

## FSM TESTBENCH

-- **** STUDENT: 64000225 .....	2
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	2
-- **** STUDENT: 64190088 .....	5
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	5
-- **** STUDENT: 64200163 .....	8
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	8
-- **** STUDENT: 64200296 .....	11
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	11
-- **** STUDENT: 64200385 .....	14
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	14
-- **** STUDENT: 64210132 .....	17
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	17
-- **** STUDENT: 64210290 .....	20
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	20
-- **** STUDENT: 64210382 .....	23
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	23
-- **** STUDENT: 64210384 .....	26
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	26
-- **** STUDENT: 64210445 .....	29
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	29
-- **** STUDENT: 64210455 .....	32
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	32
-- **** STUDENT: 64210457 .....	35
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	35
-- **** PREDLOGA VAJE .....	38
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	38

```

-- *****
-- **** STUDENT: 64000225
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,      start => start,    rco => rco,  init => init,      load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
    BEGIN
      write( RES_LINE, clk, right, 1 );
      write( RES_LINE, string'( "," ) );

      write( RES_LINE, nRST, right, 1 );
      write( RES_LINE, string'( "," ) );

      write( RES_LINE, start, right, 1 );
      write( RES_LINE, string'( "," ) );

      write( RES_LINE, rco, right, 1 );
      write( RES_LINE, string'( "," ) );

      write( RES_LINE, init, right, 1 );
      write( RES_LINE, string'( "," ) );

      write( RES_LINE, load, right, 1 );
      write( RES_LINE, string'( "," ) );

      write( RES_LINE, done, right, 1 );

      writeline( RESULTS, RES_LINE );
    END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start   <= '1';
        WAIT FOR PERIOD;
        start   <= '0';
        WAIT FOR 7 * PERIOD;
        rco     <= '1';
        WAIT FOR PERIOD;
        rco     <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64190088
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,      start => start,    rco => rco,  init => init,      load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;
END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start   <= '1';
        WAIT FOR PERIOD;
        start   <= '0';
        WAIT FOR 7 * PERIOD;
        rco     <= '1';
        WAIT FOR PERIOD;
        rco     <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64200163
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```



```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;
END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start <= '1';
        WAIT FOR PERIOD;
        start <= '0';
        WAIT FOR 7 * PERIOD;
        rco    <= '1';
        WAIT FOR PERIOD;
        rco    <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64200296
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,      start => start,    rco => rco,  init => init,      load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;
END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start <= '1';
        WAIT FOR PERIOD;
        start <= '0';
        WAIT FOR 7 * PERIOD;
        rco    <= '1';
        WAIT FOR PERIOD;
        rco    <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64200385
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,      start => start,    rco => rco,  init => init,      load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;
END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start   <= '1';
        WAIT FOR PERIOD;
        start   <= '0';
        WAIT FOR 7 * PERIOD;
        rco     <= '1';
        WAIT FOR PERIOD;
        rco     <= '0';
        WAIT;
    END PROCESS;

END rtl;

```



```

-- *****
-- **** STUDENT: 64210132
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start   <= '1';
        WAIT FOR PERIOD;
        start   <= '0';
        WAIT FOR 7 * PERIOD;
        rco     <= '1';
        WAIT FOR PERIOD;
        rco     <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start   <= '1';
        WAIT FOR PERIOD;
        start   <= '0';
        WAIT FOR 7 * PERIOD;
        rco     <= '1';
        WAIT FOR PERIOD;
        rco     <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64210382
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,      start => start,    rco => rco,  init => init,      load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );

```



```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start   <= '1';
        WAIT FOR PERIOD;
        start   <= '0';
        WAIT FOR 7 * PERIOD;
        rco     <= '1';
        WAIT FOR PERIOD;
        rco     <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64210384
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;
END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start  <= '1';
        WAIT FOR PERIOD;
        start  <= '0';
        WAIT FOR 7 * PERIOD;
        rco    <= '1';
        WAIT FOR PERIOD;
        rco    <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64210445
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start  <= '1';
        WAIT FOR PERIOD;
        start  <= '0';
        WAIT FOR 7 * PERIOD;
        rco    <= '1';
        WAIT FOR PERIOD;
        rco    <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64210455
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```



```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start   <= '1';
        WAIT FOR PERIOD;
        start   <= '0';
        WAIT FOR 7 * PERIOD;
        rco     <= '1';
        WAIT FOR PERIOD;
        rco     <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** STUDENT: 64210457
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;
END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start <= '1';
        WAIT FOR PERIOD;
        start <= '0';
        WAIT FOR 7 * PERIOD;
        rco    <= '1';
        WAIT FOR PERIOD;
        rco    <= '0';
        WAIT;
    END PROCESS;

END rtl;

```

```

-- *****
-- **** PREDLOGA VAJE
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: Ni pripomb
-- *****

library IEEE;
use IEEE.numeric_std.all;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY CORDIC_FSM_tb IS
END CORDIC_FSM_tb;

ARCHITECTURE rtl OF CORDIC_FSM_tb IS
FILE RESULTS: TEXT OPEN WRITE_MODE IS "CORDIC_FSM.csv";

COMPONENT CORDIC_FSM
PORT (
    clk : In std_logic;
    nRST : In std_logic;
    start : In std_logic;
    rco : In std_logic;
    init : Out std_logic;
    load : Out std_logic;
    done : Out std_logic
);
END COMPONENT;

SIGNAL      clk : std_logic := '0';
SIGNAL      nRST : std_logic := '0';
SIGNAL      start : std_logic := '0';
SIGNAL      rco : std_logic := '0';
SIGNAL      init : std_logic := '0';
SIGNAL      load : std_logic := '0';
SIGNAL      done : std_logic := '0';

constant    PERIOD : time := 200 ns;
constant    DUTY_CYCLE : real := 0.5;

```

```

constant    OFFSET : time := 100 ns;

BEGIN
  UUT : CORDIC_FSM
  PORT MAP (
    clk => clk,  nRST => nRST,    start => start,    rco => rco,  init => init,    load => load,
    done => done
  );

  PROCESS    -- clock process for clk
    PROCEDURE Log_variables( clk, nRST, start, rco, init, load, done : std_logic ) IS
      VARIABLE RES_LINE : LINE;
      BEGIN
        write( RES_LINE, clk, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, nRST, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, start, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, rco, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, init, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, load, right, 1 );
        write( RES_LINE, string'( "," ) );

        write( RES_LINE, done, right, 1 );

        writeline( RESULTS, RES_LINE );
      END;

  BEGIN
    WAIT for OFFSET;
    CLOCK_LOOP : LOOP
      clk  <= '0';
      WAIT FOR ( PERIOD - ( PERIOD * DUTY_CYCLE ) );
    END LOOP;
  END;
END;

```

```

        clk    <= '1';
        WAIT FOR ( PERIOD * DUTY_CYCLE );
        Log_variables( clk, nRST, start, rco, init, load, done );
    END LOOP CLOCK_LOOP;
END PROCESS;

PROCESS
    variable HDR_line : LINE;
    BEGIN
        write( HDR_line, string'( " clk, nRST, start, rco, init, load, done" ) );
        writeline( RESULTS, HDR_line );
        WAIT FOR OFFSET;
        nRST    <= '1';
        WAIT FOR OFFSET;
        nRST    <= '0';
        WAIT FOR PERIOD;
        nRST    <= '1';
        WAIT FOR PERIOD;
        start <= '1';
        WAIT FOR PERIOD;
        start <= '0';
        WAIT FOR 7 * PERIOD;
        rco    <= '1';
        WAIT FOR PERIOD;
        rco    <= '0';
        WAIT;
    END PROCESS;

END rtl;

```



