Proyek Akhir Praktikum Perancangan SIstem Digital 2024/2025

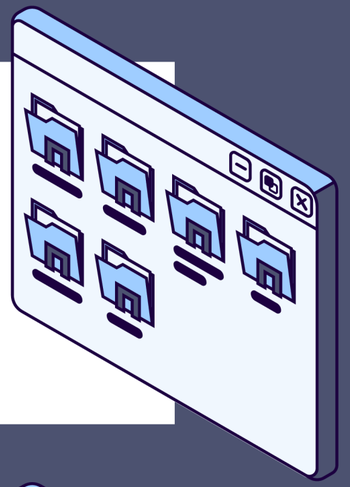
SIMPLE CALCULATOR WITH TIME CONVERTER

Kelompok PA20 (ED)

ANGGOTA KELOMPOK

Ganendra Garda Pratama

NPM: 2306250642



Jonathan Frederick Kosasih

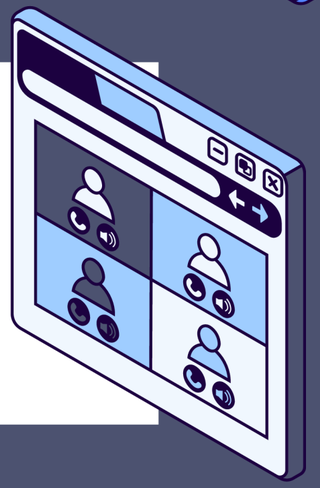
NPM: 2306225981

Rafi Naufal Aryaputra

NPM: 2306250680

Gede Rama Pradnya Widadga

NPM: 2306161914



LANDASAN TEORI

BACKGROUND

Proyek ini didasari oleh kebutuhan akan alat multifungsi yang mampu menangani operasi aritmatika dasar sekaligus mendukung pengelolaan waktu secara efisien, seperti penjumlahan, pengurangan waktu, konversi zona waktu berdasarkan GMT, dan timer. Dirancang untuk memberikan solusi praktis bagi pengguna sehari-hari, seperti pekerja internasional dan pelajar, proyek ini juga berfungsi sebagai media pembelajaran teknologi. Menggunakan pendekatan berbasis VHDL, proyek ini difokuskan pada simulasi rangkaian tanpa implementasi langsung pada perangkat keras, dengan memanfaatkan Truth Table dan Testbench. Selain memenuhi kebutuhan manajemen waktu, proyek ini membantu mahasiswa memahami cara kerja ALU dan penerapan logika digital dalam desain sistem.

LEARNING OBJECTIVES

Membuat sebuah kalkulator yang dilengkapi dengan fitur konversi waktu

Mengimplementasikan Pemrograman menggunakan bahasa VHDL untuk perancangan sebuah sistem digital

Memenuhi nilai proyek akhir dalam Praktikum Perancangan Sistem Digital

Membantu para mahasiswa untuk mengatur mengelola waktu dengan efisien

PENJELASAN COMPONENT

CALCULATOR

- Memiliki fitur add dan subtract untuk menghitung jumlah dan selisih dari 2 buah input.
- Mengambil 2 buah input berukuran 8 bit

```
ENTITY Calculator IS
    PORT (
        startOp : IN STD_LOGIC;
        opCode : IN STD_LOGIC;

        numIn_a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        numIn_b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);

        numOut : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END Calculator;
```

- opCode mengindikasikan operasi apa yang akan dilakukan
- startOp digunakan untuk menentukan kapan kalkulator dapat digunakan untuk melakukan operasi
- Hasil dari mode subtract akan selalu bernilai positif
- Output berukuran 8 bit

```
ARCHITECTURE Behavioral OF Calculator IS
    SIGNAL num_a, num_b : UNSIGNED(7 DOWNTO 0);
    SIGNAL total_a, total_b, total_res : INTEGER;
BEGIN
    -- Convert inputs to unsigned for computation
    num_a <= UNSIGNED(numIn_a);
    num_b <= UNSIGNED(numIn_b);

    -- Combine numbers into total
    total_a <= to_integer(num_a);
    total_b <= to_integer(num_b);

    PROCESS (startOp, opCode, total_a, total_b)
    BEGIN
        IF startOp = '1' THEN
            IF opCode = '1' THEN
                -- Perform addition
                total_res <= total_a + total_b;
            ELSE
                -- Perform subtraction and ensure positive result
                IF total_a > total_b THEN
                    total_res <= total_a - total_b;
                ELSE
                    total_res <= total_b - total_a;
                END IF;
            END IF;
        END IF;
    END PROCESS;

    -- Convert result back to binary for output
    numOut <= STD_LOGIC_VECTOR(TO_UNSIGNED(total_res, 8));

END Behavioral;
```

CLOCK

- Mengimplementasikan sebuah finite state machine dengan state IDLE, SET, dan RUN
- Transisi status pada FSM di-trigger berdasarkan rising edge dari clockIn
- Memiliki fitur konversi format 12-jam dengan 24-jam berdasarkan input format12
- Format 12 jam akan menggunakan Am atau PM yang diindikasikan dengan nilai meridiemStatus
- Memiliki fitur wrap-around

```
entity ClockFSM is
    port(
        clockIn  : in std_logic;
        clockSet : in std_logic;
        clockRun : in std_logic;
        format12 : in std_logic;

        hourIn : in std_logic_vector (4 downto 0);
        minIn  : in std_logic_vector (5 downto 0);
        secIn  : in std_logic_vector (4 downto 0);

        runStatus      : out std_logic;
        formatStatus   : out std_logic;
        meridiemStatus : out std_logic;

        hourOut : out std_logic_vector (4 downto 0);
        minOut  : out std_logic_vector (5 downto 0);
        secOut  : out std_logic_vector (4 downto 0)
    );
end ClockFSM;
```

```
    PROCESS (hourBuffer, format12)
    BEGIN
        IF format12 = '1' THEN
            IF hourBuffer = 0 THEN
                hourOut <= STD_LOGIC_VECTOR(to_unsigned(12, 5));
                am_pm <= '0'; -- AM
            ELSIF hourBuffer < 12 THEN
                hourOut <= STD_LOGIC_VECTOR(hourBuffer);
                am_pm <= '0'; -- AM
            ELSIF hourBuffer = 12 THEN
                hourOut <= STD_LOGIC_VECTOR(hourBuffer);
                am_pm <= '1'; -- PM
            ELSE
                hourOut <= STD_LOGIC_VECTOR(hourBuffer - 12);
                am_pm <= '1'; -- PM
            END IF;
        ELSE
            hourOut <= STD_LOGIC_VECTOR(hourBuffer);
            am_pm <= '0'; -- Placeholder for 24-hour format
        END IF;
    END PROCESS;

    minOut <= STD_LOGIC_VECTOR(minBuffer);
    secOut <= STD_LOGIC_VECTOR(secBuffer);

    runStatus <= clockRun;
    formatStatus <= format12;
    meridiemStatus <= am_pm;
END fsm;
```

```
ARCHITECTURE fsm OF ClockFSM IS
    TYPE state_type IS (IDLE, SET, RUN);
    SIGNAL state, next_state : state_type;
    SIGNAL hourBuffer : unsigned (4 DOWNTO 0);
    SIGNAL minBuffer : unsigned (5 DOWNTO 0);
    SIGNAL secBuffer : unsigned (4 DOWNTO 0);
    SIGNAL am_pm : STD_LOGIC;
BEGIN
    PROCESS (clockIn)
    BEGIN
        IF rising_edge(clockIn) THEN
            state <= next_state;
        END IF;
    END PROCESS;

    PROCESS (state, clockSet, clockRun, hourIn, minIn, secIn, hourBuffer, minBuffer, secBuffer)
    BEGIN
        CASE state IS
            WHEN IDLE =>
                IF clockSet = '1' THEN
                    next_state <= SET;
                ELSIF clockRun = '1' THEN
                    next_state <= RUN;
                ELSE
                    next_state <= IDLE;
                END IF;
            WHEN SET =>
                hourBuffer <= unsigned(hourIn);
                minBuffer <= unsigned(minIn);
                secBuffer <= unsigned(secIn);
                next_state <= IDLE;
            WHEN RUN =>
                IF secBuffer = 59 THEN
                    secBuffer <= (OTHERS => '0');
                    IF minBuffer = 59 THEN
                        minBuffer <= (OTHERS => '0');
                        IF hourBuffer = 23 THEN
                            hourBuffer <= (OTHERS => '0');
                        ELSE
                            hourBuffer <= hourBuffer + 1;
                        END IF;
                    ELSE
                        minBuffer <= minBuffer + 1;
                    END IF;
                ELSE
                    secBuffer <= secBuffer + 1;
                END IF;
                next_state <= RUN;
            WHEN OTHERS =>
                next_state <= IDLE;
        END CASE;
    END PROCESS;
```

TIMER

- Mengimplementasikan sebuah finite state machine dengan state IDLE, SET, RUN, dan DONE
- Countdown dapat diatur pada status SET dan akan memulai hitung mundur pada status RUN
- Menghitung mundur dengan sistem wrap-around ketika decrement
- doneStatus mengindikasikan bahwa proses hitung mundur telah selesai dilaksanakan oleh timer

```
ENTITY TimerFSM IS
    PORT (
        clockIn : IN STD_LOGIC;
        setTimer : IN STD_LOGIC;
        startTimer : IN STD_LOGIC;

        hourIn : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        minIn : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        secIn : IN STD_LOGIC_VECTOR (4 DOWNTO 0);

        doneStatus : OUT STD_LOGIC;

        hourOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);
        minOut : OUT STD_LOGIC_VECTOR (5 DOWNTO 0);
        secOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0)
    );
END TimerFSM;
```

```
    PROCESS (hourBuffer, minBuffer, secBuffer)
    BEGIN
        IF state /= DONE THEN
            doneStatus <= '0';
        END IF;
    END PROCESS;

    hourOut <= STD_LOGIC_VECTOR(hourBuffer);
    minOut <= STD_LOGIC_VECTOR(minBuffer);
    secOut <= STD_LOGIC_VECTOR(secBuffer);
END fsm;
```

```
ARCHITECTURE fsm OF TimerFSM IS
    TYPE state_type IS (IDLE, SET, RUN, DONE);
    SIGNAL state, next_state : state_type;
    SIGNAL hourBuffer : signed (4 DOWNTO 0);
    SIGNAL minBuffer : signed (5 DOWNTO 0);
    SIGNAL secBuffer : signed (4 DOWNTO 0);

BEGIN
    PROCESS (clockIn)
    BEGIN
        IF rising_edge(clockIn) THEN
            state <= next_state;
        END IF;
    END PROCESS;

    PROCESS (state, setTimer, startTimer, hourIn, minIn, secIn, hourBuffer, minBuffer, secBuffer)
    BEGIN
        CASE state IS
            WHEN IDLE =>
                IF setTimer = '1' THEN
                    next_state <= SET;
                ELSIF startTimer = '1' THEN
                    next_state <= RUN;
                ELSE
                    next_state <= IDLE;
                END IF;
            WHEN SET =>
                hourBuffer <= signed(hourIn);
                minBuffer <= signed(minIn);
                secBuffer <= signed(secIn);
                next_state <= IDLE;
            WHEN RUN =>
                IF secBuffer = to_signed(0, secBuffer'length) THEN
                    secBuffer <= to_signed(59, secBuffer'length);
                    IF minBuffer = to_signed(0, minBuffer'length) THEN
                        minBuffer <= to_signed(59, minBuffer'length);
                        IF hourBuffer = to_signed(0, hourBuffer'length) THEN
                            next_state <= DONE;
                        ELSE
                            hourBuffer <= hourBuffer - 1;
                        END IF;
                    ELSE
                        minBuffer <= minBuffer - 1;
                    END IF;
                ELSE
                    secBuffer <= secBuffer - 1;
                END IF;
                next_state <= RUN;
            WHEN DONE =>
                doneStatus <= '1';
                next_state <= IDLE;
            WHEN OTHERS =>
                next_state <= IDLE;
        END CASE;
    END PROCESS;
```

TIME CONVERTER

- Memiliki fitur konversi jam dan menit dari waktu lokal ke GMT
- Memiliki fitur wrap-around untuk nilai negatif dari jam dan menitnya
- startOp digunakan untuk menentukan kapan konverter dapat digunakan untuk melakukan operasi
- Mengambil input waktu lokal dan mengeluarkan output waktu yang sudah dikonversikan ke GMT
- doneStatus mengindikasikan bahwa operasi telah selesai dilaksanakan oleh konverter

```
ENTITY TimeConverter IS
    PORT (
        startOp : IN STD_LOGIC;

        inGMTHours : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        inGMTMins : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        hourIn : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        minIn : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        secIn : IN STD_LOGIC_VECTOR (4 DOWNTO 0);

        doneStatus : OUT STD_LOGIC;

        outGMTHours : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);
        outGMTMins : OUT STD_LOGIC_VECTOR (5 DOWNTO 0);
        hourOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);
        minOut : OUT STD_LOGIC_VECTOR (5 DOWNTO 0);
        secOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0)
    );
END TimeConverter;
```

```
ARCHITECTURE Behavioral OF TimeConverter IS
    SIGNAL localHours : signed (4 DOWNTO 0);
    SIGNAL localMins : signed (5 DOWNTO 0);
    SIGNAL gmtHours : signed (4 DOWNTO 0);
    SIGNAL gmtMins : signed (5 DOWNTO 0);
BEGIN
    PROCESS (startOp)
    BEGIN
        IF rising_edge(startOp) THEN
            -- Convert local time to GMT
            gmtHours <= signed(hourIn) - signed(inGMTHours);
            gmtMins <= signed(minIn) - signed(inGMTMins);

            -- Adjust for negative minutes
            IF gmtMins < 0 THEN
                gmtMins <= gmtMins + 60;
                gmtHours <= gmtHours - 1;
            END IF;

            -- Adjust for negative hours
            IF gmtHours < 0 THEN
                gmtHours <= gmtHours + 24;
            END IF;

            -- Output the converted time
            hourOut <= STD_LOGIC_VECTOR(gmtHours);
            minOut <= STD_LOGIC_VECTOR(gmtMins);
            secOut <= secIn;

            -- Set done status
            doneStatus <= '1';
        ELSE
            doneStatus <= '0';
        END IF;
    END PROCESS;

END Behavioral;
```

TIME CALCULATOR

- Memiliki fitur add dan subtract untuk menghitung jumlah dan selisih dari 2 buah input dalam satuan jam, menit, dan detik
- Input untuk jam, menit, dan detik terpisah
- opCode mengindikasikan operasi apa yang akan dilakukan
- startOp digunakan untuk menentukan kapan kalkulator dapat digunakan untuk melakukan operasi
- Hasil dari mode subtract akan selalu bernilai positif

```
ENTITY TimeCalculator IS
    PORT (
        startOp : IN STD_LOGIC;
        opCode : IN STD_LOGIC;

        hourIn_a : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        minIn_a : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        secIn_a : IN STD_LOGIC_VECTOR (4 DOWNTO 0);

        hourIn_b : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        minIn_b : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        secIn_b : IN STD_LOGIC_VECTOR (4 DOWNTO 0);

        hourOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);
        minOut : OUT STD_LOGIC_VECTOR (5 DOWNTO 0);
        secOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0)
    );
END TimeCalculator;
```

```
ARCHITECTURE Behavioral OF TimeCalculator IS
    SIGNAL hour_a, hour_b, hour_res : UNSIGNED(4 DOWNTO 0);
    SIGNAL min_a, min_b, min_res : UNSIGNED(5 DOWNTO 0);
    SIGNAL sec_a, sec_b, sec_res : UNSIGNED(4 DOWNTO 0);
    SIGNAL total_a, total_b, total_res : INTEGER;
    SIGNAL total_diff : INTEGER;
BEGIN

    -- Convert inputs to unsigned for computation
    hour_a <= UNSIGNED(hourIn_a);
    hour_b <= UNSIGNED(hourIn_b);
    min_a <= UNSIGNED(minIn_a);
    min_b <= UNSIGNED(minIn_b);
    sec_a <= UNSIGNED(secIn_a);
    sec_b <= UNSIGNED(secIn_b);

    -- Combine hours, minutes, and seconds into total seconds
    total_a <= (to_integer(hour_a) * 3600) + (to_integer(min_a) * 60) + to_integer(sec_a);
    total_b <= (to_integer(hour_b) * 3600) + (to_integer(min_b) * 60) + to_integer(sec_b);

    PROCESS (startOp, opCode, total_a, total_b)
    BEGIN
        IF startOp = '1' THEN
            IF opCode = '1' THEN
                -- Perform addition
                total_res <= total_a + total_b;
            ELSE
                -- Perform subtraction and ensure positive result
                IF total_a > total_b THEN
                    total_res <= total_a - total_b;
                ELSE
                    total_res <= total_b - total_a;
                END IF;
            END IF;
        END IF;
    END PROCESS;

    -- Convert total seconds back into hours, minutes, and seconds
    PROCESS (total_res)
    BEGIN
        hour_res <= to_unsigned((total_res / 3600), 5);
        min_res <= to_unsigned(((total_res MOD 3600) / 60), 6);
        sec_res <= to_unsigned((total_res MOD 60), 5);
    END PROCESS;

    -- Output the results
    hourOut <= STD_LOGIC_VECTOR(hour_res);
    minOut <= STD_LOGIC_VECTOR(min_res);
    secOut <= STD_LOGIC_VECTOR(sec_res);
END Behavioral;
```

SMART CLOCK
(TOP LEVEL)

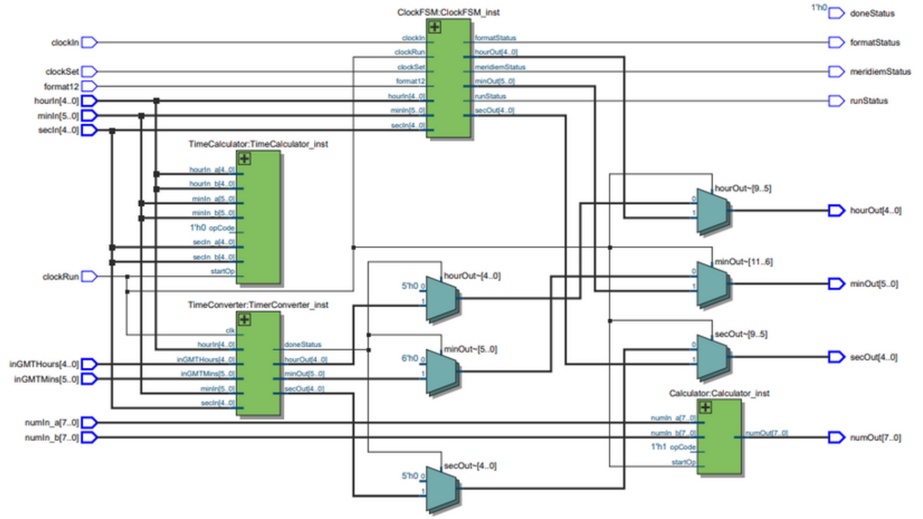
- Merupakan top level design dari 5 komponen sebelumnya
- Dapat mengoperasikan 4 submodul berbeda: TimerConverter, TimeCalculator, ClockFSM, dan Calculator
- Mengimplementasikan structural design dengan memisahkan setiap komponen menjadi flie VHDLnya sendiri
- Memberikan run status, format waktu pilihan, dan format 12/24 jam

```vhdl
-- Define the top-level entity
ENTITY SmartClock IS
    PORT (
        -- Inputs
        clockIn : IN STD_LOGIC;
        clockSet : IN STD_LOGIC;
        clockRun : IN STD_LOGIC;
        format12 : IN STD_LOGIC;

        hourIn : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        minIn : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
        secIn : IN STD_LOGIC_VECTOR (4 DOWNTO 0);

        inGMTHours : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
        inGMTMins : IN STD_LOGIC_VECTOR (5 DOWNTO 0);

        numIn_a : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
        numIn_b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);

        -- Outputs
        doneStatus : OUT STD_LOGIC;

        hourOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);
        minOut : OUT STD_LOGIC_VECTOR (5 DOWNTO 0);
        secOut : OUT STD_LOGIC_VECTOR (4 DOWNTO 0);

        runStatus : OUT STD_LOGIC;
        formatStatus : OUT STD_LOGIC;
        meridiemStatus : OUT STD_LOGIC;

        numOut : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
END SmartClock;

ARCHITECTURE Behavioral OF SmartClock IS
    -- Signals for internal connections
    SIGNAL timer_doneStatus : STD_LOGIC;
    SIGNAL timer_hourOut : STD_LOGIC_VECTOR (4 DOWNTO 0);
    SIGNAL timer_minOut : STD_LOGIC_VECTOR (5 DOWNTO 0);
    SIGNAL timer_secOut : STD_LOGIC_VECTOR (4 DOWNTO 0);

    SIGNAL calc_hourOut : STD_LOGIC_VECTOR (4 DOWNTO 0);
    SIGNAL calc_minOut : STD_LOGIC_VECTOR (5 DOWNTO 0);
    SIGNAL calc_secOut : STD_LOGIC_VECTOR (4 DOWNTO 0);

    SIGNAL clock_hourOut : STD_LOGIC_VECTOR (4 DOWNTO 0);
    SIGNAL clock_minOut : STD_LOGIC_VECTOR (5 DOWNTO 0);
    SIGNAL clock_secOut : STD_LOGIC_VECTOR (4 DOWNTO 0);

    SIGNAL calculator_numOut : STD_LOGIC_VECTOR (7 DOWNTO 0);
```

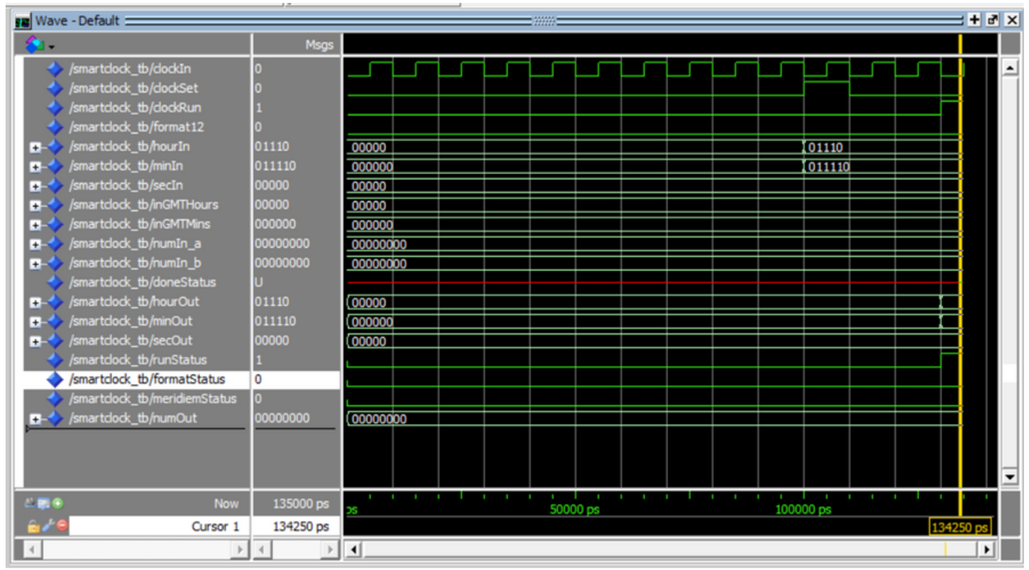
```vhdl
BEGIN
    -- Timer Converter
    TimerConverter_inst : ENTITY work.TimeConverter
        PORT MAP(
            startOp => clockRun,
            inGMTHours => inGMTHours,
            inGMTMins => inGMTMins,
            hourIn => hourIn,
            minIn => minIn,
            secIn => secIn,
            doneStatus => timer_doneStatus,
            outGMTHours => OPEN, -- Unused
            outGMTMins => OPEN, -- Unused
            hourOut => timer_hourOut,
            minOut => timer_minOut,
            secOut => timer_secOut
        );
    -- Time Calculator
    TimeCalculator_inst : ENTITY work.TimeCalculator
        PORT MAP(
            startOp => clockRun,
            -- Example: 0 for subtraction, 1 for addition
            opCode => '0',
            hourIn_a => hourIn,
            minIn_a => minIn,
            secIn_a => secIn,
            hourIn_b => hourIn,
            minIn_b => minIn,
            secIn_b => secIn,
            hourOut => calc_hourOut,
            minOut => calc_minOut,
            secOut => calc_secOut
        );
    -- Clock FSM
    ClockFSM_inst : ENTITY work.ClockFSM
        PORT MAP(
            clockIn => clockIn,
            clockSet => clockSet,
            clockRun => clockRun,
            format12 => format12,
            hourIn => hourIn,
            minIn => minIn,
            secIn => secIn,
            runStatus => runStatus,
            formatStatus => formatStatus,
            meridiemStatus => meridiemStatus,
            hourOut => clock_hourOut,
            minOut => clock_minOut,
            secOut => clock_secOut
        );
```

```vhdl
    -- Calculator
    Calculator_inst : ENTITY work.Calculator
        PORT MAP(
            startOp => clockRun,
            -- Example: 1 for addition, 0 for subtraction
            opCode => '1',
            numIn_a => numIn_a,
            numIn_b => numIn_b,
            numOut => numOut
        );
    -- Choose which output to drive based on requirements
    PROCESS (timer_doneStatus, clockRun)
    BEGIN
        IF clockRun = '1' THEN
            -- Output the result of the Clock FSM
            hourOut <= clock_hourOut;
            minOut <= clock_minOut;
            secOut <= clock_secOut;
        ELSIF timer_doneStatus = '1' THEN
            -- Output the result from the TimeConverter
            hourOut <= timer_hourOut;
            minOut <= timer_minOut;
            secOut <= timer_secOut;
        ELSE
            -- Default or error state
            hourOut <= (OTHERS => '0');
            minOut <= (OTHERS => '0');
            secOut <= (OTHERS => '0');
        END IF;
    END PROCESS;
END Behavioral;
```

TESTING & SINTHESIS



Sinthesis (Quartus)



Waveform (ModelSim)

TESTBENCH FILE FOR TOP LEVEL

```vhdl
ENTITY SmartClock_TB IS
END SmartClock_TB;

ARCHITECTURE behavior OF SmartClock_TB IS
    -- Component Declaration
    COMPONENT SmartClock
        PORT (
            clockIn : IN STD_LOGIC;
            clockSet : IN STD_LOGIC;
            clockRun : IN STD_LOGIC;
            format12 : IN STD_LOGIC;
            hourIn : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            minIn : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
            secIn : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            inGMTHours : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
            inGMTMins : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
            numIn_a : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
            numIn_b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
            doneStatus : OUT STD_LOGIC;
            hourOut : OUT STD_LOGIC_VECTOR(4 DOWNTO 0);
            minOut : OUT STD_LOGIC_VECTOR(5 DOWNTO 0);
            secOut : OUT STD_LOGIC_VECTOR(4 DOWNTO 0);
            runStatus : OUT STD_LOGIC;
            formatStatus : OUT STD_LOGIC;
            meridiemStatus : OUT STD_LOGIC;
            numOut : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
        );
    END COMPONENT;

    -- Input signals
    SIGNAL clockIn : STD_LOGIC := '0';
    SIGNAL clockSet : STD_LOGIC := '0';
    SIGNAL clockRun : STD_LOGIC := '0';
    SIGNAL format12 : STD_LOGIC := '0';
    SIGNAL hourIn : STD_LOGIC_VECTOR(4 DOWNTO 0) := (OTHERS => '0');
    SIGNAL minIn : STD_LOGIC_VECTOR(5 DOWNTO 0) := (OTHERS => '0');
    SIGNAL secIn : STD_LOGIC_VECTOR(4 DOWNTO 0) := (OTHERS => '0');
    SIGNAL inGMTHours : STD_LOGIC_VECTOR(4 DOWNTO 0) := (OTHERS => '0');
    SIGNAL inGMTMins : STD_LOGIC_VECTOR(5 DOWNTO 0) := (OTHERS => '0');
    SIGNAL numIn_a : STD_LOGIC_VECTOR(7 DOWNTO 0) := (OTHERS => '0');
    SIGNAL numIn_b : STD_LOGIC_VECTOR(7 DOWNTO 0) := (OTHERS => '0');
```

```vhdl
    -- Output signals
    SIGNAL doneStatus : STD_LOGIC;
    SIGNAL hourOut : STD_LOGIC_VECTOR(4 DOWNTO 0);
    SIGNAL minOut : STD_LOGIC_VECTOR(5 DOWNTO 0);
    SIGNAL secOut : STD_LOGIC_VECTOR(4 DOWNTO 0);
    SIGNAL runStatus : STD_LOGIC;
    SIGNAL formatStatus : STD_LOGIC;
    SIGNAL meridiemStatus : STD_LOGIC;
    SIGNAL numOut : STD_LOGIC_VECTOR(7 DOWNTO 0);

    -- Clock period definitions
    CONSTANT clock_period : TIME := 10 ns;

BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut : SmartClock PORT MAP(
        clockIn => clockIn,
        clockSet => clockSet,
        clockRun => clockRun,
        format12 => format12,
        hourIn => hourIn,
        minIn => minIn,
        secIn => secIn,
        inGMTHours => inGMTHours,
        inGMTMins => inGMTMins,
        numIn_a => numIn_a,
        numIn_b => numIn_b,
        doneStatus => doneStatus,
        hourOut => hourOut,
        minOut => minOut,
        secOut => secOut,
        runStatus => runStatus,
        formatStatus => formatStatus,
        meridiemStatus => meridiemStatus,
        numOut => numOut
    );

    -- Clock process
    clock_process : PROCESS
    BEGIN
        clockIn <= '0';
        WAIT FOR clock_period/2;
        clockIn <= '1';
        WAIT FOR clock_period/2;
    END PROCESS;

    -- Stimulus process
    stim_proc : PROCESS
    BEGIN
        -- Initial state
        WAIT FOR 100 ns;

        -- Test 1: Set time to 14:30:00
        clockSet <= '1';
        hourIn <= "01110"; -- 14
        minIn <= "011110"; -- 30
        secIn <= "00000"; -- 00
        WAIT FOR clock_period;
        clockSet <= '0';
        WAIT FOR clock_period * 2;
```

```vhdl
        -- Test 2: Start clock running
        clockRun <= '1';
        WAIT FOR clock_period * 10;

        -- Test 3: Switch to 12-hour format
        format12 <= '1';
        WAIT FOR clock_period * 2;

        -- Test 4: Test calculator functionality
        numIn_a <= "00000101"; -- 5
        numIn_b <= "00000011"; -- 3
        WAIT FOR clock_period * 2;

        -- Test 5: Test time zone conversion
        inGMTHours <= "00010"; -- GMT+2
        inGMTMins <= "000000";
        WAIT FOR clock_period * 2;

        -- Test 6: Stop clock
        clockRun <= '0';
        WAIT FOR clock_period * 2;

        -- Test 7: TimeCalculator Testing
        WAIT FOR clock_period * 2;

        -- Set first time (01:30:00)
        clockSet <= '1';
        hourIn <= "00001"; -- 1 hour
        minIn <= "011110"; -- 30 minutes
        secIn <= "00000"; -- 00 seconds
        WAIT FOR clock_period;
        clockSet <= '0';

        -- Set second time (00:30:00) and start calculation
        clockRun <= '1';
        hourIn <= "00000"; -- 0 hours
        minIn <= "011110"; -- 30 minutes
        secIn <= "00000"; -- 00 seconds
        WAIT FOR clock_period * 4;

        -- Verify calculation (should subtract: 01:00:00)
        ASSERT hourOut = "00001"
        REPORT "TimeCalculator subtraction failed"
            SEVERITY ERROR;

        -- Reset
        clockRun <= '0';
        WAIT FOR clock_period * 2;

        -- End simulation
        WAIT;
    END PROCESS;

END behavior;
```

SUMMARY

- Rangkaian ini memiliki beberapa opsi yaitu Timer, Clock, Calculator, Time Calculator, dan Time Converter.
- Opsi Timer digunakan untuk melakukan rangkaian Timer pada umumnya
- Opsi Clock digunakan untuk menjalankan waktu seperti pada umumnya
- Opsi Calculator untuk melakukan penjumlahan angka biasa seperti penjumlahan dan pengurangan
- Opsi Time Calculator untuk melakukan penjumlahan waktu seperti jam 1.30 apabila dikurang 30 menit akan menjadi jam 1.00
- Opsi Time Converter untuk melakukan konversi antar zona waktu, seperti misalkan di WIB jam 14.30, maka di WIT jam 16.30 (menambah 2 zona waktu)

THANK YOU

Kelompok PA20 (ED) - Proyek Akhir Praktikum Perancangan SIstem Digital 2024/2025