

FSM

-- **** STUDENT: 64000225	2
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	2
-- **** STUDENT: 64190088	4
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	4
-- **** STUDENT: 64200163	6
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	6
-- **** STUDENT: 64200296	8
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	8
-- **** STUDENT: 64200385	10
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	10
-- **** STUDENT: 64210132	12
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	12
-- **** STUDENT: 64210290	14
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	14
-- **** STUDENT: 64210382	16
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	16
-- **** STUDENT: 64210384	18
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	18
-- **** STUDENT: 64210445	20
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	20
-- **** STUDENT: 64210455	22
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	22
-- **** STUDENT: 64210457	24
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	24
-- **** PREDLOGA VAJE	26
-- KOMENTARJI K OCENI NALOGE -- Matej Možek: Ni pripomb.....	26

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;

BEGIN

    state_ctrl: process ( clk, nRST ) is
    begin
        if nRST = '0' then
            state <= IDLE;
        elsif rising_edge( clk ) then
            state <= next_state;
        end if;
    end process;

    next_state_calc: process ( state, start, rco ) is
    begin
        case state is
            when IDLE =>
                if start = '1' then
                    next_state <= CALC;
                end if;
            when CALC =>
                if rco = '0' then
                    next_state <= FINISHED;
                end if;
            when FINISHED =>
                if load = '1' then
                    state <= IDLE;
                end if;
            when others =>
                next_state <= state;
            end case;
        end process;
    end next_state_calc;

```

```

        else
            next_state <= IDLE;
        end if;
        when CALC =>
            if rco = '1' then
                next_state <= FINISHED;
            else
                next_state <= CALC;
            end if;
        when FINISHED =>
            if start = '1' then
                next_state <= FINISHED;
            else
                next_state <= IDLE;
            end if;
        end case;
    end process;

    init <= '1' when state = IDLE else '0';
    load <= '1' when state = CALC else '0';
    done <= '1' when state = FINISHED else '0';

```

```

END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN

        process( clk, start, rco )
        begin
            case state is
                when IDLE =>
                    if start = '1' then
                        next_state <= CALC;
                    else
                        next_state <= IDLE;
                    end if;

                when CALC =>
                    if rco = '1' then
                        next_state <= FINISHED;
                    else
                        next_state <= CALC;
                    end if;
            end case;
        end process;
    end rtl;

```

```

        when FINISHED =>
            if rco = '1' then
                next_state <= FINISHED;
            else
                next_state <= IDLE;
            end if;
        end case;
    end process;

    process( clk, nRST )
    begin
        if nRST = '0' then
            state <= IDLE;
        elsif rising_edge( clk ) then
            state <= next_state;
        end if;
    end process;

    init <= '1' when state = IDLE else '0';
    load <= '1' when state = CALC else '0';
    done <= '1' when state = FINISHED else '0';

END rtl;

```

```

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

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

entity CORDIC_FSM is
    port(
        clk, nRST, start, rco : in std_logic;
        init, load, done : out std_logic
    );
end CORDIC_FSM;

architecture rtl of CORDIC_FSM is
    type STATE_TYPE is ( IDLE, CALC, FINISHED );
    signal state, next_state : STATE_TYPE := IDLE;
begin
    vstop_v_stanje : process( clk, nRST ) is
    begin
        if nRST = '0' then
            state <= IDLE;
        elsif rising_edge( clk ) then
            state <= next_state;
        end if;
    end process;
    prehanje_stanj : process( start, state, rco ) is
    begin
        case state is
            when IDLE =>
                if start = '1' then
                    next_state <= CALC;
                else
                    next_state <= IDLE;
                end if;
            when CALC =>
                if rco = '1' then

```

```

        next_state    <= FINISHED;
    else
        next_state    <= CALC;
    end if;
    when FINISHED =>
        if start = '0' then
            next_state    <= IDLE;
        else
            next_state    <= FINISHED;
        end if;
    end case;
end process;
init    <= '1' when state = IDLE else '0';
load    <= '1' when state = CALC else '0';
done    <= '1' when state = FINISHED else '0';
end rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN

        state_proc : process( clk, nRST )
            begin
                if nRST = '0' then
                    state <= IDLE;
                elsif rising_edge( clk ) then
                    state <= next_state;
                end if;
            end process;

        next_state_proc : process( nRST, start, rco, state )
            begin
                if nRST = '0' then
                    next_state <= IDLE;
                elsif state = IDLE then
                    if start = '1' then
                        next_state <= CALC;
                    else

```



```

        next_state    <= IDLE;
    end if;
    elsif state = CALC then
        if rco = '1' then
            next_state    <= FINISHED;
        else
            next_state    <= CALC;
        end if;
    elsif state = FINISHED then
        if start = '0' then
            next_state    <= IDLE;
        else
            next_state    <= FINISHED;
        end if;
    else
        next_state    <= IDLE;
    end if;
end process;

init    <= '1' when state = IDLE else '0';
load    <= '1' when state = CALC else '0';
done    <= '1' when state = FINISHED else '0';

```

```

END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN

        cs: process( nRST, clk )
        begin
            if nRST = '0' then
                state <= IDLE;
            elsif ( clk'event and clk = '1' ) then
                state <= next_state;
            end if;
        end process;

        cns: process( start, rco, state )
        begin
            case state is
            when IDLE =>
                if start = '1' then
                    next_state <= CALC;
                else
                    next_state <= IDLE;
                end if;
            end case;
        end process;
    end architecture rtl;

```

```
end if;
when CALC =>
if rco = '1' then
next_state <= FINISHED;
else
next_state <= CALC;
end if;
when FINISHED =>
if start = '0' then
next_state <= IDLE;
else
next_state <= FINISHED;
end if;
end case;
end process;

load <= '1' when state = CALC else '0';
done <= '1' when state = FINISHED else '0';
init <= '1' when state = IDLE else '0';

END rtl;
```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN

        nRST_proc: process ( clk, nRST ) is
        begin
            if nRST = '0' then
                state <= IDLE;
            elsif rising_edge( clk ) then
                state <= next_state;
            end if;
        end process;

        next_state_proc: process ( state, start, rco ) is
        begin
            case state is
                when IDLE =>
                    if start = '1' then
                        next_state <= CALC;
                    end if;
            end case;
        end process;
    end architecture rtl;

```

```

        when CALC =>
            if rco = '1' then
                next_state    <= FINISHED;
            end if;

            when FINISHED =>
                if start = '0' then
                    next_state    <= IDLE;
                end if;

        end case;
    end process;

    init    <= '1' when state = IDLE else '0';
    load    <= '1' when state = CALC else '0';
    done    <= '1' when state = FINISHED else '0';

END rtl;

```

```

-- *****
-- **** STUDENT: 64210290
-- *****
-- KOMENTARJI K OCENI NALOGE
-- Matej Možek: FSM je sicer pravilno kodiran – razlika je v dveh procesih, ki omogočajo analizo signala next_state.
Tega pri direktni realizaciji (en proces) nimate.
-- *****

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE IEEE.numeric_std.all;

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN STD_LOGIC;
        init, load, done           : OUT STD_LOGIC
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    type STATE_TYPE is ( IDLE, CALC, FINISHED );
    signal      state, next_state : STATE_TYPE := IDLE;

BEGIN

    proc : process ( clk )
    begin
        if nRST='0' then
            state <= IDLE;
        elsif rising_edge( clk ) then
            case state is
                when IDLE =>
                    if start='1' then
                        state <= CALC;
                    else
                        state <= IDLE;
                    end if;
                when CALC =>
                    if rco='1' then
                        state <= FINISHED;
                    end if;
            end case;
        end if;
    end proc;

```

```

        else
            state <= CALC;
        end if;
    when FINISHED =>
        if start='0' then
            state <= IDLE;
        else
            state <= FINISHED;
        end if;
    end case;
end if;
end process;

```

```

init   <= '1' when state=IDLE else '0';
load   <= '1' when state=CALC  else '0';
done   <= '1' when state=FINISHED else '0';

```

```

END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN
    state_ctrl: process ( clk, nRST ) is
        begin
            if nRST = '0' then
                state <= IDLE;
            elsif rising_edge( clk ) then
                state <= next_state;
            end if;
        end process;

        next_state_calc: process ( state, start, rco ) is
            begin
                case state is
                    when IDLE =>
                        if start = '1' then
                            next_state <= CALC;
                        else
                            next_state <= IDLE;
                        end if;
                end case;
            end process;
        end process;
    end architecture rtl;

```



```

        when CALC =>
            if rco = '1' then
                next_state <= FINISHED;
            else
                next_state <= CALC;
            end if;
        when FINISHED =>
            if start = '1' then
                next_state <= FINISHED;
            else
                next_state <= IDLE;
            end if;
        end case;
    end process;

    init <= '1' when state = IDLE else '0';
    load <= '1' when state = CALC else '0';
    done <= '1' when state = FINISHED else '0';
END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;

BEGIN
    state_ctrl: process ( clk, nRST ) is
    begin
        if nRST = '0' then
            state <= IDLE;
        elsif rising_edge( clk ) then
            state <= next_state;
        end if;
    end process;
    next_state_calc: process ( state, start, rco ) is
    begin
        case state is
            when IDLE =>
                if start = '1' then
                    next_state <= CALC;
                else
                    next_state <= IDLE;
                end if;
            --
        end case;
    end process;

```

```

        when CALC =>
            if rco = '1' then
                next_state <= FINISHED;
            else
                next_state <= CALC;
            end if;
        when FINISHED =>
            if start = '1' then
                next_state <= FINISHED;
            else
                next_state <= IDLE;
            end if;
        end case;
    end process;
    init <= '1' when state = IDLE else '0';
    load <= '1' when state = CALC else '0';
    done <= '1' when state = FINISHED else '0';
END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN

        state_ctrl: process ( clk, nRST ) is
        begin
            if nRST = '0' then
                state <= IDLE;
            elsif rising_edge( clk ) then
                state <= next_state;
            end if;
        end process;

        next_state_calc: process ( state, start, rco ) is
        begin
            case state is
                when IDLE =>
                    if start = '1' then
                        next_state <= CALC;
                    else
                        next_state <= IDLE;
                    end if;
            end case;
        end process;
    end architecture rtl;

```

```

        end if;
        when CALC =>
            if rco = '1' then
                next_state <= FINISHED;
            else
                next_state <= CALC;
            end if;
        when FINISHED =>
            if start = '1' then
                next_state <= FINISHED;
            else
                next_state <= IDLE;
            end if;
        end case;
    end process;

    init <= '1' when state = IDLE else '0';
    load <= '1' when state = CALC else '0';
    done <= '1' when state = FINISHED else '0';

```

```

END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN

        cs: process( nRST, clk )
        begin
            if nRST = '0' then
                state <= IDLE;
            elsif ( clk'event and clk = '1' ) then
                state <= next_state;
            end if;
        end process;

        cns: process( start, rco, state )
        begin
            case state is
            when IDLE =>
                if start = '1' then
                    next_state <= CALC;
                else
                    next_state <= IDLE;
                end if;
            end case;
        end process;
    end architecture rtl;

```

```

        end if;
when CALC =>
    if rco = '1' then
        next_state <= FINISHED;
    else
        next_state <= CALC;
    end if;
when FINISHED =>
    if start = '0' then
        next_state <= IDLE;
    else
        next_state <= FINISHED;
    end if;
end case;
end process;

load <= '1' when state = CALC else '0';
done <= '1' when state = FINISHED else '0';
init <= '1' when state = IDLE else '0';
END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;
    BEGIN
        state_ctrl1: process ( clk, nRST ) is
        begin
            if nRST = '0' then
                state <= IDLE;
            elsif rising_edge( clk ) then
                state <= next_state;
            end if;
        end process;

        next_state_calc: process ( state, start, rco ) is
        begin
            case state is
                when IDLE =>
                    if start = '1' then
                        next_state <= CALC;
                    else
                        next_state <= IDLE;
                    end if;
            end case;
        end process;
    end
end

```



```

        when CALC =>
            if rco = '1' then
                next_state <= FINISHED;
            else
                next_state <= CALC;
            end if;
        when FINISHED =>
            if start = '1' then
                next_state <= FINISHED;
            else
                next_state <= IDLE;
            end if;
        end case;
    end process;

    init <= '1' when state = IDLE else '0';
    load <= '1' when state = CALC else '0';
    done <= '1' when state = FINISHED else '0';
END rtl;

```

```

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

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

ENTITY CORDIC_FSM IS
    PORT(
        clk, nRST, start, rco      : IN std_logic;
        init, load, done           : OUT std_logic
    );
END CORDIC_FSM;

ARCHITECTURE rtl OF CORDIC_FSM IS

    TYPE STATE_TYPE IS ( IDLE, CALC, FINISHED );
    SIGNAL      state, next_state : STATE_TYPE := IDLE;

BEGIN
    PROCESS( clk, nRST )
    BEGIN
        IF ( nRST = '0' ) THEN
            state <= IDLE;
        ELSIF ( rising_edge( clk ) ) THEN
            state <= next_state;
        END IF;
    END PROCESS;

    transition : PROCESS ( start, rco, state )
    BEGIN
        CASE state IS
            WHEN IDLE =>
                IF ( start='1' ) THEN
                    next_state <= CALC;
                ELSE
                    next_state <= IDLE;
                END IF;
            END CASE;
        END PROCESS;
    END transition;
END ARCHITECTURE;

```

```

        END IF;
    WHEN CALC =>
        IF ( rco = '1' ) THEN
            next_state <= FINISHED;
        ELSE
            next_state <= CALC;
        END IF;
    WHEN FINISHED =>
        IF ( start='0' ) THEN
            next_state <= IDLE;
        ELSE
            next_state <= FINISHED;
        END IF;
    WHEN OTHERS =>
        next_state <= IDLE;
    END CASE;
END PROCESS transition;

```

```

init <= '1' when ( state = IDLE ) else '0';
load <= '1' when ( state = CALC ) else '0';
done <= '1' when ( state = FINISHED ) else '0';

```

```

END rtl;

```

