UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

VHDL - Komponente, tasteri, serijski protokoli

Dr. Sc. Asmir Gogić, vanr. prof.

Tuzla, 2020.

VHDL - Component

- Reupotreba koda/funkcije/modela koji je prethodno razvijen u nekom drugom kodu je česta praksa.
- U VHDL-u ***reupotreba koda*** se postiže kroz koncept komponente ***component***.
- Kreiranjem komponente dobija se modul koji je moguće instancirati proizvoljan broj puta.
- ***component*** segmenat kao i entity definira samo interface modela u ovom slučaju modela koji se instancira.
- Identifikator ***component*** modela koji se deklariše ***mora imati isto ime kao i implementirani entitet***.
- Opšta sintaksa mehanizma instanciranja komponenti uključuje:
  - ***label_name*** - labelu/identifikator instance
  - ***entity_name*** - ime entiteta komponente koju instanciramo,
  - princip mapiranja porta tj mapiranja formalnih signala (signala iz komponente koju instanciramo) u stvarne signale nove arhitekture koje kreiramo na bazi instanci komponenti.

```
label_name: entity port map(
    formal_signal=>actual_signal,
    formal_signal=>actual_signal,
    .
    .
    .
    );
```

VHDL - Component

- Koncept razvoja komponente i instanciranja ćemo pokazati na jednostavnom primjeru izrade PWM modulatora.

PRIMJER 1

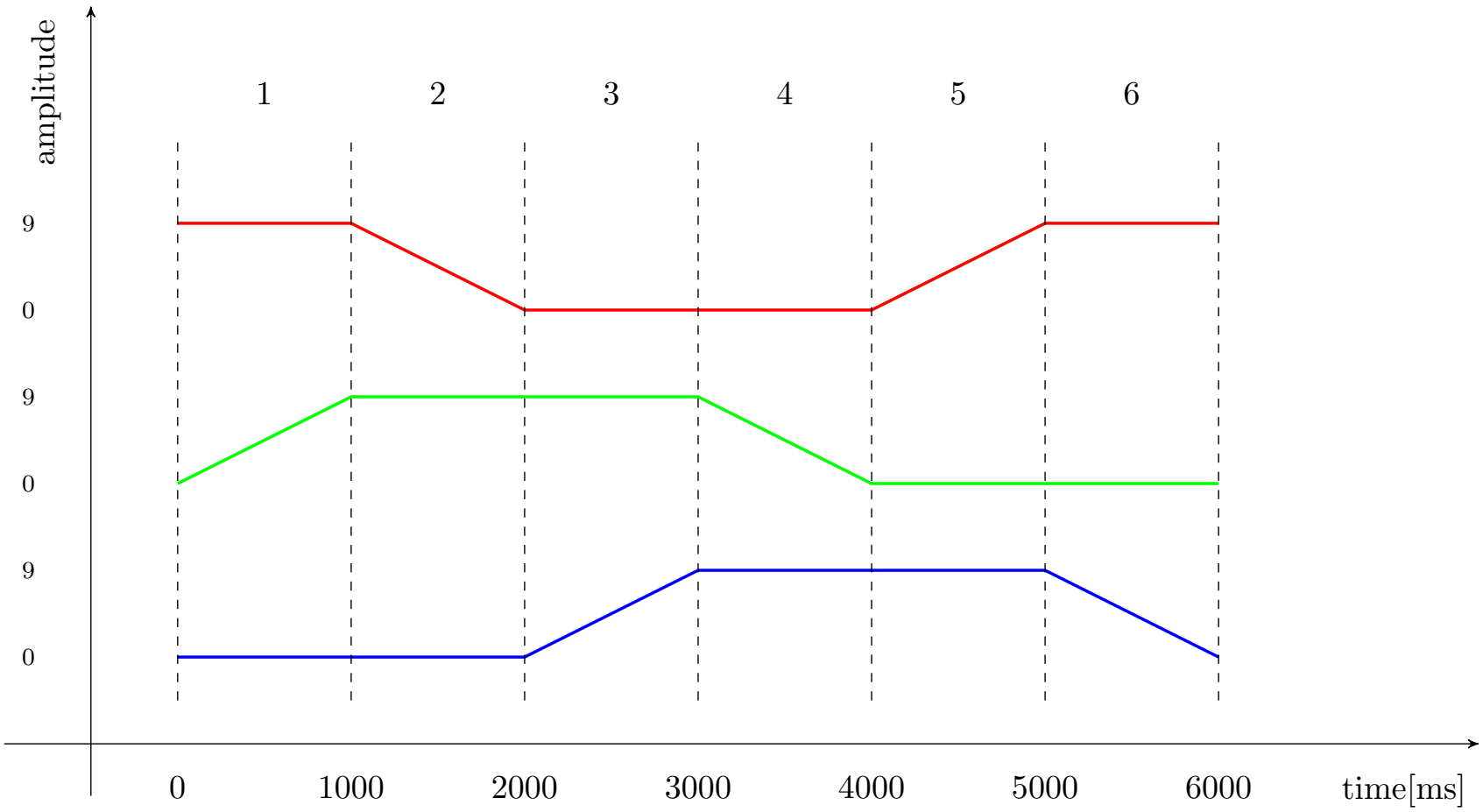
Napisati VHDL kod koji će implementirati PWM modulator kao komponentu koja će imati sljedeći interface:

- IO pin,
- clock signal,
- popunjenost PWM impulsa,
- frekvencija PWM impulsa.

Implementiranu komponentu instancirati i testirati na primjeru upravljanja RGB LED diode koja je spojena IO pinove P118, P116 i P117. Promjenu boje RGB LED dioda ciklično ponavljati prema šablonu predstavljenom na narednoj slici.

NOTE:

VHDL - Component



VHDL - Component: PWM modulator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
entity pwm is
  port(
  ipin:  out std_logic := '0';
  iclk:  in std_logic;
  iduty: in unsigned (31 downto 0) := x"00000000";
  ifreq: in unsigned (31 downto 0) := x"00000000");
end pwm;

architecture a_pwm of pwm is
signal cnt: unsigned (31 downto 0) := x"00000000";
begin
  process
  begin
    if iclk'event and iclk='1' then
      if cnt<iduty then
        cnt<=cnt+1;
        ipin<='1';
      elsif cnt<ifreq then
        cnt<=cnt+1;
        ipin<='0';
      else
        cnt<=x"00000000";
        ipin<='0';
      end if;
    end if;
  end process;
end a_pwm;
```

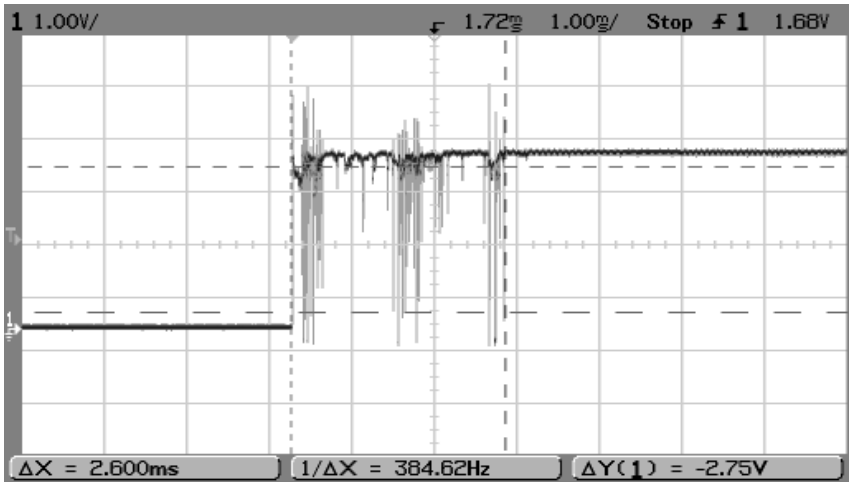
VHDL - Synthesys result

- Statistika rezultata sinteze VHDL koda za konkretan FPGA čip data je u ***.map*** odnosno ***.mrp*** fajlu.
- Statistika uključuje:
  - Ukupan broj grešaka (errors) i upozorenja (warnings),
  - Broj iskorištenih slice-ova kao registri - FF, Latch i logička kola,
  - Broj iskorištenih slice-ova kao LUT-ovi,
  - Broj iskorištenih slice-ova kao memorija,
  - Raspodjela slice-ova.
  - informacije o korištenim IOB[1] - tip, IO standard, max struja, slew rate, kašnjenje bloka, postojanje pull up/down, differencijalni ulaz/izlaz.

[1] IO Blocks

Debouncing problem

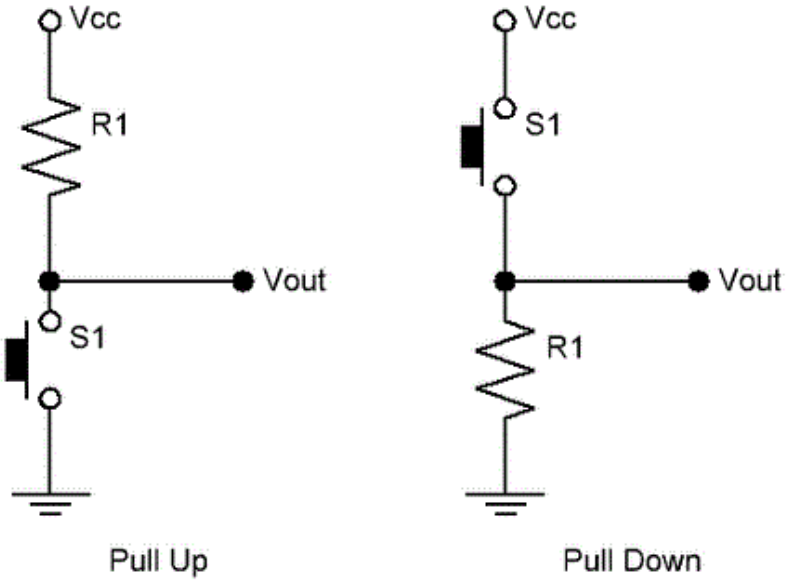
- Sve mehaničke i električne (tranzistori) prekidače karakteritira prijelazni procesa pri prelasku iz jednog stanja u drugo (HIGH u LOW i obratno). Kao rezultat imamo pojavu da se na krajevima tastera generiraju oscilacije u nivou napona zbog nesavršenosti dodirne površine.
- ***Posljedica***: taster prelazi iz stanja on-off i obratno par stotina puta u nekoliko milisekundi.



- Problema možemo tretirati na način da koristimo softversko (debouncing rutine) ili hardversko rješenje (pull-up ili pull-down sa debouncing kondenzatorom).
- Najjednostavnije rješenje debouncinga je upotreba rutina za kašnjenje. Međutim, trajanje oscilacija je specifično za aplikaciju i korisnika.
- Rutine za kašnjenje blokiraju izvršenje ostalih rutina i neefikasne su.
- ***Rješenje:*** upotreba rutina za kašnjenje koje ne blokiraju izvršenje ostalih procesa kao i analiza utjecaja stanja IO ulaznog pina.

Pull Up/Down

- U cilju osiguranja ***determinističkih uslova*** na IO pinovima koji su proglašeni kao ***ulazi*** neophodno ne istim dodati PULL UP ili PULL DOWN otpornike.



Pull Up Pull Down

- Vrijednost otpornika je obično reda desetina $k\Omega$ pa do $100k\Omega$.

Pull Up/Down

- Pull up/down možemo ***interno instalirati*** na IO pinovima.
- Instaliranje internih pull up/down otpornika izvodi se kroz ****.ucf*** file na sljedeći način:
  NET pad_net_name PULLUP;
  NET pad_net_name PULLDOWN;
- ***Vrijednost internih*** pull up/down otpornika je specifična za ***konkretan čip kao i za napon napajanja***
- Vrijednosti otpora se kreću od reda par $k\Omega$ do par desetina $k\Omega$[2].

- ***Prednost eksternih*** pull up/down otpornika je u mogućnosti kontrole vrijednost otpornika što je najčešće specifično za aplikaciju.
- ***Prednost internih*** pull up/down otpornika je u redukovanoj cijeni BoM-a[3].

[2] Više detalja u vezi konkretnih vrijednosti Spartan 6 FPGA čipa možete naći u ds162 dokumentu.

[3] BoM - Bill of Material

Pull Up/Down - specifikacija

I_{REF}	V_{REF} leakage current per pin for commercial (C) and industrial (I) devices		–10	–	10	µA
	V_{REF} leakage current per pin for expanded (Q) devices		–15	–	15	µA
I_L	Input or output leakage current per pin (sample-tested) for commercial (C) and industrial (I) devices		–10	–	10	µA
	Input or output leakage current per pin (sample-tested) for expanded (Q) devices		–15	–	15	µA
I_{HS}	Leakage current on pins during hot socketing with FPGA unpowered	All pins except PROGRAM_B, DONE, and JTAG pins when HSWAPEN = 1	–20	–	20	µA
		PROGRAM_B, DONE, and JTAG pins, or other pins when HSWAPEN = 0	$I_{HS(HSWAPEN = 1)} + I_{RPU}$			µA
C_{IN}[(1)]	Die input capacitance at the pad		–	–	10	pF
I_{RPU}	Pad pull-up (when selected) @ V_{IN} = 0V, V_{CCO} = 3.3V or V_{CCAUX} = 3.3V		200	–	500	µA
	Pad pull-up (when selected) @ V_{IN} = 0V, V_{CCO} = 2.5V or V_{CCAUX} = 2.5V		120	–	350	µA
	Pad pull-up (when selected) @ V_{IN} = 0V, V_{CCO} = 1.8V		60	–	200	µA
	Pad pull-up (when selected) @ V_{IN} = 0V, V_{CCO} = 1.5V		40	–	150	µA
	Pad pull-up (when selected) @ V_{IN} = 0V, V_{CCO} = 1.2V		12	–	100	µA
I_{RPD}	Pad pull-down (when selected) @ V_{IN} = V_{CCO}, V_{CCAUX} = 3.3V		200	–	550	µA
	Pad pull-down (when selected) @ V_{IN} = V_{CCO}, V_{CCAUX} = 2.5V		140	–	400	µA
I_{BATT}[(2)]	Battery supply current		–	–	150	nA
R_{DT}[(3)]	Resistance of optional input differential termination circuit, V_{CCAUX} = 3.3V		–	100	–	Ω

Debouncing problem - primjer

Primjer 2

Napisati VHDL kod koji će implementirati algoritam za prigušenja oscilacija tastera. Algoritam testirati na način da se LED dioda D7 (na pinu P133) postavi na stanje ON prilikom prve ON-OFF tranzicije tastera a zatim u stanje OFF nakon druge ON-OFF tranzicije tastera.

Primjer 3

Unaprijediti prethodni primjer tako da se prigušuju smetnje priliko pritiska i otpustanja tastera sa periodima prigušenja 25 ms.

Primjer 4

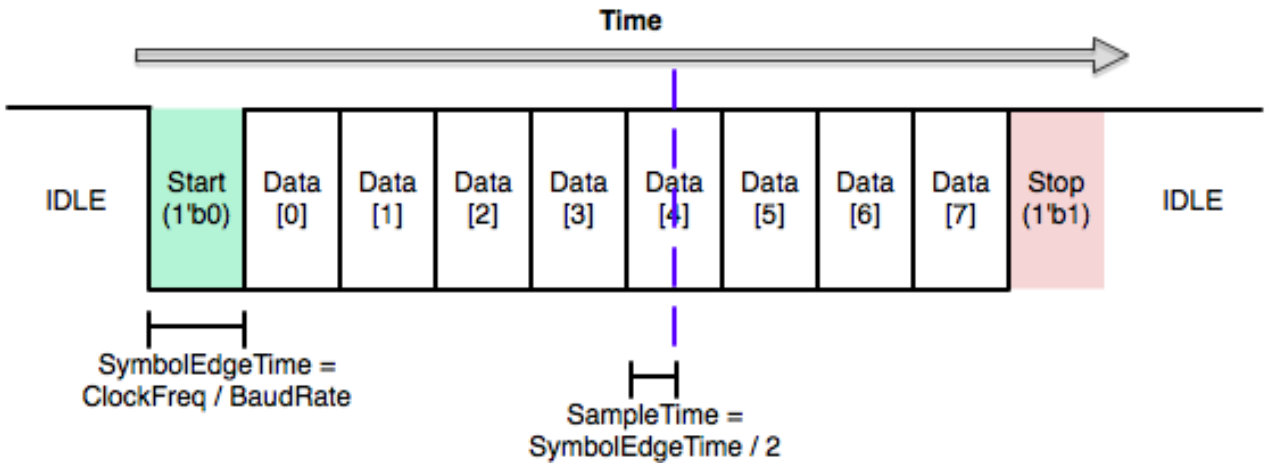
Ponoviti prethodni primjer tako da se debouncing rutina implementira kao VHDL komponenta. Šta je potrebno specificirati u interface-u komponente koja prigušuje debouncing?

Primjer 5

Napisati VHDL kod koji će implementirati binarni 8-bitni brojač na gore/dole čija će stanja biti prikazana na LED-ovim D0-D7. Odabir moda izvodit će se sa tasterom koji je spojen na IO pinu P120 a okidanje brojača sa tasterom na IO pinu P134. Za ublažavanje debouncing-a koristiti komponentu razvijenu u prethodnom primjeru.

UART

- ***UART*** - ***U****niversal* ***A****synchronous* ***T****ransmitter* ***R****eceiver*
- UART je serijski asinhroni bidirekcionalni[4] komunikacijski protokol.
- UART je pod-varijanta USART protokola koja nema sinhronizacijski mehanizam na nivou bita već na nivou simbola.
- UART koristi RZ linijski kod pri čemu se logička jedinica kodira sa Vcc (V) a logička nula sa 0 (V)



[4] full duplex - komunikacija u oba smjera, simultano!

UART

- ***USART*** implementira sinhronizacijski mehanizam preko dodatne SCK linije.
- ***UART*** komunikacija se zasniva na ***Tx***[5], ***Rx***[6], ***SCK***[7]
- ***UART*** nema sinhronizacije na nivou bita već koristi jednostavan sistem sinhronizacije na nivou simbola sa start i stop bitima.
- Kod UART-a prijenos podataka (komunikacija) se obavlja preko dvije linije (dva voda) ***Tx*** i ***Rx*** a ***odmjeravanje i odlučivanje*** obavlja se ***na sredini bitskog intervala***.
- Očigledno je da se ispravnost komunikacije bazirana na činjenici da izvori takt impulsa (koji se koriste za generiranje talasnih oblika) na obje strane moraju biti iste frekvencije, vremenski stabilni i tačni sa određenom - maksimalnom tolerancijom.
- U suprotnom doći će do klizanja vremenskog trenutka odlučivanja (jitter) i prijemna strana će dobiti ne ispravnu informaciju. Ovaj problem evidentan je pri većim bitskim brzinama.
- Podržane simbolske brzine [Baud ps, Sps] za USART-a i UART-a su: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, ...

---

[5] Izlaz pin namijenjen izlazni tok podatka

[6] Ulazni pin namijenjen za ulazni tok podataka

[7] Serijski takt impulsi - mehanizam sinhronizacije za prijemnik

UART - primjer

Primjer 6

Napisati VHDL kod koji će implementirati UART FSM (samo Tx dio). Za Tx pin odabrati Xilinx Sparta 6 IO pin P78. Funkcionalnost testirati na način da program generira sekvencu ASCII karaktera počevši od 0x20 zaključno sa 0x7F. Baudrate je 921600 a pauza između karaktera je 100 ms. Tokom geneiranja karatera LED koji je spojen na IO pin P133 postaviti na stanje 1.

Primjer 7

Proširiti funkcionalnost prethodnog primjera na načina da se doda i Rx dio. Za Rx tj DRX pin odabrati IO pin 79 a funkcionalnost implementiranog sistema testirati tako što će model generirati echo primljenog UART karatera sa Rx linije na Tx liniju.

UART - primjer

Primjer 8

Razviti VHDL komponentu UART koja će imati sljedeći interface:

- tx - IO pin koji će obnašati funkciju DTX pina,
- rx - IO pin koji će obnašati funkciju DRX pina,
- wdata - 8 bitni podataka koji se emituje,
- rdata - 8 bitni podataka koji je primljen,
- wi - flag za iniciranje emitovanja podataka u wdata,
- wc - flag za indiciranje završetka transmisije podataka u wdata,
- rc - flag za indiciranje prijema novog podataka u rdata.

Rx i Tx dijelove UART komponente implementirati kao zasebna dva procesa. UART komponentu testirati tako da za svaki primljeni karater emitovanu sa HOST-a pri 921600 bauda, ispiše odgovorajuća binarna kombinacija na LED0 - LED7.

Literatura

- RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability, 1st Editon by Pong P. Chu, 2006.
- Digital Systems Design Using VHDL 2 nd Edition, by Charles H. Roth, Jr. and Lizy Hurian John, Thomson, 2007.
- The Designer's Guide to VHDL, Third Edition, Peter J. Ashenden, 2008.