Pengembangan Sistem Square Root Digital dengan Algoritma Newton-Raphson untuk Pengolahan Data dari ADC 16-bit

Kelompok 01

Anggota:

1. **13224001 – Yozia Gedalya Marcho Ginting**
2. **13224002 – Jeva Steve Sinaga**
3. **13224003 – Benedictus Kenneth Setiadi**

EL 2002 Sistem Digital

2025

Implementasi Modul UART 16-Bit

Implementasi dituliskan secara bottom up. Setiap modul/blok dituliskan kode VHDL nya, test vectornya, dan hasil simulasinya. Bab ini ditulis berulang sesuai dengan modul yang diimplementasikan. Termasuk juga integrasi modul.

Kode VHDL

Gunakan text box untuk menuliskan kode. Berikan penjelasan terhadap kode tersebut, baik melalui comment di dalam kode maupun paragraf tambahan setelahnya.

Tick Baud Rate Generator

```
-- @file baud_gen.vhd
-- @brief Deskripsi Fungsi:
-- Modul ini menghasilkan sinyal tick baud rate berdasarkan
-- frekuensi clock input, baud rate yang diinginkan, dan faktor
oversampling.

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

entity baud_gen is
   generic (
      BAUD_RATE      : integer := 9600;         -- Default desired baud rate
(bisa diubah sesuai kebutuhan)
      CLOCK_FREQ     : integer := 50000000;     -- Default Clock frequency
in Hz (bisa diubah sesuai kebutuhan)
      OVERSAMPLE     : integer := 16            -- Default Oversampling factor
(bisa diubah sesuai kebutuhan)
   );
   port (
      clk            : in  std_logic;           -- Input clock
      rst            : in  std_logic;           -- Reset signal
      baud_tick      : out std_logic            -- Baud rate clock output (1-clock
pulse)
   );
end entity baud_gen;

architecture Behavioral of baud_gen is
   signal counter      : integer := 0; -- Counter for baud rate generation
```

```
   constant baud_divisor: integer := CLOCK_FREQ / (BAUD_RATE *
OVERSAMPLE); -- Calculate baud divisor
begin

   process(clk, rst)
   begin
      if rst = '1' then
         counter <= 0;
         baud_tick <= '0';
      elsif rising_edge(clk) then
         if baud_divisor < 1 then
            -- invalid divisor, hold tick low
            counter <= 0;
            baud_tick <= '0';
         elsif counter = baud_divisor - 1 then
            baud_tick <= '1';          -- pulse high for one clk
            counter <= 0;              -- reset counter
         else
            baud_tick <= '0';
            counter <= counter + 1;     -- increment counter
         end if;
      end if;
   end process;

end architecture Behavioral;
```

UART RX Bit Receiver

```
-- @file uart_rx_bit_receiver.vhd
-- @brief Deskripsi Fungsi:
-- Modul ini bertanggung jawab untuk menerima bit serial dari jalur
UART
-- dan mengubahnya menjadi byte data 8-bit dengan oversampling.

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

entity uart_rx_bit is
   port (
      clk       : in  std_logic;
      rst       : in  std_logic;
      baud_tick  : in  std_logic;          -- from your baud_gen
      rx        : in  std_logic;          -- UART serial line
```

```
        rx_byte_out : out std_logic_vector(7 downto 0);
        byte_valid  : out std_logic             -- 1-cycle pulse when byte complete
    );
end entity uart_rx_bit;

architecture Behavioral of uart_rx_bit is

    type state_type is (IDLE, START, DATA, STOP);
    signal state       : state_type := IDLE;

    signal sample_cnt   : integer range 0 to 15 := 0;
    signal bit_cnt      : integer range 0 to 7 := 0;

    signal shift_reg    : std_logic_vector(7 downto 0) := (others => '0');
    signal byte_valid_r : std_logic := '0';

begin

    byte_valid <= byte_valid_r;
    rx_byte_out <= shift_reg;

    process(clk, rst)
    begin
        if rst = '1' then
            state       <= IDLE;
            sample_cnt   <= 0;
            bit_cnt      <= 0;
            shift_reg    <= (others => '0');
            byte_valid_r <= '0';

        elsif rising_edge(clk) then
            byte_valid_r <= '0';  -- default

            if baud_tick = '1' then       -- oversampling tick (1/16 bit)
                case state is

                    when IDLE =>
                        if rx = '0' then      -- start bit detected
                            state      <= START;
                            sample_cnt <= 0;
                        end if;

                    when START =>
                        if sample_cnt = 7 then   -- middle of start bit
```

```
                    if rx = '0' then
                        state      <= DATA;
                        bit_cnt    <= 0;
                        sample_cnt <= 0;
                    else
                        state <= IDLE;    -- false start bit
                    end if;
                else
                    sample_cnt <= sample_cnt + 1;
                end if;

            when DATA =>
                if sample_cnt = 15 then
                    shift_reg(bit_cnt) <= rx;
                    bit_cnt <= bit_cnt + 1;
                    sample_cnt <= 0;

                    if bit_cnt = 7 then
                        state <= STOP;
                    end if;
                else
                    sample_cnt <= sample_cnt + 1;
                end if;

            when STOP =>
                if sample_cnt = 15 then
                    byte_valid_r <= '1';     -- complete!
                    state       <= IDLE;
                else
                    sample_cnt <= sample_cnt + 1;
                end if;

        end case;
      end if;
    end if;
  end process;

end architecture Behavioral;
```

UART RX 16-Bit Datapath

```
-- @file uart_rx_datapath_16bit.vhd
-- @brief Deskripsi Fungsi:
-- Modul ini bertanggung jawab untuk menerima byte data dari modul
```

```
uart_rx_bit
-- dan menyusunnya menjadi data 16-bit berdasarkan sinyal kontrol dari
FSM.

library ieee;
use ieee.std_logic_1164.all;

entity uart_rx_datapath_16bit is
   port (
      clk        : in  std_logic;
      rst        : in  std_logic;

      rx_byte    : in  std_logic_vector(7 downto 0); -- from uart_rx_bit
      capture    : in  std_logic;
      capture_sel : in  std_logic;            -- 0 = MSB, 1 = LSB

      rx_data_out : out std_logic_vector(15 downto 0)
   );
end entity;

architecture Behavioral of uart_rx_datapath_16bit is
   signal rx_reg : std_logic_vector(15 downto 0) := (others => '0');
begin

   process(clk, rst)
   begin
      if rst = '1' then
         rx_reg <= (others => '0');

      elsif rising_edge(clk) then
         if capture = '1' then
            if capture_sel = '0' then
               rx_reg(15 downto 8) <= rx_byte;
            else
               rx_reg(7 downto 0) <= rx_byte;
            end if;
         end if;
      end if;
   end process;

   rx_data_out <= rx_reg;

end architecture Behavioral;
```

UART RX 16-Bit FSM

```
-- @file uart_rx_fsm_16bit.vhd
-- @brief Deskripsi Fungsi:
-- Modul ini mengelola logika status penerimaan data 16-bit
-- melalui UART dengan menggunakan sinyal kontrol dari penerima bit
UART.

library ieee;
use ieee.std_logic_1164.all;

entity uart_rx_fsm_16bit is
    port (
        clk        : in  std_logic;
        rst        : in  std_logic;

        rx_start    : in  std_logic;        -- start receiving 16bit
        byte_valid  : in  std_logic;        -- from uart_rx_bit

        capture     : out std_logic;
        capture_sel : out std_logic;
        rx_finished : out std_logic
    );
end entity;

architecture Behavioral of uart_rx_fsm_16bit is

    type state_type is (IDLE, BYTE1, BYTE2, DONE);
    signal state : state_type := IDLE;

begin

    process(clk, rst)
    begin
        if rst = '1' then
            state <= IDLE;

        elsif rising_edge(clk) then
            case state is

                when IDLE =>
                    if rx_start = '1' then
                        state <= BYTE1;
                    end if;

                when BYTE1 =>
```

```
                if byte_valid = '1' then
                    state <= BYTE2;
                end if;

            when BYTE2 =>
                if byte_valid = '1' then
                    state <= DONE;
                end if;

            when DONE =>
                if rx_start = '0' then
                    state <= IDLE;
                end if;

        end case;
    end if;
  end process;

  -- output logic
  capture     <= '1' when (state = BYTE1 and byte_valid = '1') or (state =
BYTE2 and byte_valid = '1')
  else '0';

  capture_sel <= '0' when state = BYTE1
  else '1' when state = BYTE2
  else '0';

  rx_finished <= '1' when state = DONE else '0';

end architecture Behavioral;
```

UART RX 16-Bit

```
-- @file uart_rx_16bit.vhd
-- @brief Deskripsi Fungsi:
-- Modul ini mengintegrasikan penerima UART 16-bit dengan
menggabungkan
-- modul-modul sub-komponen seperti baud generator, penerima bit
UART,
-- FSM 16-bit, dan datapath 16-bit.

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

```
entity uart_rx_16bit is
   port (
      clk        : in  std_logic;
      rst        : in  std_logic;

      rx         : in  std_logic;    -- UART serial input
      rx_start    : in  std_logic;    -- start receiving 16bit

      rx_data_out : out std_logic_vector(15 downto 0);
      rx_finished : out std_logic
   );
end entity;

architecture Structural of uart_rx_16bit is

   signal baud_tick  : std_logic;
   signal rx_byte    : std_logic_vector(7 downto 0);
   signal byte_valid : std_logic;

   signal capture     : std_logic;
   signal capture_sel : std_logic;

begin

   -- Your baud generator (unchanged)
   BAUD: entity work.baud_gen
      port map (
         clk      => clk,
         rst      => rst,
         baud_tick => baud_tick
      );

   -- UART 8-bit receiver
   BITRX: entity work.uart_rx_bit
      port map (
         clk        => clk,
         rst        => rst,
         baud_tick   => baud_tick,
         rx         => rx,
         rx_byte_out => rx_byte,
         byte_valid  => byte_valid
      );

   -- FSM 16-bit
   FSM: entity work.uart_rx_fsm_16bit
```

```
      port map (
         clk        => clk,
         rst        => rst,
         rx_start    => rx_start,
         byte_valid  => byte_valid,
         capture     => capture,
         capture_sel => capture_sel,
         rx_finished => rx_finished
      );

   -- Datapath 16-bit
   DP: entity work.uart_rx_datapath_16bit
      port map (
         clk        => clk,
         rst        => rst,
         rx_byte     => rx_byte,
         capture     => capture,
         capture_sel => capture_sel,
         rx_data_out => rx_data_out
      );

end architecture Structural;
```

UART TX 16-Bit Serializer

```
-- @file uart_tx_serializer.vhd
-- @brief Deskripsi Fungsi:
-- Modul ini mengubah data byte menjadi frame UART dan
mengirimkannya serially.

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

entity uart_tx_serializer is
   port (
      clk      : in  std_logic;
      rst      : in  std_logic;
      baud_tick : in  std_logic;                  -- tick from baud gen

      load     : in  std_logic;                  -- load new data byte
      data_in   : in  std_logic_vector(7 downto 0);  -- data byte

      tx_line   : out std_logic;                  -- UART TX output
```

```
      busy     : out std_logic                  -- high while sending frame
   );
end entity;

architecture Behavioral of uart_tx_serializer is
   signal shift_reg : std_logic_vector(9 downto 0);   -- start + data + stop
   signal bit_cnt   : integer range 0 to 10 := 0;
   signal tx_reg    : std_logic := '1';             -- idle = '1'
   signal busy_reg  : std_logic := '0';
begin

   process(clk, rst)
   begin
      if rst = '1' then
         tx_reg   <= '1';
         busy_reg <= '0';
         bit_cnt  <= 0;

      elsif rising_edge(clk) then
         if load = '1' and busy_reg = '0' then
            -- Load frame: start(0), data(LSB first), stop(1)
            shift_reg <= '1' & data_in & '0';      -- MSB=1(stop), LSB=0(start)
            bit_cnt   <= 0;
            busy_reg  <= '1';
         end if;

         if baud_tick = '1' and busy_reg = '1' then
            tx_reg   <= shift_reg(bit_cnt);
            bit_cnt  <= bit_cnt + 1;

            if bit_cnt = 9 then
               busy_reg <= '0';                -- done sending
            end if;
         end if;
      end if;
   end process;

   tx_line <= tx_reg;
   busy    <= busy_reg;

end architecture;
```

UART TX 16-Bit FSM

```
-- @file uart_tx_fsm_16bit.vhd
-- @brief Deskripsi Fungsi:
-- Modul ini mengelola logika status transmisi data 16-bit
-- melalui UART dengan menggunakan sinyal kontrol dari serializer
UART.

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

entity uart_tx_16bit_fsm is
   port (
      clk      : in  std_logic;
      rst      : in  std_logic;

      start_tx  : in  std_logic;
      busy_ser  : in  std_logic;

      load_msb  : out std_logic;
      load_lsb  : out std_logic;
      tx_done   : out std_logic -- from uart_tx_serializer
   );
end entity;

architecture Behavioral of uart_tx_16bit_fsm is
   type state_type is (IDLE, LOAD_MSB, WAIT_MSB, LOAD_LSB, WAIT_LSB,
DONE);
   signal state, next_state : state_type := IDLE;
begin

   process(clk, rst)
   begin
      if rst = '1' then
         state <= IDLE;
      elsif rising_edge(clk) then
         state <= next_state;
      end if;
   end process;

   process(state, start_tx, busy_ser)
   begin
      load_msb <= '0';
      load_lsb <= '0';
      tx_done  <= '0';
```

```
      case state is
         when IDLE =>
            if start_tx = '1' then
               next_state <= LOAD_MSB;
            else
               next_state <= IDLE;
            end if;

         when LOAD_MSB =>
            load_msb <= '1';
            next_state <= WAIT_MSB;

         when WAIT_MSB =>
            if busy_ser = '0' then
               next_state <= LOAD_LSB;
            else
               next_state <= WAIT_MSB;
            end if;

         when LOAD_LSB =>
            load_lsb <= '1';
            next_state <= WAIT_LSB;

         when WAIT_LSB =>
            if busy_ser = '0' then
               next_state <= DONE;
            else
               next_state <= WAIT_LSB;
            end if;

         when DONE =>
            tx_done <= '1';
            next_state <= IDLE;

      end case;
   end process;
end architecture;
```

UART TX 16-Bit

```
-- @file uart_tx_16bit.vhd
-- @brief Deskripsi Fungsi:
-- Modul top-level untuk transmitter UART 16-bit yang mengintegrasikan
-- generator baud rate, FSM 16-bit, dan serializer UART.
```

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

entity uart_tx_16bit is
    port (
        clk       : in  std_logic;
        rst       : in  std_logic;

        start_tx   : in  std_logic;
        data_16bit : in  std_logic_vector(15 downto 0);

        tx_done    : out std_logic;
        tx_line    : out std_logic
    );
end entity;

architecture Structural of uart_tx_16bit is
    signal baud_tick : std_logic;

    signal load_msb  : std_logic;
    signal load_lsb  : std_logic;
    signal busy_ser  : std_logic;

    signal tx_data_mux : std_logic_vector(7 downto 0);
begin

    -- ====================================
    -- BAUD GEN (pakai modul kamu)
    -- ====================================
    BAUD_GEN_INST : entity work.baud_gen
        generic map (
            BAUD_RATE  => 9600,
            CLOCK_FREQ => 50000000,
            OVERSAMPLE => 16
        )
        port map (
            clk      => clk,
            rst      => rst,
            baud_tick => baud_tick
        );

    -- ====================================
    -- SERIALIZER
```

```
    -- ====================================
    tx_data_mux <= data_16bit(15 downto 8) when load_msb = '1'
    else data_16bit(7 downto 0);

    SERIALIZER_INST : entity work.uart_tx_serializer
        port map (
            clk      => clk,
            rst      => rst,
            baud_tick => baud_tick,
            load     => load_msb or load_lsb,
            data_in   => tx_data_mux,

            tx_line   => tx_line,
            busy     => busy_ser
        );

    -- ====================================
    -- FSM
    -- ====================================
    FSM_INST : entity work.uart_tx_16bit_fsm
        port map (
            clk      => clk,
            rst      => rst,

            start_tx  => start_tx,
            busy_ser  => busy_ser,

            load_msb  => load_msb,
            load_lsb  => load_lsb,
            tx_done   => tx_done
        );

end architecture;
```

UART 16-Bit

```
-- @file uart_16bit.vhd
-- @brief Deskripsi Fungsi:
-- Modul top-level untuk UART 16-bit yang mengintegrasikan
-- Subsistem TX dan RX UART 16-Bit.

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

```
entity uart_16bit is
   port (
      clk          : in  std_logic;
      rst          : in  std_logic;

      -- Pin Fisik Eksternal (Physical I/O)
      TXD_pin      : out std_logic;          -- Serial Output (ke PC)
      RXD_pin      : in  std_logic;          -- Serial Input (dari PC)

      -- Antarmuka ke Sistem Utama (Contoh: FSM Akar Kuadrat)
      -- TX Interface (Untuk mengirim hasil komputasi)
      tx_start_cmd   : in  std_logic;          -- Perintah kirim 16-bit
      tx_data_in     : in  std_logic_vector(15 downto 0); -- Data 16-bit
      tx_done_status  : out std_logic;          -- Status: Pengiriman 16-bit
selesai

      -- RX Interface (Untuk menerima input/konfigurasi 16-bit)
      rx_start_cmd   : in  std_logic;          -- Perintah mulai terima 16-bit
      rx_data_out    : out std_logic_vector(15 downto 0); -- Data 16-bit yang
diterima
      rx_data_valid  : out std_logic           -- Status: Penerimaan 16-bit
selesai
   );
end entity uart_16bit;

architecture Structural of uart_16bit is

   -- Sinyal INTERNAL (Wires) untuk menghubungkan antar modul
   signal baud_tick_s     : std_logic;

   -- TX Handshake & Data Wires
   signal tx_busy_ser_s   : std_logic;
   signal tx_load_msb_s   : std_logic;
   signal tx_load_lsb_s   : std_logic;
   signal tx_load_s       : std_logic;             -- tx_load_msb_s OR
tx_load_lsb_s
   signal tx_data_mux_s   : std_logic_vector(7 downto 0); -- Hasil MUX
data 16-bit ke 8-bit

   -- RX Handshake & Data Wires
   signal rx_byte_s       : std_logic_vector(7 downto 0);
   signal rx_byte_valid_s  : std_logic;
   signal rx_capture_s    : std_logic;
   signal rx_capture_sel_s : std_logic;
```

```
begin

   --
================================================================
====
   -- 0. SHARED BAUD RATE GENERATOR (baud_gen.vhd)
   --
================================================================
====
   BAUD_GEN_INST : entity work.baud_gen
      generic map (BAUD_RATE => 9600, CLOCK_FREQ => 50000000,
OVERSAMPLE => 16)
      port map (
         clk      => clk,
         rst      => rst,
         baud_tick => baud_tick_s -- Sinyal tick utama
      );

   ----------------------------------------------------------------
   -- A. TRANSMITTER SUBSYSTEM (TX) - Instantiation & Wiring
   ----------------------------------------------------------------

   -- A1. Logika MUX Data dan Load Enable (Kombinasional)
   tx_load_s <= tx_load_msb_s or tx_load_lsb_s;

   -- Memilih MSB atau LSB berdasarkan perintah FSM (tx_load_msb_s)
   tx_data_mux_s <= tx_data_in(15 downto 8) when tx_load_msb_s = '1'
                    else tx_data_in(7 downto 0);

   -- A2. TX FSM (uart_tx_16bit_fsm.vhd)
   TX_FSM_INST : entity work.uart_tx_16bit_fsm
   port map (
      clk      => clk,
      rst      => rst,
      start_tx  => tx_start_cmd,
      busy_ser  => tx_busy_ser_s,  -- Handshake: Busy dari Serializer
      load_msb  => tx_load_msb_s,
      load_lsb  => tx_load_lsb_s,
      tx_done   => tx_done_status -- Status Selesai
   );

   -- A3. TX Serializer Core (uart_tx_serializer.vhd)
   TX_CORE_INST : entity work.uart_tx_serializer
   port map (
      clk      => clk,
```

```
    rst      => rst,
    baud_tick => baud_tick_s,
    load     => tx_load_s,                -- Load jika MSB ATAU LSB aktif
    data_in   => tx_data_mux_s,             -- Data yang sudah di-MUX
    tx_line   => TXD_pin,                 -- Pin Fisik Output
    busy     => tx_busy_ser_s              -- Output Handshake ke TX FSM
);

-----------------------------------------------------------------
-- B. RECEIVER SUBSYSTEM (RX) - Instantiation & Wiring
-----------------------------------------------------------------

-- B1. RX Bit Receiver Core (uart_rx_bit_receiver.vhd)
RX_CORE_INST : entity work.uart_rx_bit
port map (
    clk        => clk,
    rst        => rst,
    baud_tick   => baud_tick_s,
    rx         => RXD_pin,                -- Pin Fisik Input
    rx_byte_out => rx_byte_s,               -- Byte 8-bit yang diterima
    byte_valid  => rx_byte_valid_s          -- Handshake: Byte diterima
);

-- B2. RX FSM Sequencer (uart_rx_fsm_16bit.vhd)
RX_FSM_INST : entity work.uart_rx_fsm_16bit
port map (
    clk        => clk,
    rst        => rst,
    rx_start    => rx_start_cmd,
    byte_valid  => rx_byte_valid_s,         -- Input Handshake dari RX Core
    capture     => rx_capture_s,            -- Output Kontrol ke Datapath
    capture_sel => rx_capture_sel_s,        -- Output Seleksi MSB/LSB
    rx_finished => rx_data_valid            -- Status Selesai 16-bit
);

-- B3. RX Datapath Accumulator (uart_rx_datapath_16bit.vhd)
RX_DATAPATH_INST : entity work.uart_rx_datapath_16bit
port map (
    clk        => clk,
    rst        => rst,
    rx_byte     => rx_byte_s,               -- Byte 8-bit dari Core
    capture     => rx_capture_s,            -- Input Kontrol (latch)
    capture_sel => rx_capture_sel_s,        -- Input Seleksi
    rx_data_out => rx_data_out              -- Output 16-bit Akhir
);
```

```
end architecture Structural;
```

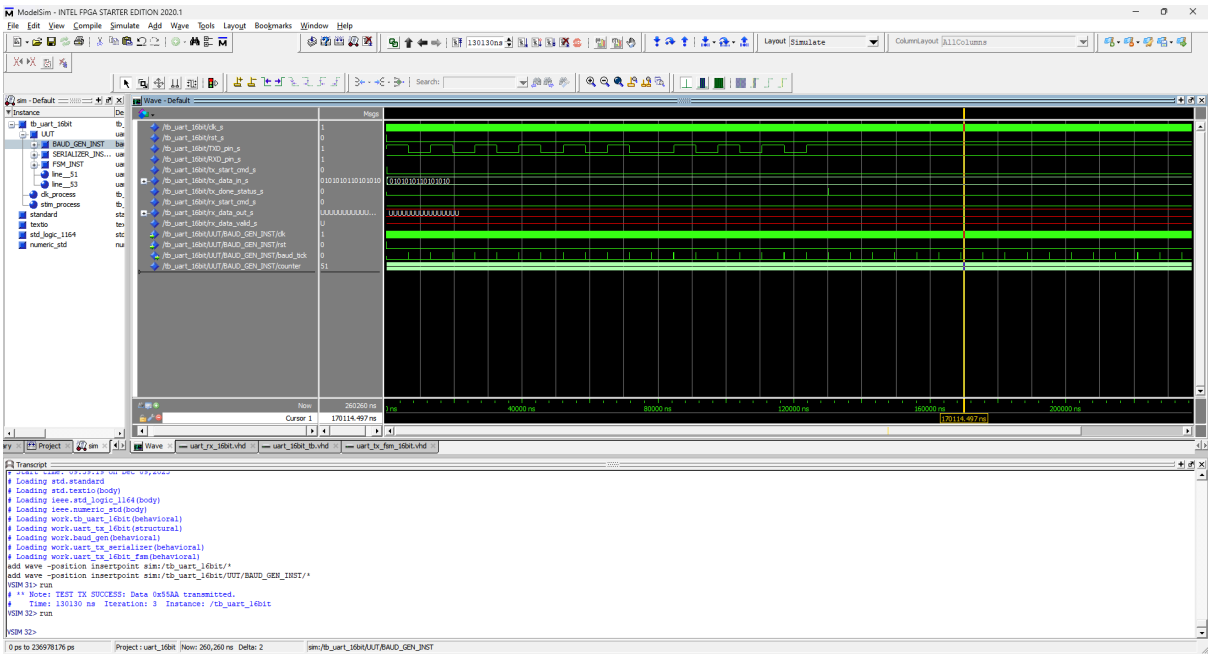
Verifikasi

Test vector

No.	Skenario Uji	Tujuan Pengujian	Input Data (Hex)	Aksi / Stimulus	Output yang Diharapkan	Kriteria Lulus (Pass)
1	**System Reset**	Memastikan sistem mulai dari kondisi bersih.	-	`rst` = '1' selama 5 cycle, lalu '0'.	Semua State FSM = `IDLE`. Output TX/RX = `0`.	Tidak ada aktivitas pada pin `TXD` saat reset.
2	**TX Case A: Distinct Bytes**	Verifikasi urutan pengiriman byte (MSB dulu, lalu LSB).	`tx_data_in` = **0x1234**	Pulse `tx_start_cmd`.	`TXD` Pin mengirim serial **0x12** $\rightarrow$ jeda $\rightarrow$ **0x34**.	Sinyal `tx_done_status` menjadi '1' setelah kedua byte terkirim.

3	**TX Case B: Bit Toggle**	Verifikasi integritas sinyal (0 ke 1 dan 1 ke 0).	tx_data_in = **0x55AA**	Pulse tx_start_cmd.	TXD Pin mengirim serial **0x55** (0101...) $\rightarrow$ **0xAA** (1010...).	Waveform TXD menunjukkan pola zigzag frekuensi tinggi yang konsisten.
4	**TX Case C: Boundary**	Verifikasi kondisi sinyal stuck High/Low.	tx_data_in = **0xFF00**	Pulse tx_start_cmd.	TXD Pin mengirim **0xFF** (All High) $\rightarrow$ **0x00** (All Low).	Sinyal TXD tetap '1' selama periode byte pertama dan '0' selama byte kedua (kecuali start/stop bit).
5	**RX Case A: Standard Data**	Verifikasi penerimaan dan penggabungan data 16-bit.	Serial RXD = **0x55** lalu **0xBB**	Simulasikan input serial eksternal ke pin RXD.	rx_data_out = **0x55BB**.	Sinyal rx_data_valid menjadi '1' tepat setelah byte kedua selesai diterima.

Simulasi



Implementasi Modul MSB Detector

Kode VHDL

```
-- @file msb_detector.vhd
-- @brief This module serves to detect the index of the MSB
-- @note Zero input already has a failsafe so no need for a redundant
failsafe

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

-- ENTITY --
entity msb_detector is
    port (
        data_in   : in  std_logic_vector(15 downto 0); -- 16-bit input
        shift     : out std_logic_vector(3 downto 0); -- 4-bit shift amount for
barrel shifter
```

```
        direction : out std_logic                   -- '0' for Right shift, '1' for Left shift
    );
end entity;

-- ARCHITECTURE --
architecture behavioral of msb_detector is
begin
    process(data_in)
        variable msb_idx_int : integer range 0 to 15;
    begin
        msb_idx_int := 0;

                    -- Searches the highest bit containing '1'
        for i in 15 downto 0 loop
            if data_in(i) = '1' then
                msb_idx_int := i;
                exit;
            end if;
        end loop;

                    -- Normalise to [0.5, 1)
        if msb_idx_int > 7 then
            direction <= '0'; -- Right Shift
            shift <= std_logic_vector(to_unsigned(msb_idx_int - 7, 4));

        elsif msb_idx_int < 7 then
            direction <= '1'; -- Left Shift
            shift <= std_logic_vector(to_unsigned(7 - msb_idx_int, 4));

        else
            direction <= '0'; -- Don't care
            shift <= "0000";
        end if;

    end process;
end architecture;
```

Modul ini berfungsi untuk mengidentifikasi letak indeks MSB untuk membantu modul barrel shifter melakukan normalisasi sesuai prosedur pembagian Goldschmidt. Perlu diperhatikan bahwa modul tidak dilengkapi dengan fail-safe masukan angka nol karena telah ditangani oleh FSM. Selain itu, sebelum mengirim jumlah bit yang harus digeser ke barrel shifter, indeks MSB dikurangi 7 agar selaras dengan tujuan untuk normalisasi ke jangkauan 0.5 sampai 1.

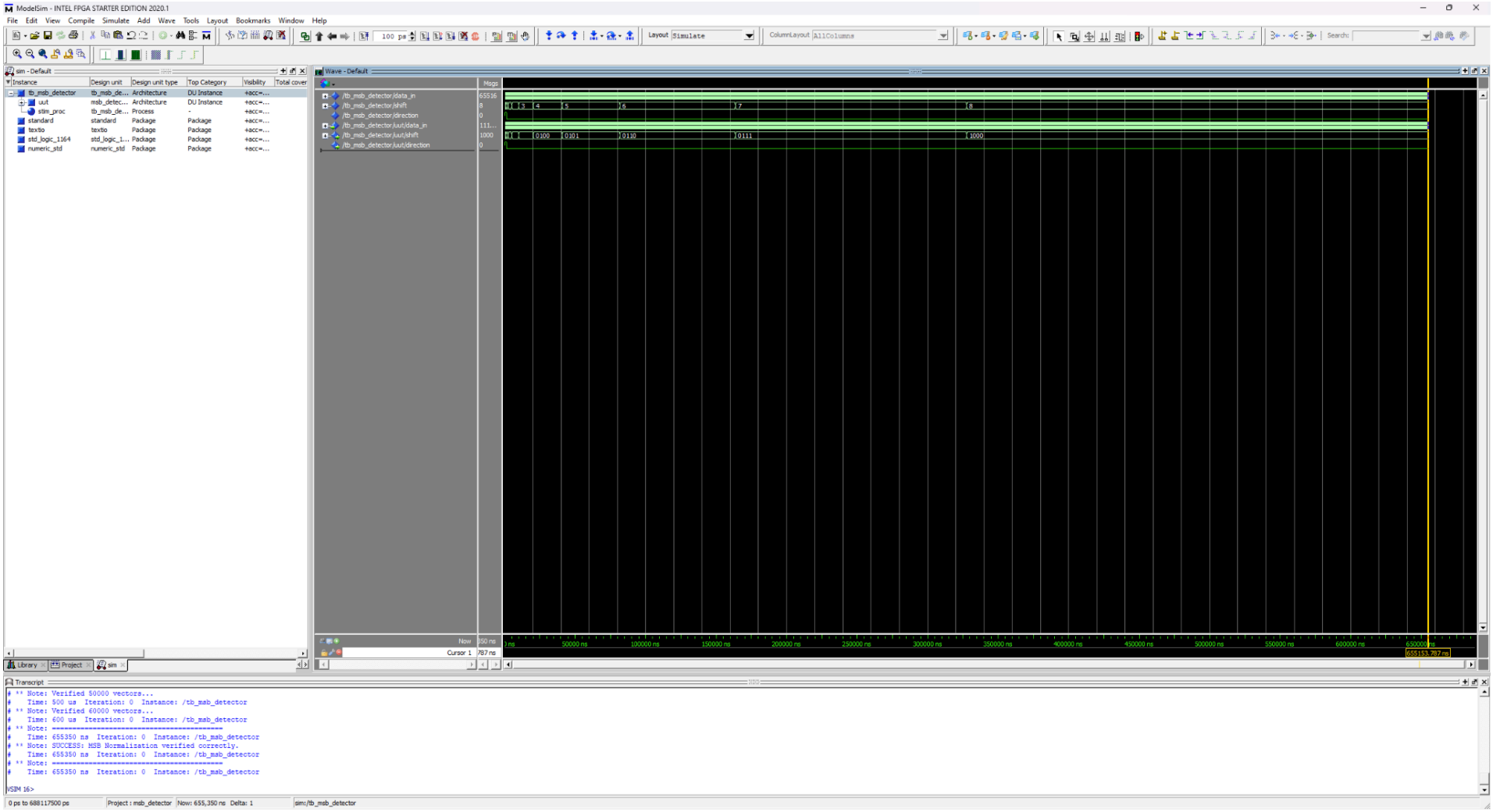
Verifikasi

Test vector

Tipe Test Vector	Input (Decimal)	Input (Biner)	Indeks MSB	Output
Nilai terkecil	1	0000...0001	0	+7
Target jangkauan	128	0000 1000 0000	7	0
Transisi	255	0000 1111 1111	7	0
Shift kanan	512	0000 0010 0000 0000	9	-2
Nilai terbesar	32768	1000 0000 0000 0000	15	-8

Tabel test vector dipersingkat untuk menghemat penulisan dokumen dengan mencantumkan kasus-kasus masukan khusus yang perlu perhatian lebih.

Simulasi


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
sim - Default
Instance
Design unit
Design unit type
Top Category
Visibility
Total cover
tb_msb_detector
uut
stim_proc
standard
textio
std_logic_1164
numeric_std
Wave - Default
/tb_msb_detector/data_in
/tb_msb_detector/shift
/tb_msb_detector/direction
/tb_msb_detector/uut/data_in
/tb_msb_detector/uut/shift
/tb_msb_detector/uut/direction
Now
Cursor 1
Transcript
** Note: Verified 50000 vectors...
Time: 500 us Iteration: 0 Instance: /tb_msb_detector
** Note: Verified 60000 vectors...
Time: 600 us Iteration: 0 Instance: /tb_msb_detector
** Note: ==========
Time: 655350 ns Iteration: 0 Instance: /tb_msb_detector
** Note: SUCCESS: MSB Normalization verified correctly.
Time: 655350 ns Iteration: 0 Instance: /tb_msb_detector
** Note: ==========
Time: 655350 ns Iteration: 0 Instance: /tb_msb_detector
VSIM 16>

Implementasi Modul Barrel Shifter

Kode VHDL

```
-- @file barrel_shifter.vhd
-- @brief This module functions as a shifter to accomodate normalisation
function

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

-- ENTITY --
entity barrel_shifter is
    Port (
        clk       : in std_logic;
        data_in   : in std_logic_vector(15 downto 0); -- 16-bit input
        shift     : in std_logic_vector(3 downto 0); -- Shift amount (0 to 15)
        direction : in std_logic;                    -- '1' for left shift, '0' for right shift
        data_out  : out std_logic_vector(15 downto 0) -- Normalised value
    );
end entity;

-- ARCHITECTURE --
architecture Behavioral of barrel_shifter is
begin
    process(clk)
    begin
        if rising_edge(clk) then
            if direction = '1' then
                data_out <= std_logic_vector(shift_left(unsigned(data_in),
to_integer(unsigned(shift))));
            else
                data_out <= std_logic_vector(shift_right(unsigned(data_in),
to_integer(unsigned(shift))));
            end if;
        end if;
    end process;
end Behavioral;
```

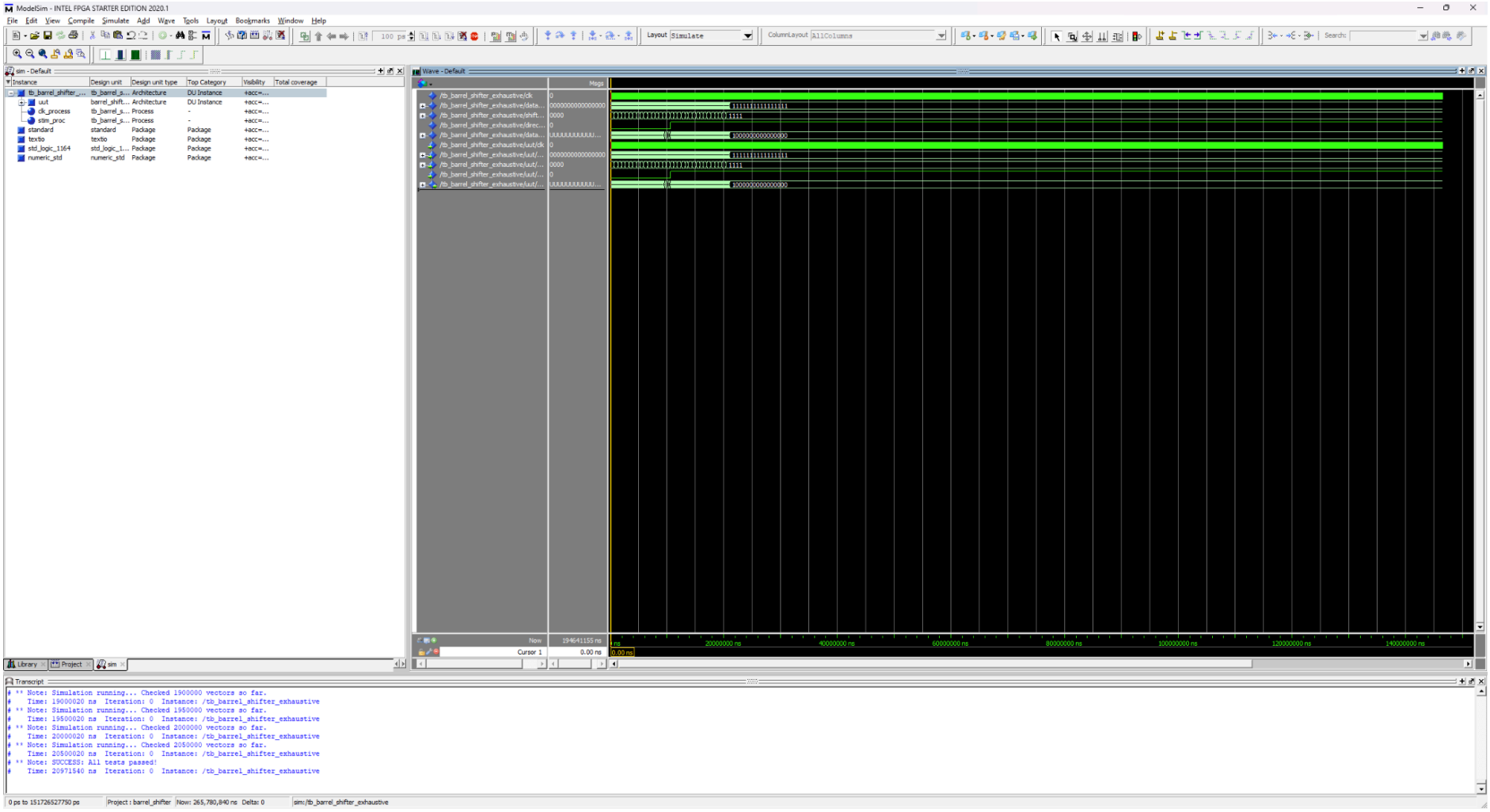
Modul ini membantu menunjang operasi normalisasi yang dibutuhkan oleh algoritma pembagian Goldschmidt dengan masukan yang diumpan oleh MSB detector.

Verifikasi

Test vector

Tipe Test Vector	Arah Shift	Jumlah Shift	Input Data (Hex)	Expected Output (Hex)
Tanpa shift	Left (1)	0	FFFF	FFFF
Shift kiri 1-bit	Left (1)	1	0001	0002
Shift kanan 1-bit	Right (0)	1	0002	0001
Nilai terbesar shift kiri	Left (1)	15	0001	8000
Nilai terbesar shift kanan	Right (0)	15	8000	0001

Simulasi


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
sim - Default
Instance
Design unit
Design unit type
Top Category
Visibility
Total coverage
tb_barrel_shifter_...
uut
clk_process
stim_proc
standard
textio
std_logic_1164
numeric_std
Wave - Default
Msgs
/tb_barrel_shifter_exhaustive/clk
/tb_barrel_shifter_exhaustive/data...
/tb_barrel_shifter_exhaustive/shift...
/tb_barrel_shifter_exhaustive/direc...
/tb_barrel_shifter_exhaustive/data...
/tb_barrel_shifter_exhaustive/uut/clk
/tb_barrel_shifter_exhaustive/uut/...
/tb_barrel_shifter_exhaustive/uut/...
/tb_barrel_shifter_exhaustive/uut/...
/tb_barrel_shifter_exhaustive/uut/...
Now
Cursor 1
20000000 ns
40000000 ns
60000000 ns
80000000 ns
100000000 ns
120000000 ns
140000000 ns
Transcript
** Note: Simulation running... Checked 1900000 vectors so far.
Time: 19000020 ns Iteration: 0 Instance: /tb_barrel_shifter_exhaustive
** Note: Simulation running... Checked 1950000 vectors so far.
Time: 19500020 ns Iteration: 0 Instance: /tb_barrel_shifter_exhaustive
** Note: Simulation running... Checked 2000000 vectors so far.
Time: 20000020 ns Iteration: 0 Instance: /tb_barrel_shifter_exhaustive
** Note: Simulation running... Checked 2050000 vectors so far.
Time: 20500020 ns Iteration: 0 Instance: /tb_barrel_shifter_exhaustive
** Note: SUCCESS: All tests passed!
Time: 20971540 ns Iteration: 0 Instance: /tb_barrel_shifter_exhaustive

Implementasi Modul Goldschmidt Divider

Kode VHDL

```
-- @file goldschmidt.vhd
-- @brief This is the goldschmid divider module

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

-- ENTITY --
entity goldschmidt is
    Port (
        clk      : in  std_logic;
        rst      : in  std_logic;
        start    : in  std_logic;
        dividend : in  std_logic_vector(15 downto 0);
        divisor  : in  std_logic_vector(15 downto 0);
        quotient : out std_logic_vector(15 downto 0);
        ready    : out std_logic
    );
end entity;

-- ARCHITECTURE --
architecture Behavioral of goldschmidt is

    signal N_reg : unsigned(15 downto 0); -- Numerator register
    signal D_reg : unsigned(15 downto 0); -- Denominator register
    signal F_reg : unsigned(15 downto 0); -- Multiplier factor register

    type state_type is (IDLE, CALC_F, MULTIPLY, CHECK_ITER); -- Internal
states
    signal state : state_type;
    signal iter_count : integer range 0 to 7; -- Size of iterations (3-bit for
flexibility)

begin

    process(clk, rst)
```

```
      variable mult_n : unsigned(31 downto 0); -- Temporary multiplier
product variable
      variable mult_d : unsigned(31 downto 0); -- Temporary multiplier
product variable
   begin
         -- Initialise by reset
      if rst = '1' then
         state <= IDLE;
         N_reg <= (others => '0');
         D_reg <= (others => '0');
         F_reg <= (others => '0');
         quotient <= (others => '0');
         ready <= '0';
         iter_count <= 0;


      elsif rising_edge(clk) then
         case state is

            when IDLE =>
               ready <= '0';
               if start = '1' then
                  N_reg <= unsigned(dividend);
                  D_reg <= unsigned(divisor);
                  iter_count <= 0;
                  state <= CALC_F;
               end if;

            when CALC_F =>
               F_reg <= (not D_reg) + 1;
               state <= MULTIPLY;

            when MULTIPLY =>
               mult_n := N_reg * F_reg;
               mult_d := D_reg * F_reg;
               N_reg <= mult_n(31 downto 16);
               D_reg <= mult_d(31 downto 16);

               state <= CHECK_ITER;

            when CHECK_ITER =>
               if iter_count = 2 then
                  quotient <= std_logic_vector(N_reg);
                  ready <= '1';
                  state <= IDLE;
```

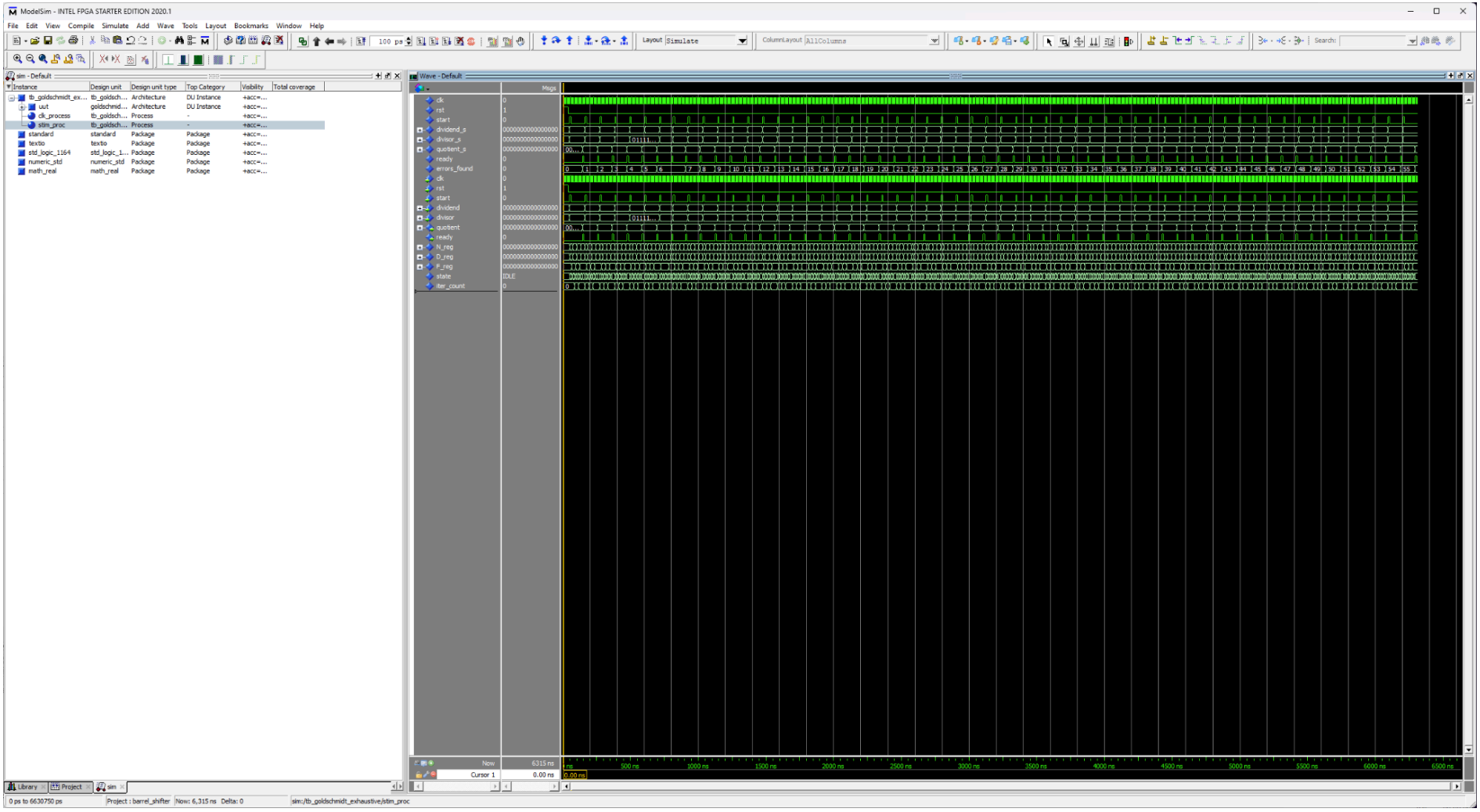
```
                else
                    iter_count <= iter_count + 1;
                    state <= CALC_F;
                end if;

            end case;
        end if;
    end process;
end architecture;
```

Test Vector

Pembilang A	Penyebut B	A Input (Hex Q8.8)	B Input (Hex Q8.8)	Hasil Pembagian ril (Decimal)	Hasil Pembagian sistem (Hex Q8.8)
10.0	2.0	**0A00**	**0200**	5.0	**0500**
1.0	4.0	**0100**	**0400**	0.25	**0040**
8.5	1.0	**0880**	**0100**	8.5	**0880**
1.0	3.0	**0100**	**0300**	0.33203125	**0055**

10.0	-2.0	**0A00**	**FE00**	-5.0	**FB00**
-10.0	2.0	**F600**	**0200**	-5.0	**FB00**
-10.0	-2.0	**F600**	**FE00**	5.0	**0500**
15.0	0.5	**0F00**	**0080**	30.0	**1E00**

Simulasi


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
sim - Default
Instance
Design unit
Design unit type
Top Category
Visibility
Total coverage
tb_goldschmidt_ex...
uut
clk_process
stim_proc
standard
textio
std_logic_1164
numeric_std
math_real
Wave - Default
Msgs
clk
rst
start
dividend_s
divisor_s
quotient_s
ready
errors_found
dividend
divisor
quotient
N_reg
D_reg
P_reg
state
iter_count
IDLE
Now
6315 ns
Cursor 1
0.00 ns
Library
Project
sim
Project : barrel_shifter
Now: 6,315 ns Delta: 0
sim:/tb_goldschmidt_exhaustive/stim_proc