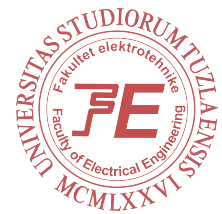
UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE



VHDL: uslovno i selektivno dodjeljivanje signala, case, enumeratori, konstante, konverzija toka podataka

Dr. Sc. Asmir Gogić, vanr. prof.

Tuzla, 2020.

Conditional Signal Assignment

- Dosada smo vidjeli da se konkurentni signali vežu za jedan izraz!
- VHDL omogućuje mehanizme uslovnog dodjeljivanja vrijednosti signala ***Conditional Signal Assignment*** po sljedećem principu

```
<target> <= <expression> when <condition> else
<expression> when <condition> else
<expression>;
```

- Princip sličan if else lancu gdje je evidentan prioritet tokom selektovanja izlaza.
- Za slučaj da imamo dva uslova da su zadovoljena, biti će izvršen prvi odozgo u implementaciji modela.

Primjer 1

Napisati VHDL kod koji će implementirati funkcionalnost digitalnog sklopa koji poredi dva dvo-bitna binarna broja. Izlaz digitalnog sklopa je 1 ako su binarni brojevi jednaki u suprotnom je u stanju 0.

Conditional Signal Assignment

- Ako pogledamo tablicu stanja logičkog sklopa...

Ulaz 1		Ulaz 0		Izlaz
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

$$Y = \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + A\overline{B}C\overline{D} + ABCD$$

Conditional Signal Assignment

- Dosadašnja spoznaja o VHDL-u bi nam omogućila sljedeću implementaciju...

```
library ieee;
use ieee.std_logic_1164.all;
entity comp is
  port (
  -- dva dvo-bitna ulaza
    a , b : in std_logic_vector (1 downto 0);
  -- jednobitni izlaz
    y : out std_logic);
end comp;

architecture fun_arch of comp is
  -- pomocni signali
  signal p0,p1,p2,p3: std_logic;

begin
  -- izlaz je suma pomocnih signala
  y <= p0 or p1 or p2 or p3;
  -- pomocni signali kao potpuni proizvodi
  p0 <= ((not a(1)) and (not b(1))) and ((not a(0)) and (not b(0)));
  p1 <= ((not a(1)) and (not b(1))) and (a(0) and b(0));
  p2 <= (a(1) and b(1)) and ((not a(0)) and (not b(0)));
  p3 <= (a(1) and b(1)) and (a(0) and b(0) ) ;
end fun_arch;
```

Conditional Signal Assignment

- ... korištenjem uslovnog dodjeljivanja imamo sljedeću implementaciju...

```
library ieee;
use ieee.std_logic_1164.all;
entity comp is
  port (
  -- dva dvo-bitna ulaza
    a , b : in std_logic_vector (1 downto 0);
  -- jednobitni izlaz
    y : out std_logic);
end comp;

architecture fun_arch of comp is
begin
y <=  '1' when (a(1)='0' and b(1)='0' and a(0)='0' and b(0)='0') else
'1' when (a(1)='0' and b(1)='1' and a(0)='0' and b(0)='1') else
'1' when (a(1)='1' and b(1)='0' and a(0)='1' and b(0)='0') else
'1' when (a(1)='1' and b(1)='1' and a(0)='1' and b(0)='1') else
'0';
end fun_arch;
```

Selected Signal Assignment

- ***Selected Signal Assignment*** - predstavlja još jedan način dodjeljivanja vrijednosti signala gdje se dodjeljivanje izvodi samo ako se pojavi odabrani set signala

```
with <choose_expression> select
  target <= <expression> when <choices>,
    <expression> when <choices>;
```

- Princip sličaj case/when iskazima gdje prioritet u izvršavanju nije bitan.
- ... primjer multipleksera 4/1:

```
entity mux_4on1_entity is
port(
        D3,D2,D1,D0 : in std_logic;
        SEL: in std_logic_vector(1 downto 0);
        MX_OUT : out std_logic);
end mux_4on1_entity;
architecture mux_4on1_arch of mux_4on1_entity is
begin
with SEL select
  MX_OUT <= D3 when "11",
            D2 when "10",
            D1 when "01",
            D0 when "00",
            '0' when others;
end mux_4on1_arch;
```

Case statement

- ***Case*** iskaz selektuje jedan od alternativnih sekvenci iskaza na bazi vrijednosti izraza.
- Za razliku od ***if*** iskaza, ***test ne mora biti bool tip***, nego može biti signal, varijabla ili izraz.
- ***Case*** iskaz se koristi kada postoji veliki broj mogućih ishoda.
- Opšta sintaksa je:

```
case expression is
  when options_1 =>
    statement_sequence
    ...
  when options_n =>
    statement_sequence
  [when others =>
    statement_sequence]
end case;
```

- ***Case*** iskaz sadrži više ***when*** klauzula i svaka od njih može imati jedan ili više izraza. Ako je izraz vrijednost, iste se odvajaju sa | karakterom

Case statement

```
case expression is
  when val =>
    statement_sequence
  when val1 | val2 | ... | valn =>
    statement_sequence
  when val3 to val4 =>
    statement_sequence
  when val5 to val6 | val7 to val8 =>
    statement_sequence
  ...
  when others =>
    statement_sequence
end case;
```

VHDL - enumerators, constants

- Predstavlja korisnički definiran ***tip podataka*** sa ***simboličkim vrijendostima***.
- U osnovi omogućuje zamjenu klasičnih decimalnih odnosno relanih vrijednosti sa simboličkim imenima.

```
type state_t is (INIT, WAIT4ACCESS, READ, ERASE, WRITE);

-- primjer
signal curr_state: state_t;
signal prev_state: state_t;
curr_state <= INIT;
```

- ***Konstantama*** za raliku od enumeratora mora eksplicitno ***dodjeliti vrijednosti***.
- Osiguravaju isti koncept kao i enumeratori.
- Nedostatak konstanti je u činjenici da možemo imati dvije različite konstante koje koristimo u FSM a da imaju iste vrijednosti.
- ... kod enumeratora se to ne može desiti

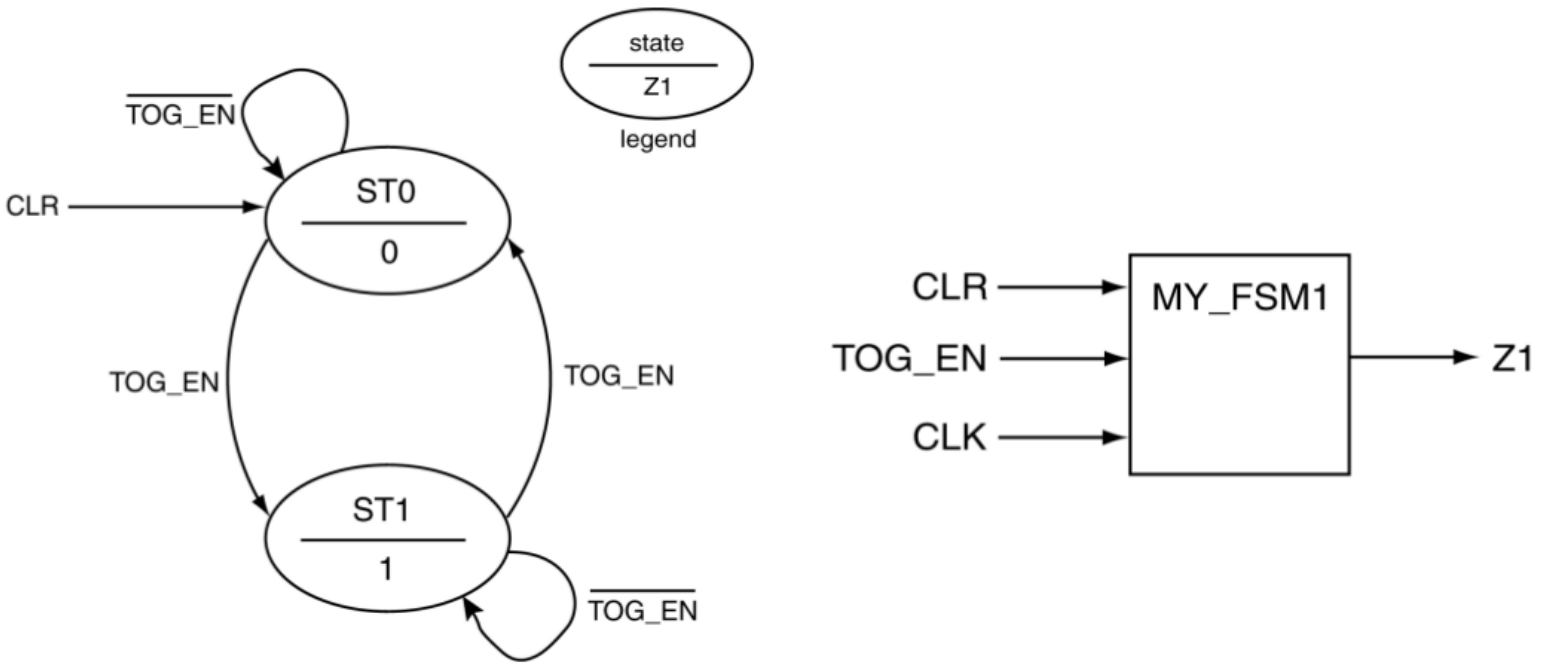
```
constant <constant_name> : <type> := <value>;

-- primjer
constant BUSWIDTH : integer := 8;
signal datain: signed(BUSWIDTH-1 downto 0) := x"00";
```

VHDL - enumerators

PRIMJER 2

Implementirati VHDL model FSM-a prikazanog na slici koristeći procese i enumeratore.



VHDL - enumerators

```
entity my_fsm1 is
  port ( TOG_EN : in std_logic;
         CLK,CLR : in std_logic;
         Z1 : out std_logic);
end my_fsm1;
architecture fsm1 of my_fsm1 is
type state_type is (ST0,ST1);  -- enumeratori
signal PS,NS : state_type; -- PS - previous state, NS - new state

begin
  sync_proc: process(CLK,NS,CLR)
  begin
    -- take care of the asynchronous input
    if (CLR = '1') then
      PS <= ST0;
    elsif (rising_edge(CLK)) then
      PS <= NS;
    end if;
  end process sync_proc;
```

VHDL - enumerators

```
  comb_proc: process(PS,TOG_EN)
  begin
    Z1 <= '0'; -- pre-assign output
    case PS is
      when ST0 => -- items regarding state ST0
        Z1 <= '0'; -- Moore output
        if (TOG_EN = '1') then NS <= ST1;
        else NS <= ST0;
        end if;
      when ST1 => -- items regarding state ST1
        Z1 <= '1'; -- Moore output
        if (TOG_EN = '1') then NS <= ST0;
        else NS <= ST1;
        end if;
      when others => -- the catch-all condition
        Z1 <= '0'; -- arbitrary; it should never
        NS <= ST0; -- make it to these two statements
     end case;
  end process comb_proc;
end fsm1;
```

PRIMJER 3

Implementirati ***testbench*** za prethodni primjer.

VHDL - enumerators

- Sljedeći primjer procesa koji mijenja vrijednost varijable *next_color* na bazi trenutne vrijednosti boje definirane u enumeratoru *type_color*.

```
type type_color is (red, yellow, green);
process (color)
  case color is
    when red =>
      next_color <= green;
    when yellow =>
      next_color <= red;
    when green =>
      next_color <= yellow;
  end case;
end process;
```

Konverzija toka podataka - zašto?

- Generalno postoje dva tipa prijenosa podataka: ***serijski i paralelni***.
- ***Serijski prijenos*** podataka karakterizira sukcesivno emitovanje[1] simbola/bita kroz komunikacijski kanal (žičani ili bežični) prema nekom predefiniranom redoslijedu[2].
- Prijenos podataka u osnovnom opsegu žičanim medijem se odvija u blokovima od Nx8 bita pri čemu sa odgovarajućim ***START*** i ***STOP*** signalima naglašava početak i kraj prijenosa bloka podataka.
- Veliki broj serijskih protokola pored informacionih/start/stop bita/simbola emituju i zaštitne bite[3]
- ***Paralelni prijenos*** omogućuje simultani prijenos cijelog bloka podataka ili dijela bloka podataka.
- Najčešći broj paralelnih komunikacijski kanala (električnih vodova) je 2,4,8,16,32,64...
- Postoje i hibridne verzije serijsko-paralelnog prijenosa kao što je primjer kod QSPI i OCTA SPI protokola, USB3, MIPI DSI/CSI, HDMI, LVDS,...

---

[1] jedan za drugim

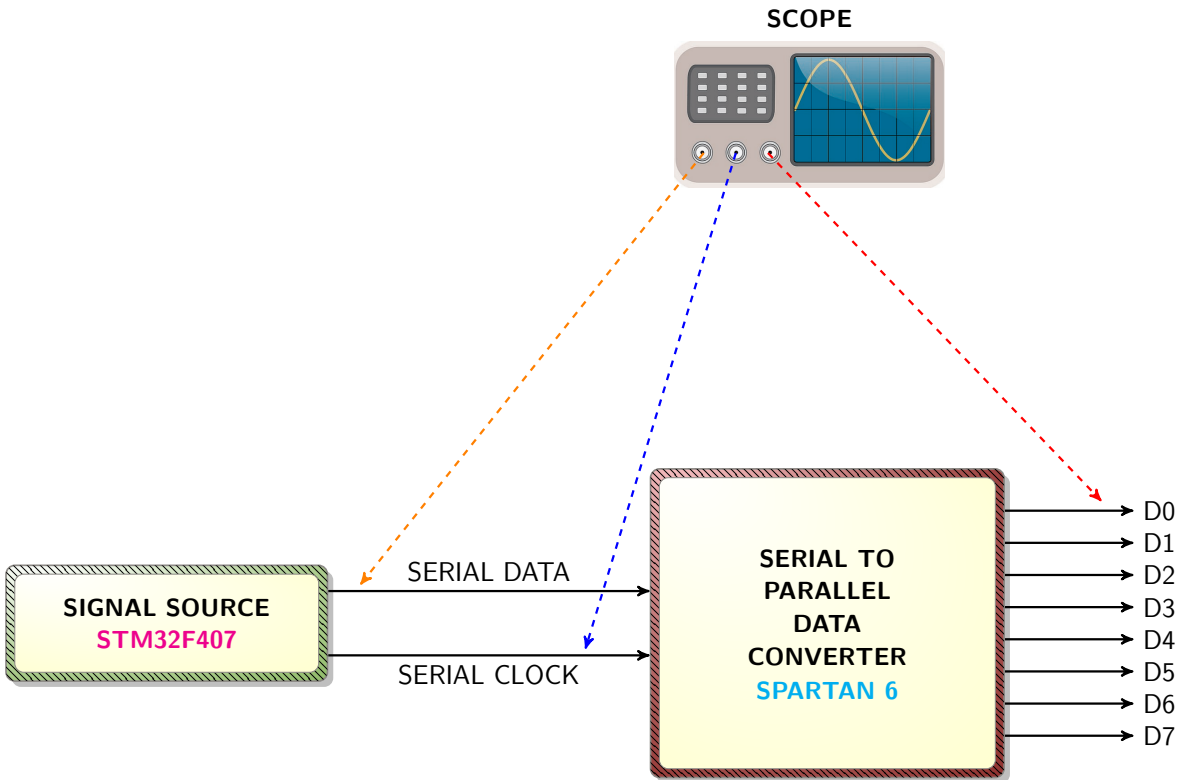
[2] najznačajniji prema najmanje značajnom i obrnuto

[3] Koji su rezultat kanalnog kodiranja, npr bit pariteta kod U(S)ART-a, CRC kod SPI, itd

Konverzija toka podataka - primjer

PRIMJER 4

Razviti VHDL model serijsko-paralelnog konvertora podataka koji ima dva ulaza SDATA i SCLK te jedan 8 bitni izlaz POUT. Signal SDATA predstavlja podatke a SCLK takt impulse nekog eksternog modula. Serijski podaci na ulazu SDATA dolaze u grupama od po 8 bita a vrijednost bita je validna na negativnoj ivici SCLK takt impulsa. Konvertor treba da nakon detekcije osmog bita postavi stanje izlaza POUT.



Konverzija toka podataka - HW

PRIMJER 5

Ponoviti prethodni primjer ali tako da se generira i PCLK signal koji predstavlja takt impulse za paralelni tok podataka čija je frekvencija 8 puta manja.

PRIMJER 6

Ponoviti prethodni primjer tako da cijeli modul bude implementiran kao komponenta ***ser2par*** koja se može instancirati.

PRIMJER 7

Razviti VHDL model paralelno-serijskog konvertora podataka koji ima šesnaest ulaza za podatke I_k $k \in [0, 15]$ i jedan ulaz za takt impulse PCLK. Modul ima jedan izlaz za podatke XDATA i izlaz za takt impulse XCLK. Odmjeravanje signala izvodi se na pozitivnoj ivici PCLK signala a frekvencija takt impulsa XCLK = 16$\times$PCLK.

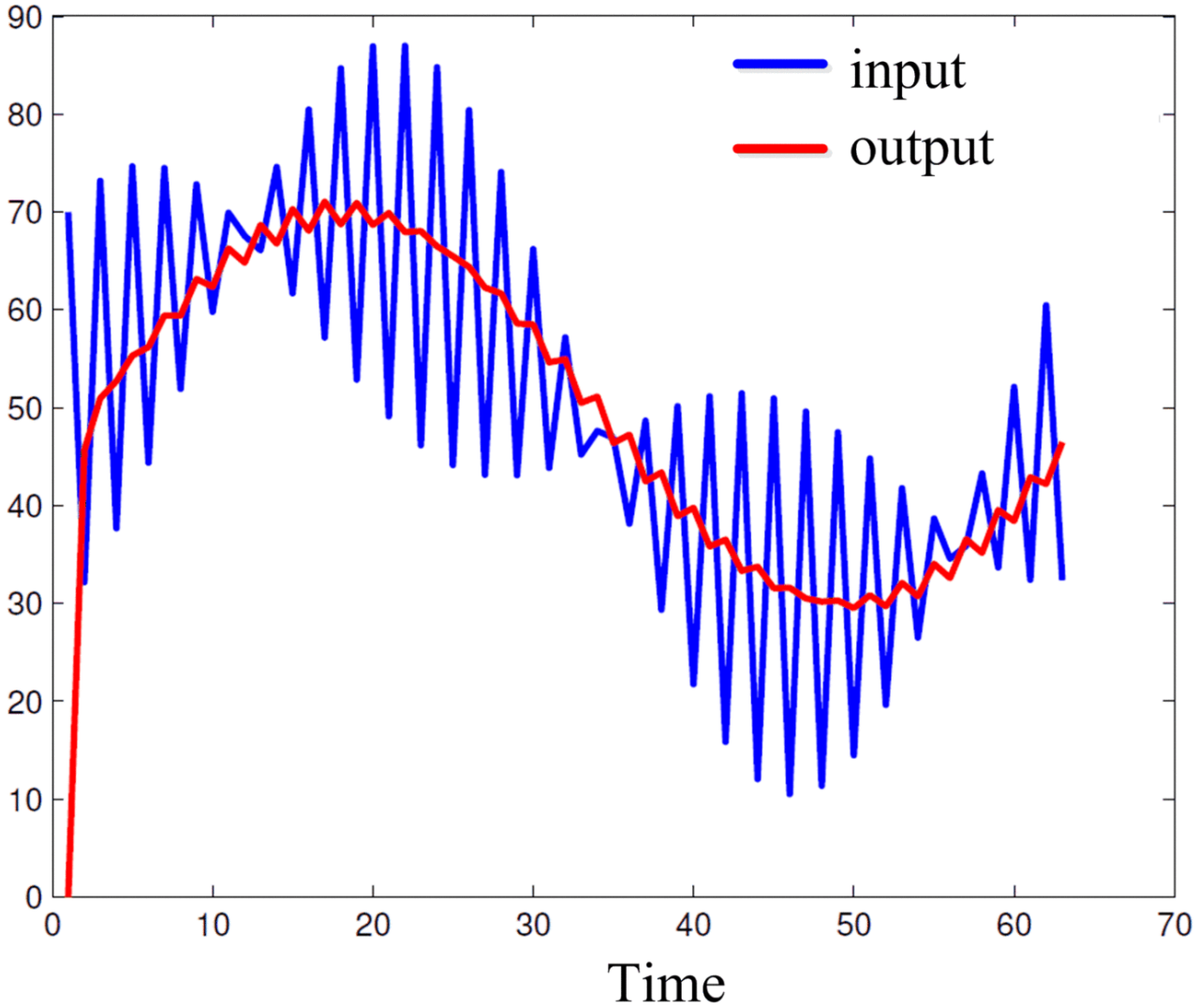
Srednja vrijednost diskretnog signala

- Proračun srednje vrijednost siganal izvodimo sa tvz. ***moving average*** filter (MAF) koji u osnovi pronalazi aritmetičku sredinu predefiniranog seta uzoraka.
- Pogodan je za odstranjivanje viših harmonijskih komponenti, slučajnog šuma odnosno za uglačavanje signala.
- Impulsni odziv FIR MAF-a je

$$h[n] = \frac{1}{N} \sum_{k=0}^{N-1} \delta[n-k] \tag{1}$$

- MAF je ustvari LPF čija je granična frekvencija f_s/N.
- Koeficijenti filtera su isti i iznose 1/N.

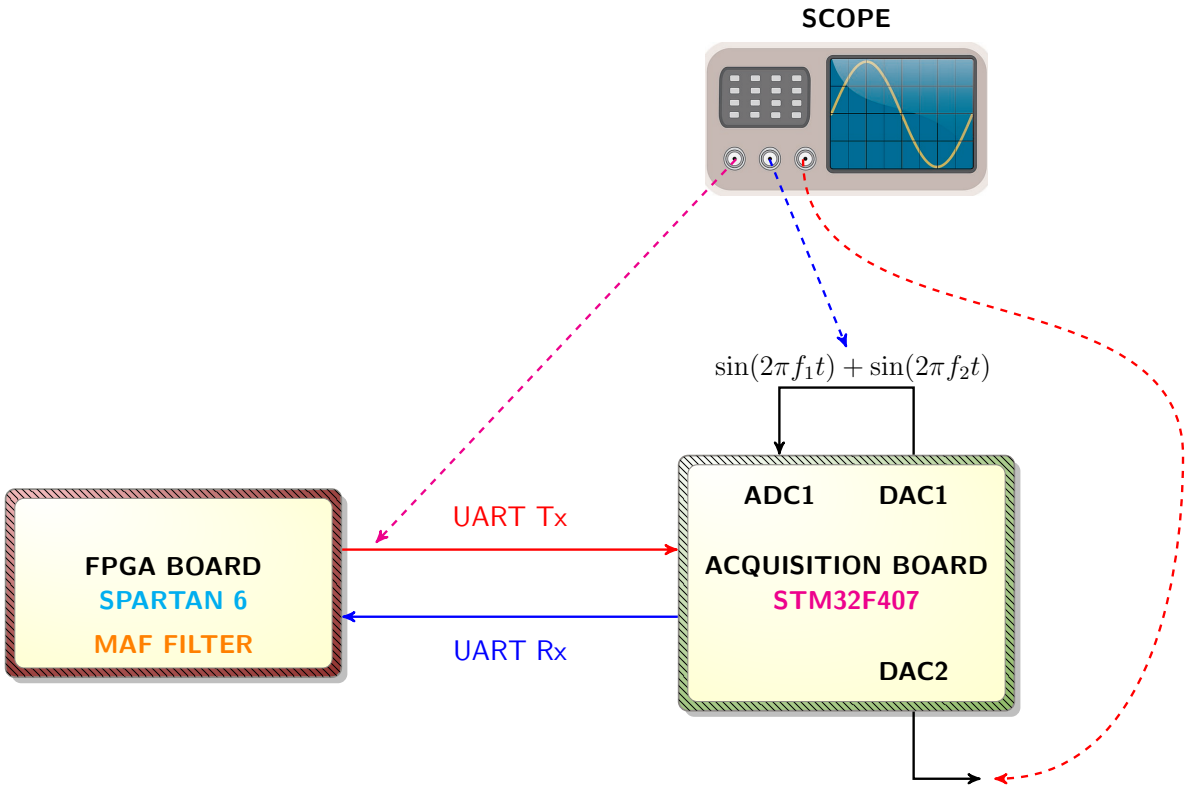
Srednja vrijednost diskretnog signala



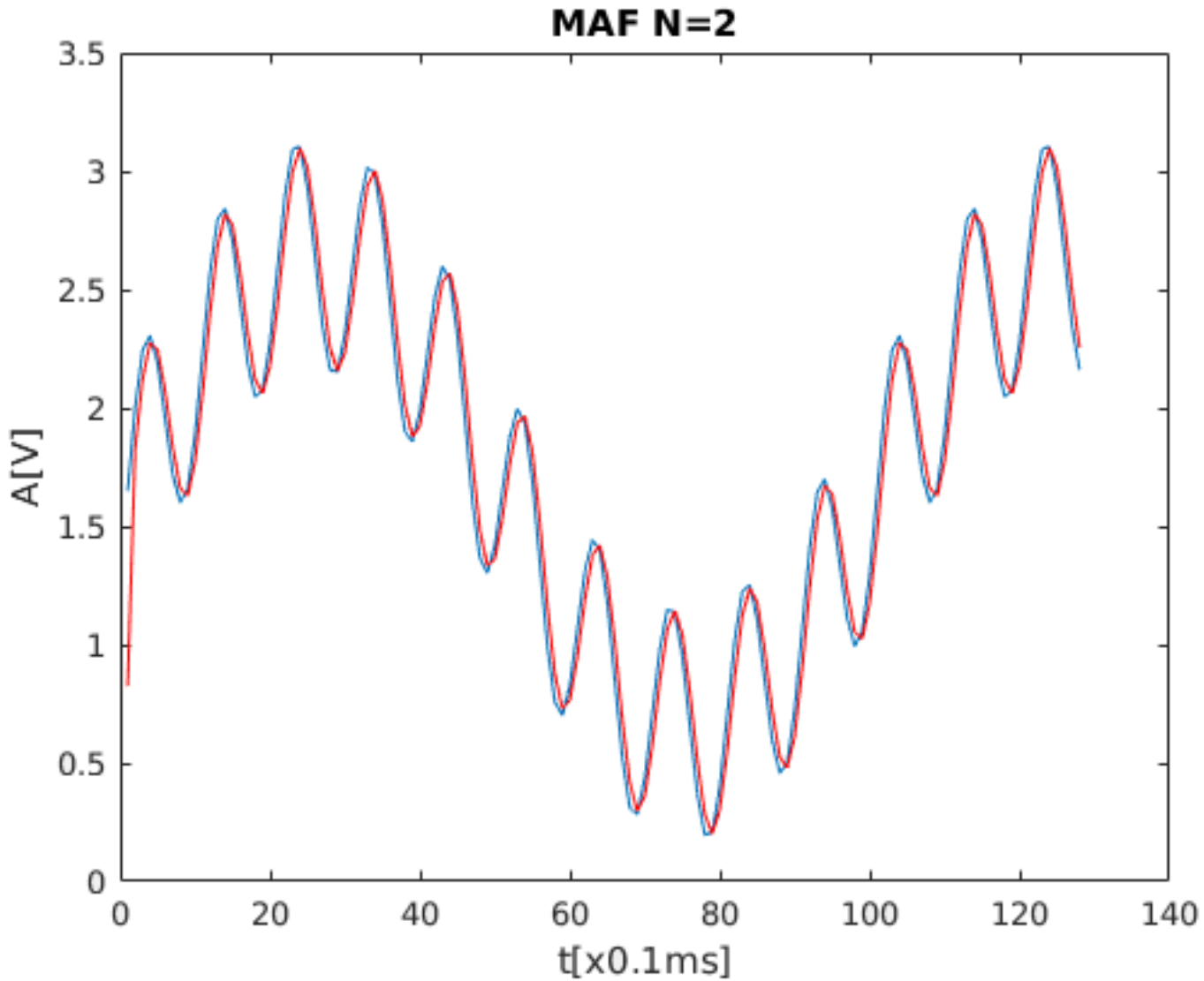
Srednja vrijednost diskretnog signala

Primjer 8

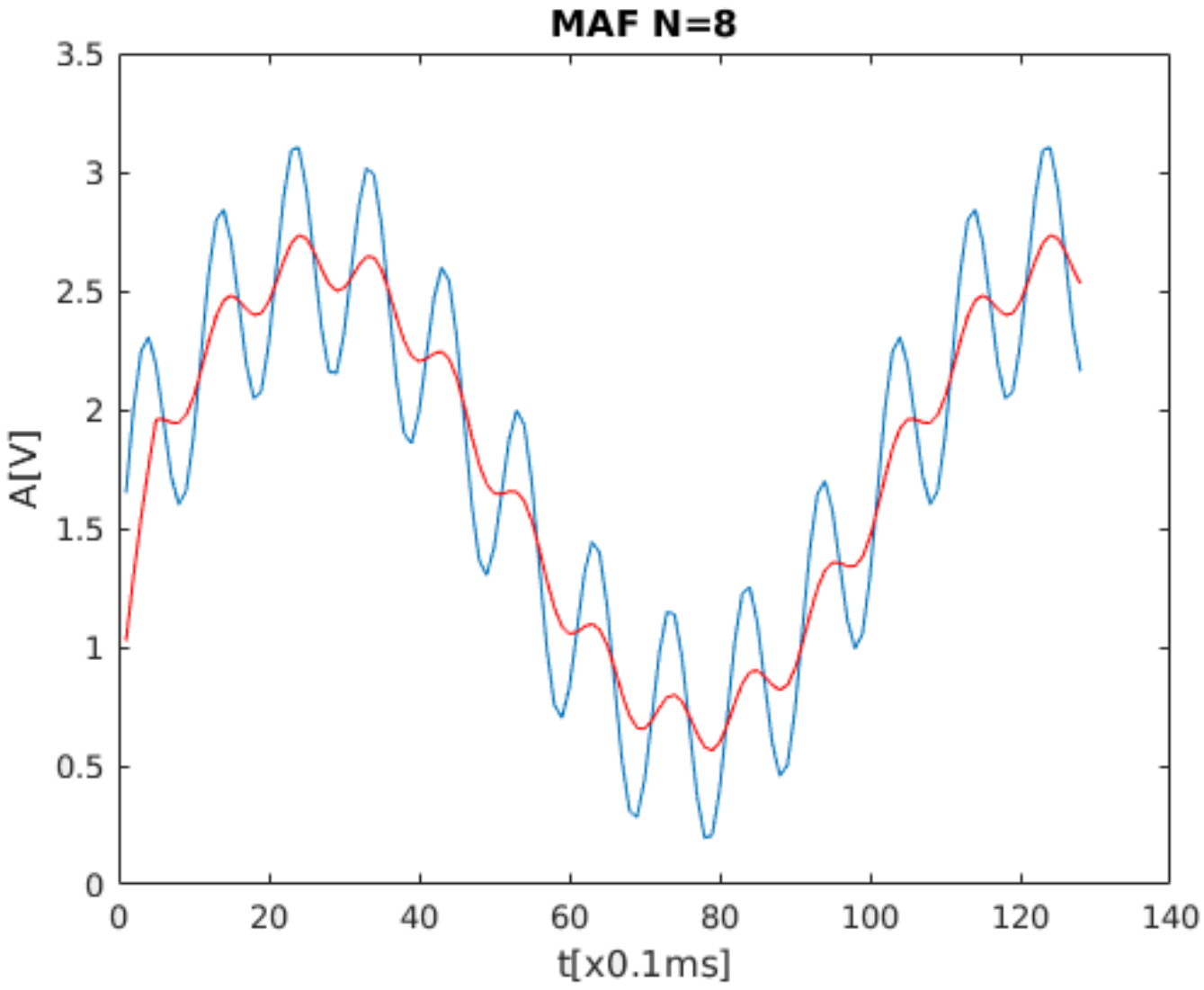
Razviti VHDL model MAF digitalnog filtera N koeficijenta. Ulaz MAF-a predstavljaju uzorci ADC-a čija je frekvencija uzorkovanja 10 kHz a signal poruke je $\sin(200\pi t) + \sin(1000\pi t)$. Razmjena podataka između ADC-a i FPGA čipa izvodi se UART serijskim protokolom čiji je baudrate 921600 bauda.. Analizirati utjecaj reda filtera na graničnu frekvenciju filtera.



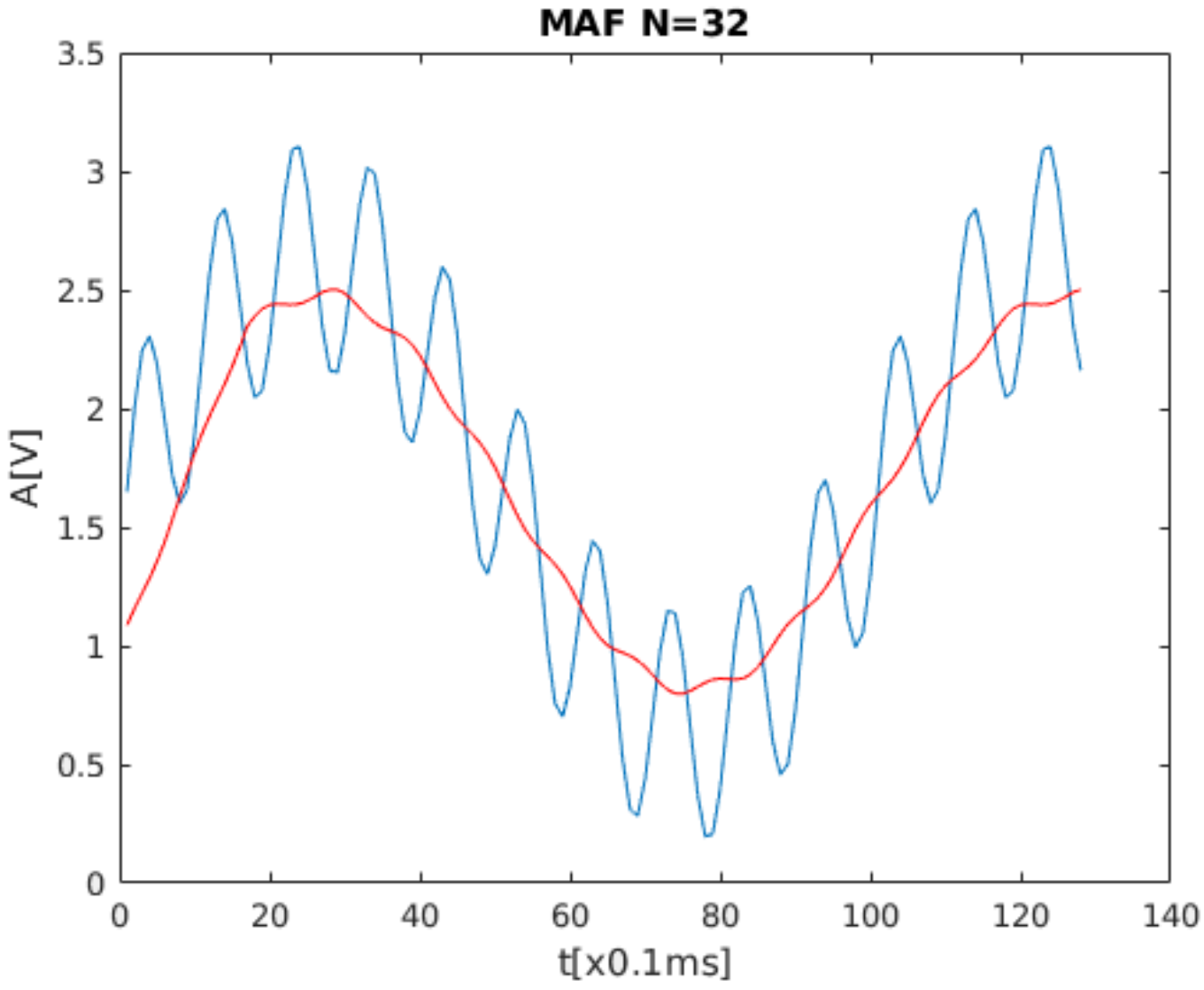
MAF - Matlab rezultati N=2



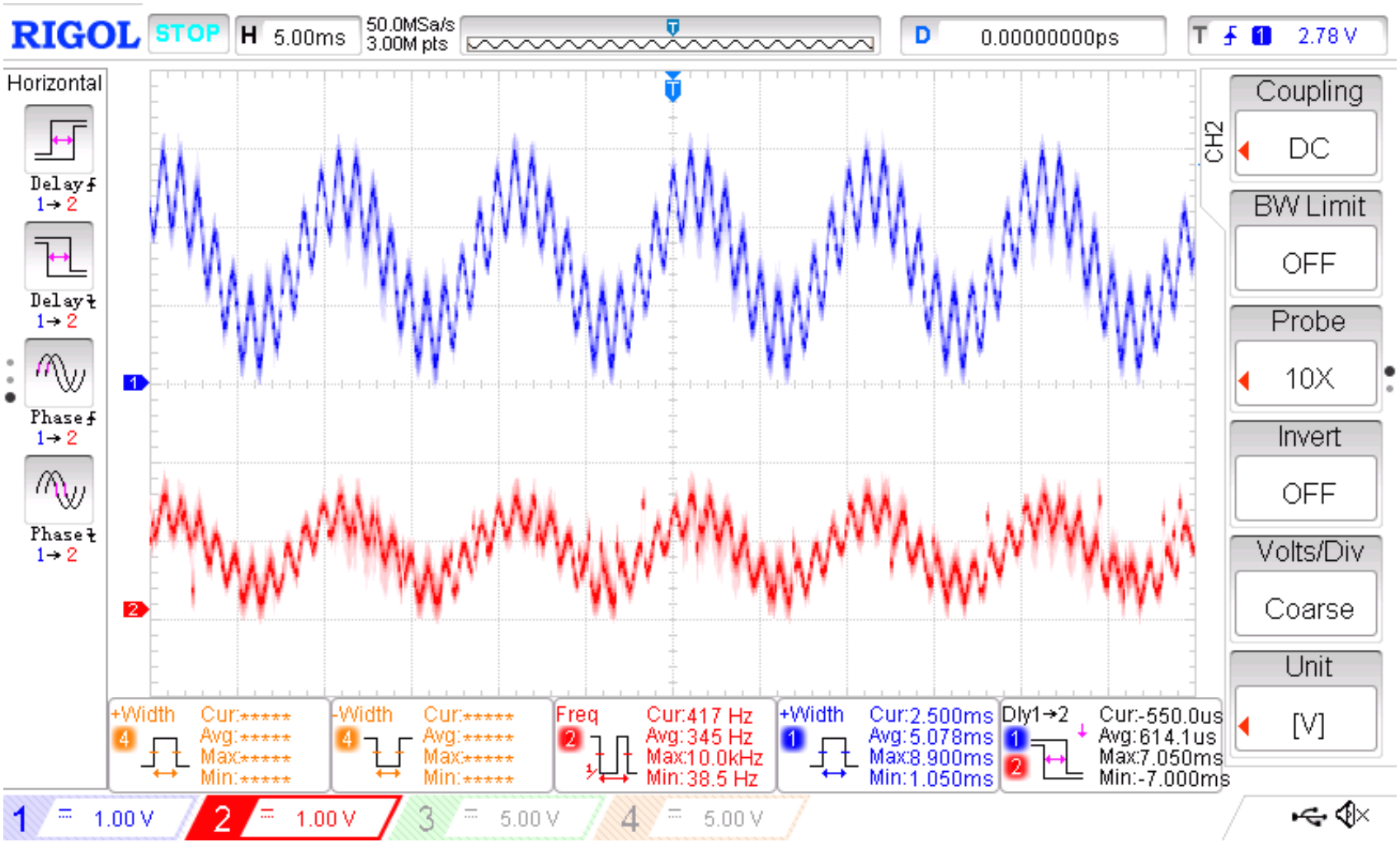
MAF - Matlab rezultati N=8



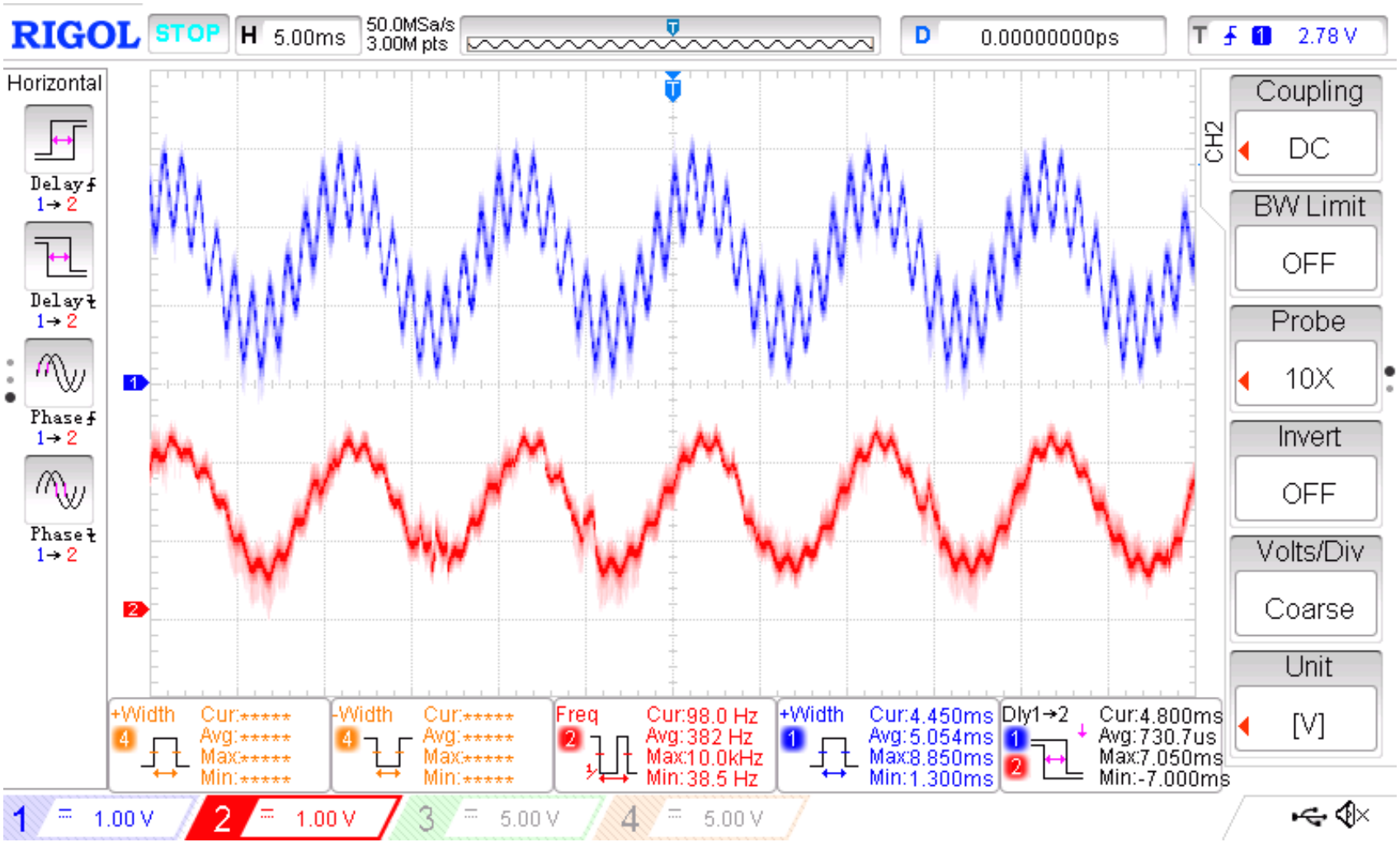
MAF - Matlab rezultati N=32



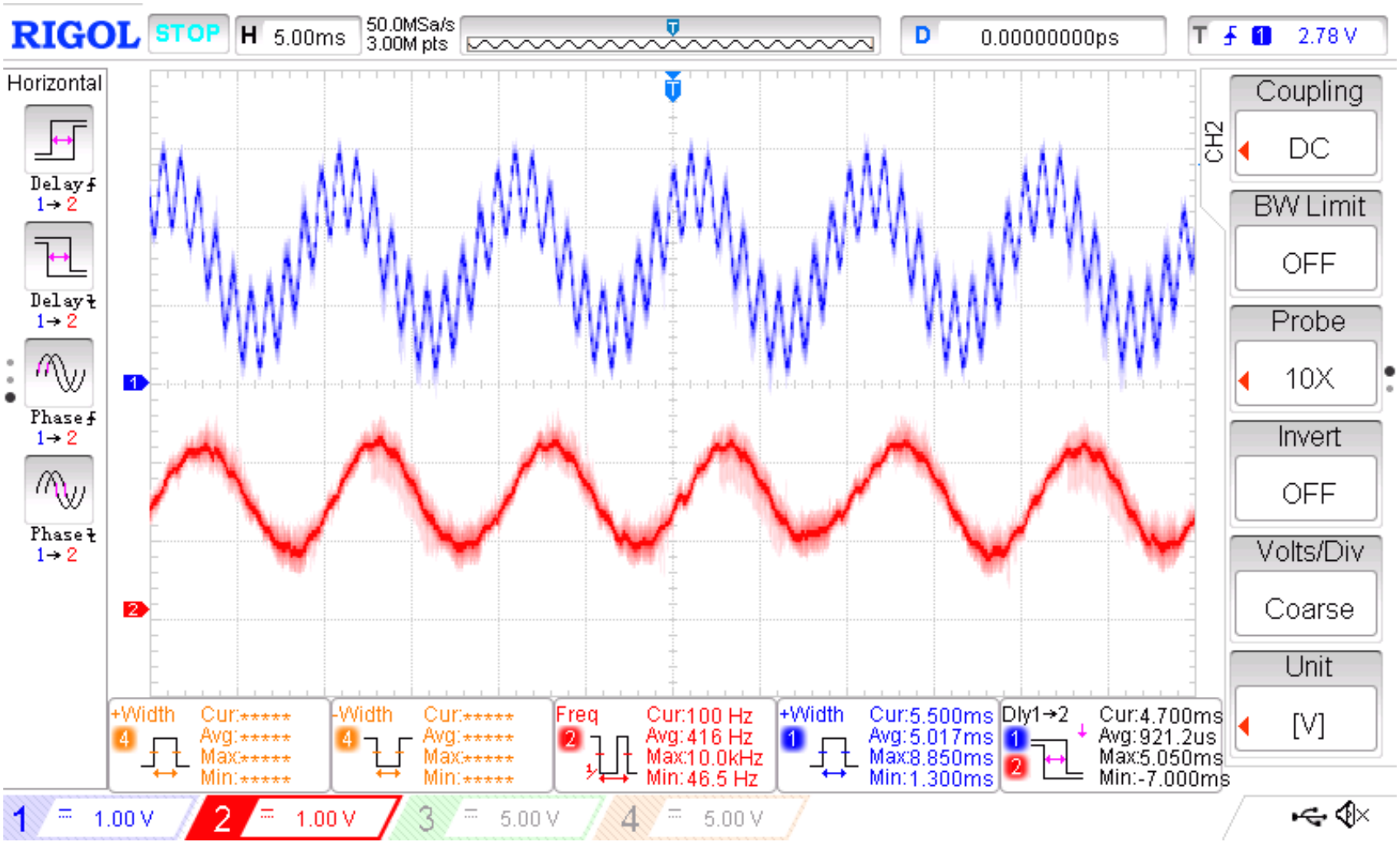
MAF - FPGA rezultati N=2



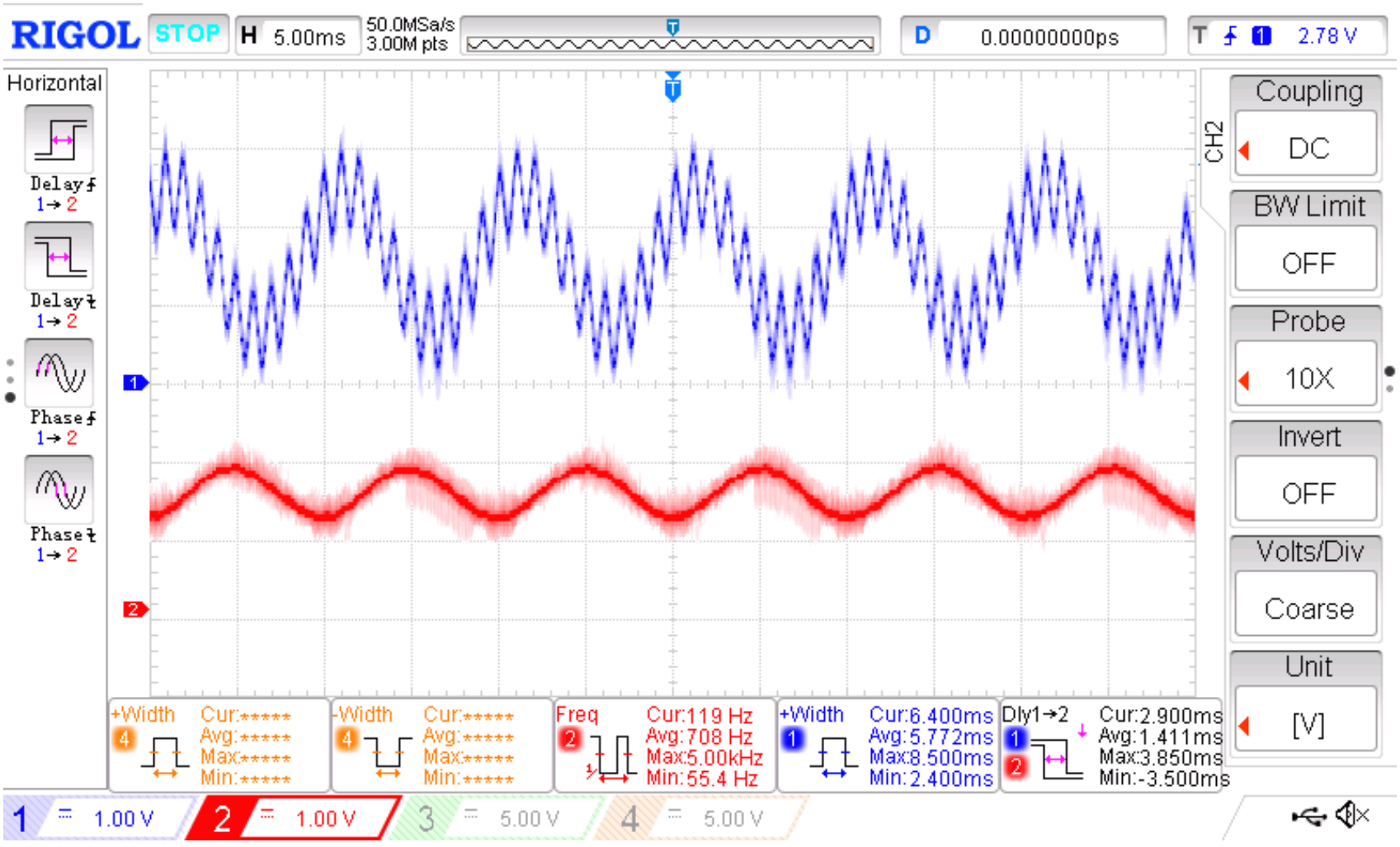
MAF - FPGA rezultati N=8



MAF - FPGA rezultati N=32



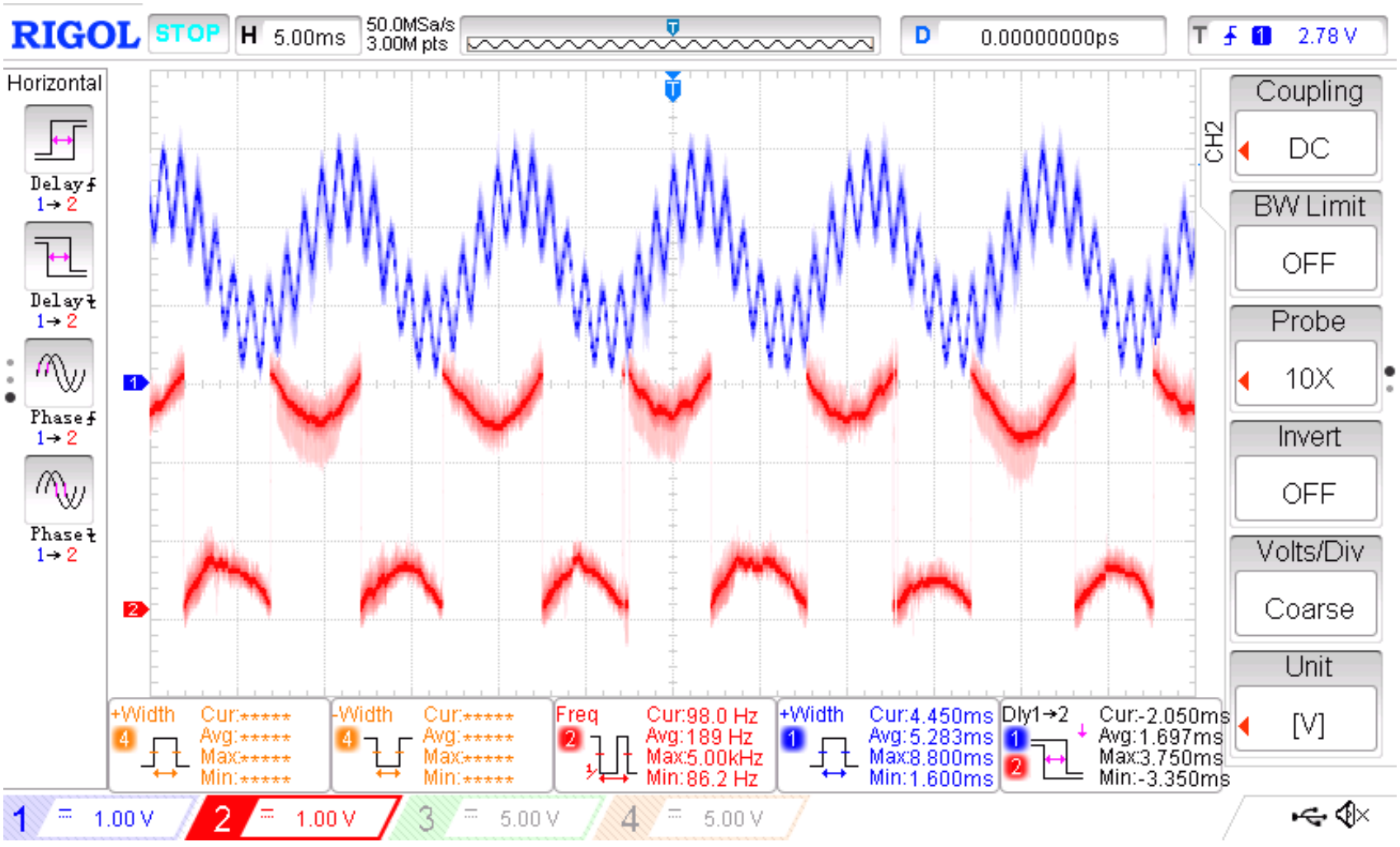
MAF - FPGA rezultati N=64



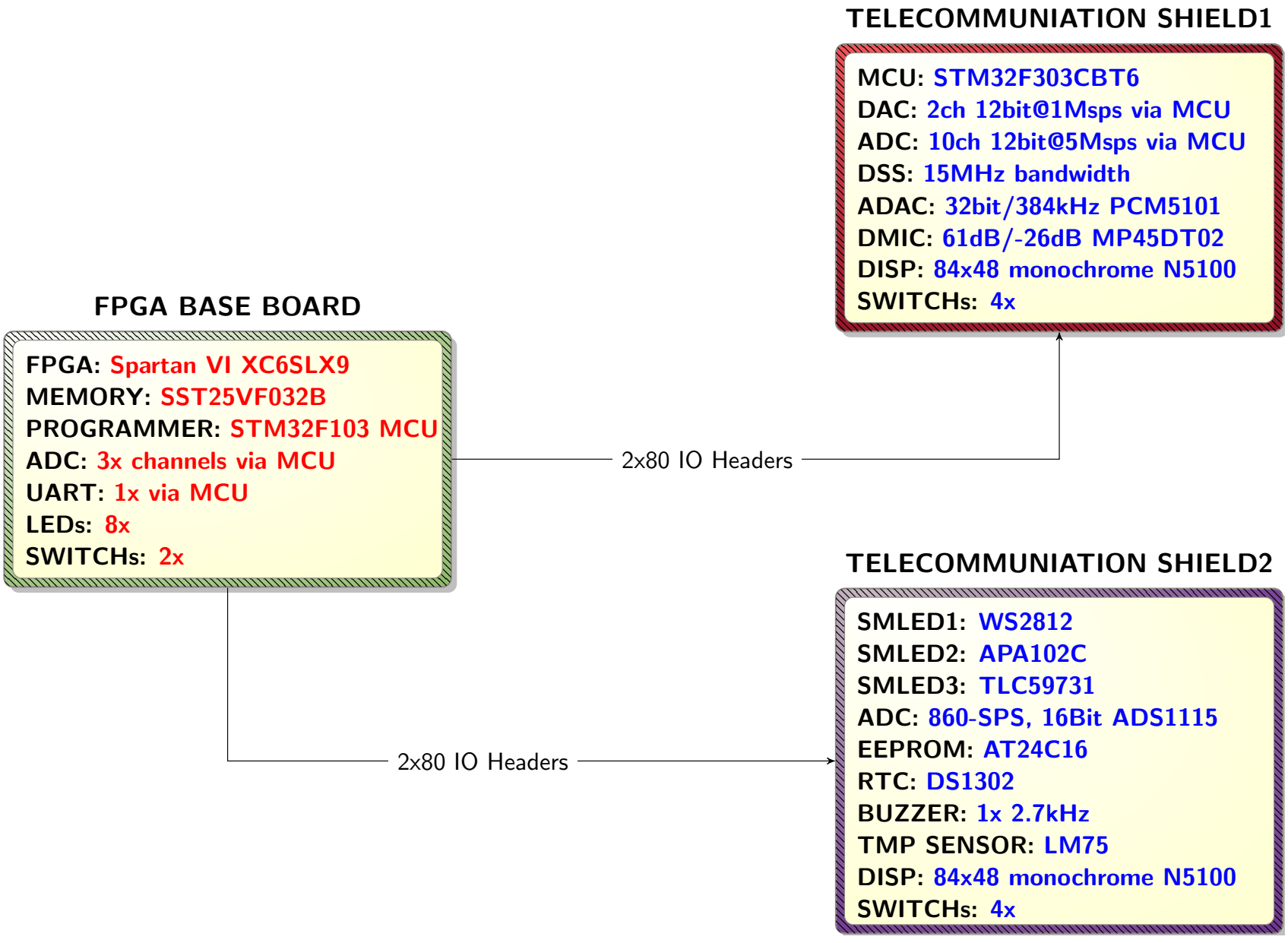
VHDL resize & conversion

From type	To type	Conversion function
std_logic_vector	unsigned	unsigned(arg)
std_logic_vector	signed	signed(arg)
unsigned	std_logic_vector	std_logic_vector(arg)
signed	std_logic_vector	std_logic_vector(arg)
integer	unsigned	to_unsigned(arg, size)
integer	signed	to_signed(arg, size)
unsigned	integer	to_integer(arg)
signed	integer	to_integer(arg)
integer	std_logic_vector	integer -> unsigned/signed -> std_logic_vector
std_logic_vector	integer	std_logic_vector -> unsigned/signed -> integer
unsigned + unsigned	std_logic_vector	std_logic_vector(arg1 + arg2)
signed + signed	std_logic_vector	std_logic_vector(arg1 + arg2)
Type		**Resize function**
unsigned		resize(arg, size)
signed		resize(arg, size)

MAF - overflow issue!



FPGA DEV Board



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.