
//rejestr przesuwny
//generator zezwolenia zapisu na wyjście
//rejestr wyjściowy

endmodule
```

```
//rejestr stanu
always @(posedge clk, posedge rst)
     if(rst)
          st <= idle;
     else
          st <= nst;

//logika automatu
always @* begin
     nst = idle;
     cnten = 1'b1;
     case(st)
          idle: begin
                    cnten = 1'b0;
                    nst = en? (clr_ctrl?shdown:start):idle;
               end
          shdown:  nst = (cnt == m-1)?start:shdown;
          start:   nst = (cnt == d)?progr:start;
          progr:   nst = (dcnt == {(bdcnt+1){1'd0}})?idle:progr;
     endcase
end
```

```
//licznik czasu trwania stanów
//i poziomów zegara transmisji
always @(posedge clk, posedge rst)
    if(rst)
        cnt <= {bm{1'b0}};
    else if(cnten)
        if(cnt == m | dcnt == {(bdcnt+1){1'd0}}) cnt <= {bm{1'b0}};
        else cnt <= cnt + 1'b1;
```

```
//logika sygnałów wyjściowych
assign clr = (st == shdown)?1'b1:1'b0;
assign ss = ((st == start) | (st == progr))?1'b0:1'b1;
assign sclk = ((st == progr) & (cnt < (m/2 + 1)))?1'b1:1'b0;
```

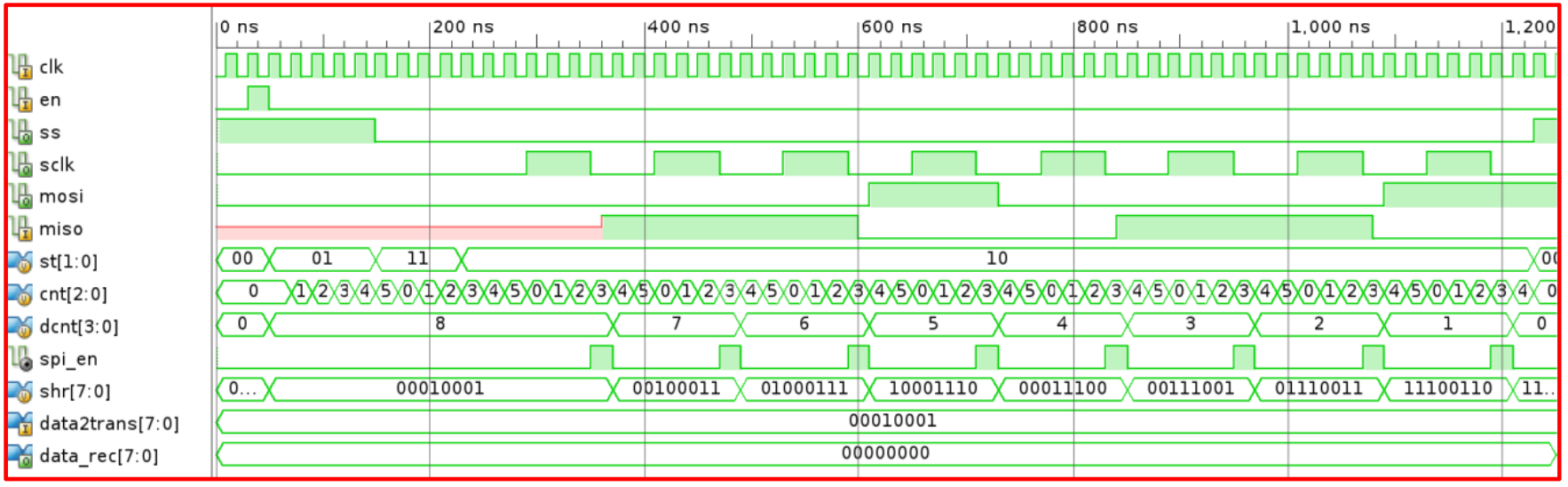
```
//generator zezwolenie dla rejestru przesuwnego
always @(posedge clk, posedge rst)
    if(rst)
        tmp <= 1'b0;
    else
        tmp <= sclk;
assign spi_en = ~sclk & tmp;

//licznik bitów
always @(posedge clk, posedge rst)
    if(rst)
        dcnt <= bits;
    else if(spi_en)
            dcnt <= dcnt - 1'b1;
        else
            if(en & dcnt == {(bdcnt+1){1'd0}}) dcnt <= bits;
```

```
//rejestr przesuwny
assign mosi = shr[bits-1];
always @(posedge clk, posedge rst)
    if(rst)
        shr <= {bits{1'b0}};
    else if(en)
            shr <= data2trans;
        else if(spi_en)
            shr <= {shr[bits-2:0],miso};
```

```
//generator zezwolenia zapisu na wyjście
always @(posedge clk, posedge rst)
    if(rst)
        tm <= 1'b0;
    else
        tm <= ss;
assign en_out = ss & ~tm;

//rejestr wyjściowy
always @(posedge clk, posedge rst)
    if(rst)
        data_rec <= {bits{1'b0}};
    else if(en_out)
        data_rec <= shr;
```



Parametry modułu:

Liczba transmitowanych bitów: bits = 8

Parametry lokalne:

Czas transmisji jednego bitu: m = 5

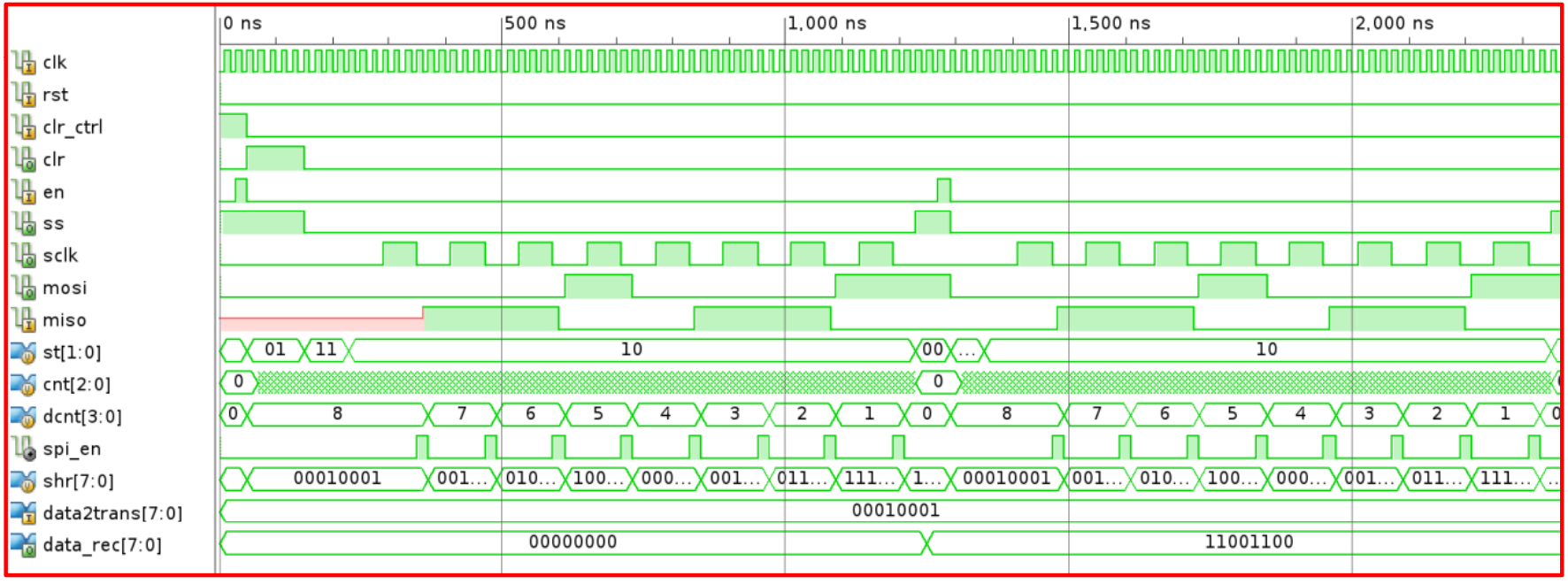
Czas opóźnienia na początku: d = 2

Dane do testowania:

Dane wysyłane przez Master: 00010001

Dane wysyłane przez Slave: 11001100

Symulacja przykładu z dwiema transakcjami



Dane do testowania:

Dane wysyłane przez Master: 00010001

Dane przyjęte przez Master : 11001100