Final Project PSD

MORSE TO ASCII

Kelompok 4

ANGGOTA KELOMPOK

1	Arkaan Pasya Seplitara	[Ketua]
2	Qais Ismail	[Anggota]
3	Danish Al-Fayyadh Sunarta	[Anggota]
3	Raihan Muhammad Nafis Al-Kautsar	[Anggota]

PENDAHULUAN

LATAR BELAKANG

Morse Code merupakan meode telekomunikasi yang sudah ada sejak dahulu dan masih cukup sering dipakai hingga saat ini, namun pada praktiknya, kita terkadang kesulitan untuk menerjemahkan morse menjadi bahasa yang sehari-hari digunakan.

Disini, kami mencoba mengatasi masalah tersebut dengan membuat decoder morse to ASCII untuk memudahkan sekaligus mengurangi probabilitas terjadinya kesalahan decode yang dapat terjadi saat morse code diterjemahkan oleh manusia

HOW IT WORKS?

Sistem akan menerima input berupa kode morse yang disetarakan dengan bentuk biner yang dipisahkan dengan ‘**0**‘ setiap huruf dan katanya.

Yang kemudian penyetaraan dari morse inilah yang akan di decode oleh sistem.

HOW IT WORKS?

Namun, disini ada sedikit mekanisme tambahan agar sistem bisa lebih stabil dalam menerjemahkan morse codenya:

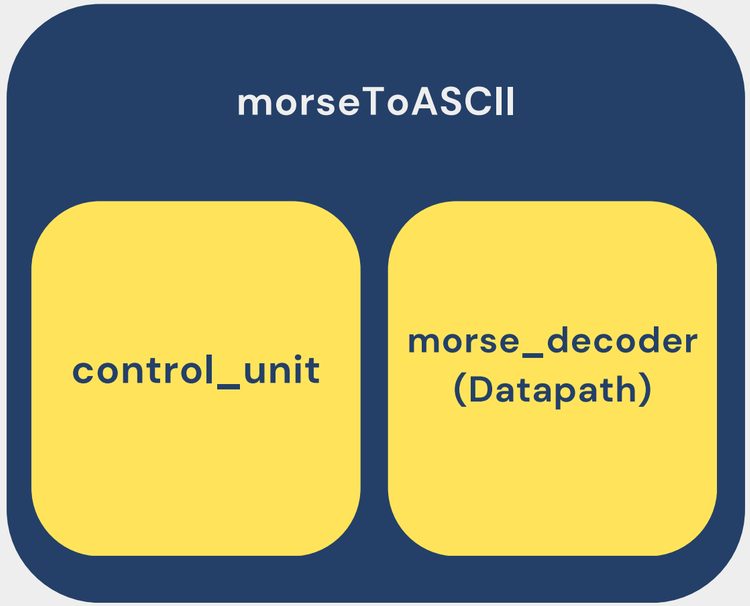
1011000 → 1011000000

Ketika user memberi input sebuah huruf dan huruf tersebut hanya menggunakan sedikit bit (<10), maka sistem akan langsung mengisi sisa bit yang kosong menjadi '**0**'.

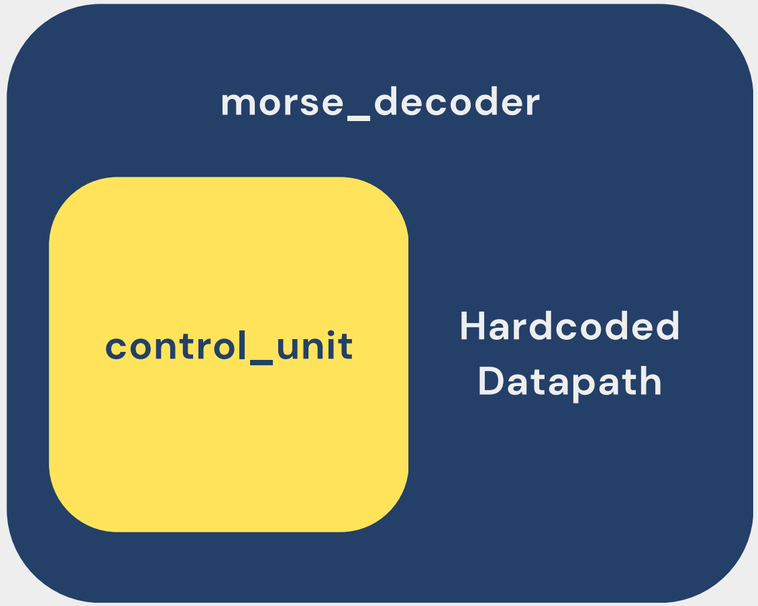
IMPLEMENTASI

STRUKTUR

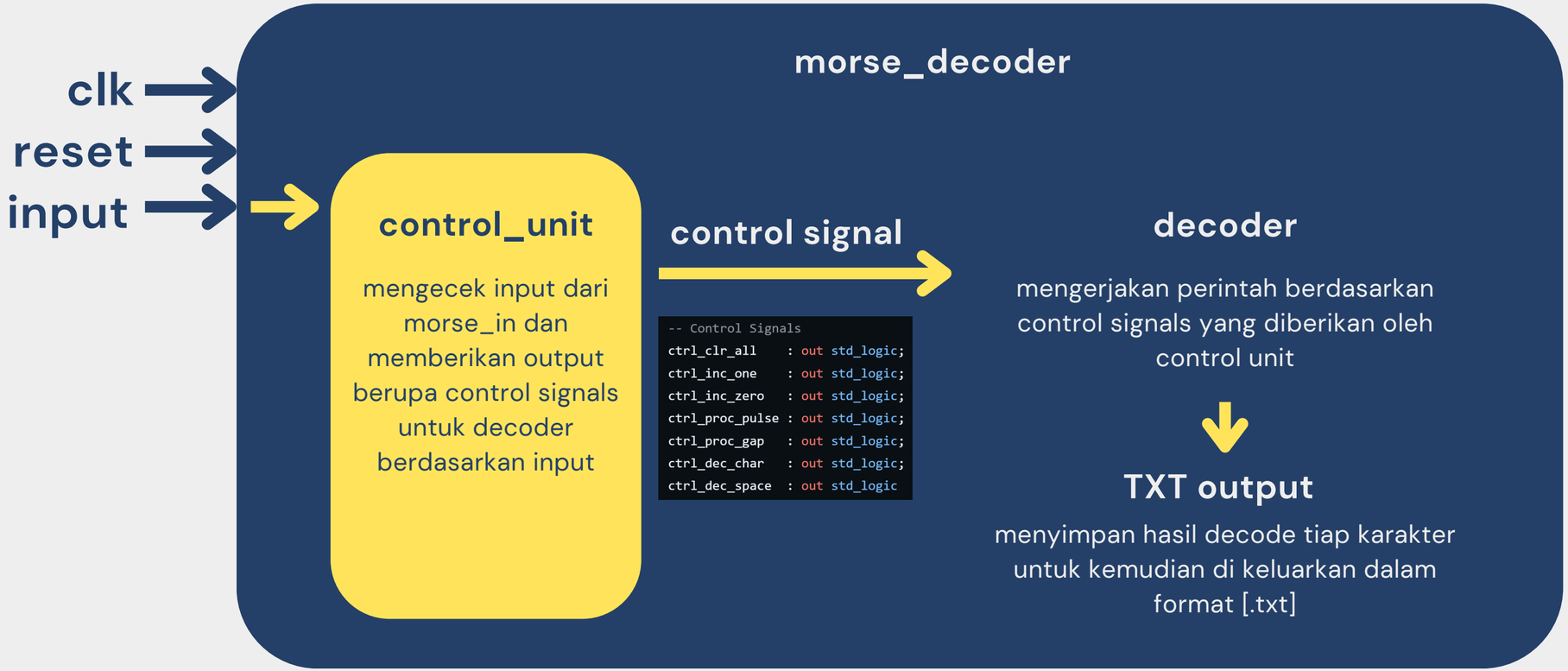
Konsep Awal



Hasil Akhir



ALUR


clk
reset
input
morse_decoder
control_unit
mengecek input dari morse_in dan memberikan output berupa control signals untuk decoder berdasarkan input
control signal
-- Control Signals
ctrl_clr_all : out std_logic;
ctrl_inc_one : out std_logic;
ctrl_inc_zero : out std_logic;
ctrl_proc_pulse : out std_logic;
ctrl_proc_gap : out std_logic;
ctrl_dec_char : out std_logic;
ctrl_dec_space : out std_logic
decoder
mengerjakan perintah berdasarkan control signals yang diberikan oleh control unit
TXT output
menyimpan hasil decode tiap karakter untuk kemudian di keluarkan dalam format [.txt]

DIVE INTO CONTROL UNIT

uCode

XXX | XXXXXXX

di dalam control unit, kita akan banyak bersinggungan dengan uCode, dimana uCode ini merepresentasikan Sequence [3 bit awal] dan Control signal [7 bit akhir] yang akan di pass ke datapath

```
-- [Sequence Control (3 bits)] | [Control Signals (7 bits)]
constant C_SEQ_BITS  : integer := 3;
constant C_CTRL_BITS : integer := 7;
constant C_UCODE_WIDTH : integer := C_SEQ_BITS + C_CTRL_BITS;
```

Control Signal

XXX | XXXXXXX

merupakan bagian ekivalen dari state dari FSM di decoder. bagian ini menentukan apa yang saat ini harus dilakukan oleh decoder.

Sequence

XXX | XXXXXXX

merupakan bagian yang menentukan akan apa yang selanjutnya harus dilakukan oleh decoder atau ekivalen untuk mekanisme transisi state.

```
case seq_cmd is
    when SEQ_NEXT => Next_uPC <= uPC + 1;
    when SEQ_GOTO_IDLE => Next_uPC <= 0;

    when SEQ_GOTO_LOW =>
        Next_uPC <= 4; -- Return to Main Low Loop

    when SEQ_JMP_HIGH => -- IDLE Logic
        if morse_in = '1' then
            Next_uPC <= 2; -- Go to Main High (Skip Entry proc logic for first pulse)
        else
            Next_uPC <= 0;
        end if;

    when SEQ_CHK_FALL => -- HIGH Logic
        if morse_in = '0' then
            Next_uPC <= 3; -- Fall detected! Go to ENTRY LOW (Process Pulse)
        else
            Next_uPC <= 2; -- Stay in Main High
        end if;

    when SEQ_CHK_RISE => -- LOW Logic
        if morse_in = '1' then
            Next_uPC <= 1; -- Rise detected! Go to ENTRY HIGH (Process Gap)
        else
            if flag_space_end = '1' then
                Next_uPC <= 6;
            elsif flag_char_end = '1' then
                Next_uPC <= 5;
            else
                Next_uPC <= 4; -- Stay in Main Low
            end if;
        end if;

    when others => Next_uPC <= 0;
end case;
```

control signal

```
-- Control Signals
ctrl_clr_all    : out std_logic;
ctrl_inc_one    : out std_logic;
ctrl_inc_zero   : out std_logic;
ctrl_proc_pulse : out std_logic;
ctrl_proc_gap   : out std_logic;
ctrl_dec_char   : out std_logic;
ctrl_dec_space  : out std_logic
```

DIVE INTO DECODER

control signal

```
-- Control Signals
ctrl_clr_all    : out std_logic;
ctrl_inc_one    : out std_logic;
ctrl_inc_zero   : out std_logic;
ctrl_proc_pulse : out std_logic;
ctrl_proc_gap   : out std_logic;
ctrl_dec_char   : out std_logic;
ctrl_dec_space  : out std_logic
```

Control Signal ??? → State!

```
-- Control Signals from CU
signal ctrl_clr_all    : std_logic;  [1]
signal ctrl_inc_one    : std_logic;  [2]
signal ctrl_inc_zero   : std_logic;  [3]
signal ctrl_proc_pulse : std_logic;  [4]
signal ctrl_proc_gap   : std_logic;  [5]
signal ctrl_dec_char   : std_logic;  [6]
signal ctrl_dec_space  : std_logic;  [7]
```

```
-- 1. Clear All (IDLE)
if ctrl_clr_all = '1' then
    counter_zero <= 0;
    counter_one  <= 0;
end if;
```

```
-- 2. Increment High Counter
if ctrl_inc_one = '1' then
    counter_one <= counter_one + 1;
end if;
```

```
-- 3. Process Pulse (Transition High -> Low)
-- Logic: Sinyal jadi low. Analisis durasi '1' nya.
if ctrl_proc_pulse = '1' then
    if counter_one < THRESH_DASH then
        -- Input merupakan DOT ('1')
        counter_signal <= counter_signal + 1;
        shiftreg_inb   <= shiftreg_inb(0 downto 0) & '1';
    else
        -- Input merupakan DASH ('1')
        counter_signal <= counter_signal + 2;
        shiftreg_inb   <= shiftreg_inb(7 downto 0) & "11";
    end if;
    counter_one  <= 0;
    counter_zero <= counter_zero + 1; -- Start counting gap
end if;
```

.....

Bagian control signal ini sebenarnya ekivalen dari state di FSM, setiap signal memiliki responsibility masing-masing

Decode

```
case to_integer(shiftreg_inb) is
    -- 5 simbol
    when 512 => ascii_out_int <= x"45"; -- E (.) = 1
    when 768 => ascii_out_int <= x"54"; -- T (-) = 11
    -- 2 simbol
    when 640 => ascii_out_int <= x"49"; -- I (..) = 101
    when 704 => ascii_out_int <= x"41"; -- A (.-) = 1011
    when 832 => ascii_out_int <= x"4E"; -- N (-.) = 1101
    when 864 => ascii_out_int <= x"4D"; -- M (--) = 11011
    -- 3 simbol
    when 672 => ascii_out_int <= x"53"; -- S (...) = 10101
    when 688 => ascii_out_int <= x"55"; -- U (..-) = 101011
    when 720 => ascii_out_int <= x"52"; -- R (.-.) = 101101
    when 728 => ascii_out_int <= x"57"; -- W (.--) = 101111
    when 848 => ascii_out_int <= x"44"; -- D (-..) = 110101
    when 856 => ascii_out_int <= x"4B"; -- K (-.-) = 110111
    when 872 => ascii_out_int <= x"47"; -- G (--.) = 110101
    when 876 => ascii_out_int <= x"4F"; -- O (---) = 110111011
    -- 4 simbol
    when 680 => ascii_out_int <= x"48"; -- H (....) = 1010101
    when 684 => ascii_out_int <= x"56"; -- V (...-) = 1010111
    when 692 => ascii_out_int <= x"46"; -- F (..-.) = 1010110
    when 724 => ascii_out_int <= x"4C"; -- L (.-..) = 1011010
    when 730 => ascii_out_int <= x"50"; -- P (.--.) = 1011011
    when 874 => ascii_out_int <= x"5A"; -- Z (--..) = 1101101
    when 731 => ascii_out_int <= x"4A"; -- J (.---) = 1011011
    when 852 => ascii_out_int <= x"42"; -- B (-...) = 1101011
    when 854 => ascii_out_int <= x"58"; -- X (-..-) = 1101011
    when 858 => ascii_out_int <= x"43"; -- C (-.-.) = 1101011
    when 859 => ascii_out_int <= x"59"; -- Y (-.--) = 1101011
    when 875 => ascii_out_int <= x"51"; -- Q (--.-) = 1101101
    -- kalo morse tidak dikenal
    when others => ascii_out_int <= x"3F"; -- ?
end case;
```

```
-- 7. Decode Space
if ctrl_dec_space = '1' then
    ascii_out_int <= x"20"; -- ASCII space
    valid_out_int <= '1';
    counter_zero  <= 0;
    counter_one   <= 0;
    NW <= '0';
    NL <= '0';
end if;
```

Output

```
TXT_OUTPUT: process(clk)
    file output_file       : text open write_mode is "output.txt";
    variable current_line  : line;
    variable char_count    : integer := 0;
    variable word_count    : integer := 0;
    variable ascii_int     : integer;
    variable char          : character;
begin
    if rising_edge(clk) then
        if reset = '1' then
            if word_count > 0 or char_count > 0 then
                writeline(output_file, current_line);
            end if;
            char_count := 0;
            word_count := 0;
        elsif valid_out_int = '1' then
            ascii_int := to_integer(unsigned(ascii_out_int));
            char := character'val(ascii_int);
            if char = ' ' then
                write(current_line, char);
                word_count := word_count + 1;
                char_count := 0;
                if word_count >= 16 then
                    writeline(output_file, current_line);
                    word_count := 0;
                end if;
            else
                if char_count < 32 then
                    write(current_line, char);
                    char_count := char_count + 1;
                else
                    null;
                end if;
            end if;
        end if;
    end if;
end process TXT_OUTPUT;
```

hasil akan disimpan pada ‘output.txt’

COUNTER?

```
if flag_char_end = '1' then
    if counter_signal = 9 then shiftreg_inb <= shiftreg_inb(8 downto 0) & '0';
    elsif counter_signal = 8 then shiftreg_inb <= shiftreg_inb(7 downto 0) & "00";
    elsif counter_signal = 7 then shiftreg_inb <= shiftreg_inb(6 downto 0) & "000";
    elsif counter_signal = 6 then shiftreg_inb <= shiftreg_inb(5 downto 0) & "0000";
    elsif counter_signal = 5 then shiftreg_inb <= shiftreg_inb(4 downto 0) & "00000";
    elsif counter_signal = 4 then shiftreg_inb <= shiftreg_inb(3 downto 0) & "000000";
    elsif counter_signal = 3 then shiftreg_inb <= shiftreg_inb(2 downto 0) & "0000000";
    elsif counter_signal = 2 then shiftreg_inb <= shiftreg_inb(1 downto 0) & "00000000";
    elsif counter_signal = 1 then shiftreg_inb <= shiftreg_inb(0) & "00000000";
    end if;

    counter_signal <= 0;
    NL <= '1'; -- Next Letter Flag
end if;
```

```
if counter_zero < THRESH_SPACE then
    counter_zero <= counter_zero + 1;
end if;
```

```
when SEQ_CHK_RISE => -- LOW Logic
    if morse_in = '1' then
        Next_uPC <= 1; -- Rise detected! Go to ENTRY HIGH (Process Gap)
    else
        if flag_space_end = '1' then
            Next_uPC <= 6;
        elsif flag_char_end = '1' then
            Next_uPC <= 5;
        else
            Next_uPC <= 4; -- Stay in Main Low
        end if;
    end if;
```

```
if counter_one < THRESH_DASH then
    -- Input merupakan DOT ('1')
    counter_signal <= counter_signal + 1;
    shiftreg_inb   <= shiftreg_inb(8 downto 0) & '1';
else
    -- Input merupakan DASH ('1')
    counter_signal <= counter_signal + 2;
    shiftreg_inb   <= shiftreg_inb(7 downto 0) & "11";
end if;
counter_one  <= 0;
counter_zero <= counter_zero + 1; -- Start counting gap
```

Counter_Signal ???

counter ini digunakan untuk menghitung sudah berapa banyak input yang diberikan oleh user hingga saat ini. Counter ini penting untuk mengetahui berapa ‘**0**’ yang perlu di padding saat input untuk huruf < 10 bit

Counter_one ???

counter ini digunakan untuk menghitung sudah berapa banyak input ‘1’ yang diberikan oleh user hingga saat ini. Counter ini penting untuk mengetahui apakah user bermaksud untuk menginput dash (-) atau dot (.)

Counter_zero ???

counter ini digunakan untuk menghitung sudah berapa banyak input ‘0’ yang diberikan oleh user hingga saat ini. Counter ini penting untuk mengetahui apakah user bermaksud untuk masuk ke simbol morse/huruf/kalimat selanjutnya

TESTING

```
procedure send_character(c : character) is
begin
    case c is
        when 'l' => send_dot; send_dash; send_dot; send_dot; end_char;
        when 'i' => send_dot; send_dot; end_char;
        when 's' => send_dot; send_dot; send_dot; end_char;
        when 't' => send_dash; end_char;
        when 'o' => send_dash; send_dash; send_dash; end_char;
        when 'f' => send_dot; send_dot; send_dash; send_dot; end_char;
        when 'h' => send_dot; send_dot; send_dot; send_dot; end_char;
        when 'e' => send_dot; end_char;
        when 'n' => send_dash; send_dot; end_char;
        when 'g' => send_dash; send_dash; send_dot; end_char;
        when 'a' => send_dot; send_dash; end_char;
        when 'm' => send_dash; send_dash; end_char;
        when ' ' => send_space;
        when others => report "Skipping unknown char" severity note;
    end case;
end procedure;

-- Kalimat
constant my_string : string := "list of the things that i hate the most";
```

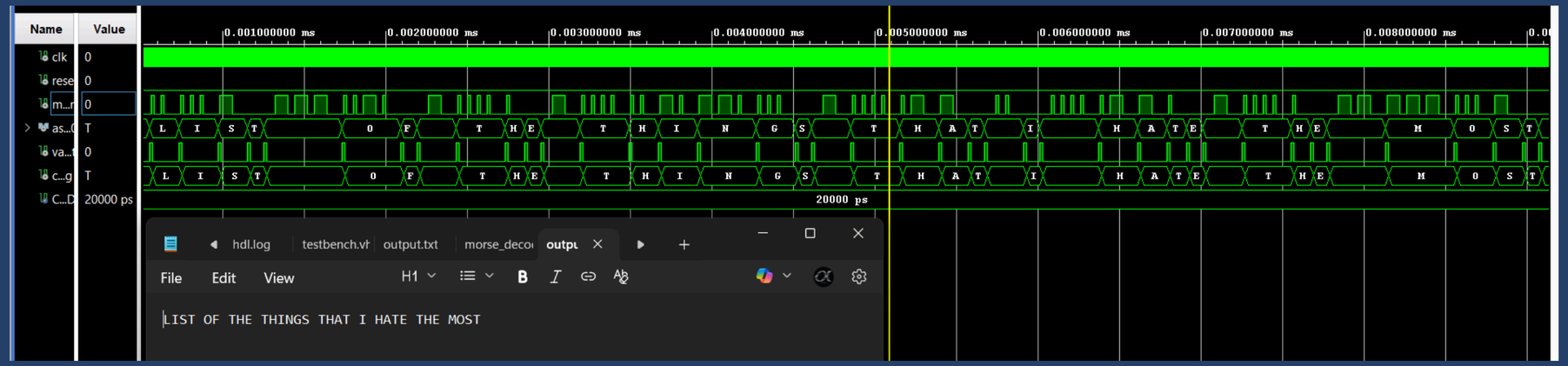
```
-- mengirim morse sequence
for i in my_string'range loop
    send_character(my_string(i));
end loop;
```

Testbench

Mengirimkan morse sequence dari kalimat “LIST OF THE THINGS THAT I HATE THE MOST”

TESTING

Output



TANYA JAWAB

Final Project PSD

TERIMA KASIH

Kelompok 4