



Course Code:	ENSC 252 1207
Course Title:	Fundamentals of Digital Logic
Semester Code:	Fall 2020 1207

Instructor:	Prof. Anita Tino
-------------	------------------

Asst/Lab No. :	5
Asst/Lab Title:	Distress Box

Submission Date:	November 15 2020
------------------	------------------

Student Name:	Devon Burnham
SFU ID:	301394066
Signature*:	(Devon Burnham)

*By signing above, I certify that the contribution made to this lab/assignment is solely mine and confirm that all work completed is my own. Any suspicion of plagiarism and/or copying will result in an investigation in the ENSC department. A mark of zero or F may be assigned, and depending on the level of severity, an FD grade on my transcript and/or expulsion from the school, according SFU's Code of Academic Integrity found online at <http://www.sfu.ca/policies/gazette/student/s10-01.html>

Sequencer VHDL Code
(genericized to support any data width and number of states)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY sequencer IS
    GENERIC (
        data_width : INTEGER := 6;
        N : INTEGER := 33);
    PORT (
        clk : IN STD_LOGIC;
        reset : IN STD_LOGIC;
        count : OUT unsigned(data_width-1 DOWNTO 0));
END sequencer;

ARCHITECTURE behaviour OF sequencer IS

    SIGNAL countSig : unsigned(data_width-1 DOWNTO 0);

BEGIN
    PROCESS (reset, clk) IS
    BEGIN
        IF (reset = '1') THEN
            countSig <= to_unsigned(N, data_width);
        ELSIF (rising_edge(clk)) THEN
            IF (countSig = to_unsigned(0, data_width)) THEN
                countSig <= to_unsigned(N, data_width);
            ELSE
                countSig <= countSig - to_unsigned(1, data_width);
            END IF;
        ELSE
            countSig <= countSig;
        END IF;
    END PROCESS;

    count <= countSig;

END behaviour;
```

SOS Sequencer Testbench Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_sequencer1 IS
END ENTITY;

ARCHITECTURE test OF tb_sequencer1 IS

    COMPONENT sequencer IS
        GENERIC (
            data_width : INTEGER := 6;
            N : INTEGER := 33);
        PORT (
            clk : IN STD_LOGIC;
            reset : IN STD_LOGIC;
            count : OUT unsigned(data_width - 1 DOWNTO 0));
    END COMPONENT;

    CONSTANT HALF_PERIOD : TIME := 10 ns;
    CONSTANT PERIOD : TIME := 20 ns;

    SIGNAL sigclk : STD_LOGIC := '1';
    SIGNAL sigreset : STD_LOGIC;
    SIGNAL sigcount : unsigned(5 DOWNTO 0);

BEGIN

    DUT : sequencer [GENERIC MAP(6, 33)]
    PORT MAP(sigclk, sigreset, sigcount);

    --to cycle clk for the duration of the test
    sigclk <= NOT sigclk AFTER HALF_PERIOD;

    PROCESS IS
    BEGIN

        --reset
        sigreset <= '1';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';
        WAIT;

    END PROCESS;

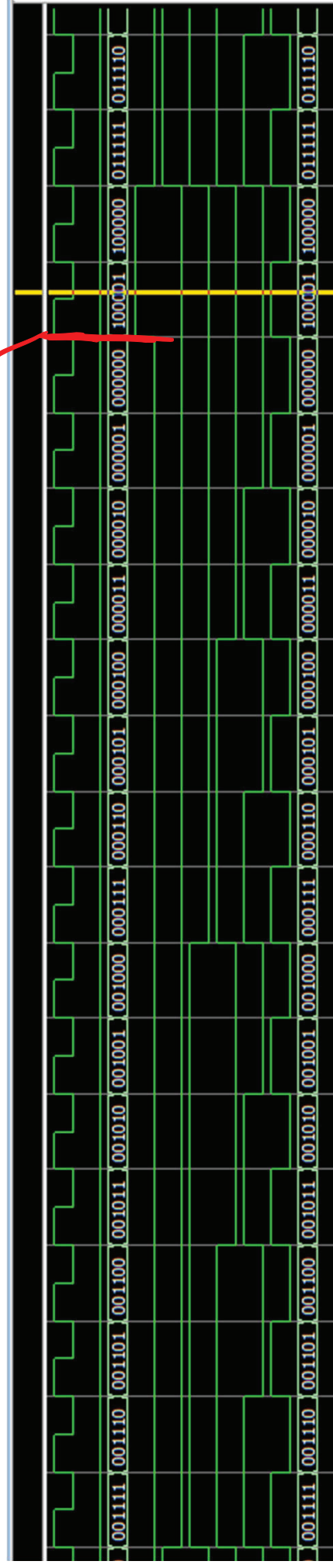
END ARCHITECTURE;
```

6-wide, 34 states

reset



Underflow!



DEVO Sequencer Testbench Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_sequencer2 IS
END ENTITY;

ARCHITECTURE test OF tb_sequencer2 IS

    COMPONENT sequencer IS
        GENERIC (
            data_width : INTEGER := 6;
            N : INTEGER := 33);
        PORT (
            clk : IN STD_LOGIC;
            reset : IN STD_LOGIC;
            count : OUT unsigned(data_width - 1 DOWNTO 0));
    END COMPONENT;

    CONSTANT HALF_PERIOD : TIME := 10 ns;
    CONSTANT PERIOD : TIME := 20 ns;

    SIGNAL sigclk : STD_LOGIC := '1';
    SIGNAL sigreset : STD_LOGIC;
    SIGNAL sigcount : unsigned(5 DOWNTO 0);

BEGIN

    DUT : sequencer GENERIC MAP(6, 43)
    PORT MAP(sigclk, sigreset, sigcount);

    --to cycle clk for the duration of the test
    sigclk <= NOT sigclk AFTER HALF_PERIOD;

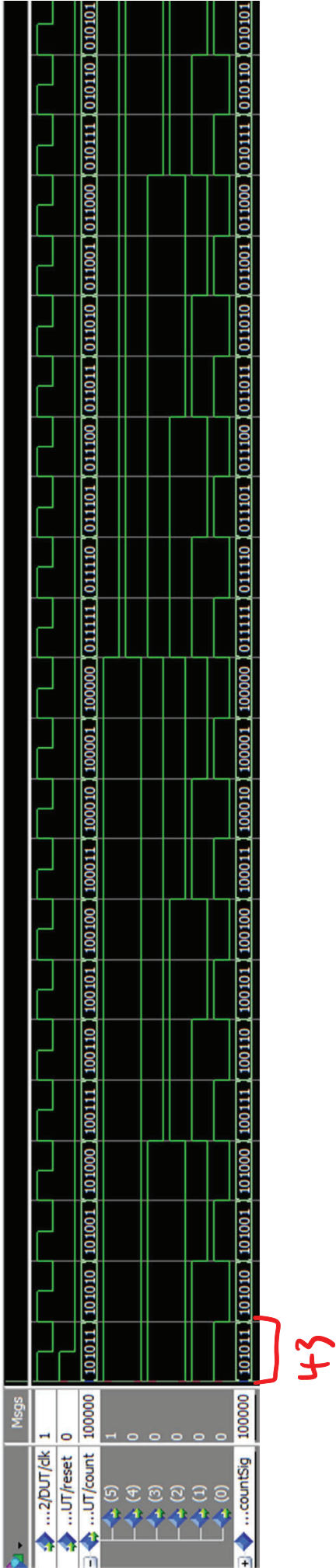
    PROCESS IS
    BEGIN

        --reset
        sigreset <= '1';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';
        WAIT;
    END PROCESS;

END ARCHITECTURE;
```

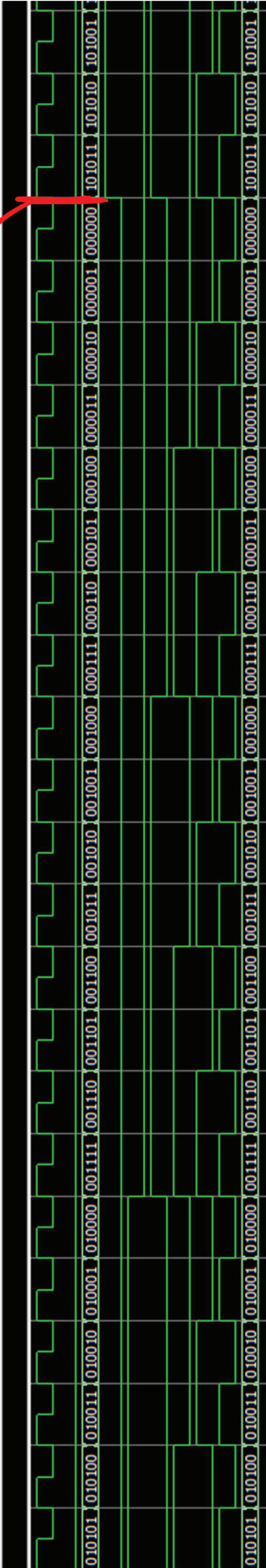
44 states
(only difference vs. SOS sequencer)

DEVO Sequencer Waveforms



This set of waveforms is identical to the previous set of waveforms, the sole difference is that it begins and underflows to 101011 (43), instead of 100001 (33). This is because the morse code for DEVO, the first 4 letters of my name (Devon), is 44 units of time long, including the 7 units at the end of each transmission to signify the end of a word.

underflow



Decoder VHDL Code
(genericized to support any decoding key)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY decoder IS
    GENERIC (
        morse : STD_LOGIC_VECTOR);
    PORT (
        seq : IN unsigned(5 DOWNTO 0);
        WaveOut : OUT STD_LOGIC);
END ENTITY;

ARCHITECTURE behaviour OF decoder IS
BEGIN
    WaveOut <= morse((morse'length - 1) - to_integer(seq));
END ARCHITECTURE;
```

.

SOS Decoder Testbench Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_decoder1 IS
END ENTITY;

ARCHITECTURE test OF tb_decoder1 IS

    COMPONENT decoder IS
        GENERIC (
            morse : STD_LOGIC_VECTOR);
        PORT (
            seq : IN unsigned;
            WaveOut : OUT STD_LOGIC);
    END COMPONENT;

    CONSTANT HALF_PERIOD : TIME := 10 ns;
    CONSTANT PERIOD : TIME := 20 ns;

    SIGNAL sigseq : unsigned(5 DOWNTO 0);
    SIGNAL sigwave : STD_LOGIC;

BEGIN

    DUT : decoder GENERIC MAP("1010100011101110111000101010000000")
    PORT MAP(sigseq, sigwave);

    PROCESS IS
    BEGIN

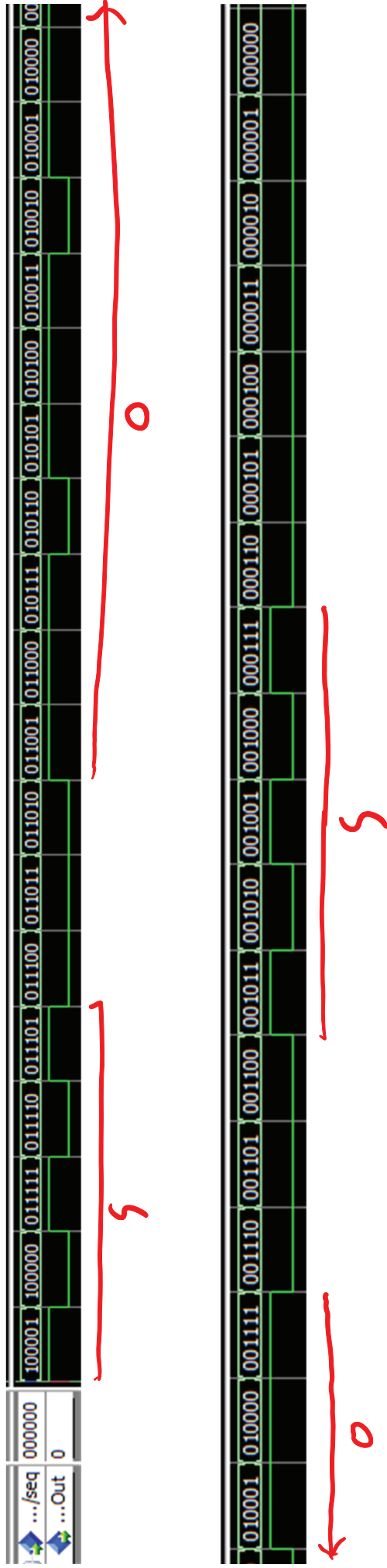
        FOR i IN 33 DOWNTO 0 LOOP
            sigseq <= to_unsigned(i, 6);
            WAIT FOR HALF_PERIOD;
        END LOOP;

        WAIT;
    END PROCESS;

END ARCHITECTURE;
```

Morse code for "SOS"

SOS Decoder Testbench Waveforms



This waveform provides an unsigned signal decrementing from 33 down to 0, and the expected output is the morse code for "SOS", which is what is produced. The waveform has been split here for easier viewing; the first waveform ends at 010000 and the second begins a little earlier at 010001. Nevertheless, the expected pattern of SOS is clearly visible in the output.

DEVO Decoder Testbench Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_decoder2 IS
END ENTITY;

ARCHITECTURE test OF tb_decoder2 IS

    COMPONENT decoder IS
        GENERIC (
            morse : STD_LOGIC_VECTOR);
        PORT (
            seq : IN unsigned;
            WaveOut : OUT STD_LOGIC);
    END COMPONENT;

    CONSTANT HALF_PERIOD : TIME := 10 ns;
    CONSTANT PERIOD : TIME := 20 ns;

    SIGNAL sigseq : unsigned(5 DOWNTO 0);
    SIGNAL sigwave : STD_LOGIC;

BEGIN

    DUT : decoder GENERIC MAP("11101010001000101010111000111011101110000000")
    PORT MAP(sigseq, sigwave);

    PROCESS IS
    BEGIN

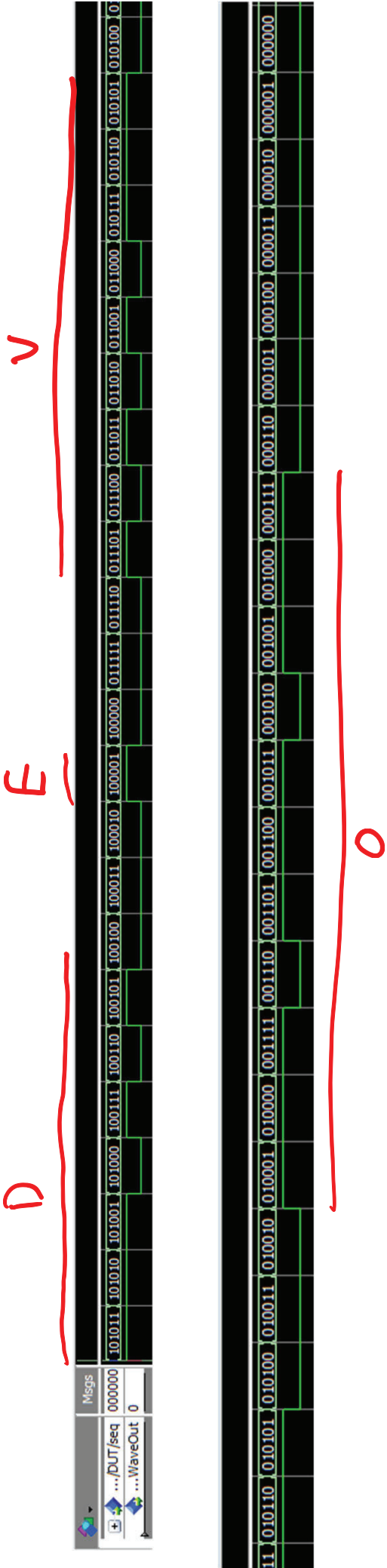
        FOR i IN 43 DOWNTO 0 LOOP
            sigseq <= to_unsigned(i, 6);
            WAIT FOR HALF_PERIOD;
        END LOOP;

        WAIT;
    END PROCESS;

END ARCHITECTURE;
```

Morse for "DEVO"
(only difference vs. SOS decoder)

DEVO Decoder Testbench Waveforms



This waveform is the same as the previous one, except it decodes a pattern of 44 down to 0 into the morse code representation for DEVO. This is successfully produced, as seen above.

Box VHDL Code
(genericized to support any repeating output)

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY box IS
    GENERIC (
        data_width : INTEGER := 6;
        N : INTEGER := 33;
        morse : STD_LOGIC_VECTOR);
    PORT (
        clk : IN STD_LOGIC;
        reset : IN STD_LOGIC;
        enable : IN STD_LOGIC;
        code_out : OUT STD_LOGIC);
END ENTITY;

ARCHITECTURE structural OF box IS

    COMPONENT sequencer IS
        GENERIC (
            data_width : INTEGER := 6;
            N : INTEGER := 33);
        PORT (
            clk : IN STD_LOGIC;
            reset : IN STD_LOGIC;
            count : OUT unsigned(data_width-1 DOWNTO 0));
    END COMPONENT;

    COMPONENT decoder IS
        GENERIC (
            morse : STD_LOGIC_VECTOR);
        PORT (
            seq : IN unsigned(data_width-1 DOWNTO 0);
            WaveOut : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL countSig : unsigned(data_width-1 DOWNTO 0);
    SIGNAL waveSig : STD_LOGIC;

BEGIN

    sequ : sequencer GENERIC MAP(data_width, N)
    PORT MAP(clk, reset, countSig);
    dec : decoder GENERIC MAP(morse)
    PORT MAP(countSig, waveSig);

    code_out <= waveSig WHEN (enable = '1') ELSE
        '0';

END ARCHITECTURE;
```

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_box1 IS
END ENTITY;

ARCHITECTURE test OF tb_box1 IS

    COMPONENT box IS
        GENERIC (
            data_width : INTEGER := 6;
            N : INTEGER := 33;
            morse : STD_LOGIC_VECTOR);
        PORT (
            clk : IN STD_LOGIC;
            reset : IN STD_LOGIC;
            enable : IN STD_LOGIC;
            code_out : OUT STD_LOGIC);
    END COMPONENT;

    CONSTANT HALF_PERIOD : TIME := 10 ns;
    CONSTANT PERIOD : TIME := 20 ns;

    SIGNAL sigclk : STD_LOGIC := '1';
    SIGNAL sigreset : STD_LOGIC;
    SIGNAL sigenable : STD_LOGIC;
    SIGNAL sigcode_out : STD_LOGIC;

BEGIN

    DUT : box GENERIC MAP(6, 33, "1010100011101110111000101010000000")
    PORT MAP(sigclk, sigreset, sigenable, sigcode_out);

    --to cycle clk for the duration of the test
    sigclk <= NOT sigclk AFTER HALF_PERIOD;

    PROCESS IS
    BEGIN

        --reset
        sigreset <= '1';
        sigenable <= '1';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';

        WAIT FOR PERIOD * 34; --to ensure looping works

        --reset
        sigreset <= '1';
        sigenable <= '0';
        WAIT FOR PERIOD;
        sigreset <= '0';

        wait;

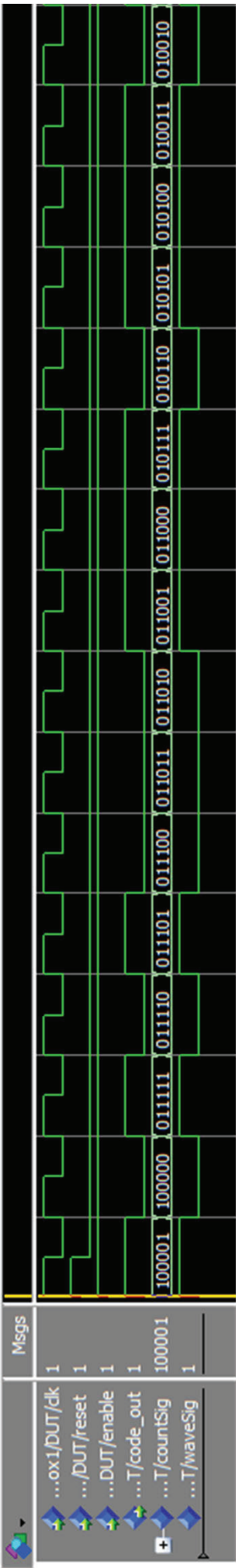
    END PROCESS;

END ARCHITECTURE;

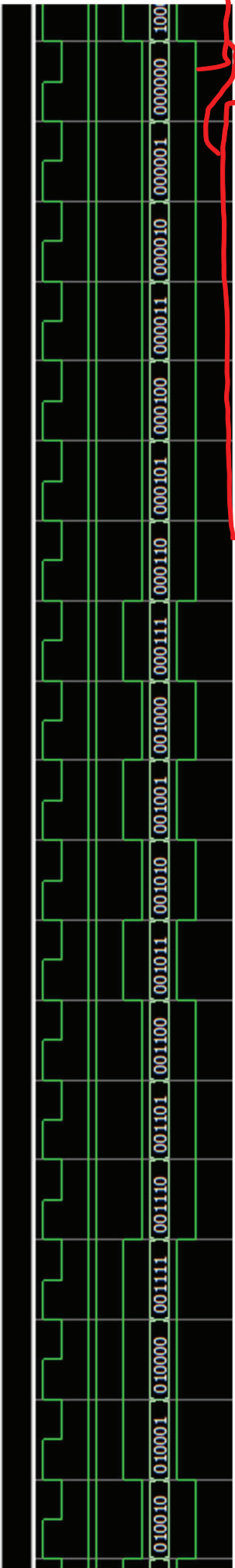
```

34 states, morse for
"sos"

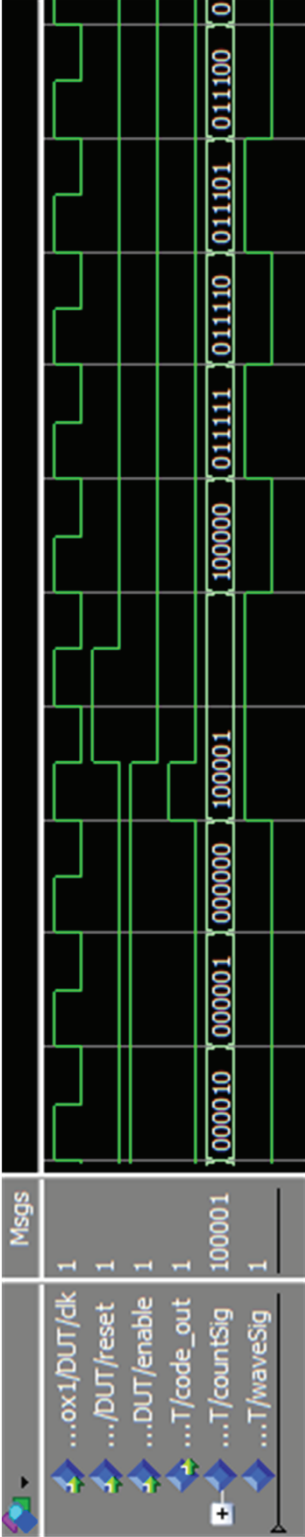
SOS Box Testbench Waveforms



With only clk and reset inputs, this circuit produces the correct waveform for the morse code representation of SOS, with the rising edge of the clk signal corresponding to the next unit of time.



Visible at the end of this waveform is the beginning of the next transmission, after the 7 spaces it resets to state 33 and begins to transmit 'S'. This is swiftly interrupted to demonstrate the asynchronous reset and enable/disable functionality.



This screenshot shows how setting enable to '0' suppresses the output, regardless of what waveSig is showing.

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_box2 IS
END ENTITY;

ARCHITECTURE test OF tb_box2 IS

    COMPONENT box IS
        GENERIC (
            data_width : INTEGER := 6;
            N : INTEGER := 33;
            morse : STD_LOGIC_VECTOR);
        PORT (
            clk : IN STD_LOGIC;
            reset : IN STD_LOGIC;
            enable : IN STD_LOGIC;
            code_out : OUT STD_LOGIC);
    END COMPONENT;

    CONSTANT HALF_PERIOD : TIME := 10 ns;
    CONSTANT PERIOD : TIME := 20 ns;

    SIGNAL sigclk : STD_LOGIC := '1';
    SIGNAL sigreset : STD_LOGIC;
    SIGNAL sigenable : STD_LOGIC;
    SIGNAL sigcode_out : STD_LOGIC;

BEGIN

    DUT : box GENERIC MAP(6, 43, "11101010001000101010111000111011101110000000")
    PORT MAP(sigclk, sigreset, sigenable, sigcode_out);

    --to cycle clk for the duration of the test
    sigclk <= NOT sigclk AFTER HALF_PERIOD;

    PROCESS IS
    BEGIN

        --reset
        sigreset <= '1';
        sigenable <= '1';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';

        WAIT FOR period * 44; --to ensure looping works

        --reset
        sigreset <= '1';
        sigenable <= '0';
        WAIT FOR period;
        sigreset <= '0';

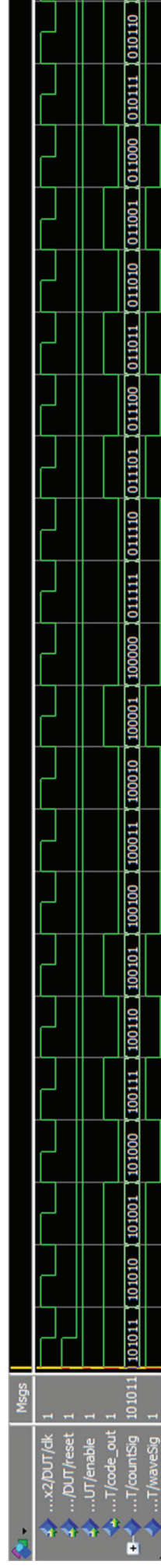
    wait;
    END PROCESS;

END ARCHITECTURE;

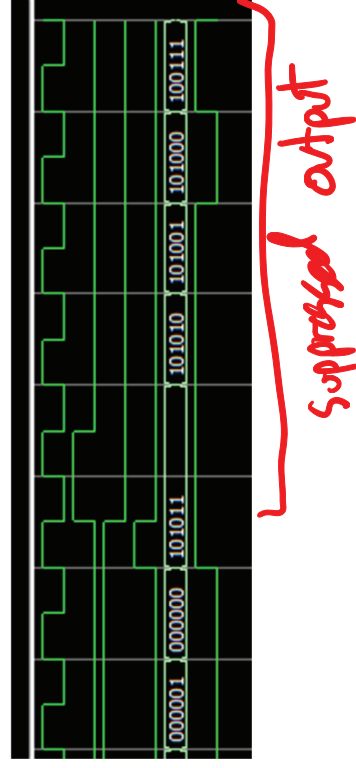
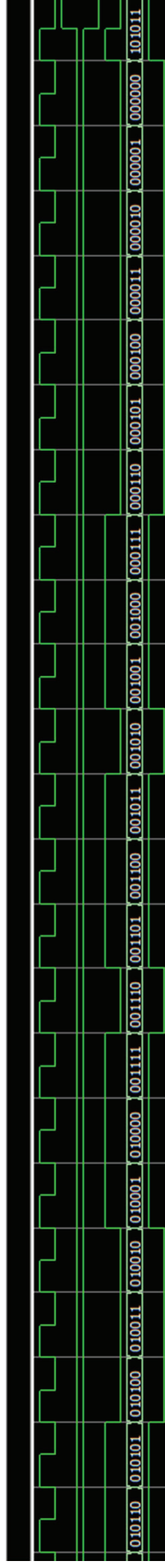
```

44 states, morse for
"DEVO" (only difference
vs. box1)

DEVO Box Testbench Waveforms



This waveform has identical characteristics to the previous waveform, except it produces the morse code for DEVO instead of SOS. Reset, clk, and enable all work the same as before, as shown in these screenshots.



Distress Box VHDL Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY distress_box IS
    PORT (
        clk : IN STD_LOGIC;
        reset : IN STD_LOGIC;
        enable : IN STD_LOGIC;
        sel : IN STD_LOGIC;
        code_out : OUT STD_LOGIC
    );
END ENTITY;

ARCHITECTURE structural OF distress_box IS

    COMPONENT box IS
        GENERIC (
            data_width : INTEGER := 6;
            N : INTEGER := 33;
            morse : STD_LOGIC_VECTOR
        );
        PORT (
            clk : IN STD_LOGIC;
            reset : IN STD_LOGIC;
            enable : IN STD_LOGIC;
            code_out : OUT STD_LOGIC
        );
    END COMPONENT;

    SIGNAL code1 : STD_LOGIC;
    SIGNAL code2 : STD_LOGIC;

BEGIN

    b1 : box GENERIC MAP(6, 33, "1010100011101110111000101010000000")
        PORT MAP(clk, reset, enable, code1);
    b2 : box GENERIC MAP(6, 43, "11101010001000101010111000111011101110000000")
        PORT MAP(clk, reset, enable, code2);

    code_out <= code1 WHEN (sel = '0') ELSE
        code2;

END ARCHITECTURE;
```

SOS

DEVO

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
ENTITY tb_distress_box IS
END ENTITY;
ARCHITECTURE test OF tb_distress_box IS
    COMPONENT distress_box IS
        PORT (
            clk : IN STD_LOGIC;
            reset : IN STD_LOGIC;
            enable : IN STD_LOGIC;
            sel : IN STD_LOGIC;
            code_out : OUT STD_LOGIC );
    END COMPONENT;

    CONSTANT HALF_PERIOD : TIME := 10 ns;
    CONSTANT PERIOD : TIME := 20 ns;

    SIGNAL sigclk : STD_LOGIC := '1';
    SIGNAL sigreset : STD_LOGIC;
    SIGNAL sigenable : STD_LOGIC;
    SIGNAL sigsel : STD_LOGIC;
    SIGNAL sigcode_out : STD_LOGIC;

```

Distress Box Testbench Code
(some linebreaks removed to fit on one page)

```

BEGIN
    DUT : distress_box PORT MAP(sigclk, sigreset, sigenable, sigsel, sigcode_out);
    --to cycle clk for the duration of the test
    sigclk <= NOT sigclk AFTER HALF_PERIOD;

    PROCESS IS
    BEGIN
        --reset
        sigreset <= '1';
        sigenable <= '1';
        sigsel <= '0';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';

        WAIT FOR period * 34; --to ensure looping works

        --reset
        sigreset <= '1';
        sigenable <= '0';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';

        WAIT FOR period * 5;

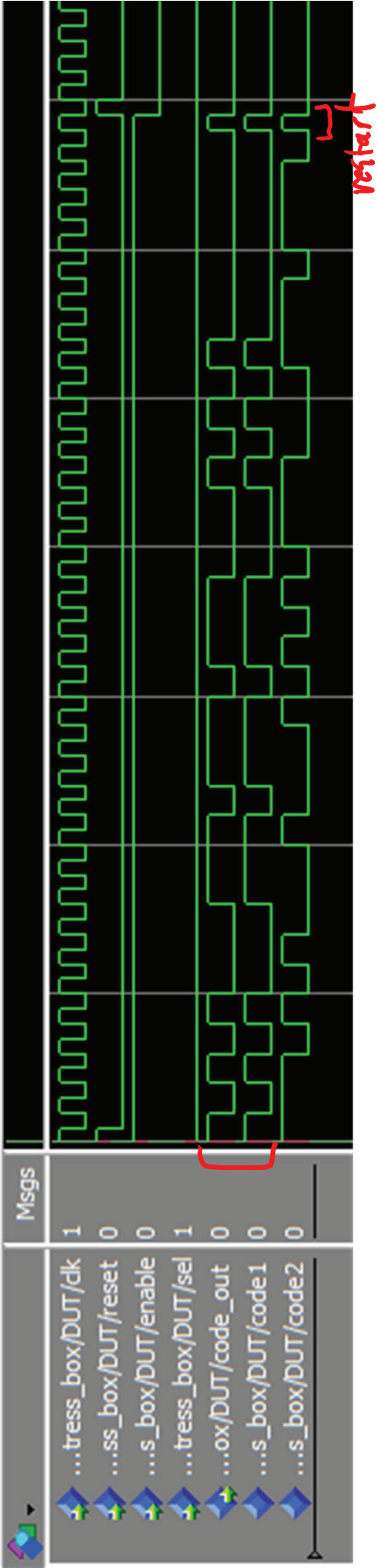
        --reset
        sigreset <= '1';
        sigenable <= '1';
        sigsel <= '1';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';

        WAIT FOR period * 44; --to ensure looping works

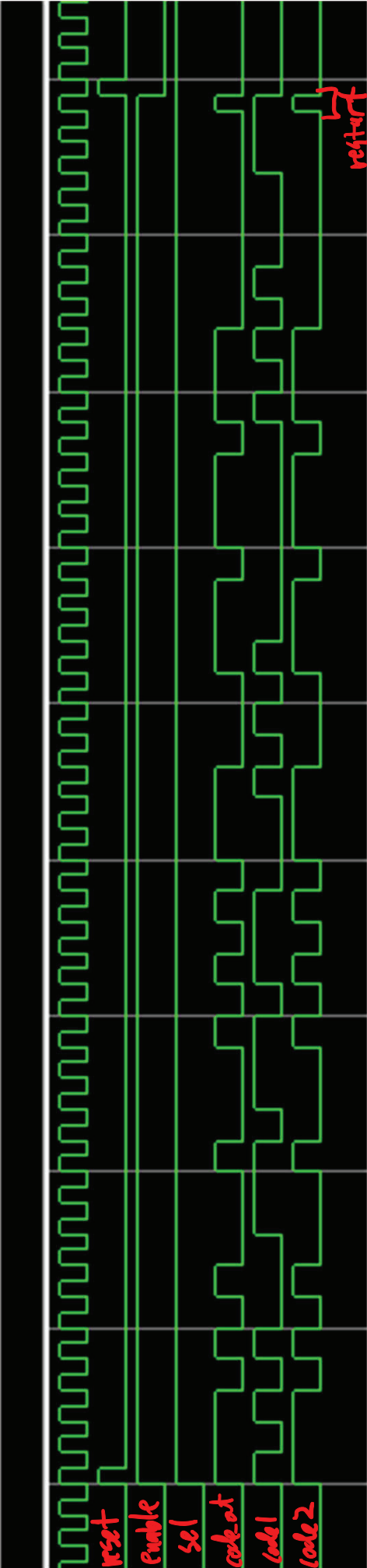
        --reset
        sigreset <= '1';
        sigenable <= '0';
        WAIT FOR HALF_PERIOD;
        sigreset <= '0';
        wait;
    END PROCESS;
END ARCHITECTURE;

```

Distress Box Testbench Waveforms



In these waveforms, the complete functionality of a distress box is demonstrated. Initially, with `sel` set to 0, the box transmits SOS in morse code. As such, `code1`, which corresponds to the SOS box, is copied to `code_out`. The box begins to restart the transmission before we disable transmission by setting `enable` to 0.



Next, the box is reset and reenabled, and we switch from SOS mode to DEVO mode by switching `sel` to 1. This means `code2` is copied to `code_out`, which produces the morse code for DEVO. Similar behaviour to the above waveform is exhibited: it begins a retransmission before we disable transmission.