KONTROL UNITATEA

```
library IEEE;
use IEEE.std_logic_1164.all;

-- Deskribapen hau ASM algoritmoari dagokio.

entity KU_AU_OUT is
port (
clk,reset: in std_logic;
ENABLE, DACLRCSINK, BCLKSINK, AZKENA: in std_logic;
LDSAMPLE, LDBEG, DESEZK, DEKBEG, READYOUT, AUKBIT: out std_logic
);
end KU_AU_OUT;

architecture arki of KU_AU_OUT is

type EGOERA is (e0,e1,e2,e3,e4,e5,e6,e7,e8);
signal UE,HE: EGOERA;

begin -- arkitekturaren definizioa

process (ENABLE, DACLRCSINK, BCLKSINK, AZKENA)
begin
case UE is
 when e0 => if ENABLE='1' then HE<=e1;
                    else HE<=e0;
           end if;
 when e1 => HE<= e2;
 when e2 => if DACLRCSINK='0' then HE<= e3;
                    else HE<= e2;
              end if;
 when e3 => if DACLRCSINK='1' and BCLKSINK='0' then HE<=e4;
else HE<= e3;
   end if;
 when e4 => if BCLKSINK='1' then HE<=e5;
        else HE<=e4;
   end if;
  when e5 => if BCLKSINK='0' then HE<=e6;
```

```
       else HE<=e5;
   end if;
  when e6 => if AZKENA='0' then HE<=e4;
else HE<=e7;
      end if;
  when e7 => if DACLRCSINK='1' then HE<=e8;
else HE<=e0;
end if;
  when e8 => if DACLRCSINK='0' and BCLKSINK='0' then HE<=e4;
else HE<=e8;
end if;

end case;
end process;

-- prozesu sinkrono batean, uneko egoera (UE) erregistratzen da erloju-ertzero.
-- reset asinkronoa
process (clk,reset)
begin
if reset='1' then UE<=e0;
          elsif (clk'event and clk='1') then UE<=HE;
end if;
end process;
-- kontrol-seinale bakoitzeko, esleipen konkurrente bat
LDSAMPLE <= '1' when (UE=e0 and ENABLE='1') else '0';
LDBEG <= '1' when ((UE=e0 and ENABLE='1') or (UE=e8 and DACLRCSINK='0' and
BCLKSINK='0')) else '0';
DESEZK <= '1' when (UE=e5 and BCLKSINK='0') else '0';
DEKBEG <= '1' when (UE=e5 and  BCLKSINK='0') else '0';
AUKBIT <= '1' when (UE=e4) else '0';
READYOUT <=  '1' when (UE=e1) else '0';

end arki; -- arkitekturaren bukaera
```

KONTROL UNITATEA ETA PROZESU UNITATEA

```
library IEEE;
use IEEE.std_logic_1164.all;
use ieee.numeric_std.all;


-- Kronograma eskuz betetzeko erabili dugun ariketa (bigarrena).
-- Diseinuaren helburua: bi zenbaki osoen arteko biderketa egitea
-- Zenbakiak 2rako osagarrian adierazita daude.
-- Diseinua (ASM+PU) eta kronograma betetzeko eskema egelan daude eskura.
-- Deskribapen hau ASM algoritmoari dagokio.


entity AU_OUT is
port (
clk,reset: in std_logic;
ENABLE: in std_logic;
SAMPLE: in std_logic_vector (15 downto 0);
DACDAT: out std_logic;
BCLK: in std_logic;
DACLRC: in std_logic
);
end AU_OUT;


architecture arki_out of AU_OUT is


 --erazagupenak: osagaiak eta barne seinaleak
component KU_AU_OUT
port (
clk,reset: in std_logic;
ENABLE, AZKENA, DACLRCSINK, BCLKSINK: in std_logic;
LDSAMPLE, LDBEG, DESEZK, DEKBEG, READYOUT, AUKBIT: out std_logic
);
end component;


 signal LDSAMPLE,LDBEG,AUKBIT : std_logic;
 signal DESEZK,DEKBEG: std_logic;
 signal DACLRCSINK, BCLKSINK, AZKENA, READYOUT: std_logic;
 signal Q_KONT: unsigned (4 downto 0);
 signal Q_SAMPLE: std_logic_vector (15 downto 0);
```

```
begin -- arkitekturaren hasiera

 ku: KU_AU_OUT
 port map (
    clk => clk,
    reset => reset, ENABLE => ENABLE,
    AZKENA => AZKENA, LDSAMPLE => LDSAMPLE, LDBEG => LDBEG, DESEZK => DESEZK,
DEKBEG => DEKBEG, AUKBIT => AUKBIT, READYOUT => READYOUT, BCLKSINK =>
BCLKSINK, DACLRCSINK => DACLRCSINK
    );


  process(clk)
  begin
    if clk'event and clk='1'  then
      if LDSAMPLE='1'  then Q_SAMPLE<=SAMPLE;
      elsif DESEZK='1' then
            for I in 1 to 15 loop
               Q_SAMPLE(I)<=Q_SAMPLE(I-1);
            end loop;
         Q_SAMPLE(0)<=Q_SAMPLE(15);
         end if;
      end if;
end process;



-- D biegonkorrak
process(clk)
begin
if CLK'event and CLK='1' then
BCLKSINK <= BCLK;
end if;
end process;

process(clk)
begin
if CLK'event and CLK='1' then
DACLRCSINK<= DACLRC;
end if;
end process;
```

```
    -- emaitza-erregistroaren aurreko multiplexorea
    DACDAT<= Q_SAMPLE(15) when AUKBIT='1' else '0';

    -- Kontagailua
    process(clk)
    begin
     if clk'event and clk='1' then
       if LDBEG='1' then Q_KONT<="10000";
         elsif DEKBEG='1' then Q_KONT<= Q_KONT-1;
       end if;
     end if;
    end process;
  AZKENA<='1' when (Q_KONT="00000") else '0';


end arki_out;
```

TESTBENCH-A

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

--Biderkagailua simulatzeko testbench-a

entity AU_OUT_TB IS

end AU_OUT_TB;

architecture AU_OUT_TB_ARCH of AU_OUT_TB is
    signal reset : std_logic;
      signal clk : std_logic:='0';
      signal ENABLE : std_logic;
    signal DACDAT: std_logic;
    signal BCLK: std_logic:='1';
    signal DACLRC: std_logic:='0';
    signal SAMPLE : std_logic_vector (15 downto 0);

    component AU_OUT IS
     port (
            clk,reset: in std_logic;
        ENABLE: in std_logic;
        SAMPLE: in std_logic_vector (15 downto 0);
        DACDAT: out std_logic;
        BCLK: in std_logic;
        DACLRC: in std_logic
          );
    end component;

begin

U1: AU_OUT port map (
  reset=>reset,clk=>clk,ENABLE=>ENABLE,SAMPLE=>SAMPLE,DACDAT=>DACDAT,
BCLK=>BCLK, DACLRC=>DACLRC
```

```
  );

clk<= not clk after 10 ns;
BCLK<= not BCLK after 160 ns;
DACLRC<= not DACLRC after 61440 ns; --200 ns;--

process
begin
 SAMPLE<="1010101001010101";
 wait for 20 ns;
 ENABLE<='0';
 reset<='1';
 wait for 5 ns;
 reset<='0';
 wait for 15 ns;
 ENABLE<='1';
 wait for 20 ns;
 ENABLE<='0';
 wait;
end process;

end AU_OUT_TB_ARCH;
```