SigTest1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity SigTest1 is
  Port ( Ind :   in  std_logic_vector(1 downto 0);
         Ud : out  std_logic_vector(1 downto 0);
         Clk :  in  std_logic);
end SigTest1;

architecture Behavioral of SigTest1 is
   Signal A, B, C: std_logic_vector(1 downto 0);
begin

   process( Clk)
      -- Variable A, B, C: std_logic_vector(1 downto 0);
   begin
      if Rising_edge( Clk) then
         Ud <=  C;
         C  <=  B;
         B  <=  A;
         A  <=  Ind;
      end if;
   end process;

end Behavioral;
```

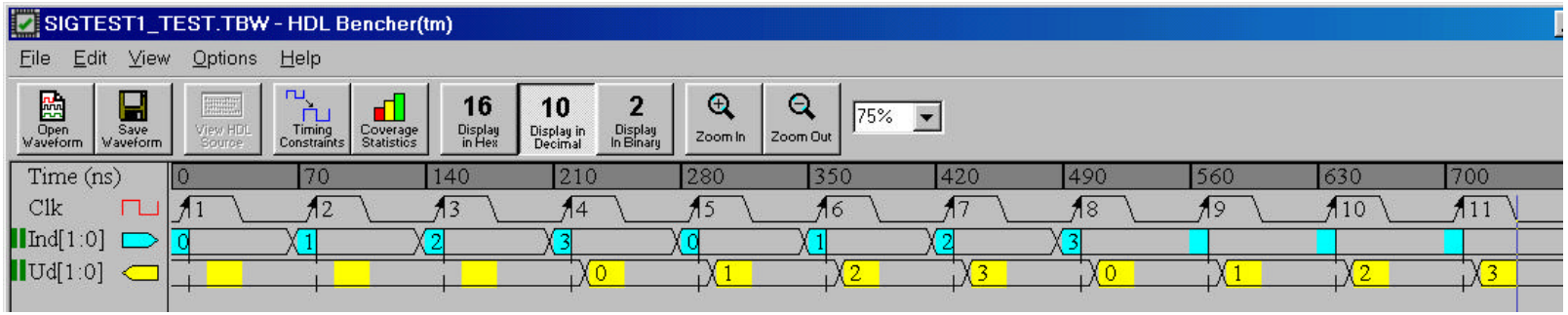
Bemærk rækkefølge

SigTest2

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity SigTest2 is
  Port ( Ind :   in  std_logic_vector(1 downto 0);
         Ud : out  std_logic_vector(1 downto 0);
         Clk :  in  std_logic);
end SigTest2;

architecture Behavioral of SigTest2 is
   Signal A, B, C: std_logic_vector(1 downto 0);
begin

   process( Clk)
      -- Variable A, B, C: std_logic_vector(1 downto 0);
   begin
      if Rising_edge( Clk) then
         A  <=  Ind;
         B  <=  A;
         C  <=  B;
         Ud <=  C;
      end if;
   end process;

end Behavioral;
```

Bemærk rækkefølge

SigTest1 og SigTest2



VarTest1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity VarTest1 is
  Port ( Ind :   in std_logic_vector(1 downto 0);
         Ud : out std_logic_vector(1 downto 0);
         Clk :  in std_logic);
end VarTest1;

architecture Behavioral of VarTest1 is
  -- Signal A, B, C: std_logic_vector(1 downto 0);
begin

  process( Clk)
     Variable A, B, C: std_logic_vector(1 downto 0);
  begin
     if Rising_edge( Clk) then
        Ud <=  C;
        C  :=  B;
        B  :=  A;
        A  :=  Ind;
     end if;
  end process;

end Behavioral;
```

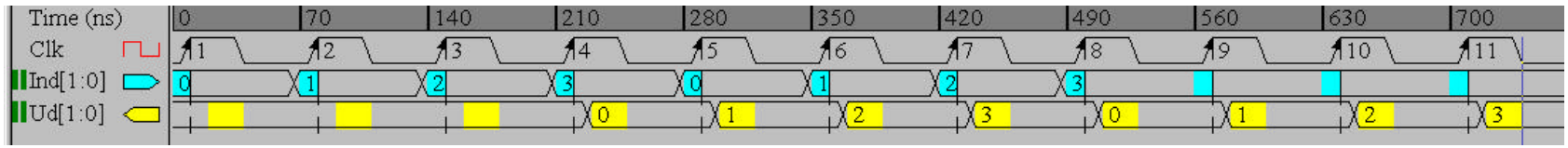
Bemærk rækkefølge

VarTest2

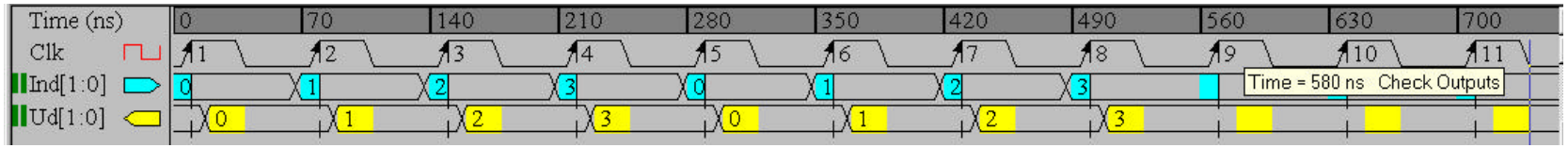
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity VarTest2 is
  Port ( Ind :   in std_logic_vector(1 downto 0);
         Ud : out std_logic_vector(1 downto 0);
         Clk :  in std_logic);
end VarTest2;

architecture Behavioral of VarTest2 is
  -- Signal A, B, C: std_logic_vector(1 downto 0);
begin

  process( Clk)
     Variable A, B, C: std_logic_vector(1 downto 0);
  begin
     if Rising_edge( Clk) then
        A  :=  Ind;
        B  :=  A;
        C  :=  B;
        Ud <=  C;
     end if;
  end process;

end Behavioral;
```

Bemærk rækkefølge

VarTest1



VarTest2

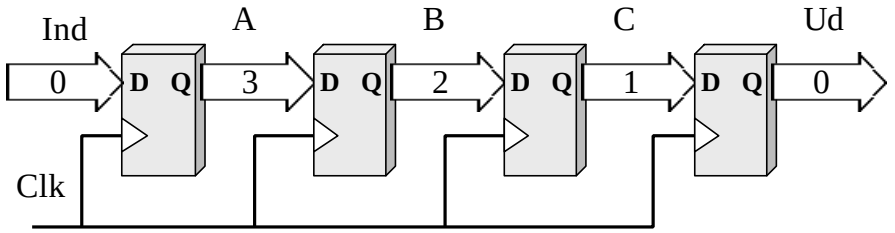


VarTest3

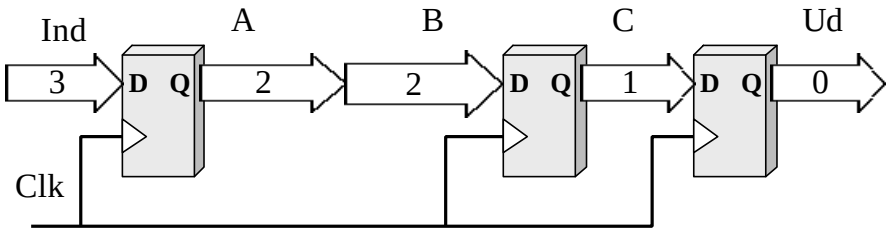
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity VarTest3 is
  Port ( Ind :  in std_logic_vector(1 downto 0);
         Ud : out std_logic_vector(1 downto 0);
         Clk :  in std_logic);
end VarTest3;

architecture Behavioral of VarTest3 is
  -- Signal A, B, C: std_logic_vector(1 downto 0);
begin

  process( Clk)
      Variable A, B, C: std_logic_vector(1 downto 0);
   begin
      if Rising_edge( Clk) then
         Ud <=  C;
         B  :=  A;
         C  :=  B;
         A  :=  Ind;
      end if;
   end process;

end Behavioral;
```

Bemærk rækkefølge

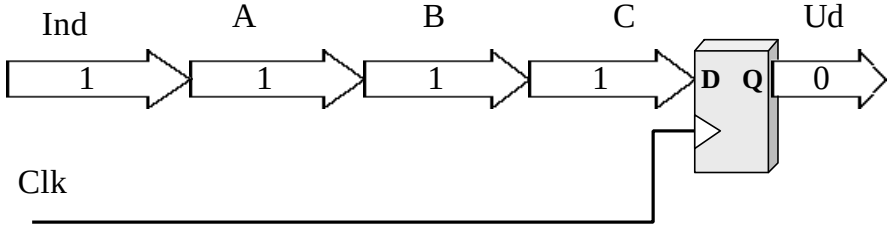
SigTest1, SigTest2 og VarTest1



VarTest3



VarTest2



VarTest3