Návrh číslicových systémů (INC): Projekt: Ukázka implementace

Lukáš Kekely

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 00 Brno - Královo Pole
ikekely@fit.vutbr.cz

01. 02. 2023

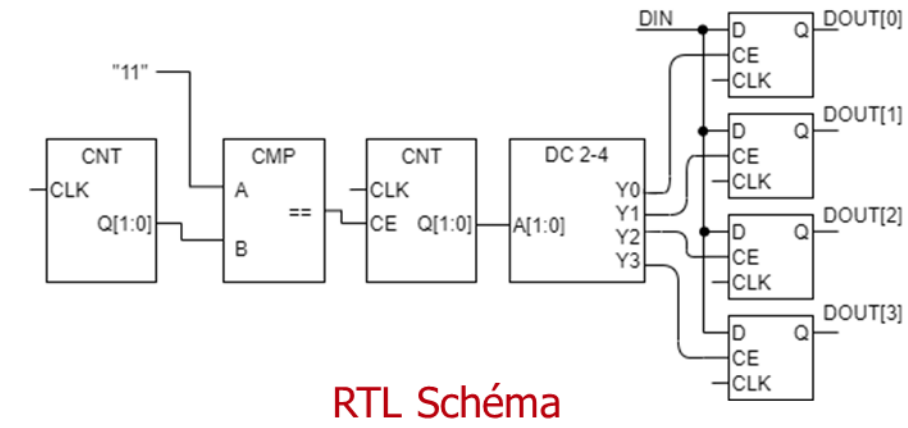
Zásady HDL programování

- Číslicový návrh nefunguje jako imperativní programování
  - komplexní funkcionalita se nevyjadřuje jako sekvence příkazů
  - všechny komponenty pracují neustále a vzájemně paralelně
  - obvody skládáme ze známých kousků jako puzzle nebo lego
  - spojením jednoduchých komponent postavíme složitější

- Obvod nepopisovat jako jeden velký VHDL proces řízený CLK
  - všechny výstupy jsou registrovány a o takt později
  - RTL schéma nelze v kódu snadno rozpoznat
  - kód je obecně hůře čitelný a pochopitelný
- **Každá RTL součástka popsána samostatným procesem**

VHDL procesy a RTL komponenty

```
p_all: process(CLK, RST)
begin
   if (RST='1') then
      cnt <= "00";
      cnt_out <= "00";
      dout <= "0000";
   elsif (CLK'event) and (CLK='1') then
      cnt <= cnt + 1;
      if (cnt = "11") then
         cnt_out_ce <= '1';
      else
         cnt_out_ce <= '0';
      end if;
      if (cnt_out_ce='1') then
         cnt_out <= cnt_out + 1;
      end if;
      dout <= "0000";
      case cnt_out is
         when "00" => dout(0) <= din;
         when "01" => dout(1) <= din;
         when "10" => dout(2) <= din;
         when "11" => dout(3) <= din;
         when others => dout <= "0000";
      end case;
   end if;
end process;
```



RTL Schéma

```
p_cnt1: process(CLK, RST)
begin
   if (RST='1') then
      cnt_out <= "00";
   elsif (CLK'event) and (CLK='1') then
      cnt <= cnt + 1;
   end if;
end process;
```

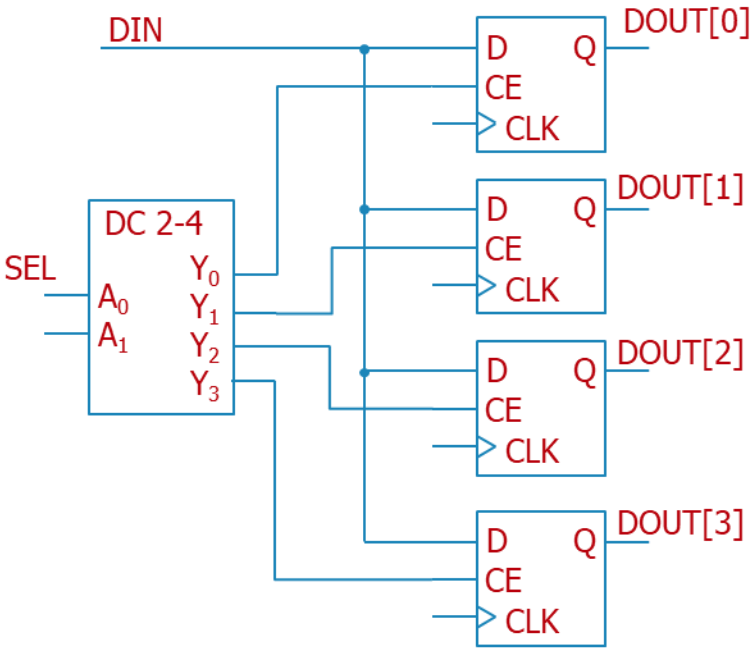
```
p_cmp: process(cnt)
begin
   if (cnt = "11") then
      cnt_out_ce <= '1';
   else
      cnt_out_ce <= '0';
   end if;
end process;
```

```
p_cnt2: process(CLK)
Begin
   if (RST='1') then
      cnt_out <= "00";
   elsif (CLK'event) and (CLK='1') then
      if (cnt_out_ce='1') then
         cnt_out <= cnt_out + 1;
      end if;
   end if;
end process;
```

```
p_dec_reg: process(CLK, RST)
begin
   if (RST='1') then
      dout <= "0000";
   elsif (CLK'event) and (CLK='1') then
      case sel is
         when "00" => dout(0) <= din;
         when "01" => dout(1) <= din;
         when "10" => dout(2) <= din;
         when "11" => dout(3) <= din;
         when others => dout <= "0000";
      end case;
   end if;
end process;
```

VHDL procesy a RTL komponenty

```
p_dec_reg: process(CLK, RST)
begin
   if (RST='1') then
      dout <= "0000";
   elsif (CLK'event) and (CLK='1') then
      case sel is
         when "00" => dout(0) <= din;
         when "01" => dout(1) <= din;
         when "10" => dout(2) <= din;
         when "11" => dout(3) <= din;
      end case;
   end if;
end process;
```




```
p_dec: process(sel)
begin
   dout_ce <= "0000";
   case sel is
      when "00" => dout_ce <= "0001";
      when "01" => dout_ce <= "0010";
      when "10" => dout_ce <= "0100";
      when "11" => dout_ce <= "1000";
      when others => dout_ce <= "0000";
   end case;
end process;
```

```
p_reg0: process(CLK, RST)
begin
   if (RST='1') then
      dout(0) <= '0';
   elsif (CLK'event) and (CLK='1') then
      if (dout_ce(0)='1') then
         dout(0) <= din
      end if;
   end if;
end process;
```

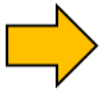
...

```
p_reg3: process(CLK, RST)
begin
   if (RST='1') then
      dout(3) <= '0';
   elsif (CLK'event) and (CLK='1') then
      if (dout_ce(3)='1') then
         dout(3) <= din
      end if;
   end if;
end process;
```

Zkrácené formy zápisu (bez procesu)

Komparátor

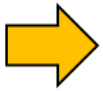
```
p_cmp: process(X)
begin
   if (X = "11") then
      Y <= '1';
   else
      Y <= '0';
   end if;
end process;
```



```
Y <= '1' when (X = "11") else '0';
```

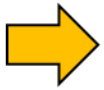
Multiplexor

```
p_mux2: process(SEL, X0, X1)
begin
   if (SEL = '1') then
      Y <= X1;
   else
      Y <= X0;
   end if;
end process;
```



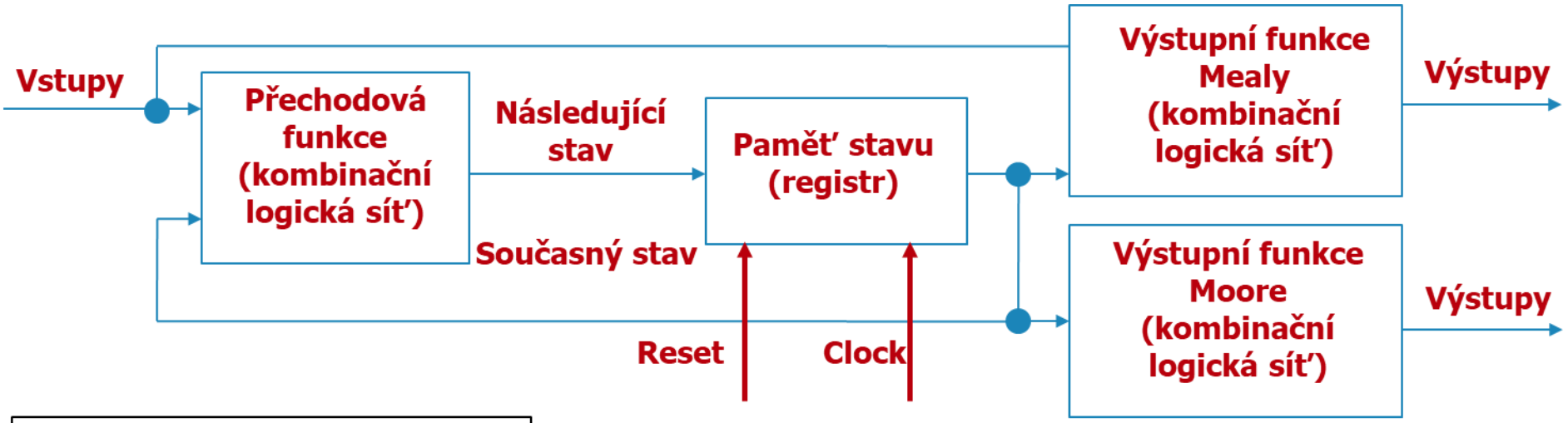
```
Y <= X1 when (SEL = '1') else X0;
```

```
p_mux4 : process(X1,X2,X3,X4,SEL)
begin
  case SEL is
    when "00" => Y <= X0 ;
    when "01" => Y <= X1 ;
    when "10" => Y <= X2 ;
    when "11" => Y <= X3 ;
    when others => Y <= '0';
  end case;
end process p_mux;
```



```
with SEL select
    Y <= X0 when "00",
         X1 when "01",
         X2 when "10",
         X3 when "11",
        '0' when others;
```

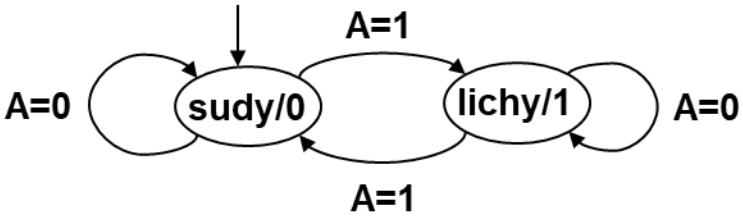
Správná implementace automatu (3 procesy)




```
nstate_logic: process(pstate, A)
begin
   case pstate is
      when S_sudy =>
         nstate <= S_sudy;
         if (A='1') then
            nstate <= S_lichy;
         end if;
      when S_lichy =>
         nstate <= S_lichy;
         if (A='1') then
            nstate <= S_sudy;
         end if;
      when others =>
         nstate <= S_sudy;
   end case;
end process;
```

```
pstatereg: process(RST, CLK)
begin
   if (RST='1') then
      pstate <= S_sudy;
   elsif (CLK'event) and
         (CLK='1') then
      pstate <= nstate;
   end if;
end process;
```

```
output_logic: process(pstate)
begin
   case pstate is
      when S_sudy =>
         Y <= '0';
      when S_lichy =>
         Y <= '1';
      when others =>
         Y <= '0';
   end case;
end process;
```



Zákeřné chyby implementace

- Neúplný sensitivity list procesu
  - **Projev:** chování obvodu v simulaci je jiné než očekáváte
  - **Odhalení:** varování ve výpisu průběhu syntézy obvodu
  - **Nejčastější výskyt:**
    - procesy FSM, kromě vstupů FSM se používá také aktuální stav
    - kombinační obvody, např. SEL nebo D při MUX/DEMUX
    - registry, reset signál při implementaci s asynchronním resetem

```
nstate_logic: process(pstate, A)
begin
   case pstate is
      when S_sudy =>
         nstate <= S_sudy;
         if (A='1') then
            nstate <= S_lichy;
         end if;
      when S_lichy =>
         nstate <= S_lichy;
         if (A='1') then
            nstate <= S_sudy;
         end if;
      when others =>
         nstate <= S_sudy;
   end case;
end process;
```

```
p_mux4 : process(X1,X2,X3,X4,SEL)
begin
  case SEL is
     when "00" => Y <= X0 ;
     when "01" => Y <= X1 ;
     when "10" => Y <= X2 ;
     when "11" => Y <= X3 ;
     when others => Y <= '0';
  end case;
end process p_mux;
```

```
pstatereg: process(RST, CLK)
begin
   if (RST='1') then
      pstate <= S_sudy;
   elsif (CLK'event) and
         (CLK='1') then
      pstate <= nstate;
   end if;
end process;
```

Zákeřné chyby implementace

- Nechtěný latch v kombinačním obvodu nebo místo registru
  - **Projev:** chování po syntéze může být jiné než očekáváte
  - **Odhalení:** varování ve výpisu průběhu syntézy obvodu
  - **Nejčastější výskyt:**
    - procesy FSM, definovat přirazení implicitní hodnoty všech výstupů nebo definovat jejich hodnotu ve všech situacích
    - kombinační obvody, nepokrytí všech možností if nebo case
    - registry, nepoužívat hladinové, všechny taktovat hodinami CLK

```
output_logic: process(pstate)
begin
    case pstate is
        when S_sudy =>
            Y <= '0';
        when S_lichy =>
            Y <= '1';
        when others =>
            Y <= '0';
    end case;
end process;
```

```
p_demux: process(din, sel)
begin
    dout <= "0000";
    case sel is
        when "00" => dout(0) <= din;
        when "01" => dout(1) <= din;
        when "10" => dout(2) <= din;
        when "11" => dout(3) <= din;
        when others => dout <= "0000";
    end case;
end process;
```

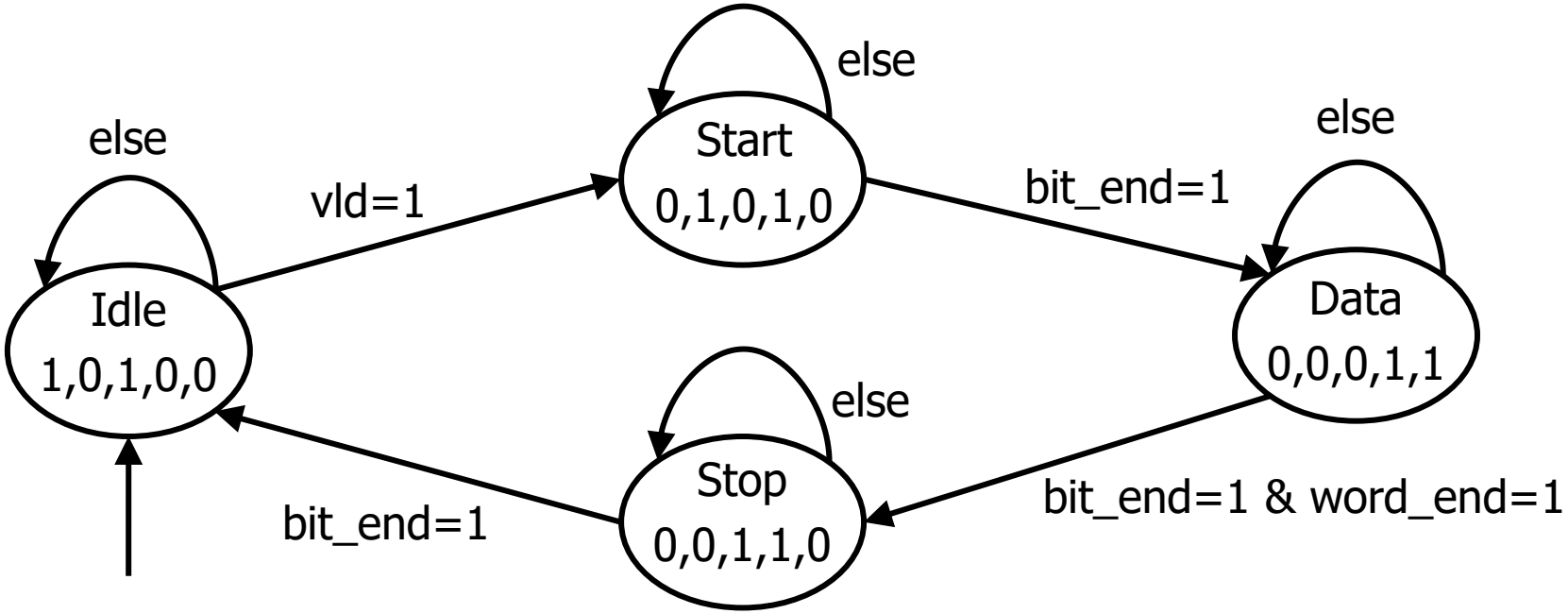
```
p_reg0: process(CLK, RST)
begin
    if (RST='1') then
        dout(0) <= '0';
    elsif (CLK'event) and
          (CLK='1') then
        if (dout_ce(0)='1') then
            dout(0) <= din
        end if;
    end if;
end process;
```

Stavy automatu: Idle, Start, Data, Stop

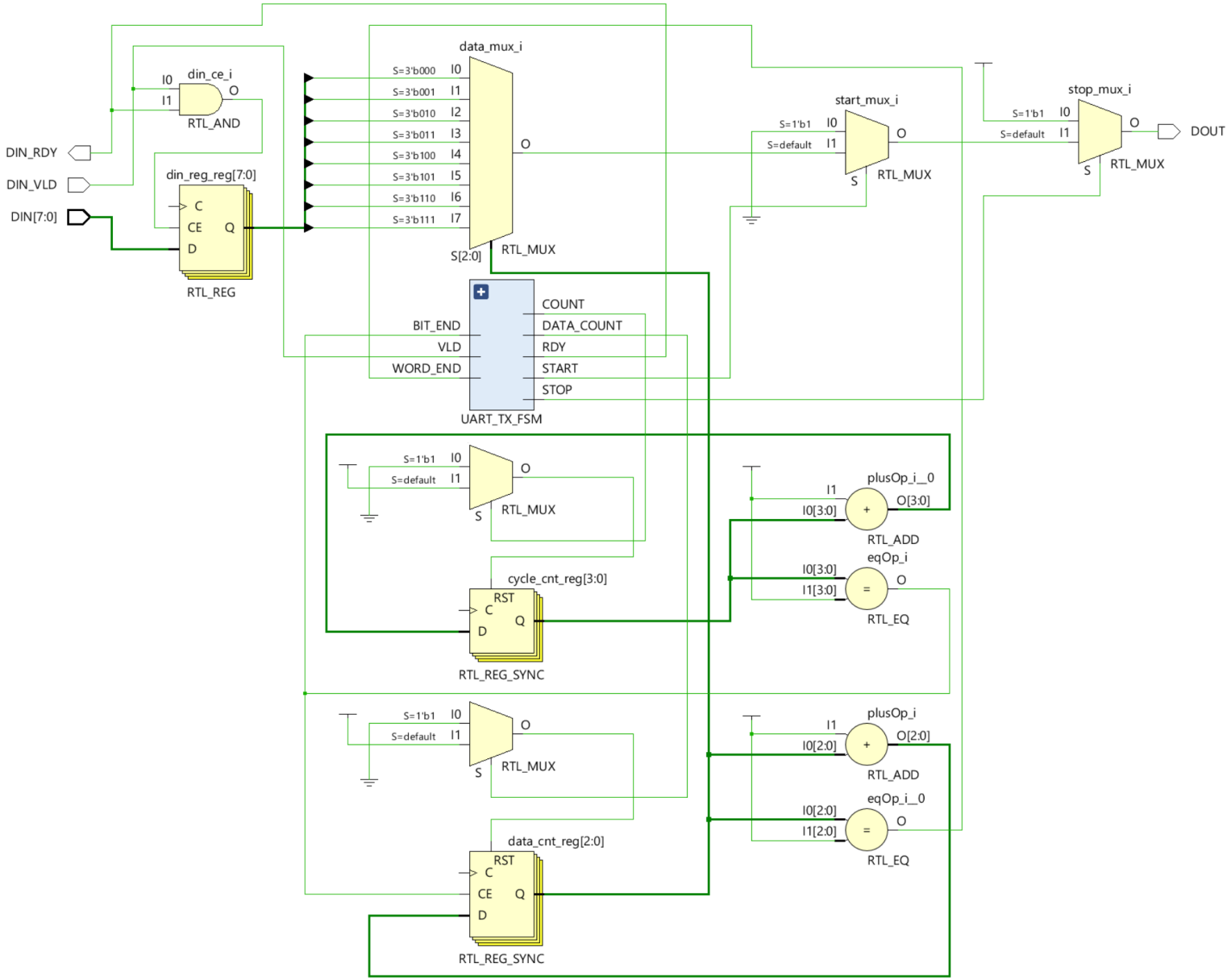
Vstupní signály: vld, bit_end, word_end

Mealyho výstupy: <žádné>

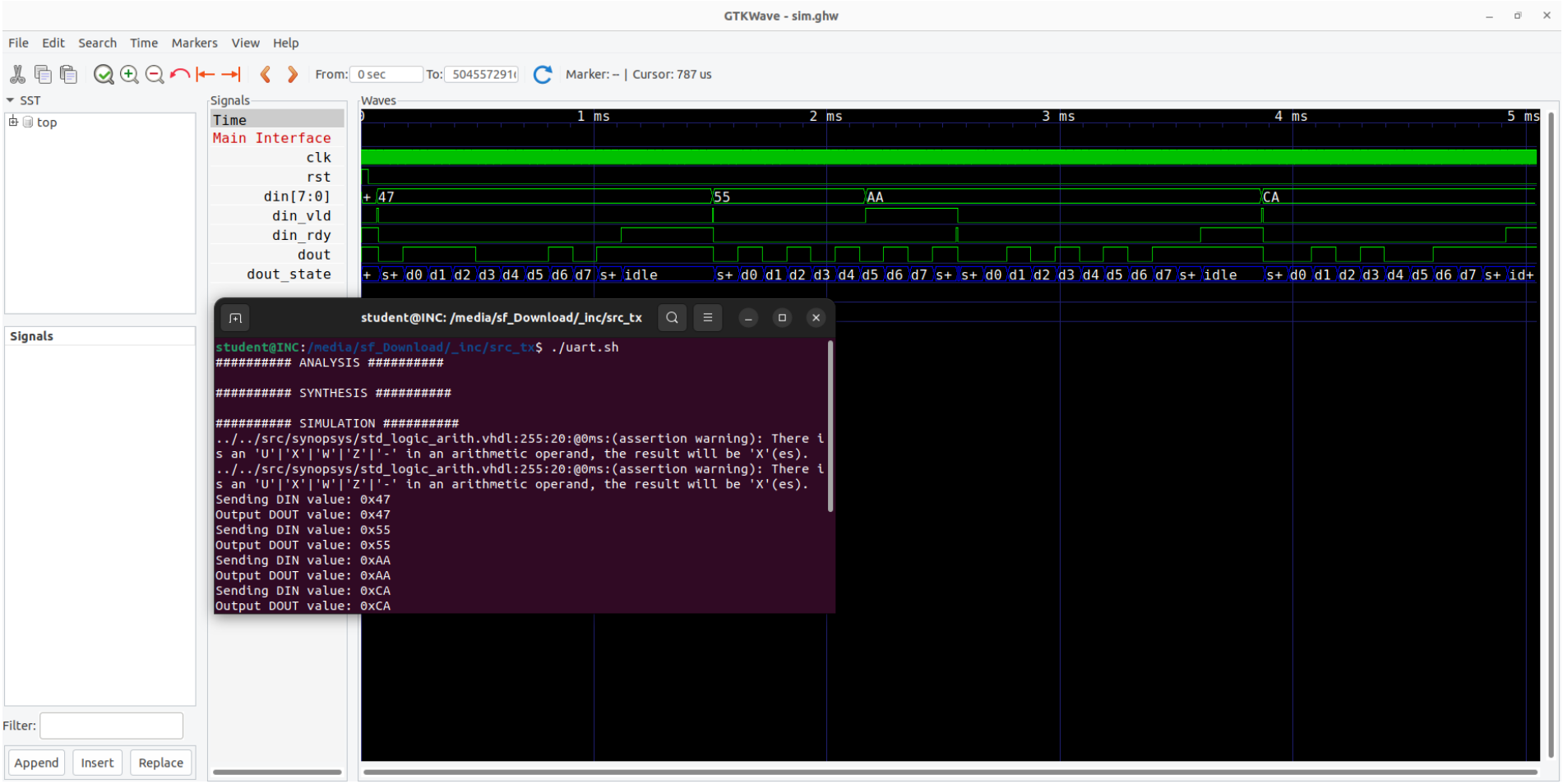
Moorovy výstupy: rdy, start, stop, count, data_count



Implementace RTL



Simulace


GTKWave - sim.ghw
File Edit Search Time Markers View Help
From: 0 sec
To:
Marker: -- | Cursor: 787 us
SST
top
Signals
Filter:
Append Insert Replace
Signals
Time
Main Interface
clk
rst
din[7:0]
din_vld
din_rdy
dout
dout_state
Waves
1 ms
2 ms
3 ms
4 ms
5 ms
47
55
AA
CA
idle
student@INC: /media/sf_Download/_inc/src_tx
student@INC:/media/sf_Download/_inc/src_tx$ ./uart.sh
########## ANALYSIS ##########
########## SYNTHESIS ##########
########## SIMULATION ##########
../../src/synopsys/std_logic_arith.vhdl:255:20:@0ms:(assertion warning): There i
s an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
../../src/synopsys/std_logic_arith.vhdl:255:20:@0ms:(assertion warning): There i
s an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
Sending DIN value: 0x47
Output DOUT value: 0x47
Sending DIN value: 0x55
Output DOUT value: 0x55
Sending DIN value: 0xAA
Output DOUT value: 0xAA
Sending DIN value: 0xCA
Output DOUT value: 0xCA

Děkuji za pozornost!