Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет електроніки

Кафедра конструювання електронно-обчислювальної апаратури

Пояснювальна записка до курсової роботи

Тема: приймач та передавач даних Азбукою Морзе

Студент, що виконав: Тисяк Є. В. _____(підпис) "___"_____20___р.

Викладач: Антонюк О. І. _____(підпис) "___"_____20___р.

Київ 2021

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

1 – напруга логічного рівня одиниці

0 – напруга логічного рівня нуля

ФОРМУЛЮВАННЯ ЗАВДАННЯ

Ускладнене завдання другого варіанту курсової роботи в оригіналі звучить як: «Спроектувати та протестувати передавач та приймач сповіщень Азбукою Морзе (використовувати кодування літер для англійського алфавіту). По зовнішньому сигналу «старт» передавач повинен сформувати та надіслати сповіщення, яке містить прізвище та ім'я студента і номер академічної групи. Приймач повинен прийняти сповіщення та дешифрувати його (перекодувати сповіщення у формат ASCII). Для зупинки приймача використовується знак «кінець передачі». Сповіщення, що надсилається, зчитується з блоку пам'яті. Сповіщення, що приймається, теж зберігається у блоці пам'яті.»

Відокремлю пункти та опишу власну реалізацію:

1. Сигналом «старт» буде слугувати одночасне натискання кнопки «reset» на приймачі та передавачі. Даний сигнал буде моделюватись одним і тим самим регістром модуля testbench. Цей регістр буде називатись «i_rst». І для передавача, і для приймача входи скидання будуть називатись так само, як і регістр, тобто «i_rst». В цих входів буде активний високий рівень, тобто 1 на цьому вході призведе до скидання приймача та передавача. Після відпускання кнопки, тобто коли на ній буде 0, приймач і передавач почнуть відповідно приймати і передавати сигнал.
2. Сигнал, що передаватиметься це – «TUSAK DK91». Зверніть увагу, що шостий символ це «space».
3. Після того, як приймач отримає сигнал, що буде дешифрований як «1», на виході «o_the_end» приймача встановиться напруга рівня логічної одиниці. Даний вихід можна підключити через обмежуючий резистор до світлодіоду і таким чином ми дізнаємось про те, що передача даних завершена. Після надходження 1 будемо вважати роботу нашої каналу зв'язку закінченою, тобто «1» це буде знак «кінець передачі».
4. В передавачі і приймачі роль пам'яті на себе візьмуть шістдесят чотири 8-розрядні регістри. При створенні реальних прототипів варто було б потурбуватись про можливість в подальшому працювати з вмістом пам'яті. Для передавачам це можливість запису в пам'ять, для приймача це можливість читати її. Оскільки це доволі складно, з урахуванням того, що при симуляції в ModelSim можливо легко працювати з регістрами, маючи доступ до будь якої його ділянки, цим можна знехтувати.
5. Для кодування Азбукою Морзе великих літер, а саме «T», «U», «S» , «A», «K», «D» та цифр «9», «1»,що передаються буду використовувати шифр наведений на сайті Азбука Морзе — (wikipedia.org) і , а саме табличку наведену на рисунку Рис.1. Основною вимогою до кодування є дотримання пропорції відношення

часу точки до тире як 1 до 3. Пауза між складовими одного знаку така само, як тривалість точки, а між знаками така, як тривалість тире. Пауза між словами в 7 разів більша за тривалість точки.

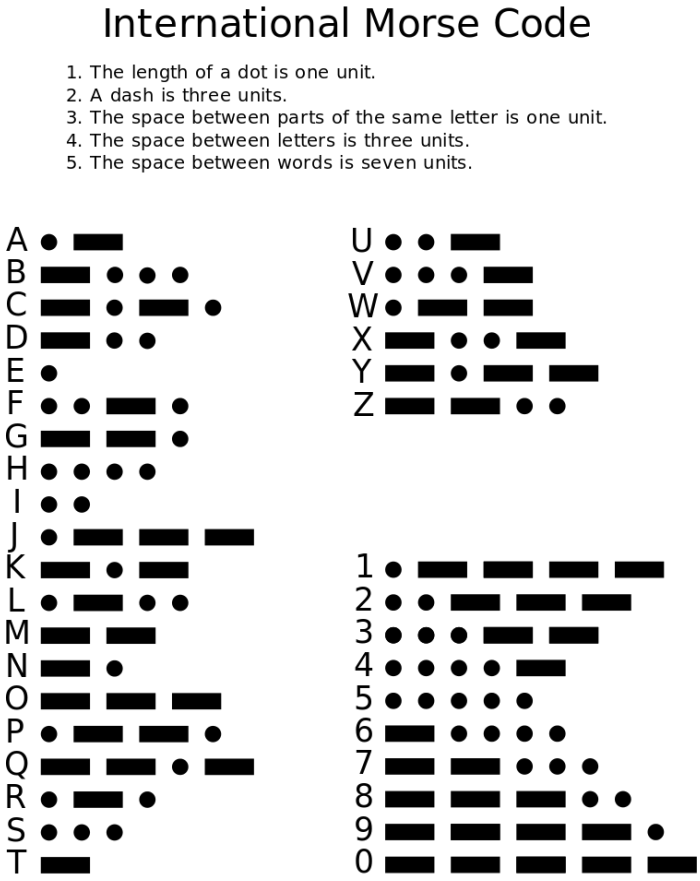

Рис.1 Табличка кодування сигналів Азбукою Морзе.

ВСТУП

Кодування повідомлень Азбукою Морзе широко застосовувалось в XIX та XX століттях. Основними перевагами є:

1. Висока завадостійкість.
2. Можливість обучений людині приймати сигнал на слух та передавати в ручну.
3. Низькі вимоги до апаратури зв’язку, відповідно дешевизна, простота та надійність.

Основними недоліком є низька швидкість передачі даних. З розвитком електротехніки і появою можливості працювати з більш складними даними кодування повідомлень Азбукою Морзе перестало застосовуватись. Зараз це швидше радіолюбительська ніша.

Завданням даного курсового проекту є показати вміння студента користуватись засобами автоматизованого проектування цифрових пристроїв, а саме знання HDL мови програмування Verilog та вміння користуватись програмним забезпеченням для перевірки спроектованих пристроїв ModelSim.

СТРУКТУРНА СХЕМА

Загалом канал зв’язку по якому повинен передаватись сигнал описаний в пункті ВСТУП скрадатиметься з двох блоків. Структура каналу зв’язку наведена на Рис.2.

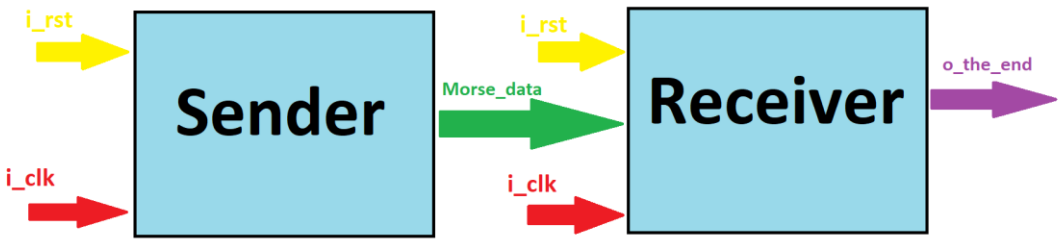

Рис.2 Структурна схема

Перший блок – це передавач (sender). Цей пристрій міститиме в пам’яті (Рис.3) двійковий код в якому буде зашифрована в ASCII форматі повідомлення, що передається, тобто «TUSAK DK91». Вміст пам’яті наведено в шістнадцятковій системі числення на Рис.4, та в форматі ASCII на Рис.5

```
reg [7:0] mem[15:0];
```

Рис.3 Оголошення регістра пам’яті в блоці sender.v на мові Verilog.

```
sim:/main_morse_tb/my_sender/mem @ 4609 ns
xx xx xx xx xx xx 31 39 4b 44 20 4b 41 53 55 54
```

Рис.4 Вміст пам’яті передавача в шістнадцятковій системі числення.

```
sim:/main_morse_tb/my_sender/mem @ 2201 ns
      1 9 K D   K A S U T
```

Рис.5 Вміст пам’яті передавача в форматі ASCII.

Для того, щоб заповнити цей регістр потрібною інформацією я використовую не синтезовану команду мови Verilog «$readmemh». Те як це виглядає в коді можна побачити на Рис.6.

```
initial $readmemh("mem.txt", mem);
```

Рис.6 Команда, що заповнить вміст регістра пам’яті.

З Рис.6 можна побачити, що в регістр будуть записані данні, що знаходяться в файлі «mem.txt», для зчитування цей файл повинен знаходитись в одній папці з рештою проекту. Структуру проекту можна відслідковувати за посиланням LAMPA23/Morse-code (github.com), а також вона наведена на Рис.7.

Имя	Дата изменения	Тип	Размер
mem	26.12.2021 1:53	Текстовый докум...	1 КБ
sim	26.12.2021 16:19	Файл "DO"	2 КБ
main_morse_tb	26.12.2021 15:38	Файл "V"	1 КБ
receiver	26.12.2021 16:21	Файл "V"	2 КБ
sender	26.12.2021 1:54	Файл "V"	3 КБ

Рис.7 Файлова структура проекту.

Всередині приймача описаний шифратор, який в залежності від значення регістру adr заповнювати робочий регістр data_morse відповідними даними. Регістр data_morse буде зсуватись право, таким чином передаючи свій вміст приймачу (вважаємо, що старші біти зліва). В передавача будуть 2 одно-розрядні входи: синхронізації (i_clk) та скидання(i_rst), та один одно-розрядний вихід – вихід даних (o_data_morse).

Другий блок – це приймач (receiver). Це блок має 3 одно-розрядні входи: синхронізації(i_clk), скидання(i_rst) та вхід даних (i_data_morse), та один вихід для сигналізації кінця передачі даних (o_the_end), що описаний в частині ВСТУП. Приймач містить в собі пам'ять такої самої розрядності як і передавач, робочі регістри та дешифратор, який в залежності від даних в робочих регістрах буде записувати в пам'ять відповідні символи зашифровані в ASCII. Адреса пам’яті в яку буде записана відповідна інформація визначатиметься одним із робочих регістрів, а саме adr. Зазначу, що вихід приймача o_the_end (як і вихід передавача o_data_morse) представляє собою регістр, а всі входи є просто привідниками (змінними типу wire). Вигляд

ініціалізації входів і виходів передавача і приймача наведені на Рис.8 та Рис.9 відповідно.

```
input i_clk, i_rst;
output reg o_data_morse;
```

Рис.8 Вигляд ініціалізації входів і виходів передавача і приймача

```
input i_clk, i_rst, i_data_morse;
output reg o_the_end;
```

Рис.9 Вигляд ініціалізації входів і виходів приймача.

РОЗРОБКА ТА ПЕРЕВІРКА СКЛАДОВИХ ЧАСТИН

Процес розробки я почав з передавача, оскільки для розробки приймача дуже зручно мати працюючий передавач. За допомогою передавача я зможу формувати сигнал для тестування приймача. Оскільки правильна робота і приймача і передавача означає виконання завдання, то це вже буде кінцеве тестування.

Принцип роботи передавача наступний: при скиданні пристрою входом i_rst, регістр adr та data_morse скидаються в нуль, а регістр counter в мінус один. Ділянка коду, що відповідає за це – Рис.10, вигляд на часовій діаграмі – Рис.11.

```
always @(posedge i_clk) begin
    if (i_rst) begin
        adr = 0;
        data_morse = 0;
        counter = -1;
```

Рис.10 Ділянка коду, що відповідає за скидання пристрою.

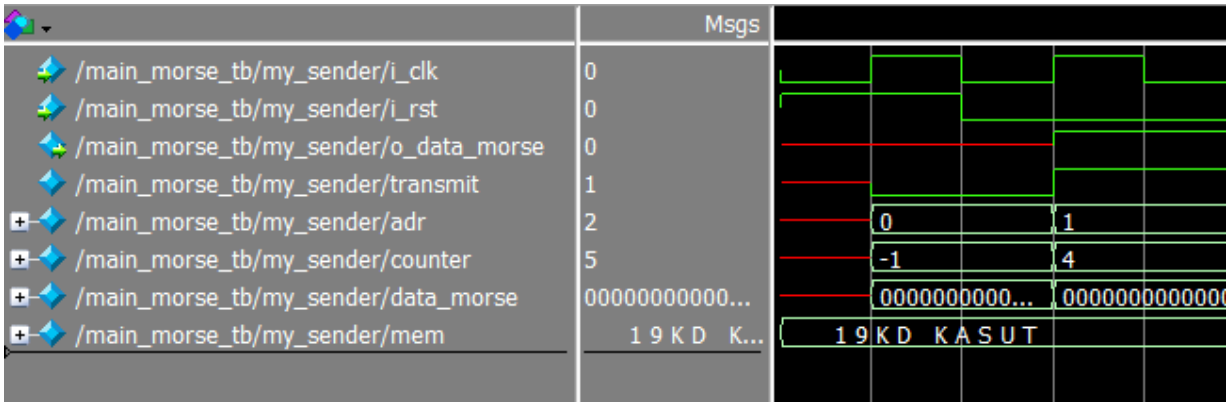

Рис.11 Вигляд скидання передавачу на часовій діаграмі.

Через те, що counter = -1 по наступному передньому фронту регістр transmit скинеться в нуль. Ділянка коду, що відповідає за це – Рис.12, вигляд на часовій діаграмі – Рис.11. Розрядності регістрів передавача наведені на Рис.13.

```
always @(posedge i_clk) begin
    if(counter != -1) begin
        o_data_morse = data_morse[counter];
        counter = counter - 1;
    end else begin
        transmit = 0;
    end
end
```

Рис.12 Ділянка коду, що відповідає за скидання регістра transmit, зсув регістра o_data_morse та спадання регістра counter на 1 з кожним зсувом регістра o_data_morse.

```
reg [7:0] mem[15:0];
reg [31:0] data_morse;
reg [63:0] adr;
reg transmit;
```

Рис.13 Розрядності регістрів передавача.

По наступному передньому фронту шифратор запише в регістр data_morse значення яке відповідає вмісту пам’яті за адресую, яка відповідає тому, що записане в регістр adr. Після цього, вміст adr одразу підніметься на 1, а transmit стане 1. Ділянка коду, що відповідає за це – Рис.15, вигляд на часовій діаграмі – Рис.14.

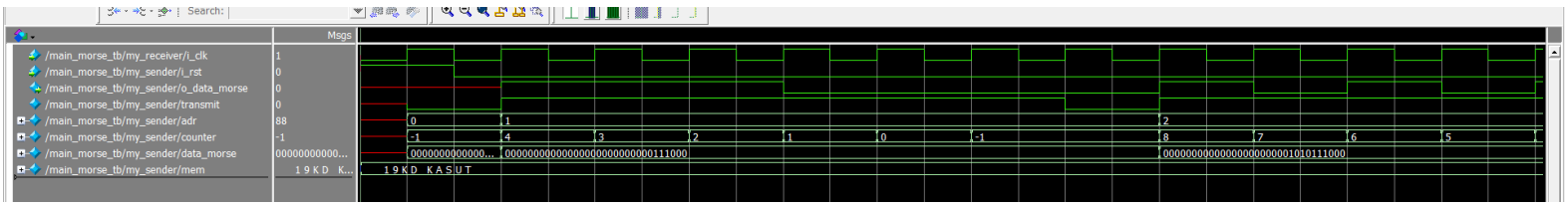

Рис.14 Часова діаграма початку передачі сигналу.

```
always @(posedge i_clk) begin
    if (i_rst) begin
        adr = 0;
        data_morse = 0;
        counter = -1;
    end else begin
        if(!transmit) begin
            case (mem[adr])
                'h 54: begin//T
                    data_morse = 'b 111000;
                    counter = 5;
                end
                'h 55: begin//U
                    data_morse = 'b 1010111000;
                    counter = 9;
                end
                'h 53: begin//S
                    data_morse = 'b 10101000;
                    counter = 7;
                end
                'h 41: begin//A
                    data_morse = 'b 10111000;
                    counter = 7;
                end
                'h 4B: begin//K
                    data_morse = 'b 111010111000;
                    counter = 11;
                end
                'h 20: begin//[space]
                    data_morse = 'b 000;
                    counter = 2;
                end
                'h 44: begin//D
                    data_morse = 'b 1110101000;
                    counter = 11;
                end
                'h 39: begin//9
                    data_morse = 'b 1110_1110_1110_1110_1000;
                    counter = 19;
                end
                'h 31: begin//1
                    data_morse = 'b 1011_1011_1011_1011_1000;
                    counter = 19;
                end
            endcase
            adr = adr + 1;
            transmit = 1;
        end
    end

end
```

Рис.15 Шифратор в передавачі та оточуюча його логіка.

Розглянемо на прикладі, як визначаються дані, що передаються в data_morse на прикладі «K». «K» в коді Морзе позначається як «тире-крапка-тире» (Рис.1). Відповідно спершу прописується тире(111), потім пауза(0), потім крапка(1), потім пауза(0), потім тире(111), потім кінець символу(000). Отримуємо 111_0_1_0_111_000. У відповідності до довжини отриманого коду виставляється своє значення counter яке повинно бути на 1 менше ніж довжина коду, оскільки потрібно рахувати не до 1, а до нуля.

Зверну увагу, що символ space передається, як окремий символ, а не додаткові 4 нулі в шифрі літери «K».

Як можна бачити з Рис.14 після передачі коду, на кінці формується додатковий нуль на тому самому такті на якому transmit = 0, тому при дешифровці слід буде до всіх цих кодів додати 0 зліва (вважаємо, що старші зліва). Виняток становить лише літера «T», в яку 2 бітом (Рис.16) записується невизначений стан.

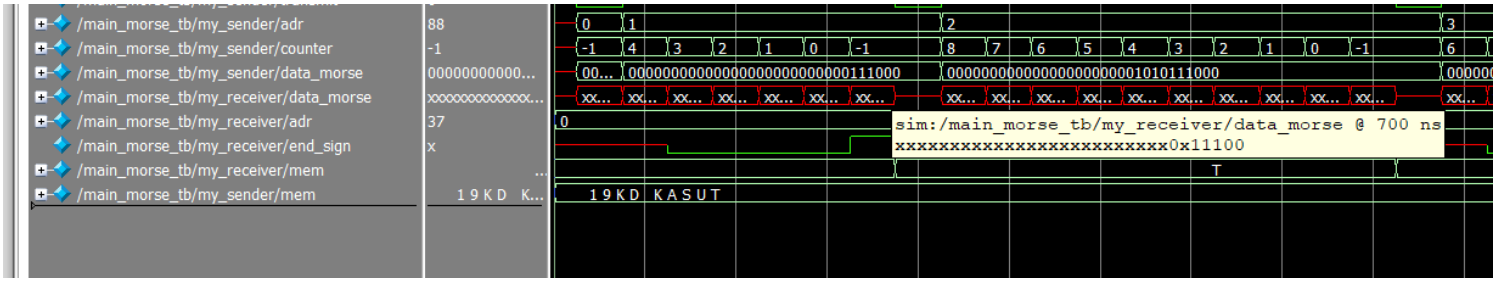

Рис.16 Запис в приймач літери «T».

Як видно з Рис.16 замість очікуваного 011100 ми отримуємо 0x11100, це спричинено знаходженням невизначеного стану на i_data_morse, що в свою чергу причинено знаходженням невизначеного стану на o_data_morse (Рис. 17).

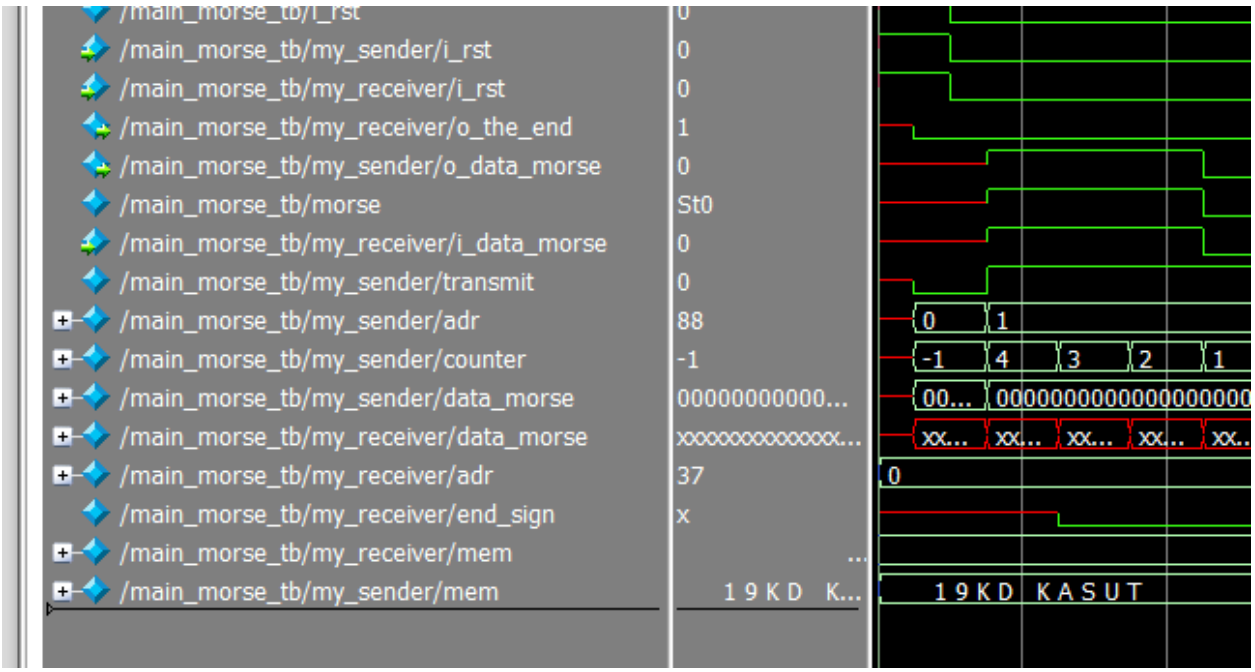

Рис.17 Сингали на початку передачі.

Саме через це слід буде записати в дешифраторі розшифровку для літери «T» так як на Рис.18.

```
'b xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_0x11_1000: mem[adr] = 'h 54;//T
```

Рис.18 Розшифровка в дешифраторі приймача для літери «T».

На цьому можна вважати опис та тестування передавача закінченим. Для приймача все робиться за аналогією, тому дуже детально зупинятись я не буду.

Код приймача, який описується в файлі receiver.v (передавач описується в файлі sender.v) наведений на Рис.19.

```
module receiver (
    i_clk,
    i_rst,
    i_data_morse,
    o_the_end
);

    input i_clk, i_rst, i_data_morse;
    output reg o_the_end;

    reg [7:0] mem[64:0];
    reg [31:0] data_morse;
    reg [7:0] adr = 0;
    reg end_sign;

    assign end_sign = ~|data_morse[1:0];

    always @(posedge i_clk) begin
        data_morse = {data_morse[30:0], i_data_morse};
        if(i_rst) begin
            adr = 0;
            o_the_end = 0;
            data_morse = 'b xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxx0;
        end else if(end_sign) begin
            case (data_morse)
                'b xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_0x11_1000: mem[adr] = 'h
54;//T
                'b xxxx_xxxx_xxxx_xxxx_xxxx_x010_1011_1000: mem[adr] = 'h
55;//U
                'b xxxx_xxxx_xxxx_xxxx_xxxx_xxx0_1010_1000: mem[adr] = 'h
53;//S
                'b xxxx_xxxx_xxxx_xxxx_xxxx_xxx0_1011_1000: mem[adr] = 'h
41;//A
                'b xxxx_xxxx_xxxx_xxxx_xxx0_1110_1011_1000: mem[adr] = 'h
4B;//K
                'b xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_0000: mem[adr] = 'h
20;//[space]
```

```
                'b xxxx_xxxx_xxxx_xxxx_xxxx_x011_1010_1000: mem[adr] = 'h
44;//D
                'b xxxx_xxxx_xxx0_1110_1110_1110_1110_1000: mem[adr] = 'h
39;//9
                'b xxxx_xxxx_xxx0_1011_1011_1011_1011_1000: begin
                    mem[adr] = 'h 31;//1
                   o_the_end = 1;
                end
            endcase
            adr = adr + 1;
            data_morse = 'hx;
        end
    end

endmodule
```

Рис.19 Код приймача, що описується в файлі receiver.v.

ПОЄДНАННЯ СКЛАДОВИХ ЧАСТИН ТА РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Як вже було зазначено вище, демонстрація правильної роботи приймача за умови правильної роботи передавача буде означати, що весь канал зв'язку працює правильно. Самостійно все можна перевірити знайшовши всі матеріали за посиланням LAMPA23/Morse-code (github.com).

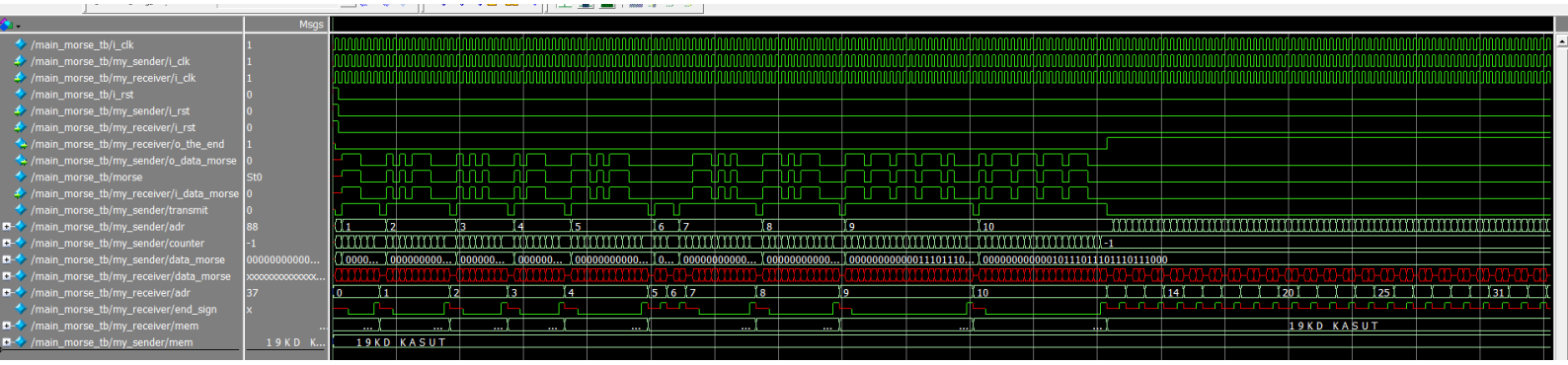

Рис.20 Часова діаграма сигналів кінцевого моделювання роботи системи.

Читати з пам'яті повідомлення слід починаючи з молодших байтів. Як можна побачити на Рис.20 було отримане правильне повідомлення.

Передавач і приймач були поєднанні в тому ж самому файлі в якому формуються вхідні сигнали скидання, синхронізації та обмеження часу симуляції. Це файл main_morse_tb.v, вміст якого наведений на Рис.21

```
`timescale 10ns/1ns
module main_morse_tb ();

    parameter t = 10;

    reg i_clk, i_rst;

    wire o_the_end;
    wire morse;

    sender my_sender(
        .i_clk(i_clk),
        .i_rst(i_rst),
        .o_data_morse(morse)
    );

    receiver my_receiver(
        .i_clk(i_clk),
        .i_rst(i_rst),
        .i_data_morse(morse),
        .o_the_end(o_the_end)
    );

    initial begin
```

```
        i_clk = 0;
        forever # ( t / 2 ) i_clk = ~i_clk;
    end



    initial begin
        i_rst = 1;
        # ( t ) i_rst = 0;
    end

    initial begin
        # ( t * 200 ) $stop;
    end

endmodule
```

Рис.21 Вміст файлу main_morse_tb.v.

Для запуску симуляції створив файл sim.do вміст якого наведений на Рис.22.

```
add wave sim:/main_morse_tb/i_clk
add wave -radix binary sim:/main_morse_tb/my_sender/i_clk
add wave -radix binary sim:/main_morse_tb/my_receiver/i_clk


add wave sim:/main_morse_tb/i_rst
add wave -radix binary sim:/main_morse_tb/my_sender/i_rst
add wave -radix binary sim:/main_morse_tb/my_receiver/i_rst
add wave -radix binary sim:/main_morse_tb/my_receiver/o_the_end




add wave -radix binary sim:/main_morse_tb/my_sender/o_data_morse
add wave sim:/main_morse_tb/morse
add wave -radix binary sim:/main_morse_tb/my_receiver/i_data_morse

add wave -radix binary sim:/main_morse_tb/my_sender/transmit
add wave -radix unsigned sim:/main_morse_tb/my_sender/adr
add wave -radix decimal sim:/main_morse_tb/my_sender/counter
add wave -radix binary sim:/main_morse_tb/my_sender/data_morse
add wave -radix binary sim:/main_morse_tb/my_receiver/data_morse
add wave -radix unsigned sim:/main_morse_tb/my_receiver/adr

add wave -radix binary sim:/main_morse_tb/my_receiver/end_sign

add wave -radix ASCII sim:/main_morse_tb/my_receiver/mem
add wave -radix ASCII sim:/main_morse_tb/my_sender/mem
```

Рис.22 Вміст файлу sim.do.

ВИСНОВКИ

В пункті ФОРМУЛЮВАННЯ ЗАВДАННЯ я достатньо детально описав все те, що намагався зробити. Думаю можна сказати, що цей план мінімум був виконаний. Звичайно даний пристрій створений для передачі наперед визначених даних. Сильним спрощенням для проектування також є можливість надавати їм один сигнал синхронізації та скидання. Цей пристрій не можна назвати основою для створення рольного пристрою, ще і через те що не реалізовані деякі необхідні речі, накшталт блоку керування для користувача чи навіть той самий DFT (design for test), але це і не було завданням даного курсового проекту.