ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

HỆ THỐNG SỐ

Đề tài

Xây dựng hệ thống phát hiện chuỗi

GVHD: Trần Ngọc Thịnh
Lê Tấn Long
SV: Nguyễn Trần Quang Minh - 1811083
Võ Ngọc Quý - 51303307
Nguyễn Duy Sơn - 1811197
Mai Đình Phúc - 1811149
Nguyễn Đức Lộc - 1811064
Nguyễn Ngọc Lan Anh - 1810812

TP. HỒ CHÍ MINH, THÁNG 5/2019

Mục lục

1 Giới thiệu

1.1 Về hệ thống phát hiện chuỗi

Hệ thống phát hiện chuỗi, như tên gọi của nó, có tác dụng phát hiện vị trĩ của một chuỗi con nằm trong một đoạn văn bản. Thông tin này qua đó có thể được dùng cho nhiều mục đích, chủ yếu trong việc xử lý tín hiệu số.

Hệ thống này có nhiều ứng dụng trong các lĩnh vực như y học, kinh tế, tài chính, chứng khoán, quản lý mạng truyền thông,v.v

1.2 Về thiết bị sử dụng (Board FPGA De2i)

Board De2i-150 là một nền tảng nhúng kết hợp giữa bộ xử lý Intel N2600 và Cyclone IV GX FPGA của hãng Altera.

Ưu điểm của board này là sự linh hoạt trong việc cấu hình thiết bị, với tốc độ cao mà vẫn giữ giá thành và nguồn năng lượng tiêu thụ ở mức thấp, cùng với khả năng tái cấu trúc phần cứng. Nhờ những ưu điểm này mà nó có thể đáp ứng mọi nhiệm vụ, ttrowr thành một công cụ tuyệt vời để mô phỏng và thiết kế phần cứng.

1.3 Về các chức năng của sản phẩm

Hệ thống phát hiện chuỗi của nhóm 5 xây dựng và mô phỏng trên board De2i-150 bao gồm các chức nay như đã được yêu cầu:

1. Phát hiện các chuỗi 4 bit từ một văn bản do người dùng nhập vào, mỗi lượt 4 bit.
2. Xoá chuỗi 4 bit vừa nhập và thay bằng chuỗi khác.
3. Hiển thị vị trí của chuỗi được tìm thầy trên LED 7 đoạn
4. Hiển thị số lượng chuỗi phát hiện được

1.4 Mô tả input/output:

1.4.1 Input

Các switch SW[3] đén SW[0] được dùng để nhập lần lượt 4 bit của văn bản cần kiểm tra vào
Các switch SW[8] đến SW[5] được dùng để nhập chuỗi pattern

Switch SW[4] dùng để xác nhận kết thúc đoạn văn bản và in số chuỗi tìm thấy ra LED 7 đoạn.

Ấn giữ KEY[2] và ấn KEY[3] để xác nhận đưa 4 bit vào đoạn văn bản.

Ấn KEY[3] để duyệt lần lượt 4 bit qua module kiểm tra.

1.4.2 Output

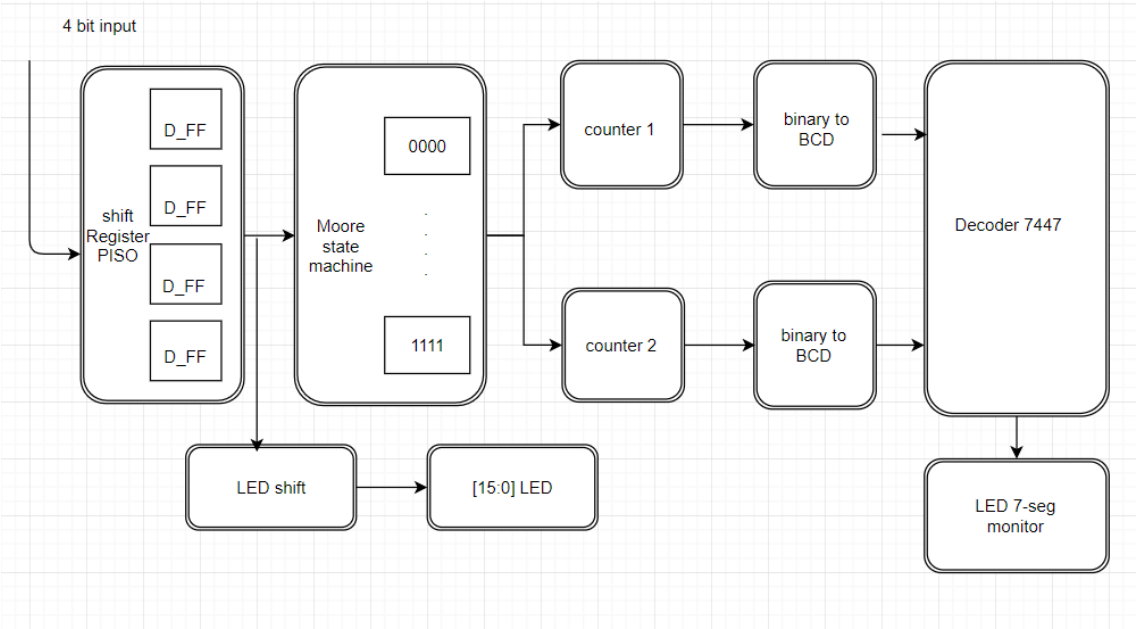
18 LED đỏ LEDR[17] đến LEDR[0] hiển thị một phần của đoạn văn bản được người dùng nhập vào (18 ký tự gần nhất).

4 LED xanh lá LEDG[3] đến LEDG[0] thể hiện trạng thái của KEY[3] đến KEY[0].

LED 7 đoạn HEX7 và HEX6 hiển thị số lượng chuỗi phát hiện được khi kết thúc văn bản.

LED 7 đoạn HEX5 và HEX4 hiển thị vị trí cảu chuỗi mẫu khi được tìm thấy.

2 Thiết kế



Hình trên mô tả sơ đồ khối của mạch.

2.1 Shift register PISO

Khối này có chức năng là 1 thanh ghi dịch bao gồm 4 khối D Flip flop để 4 bit chứa dữ liệu nhập vào từ các switch 3 đến 0.

Đồng thời khối này sẽ đẩy từng bit vào bộ xử lý chuỗi để phát hiện vị trí chuỗi mẫu.

2.2 Moore state machine

Kiểm tra dữ liệu chuyển từ thanh ghi dịch có phù hợp với chuỗi mẫu hay không.

2.3 counter1

Đếm vị trí của chuỗi con trong văn bản.

2.4 counter2

Đếm số chuỗi tìm được

2.5 binary to BCD

. Chuyển từ nhị phân sang BCD

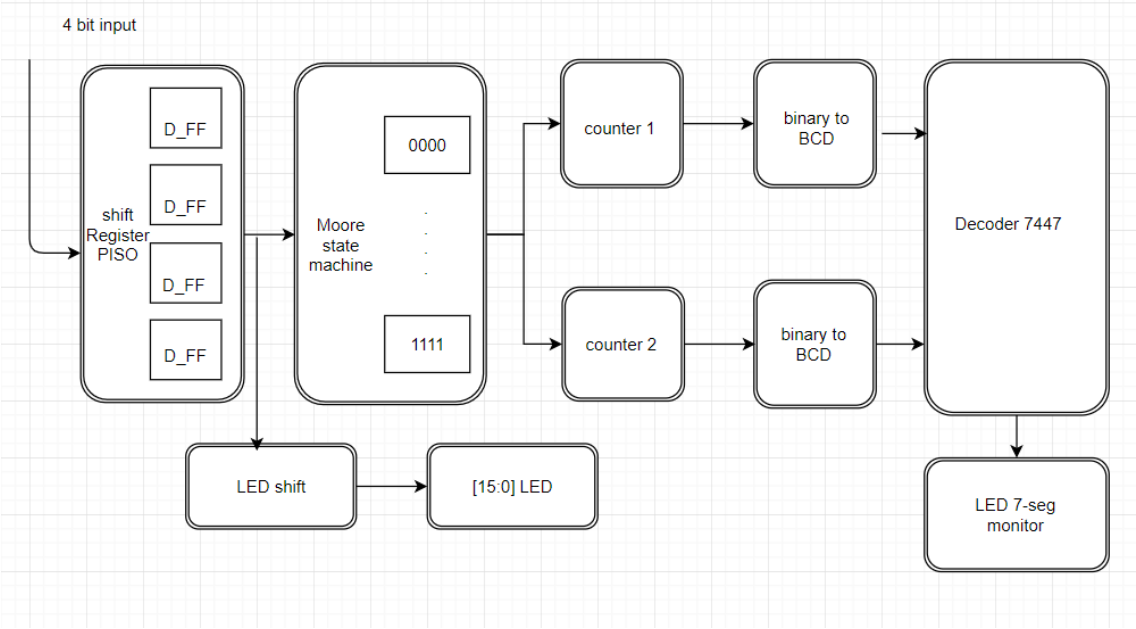
2.6 Decode 7447

Giải mã để hiển thị ra màn hình Led 7 đoạn.

2.7 LEDshift

Hiển thị 18 ký tự mới nhất của đoạn văn bản trên 18 LED đỏ của board.

3 Hiện thực



Hình trên mô tả sơ đồ khối của mạch.

3.1 Shift register PISO

Dưới đây là đoạn code Verilog mô tả thanh ghi dịch và cũng như các D Flip-Flop viết theo mô hình hành vi.

Tín hiệu so là tín hiệu đầu ra chứa giá trị của các bit trong đoạn văn bản.
B là đầu vào từ các SW[3:0]
CLK là clock cho việc dịch/load thanh ghi.

3.2 Moore state machine

Thiết kế bao gồm 16 máy trạng thái tương ứng với16 chuỗi 4 bit với cùng dữ kiệu vào và chọn 1 trong 16 dữ liệu ra.

```
1  module shiftRegPISO(so,B,SnL,CLK);
2     output so;
3     input [3:0]B;
4     input SnL;
5     input CLK;
6     wire n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13;
7
8     D_FF DFF0(CLK,B[0],n4);
9     and and0(n5,n4,SnL);
10    and and1(n1,B[1],~SnL);
11    or or0(n7,n5,n1);
12    D_FF DFF1(CLK,n7,n8);
13    and and4(n9,n8,SnL);
14    and and2(n2,B[2],~SnL);
15    or or1(n10,n2,n9);
16    D_FF DFF2(CLK,n10,n11);
17    and and5(n12,n11,SnL);
18    and and3(n3,B[3],~SnL);
19    or or2(n13,n12,n3);
20    D_FF DFF3(CLK,n13,so);
21
22 endmodule
```

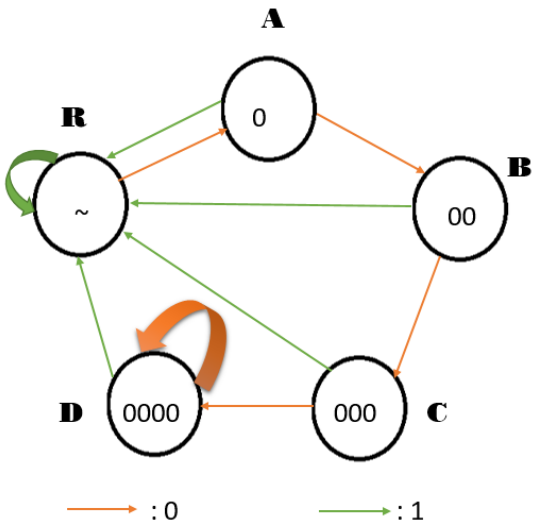
```
1  module D_FF(clock, d, q);
2  input clock, d;
3  wire clock, d;
4  output q;
5  reg q;
6  always @ (posedge clock)
7  q <= d;
8  endmodule
```

Dưới đây là đoạn code Verilog.

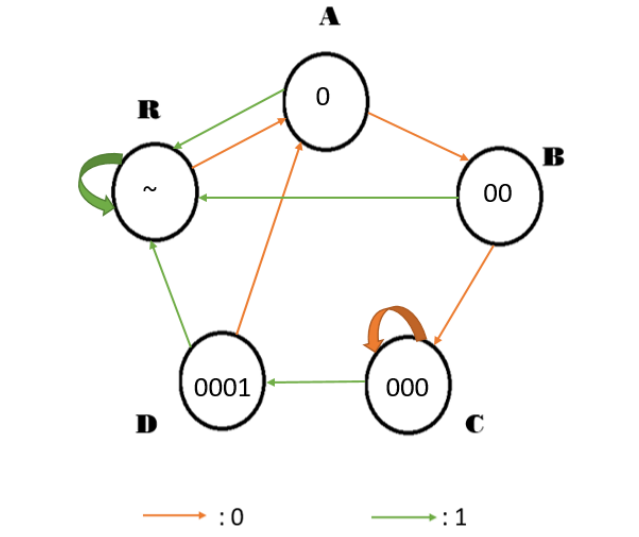
```
module onefrom16(seq_in,A,clock,reset,seq_out);
   output seq_out;
   reg seq_out;
   input [3:0] A, seq_in, clock, reset;
   wire [15:0]temp;
      moore0000 m1(seq_in,clock,reset,temp[0]);
      moore0001 m2(seq_in,clock,reset,temp[1]);
      moore0010 m3(seq_in,clock,reset,temp[2]);
      moore0011 m4(seq_in,clock,reset,temp[3]);
      moore0100 m5(seq_in,clock,reset,temp[4]);
      moore0101 m6(seq_in,clock,reset,temp[5]);
      moore0110 m7(seq_in,clock,reset,temp[6]);
      moore0111 m8(seq_in,clock,reset,temp[7]);
      moore1000 m9(seq_in,clock,reset,temp[8]);
      moore1001 m10(seq_in,clock,reset,temp[9]);
      moore1010 m11(seq_in,clock,reset,temp[10]);
      moore1011 m12(seq_in,clock,reset,temp[11]);
      moore1100 m13(seq_in,clock,reset,temp[12]);
      moore1101 m14(seq_in,clock,reset,temp[13]);
      moore1110 m15(seq_in,clock,reset,temp[14]);
      moore1111 m16(seq_in,clock,reset,temp[15]);
   always@(*)
   case (A[3:0])
      4'b0000:seq_out<=temp[0];
      4'b0001:seq_out<=temp[1];
      4'b0010:seq_out<=temp[2];
      4'b0011:seq_out<=temp[3];
      4'b0100:seq_out<=temp[4];
      4'b0101:seq_out<=temp[5];
      4'b0110:seq_out<=temp[6];
      4'b0111:seq_out<=temp[7];
      4'b1000:seq_out<=temp[8];
      4'b1001:seq_out<=temp[9];
      4'b1010:seq_out<=temp[10];
      4'b1011:seq_out<=temp[11];
      4'b1100:seq_out<=temp[12];
      4'b1101:seq_out<=temp[13];
      4'b1110:seq_out<=temp[14];
      4'b1111:seq_out<=temp[15];
   endcase

endmodule
```

Các module được thiết kế với sơ đồ trạng thái như sau:

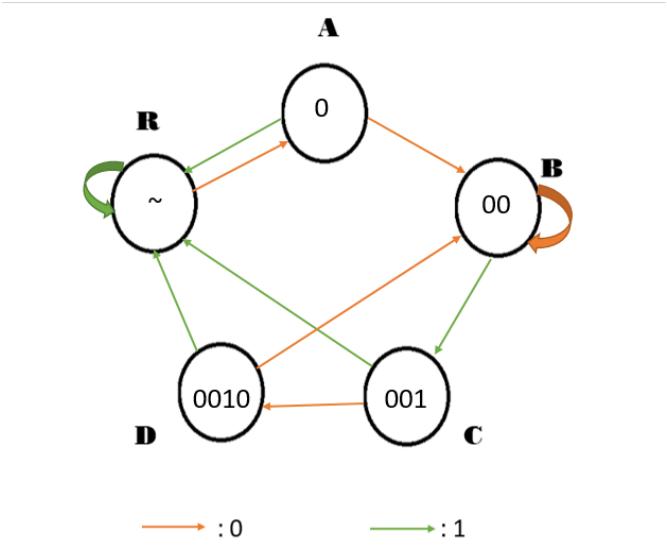
Sơ đồ trạng thái của chuỗi 0000



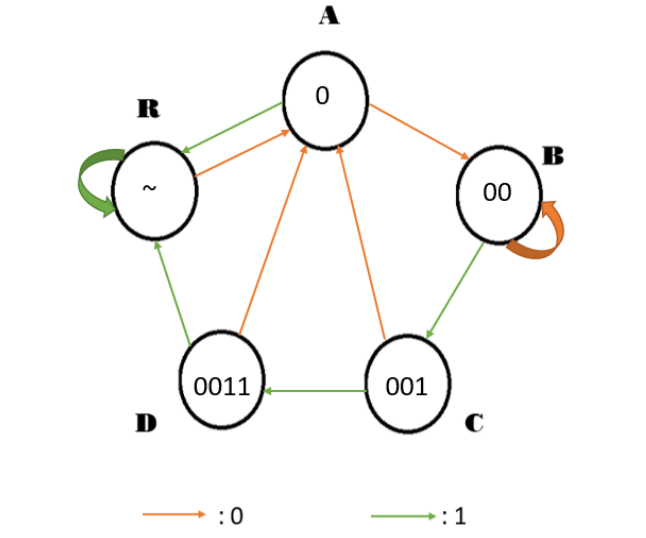
Sơ đồ trạng thái của chuỗi 0001



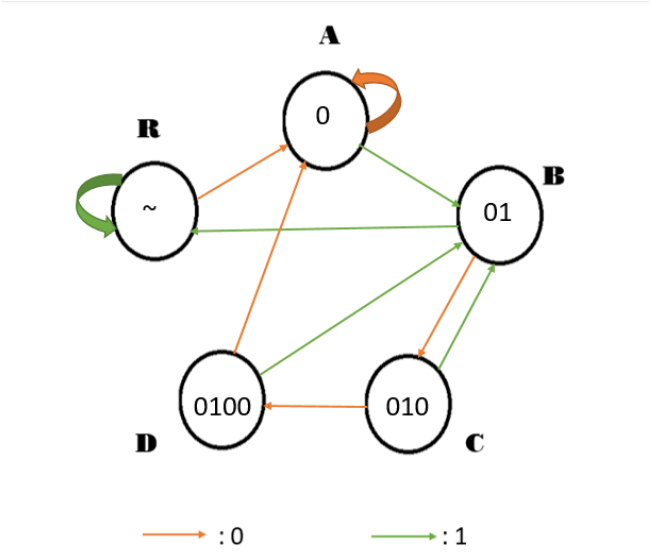
Sơ đồ trạng thái của chuỗi 0010



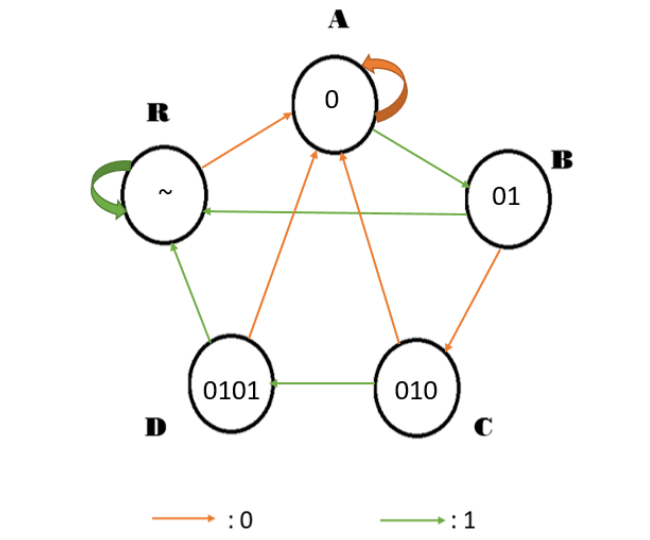
Sơ đồ trạng thái của chuỗi 0011



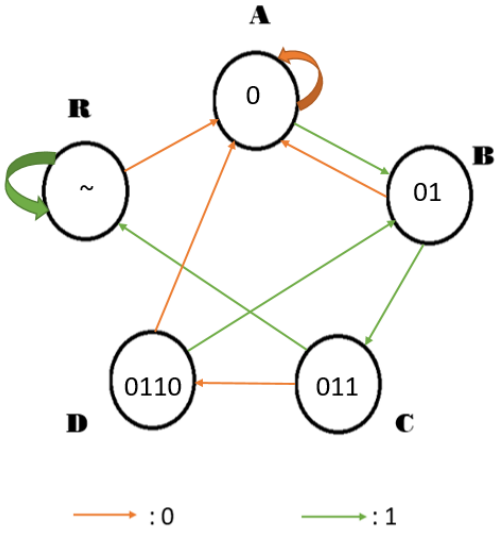
Sơ đồ trạng thái của chuỗi 0100



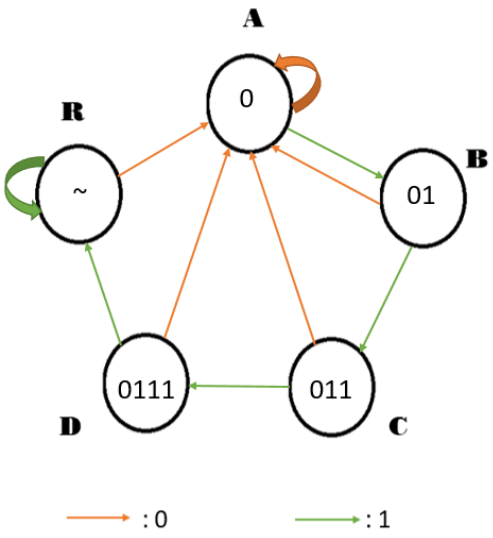
Sơ đồ trạng thái của chuỗi 0101



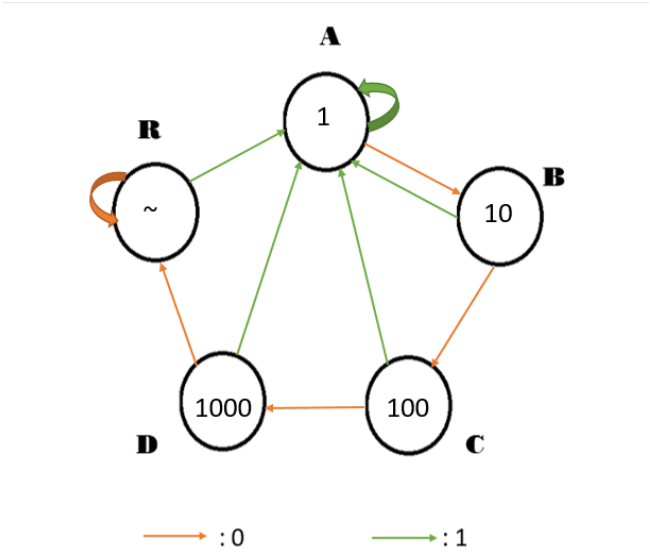
Sơ đồ trạng thái của chuỗi 0110



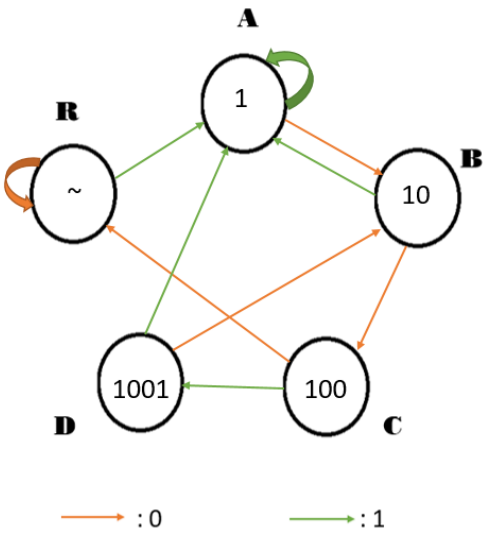
Sơ đồ trạng thái của chuỗi 0111



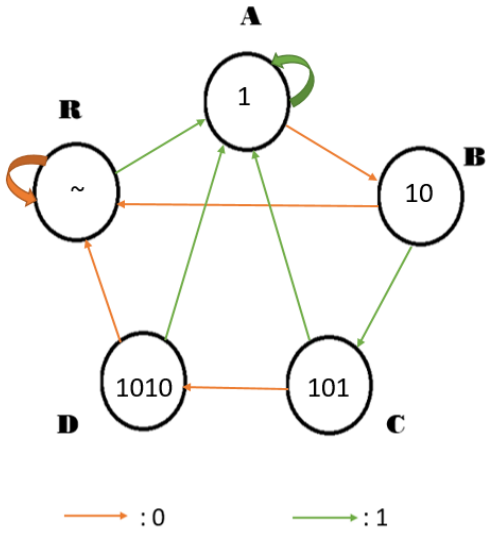
Sơ đồ trạng thái của chuỗi 1000



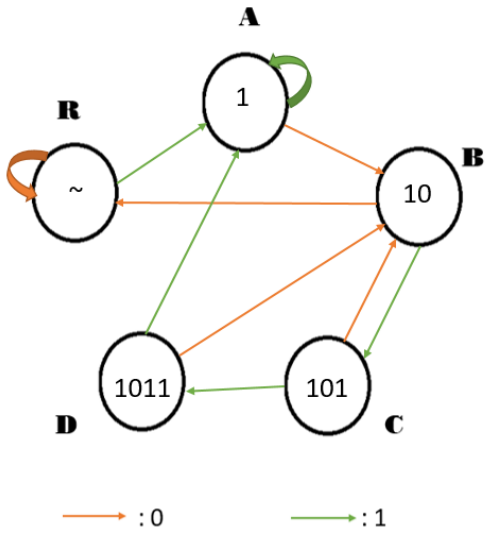
Sơ đồ trạng thái của chuỗi 1001



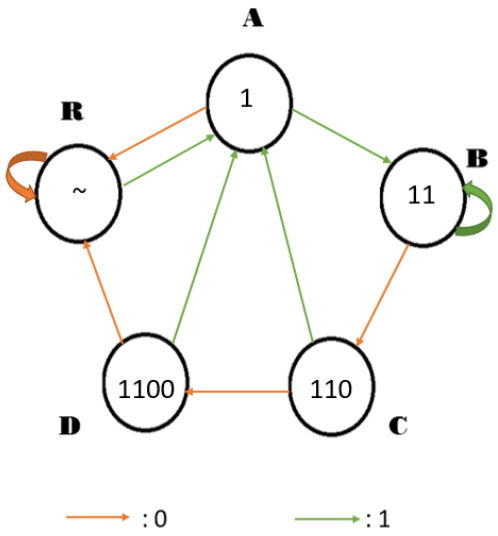
Sơ đồ trạng thái của chuỗi 1010



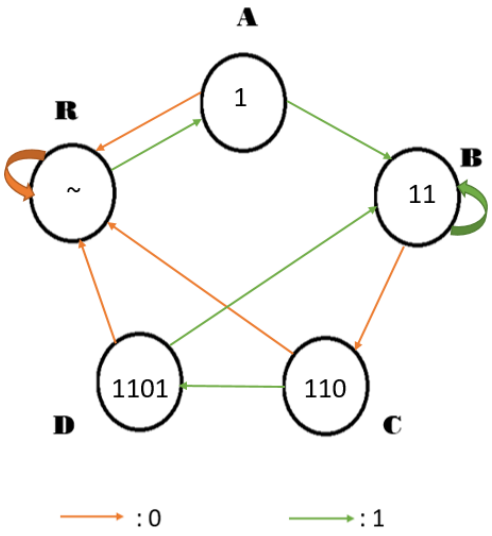
Sơ đồ trạng thái của chuỗi 1011



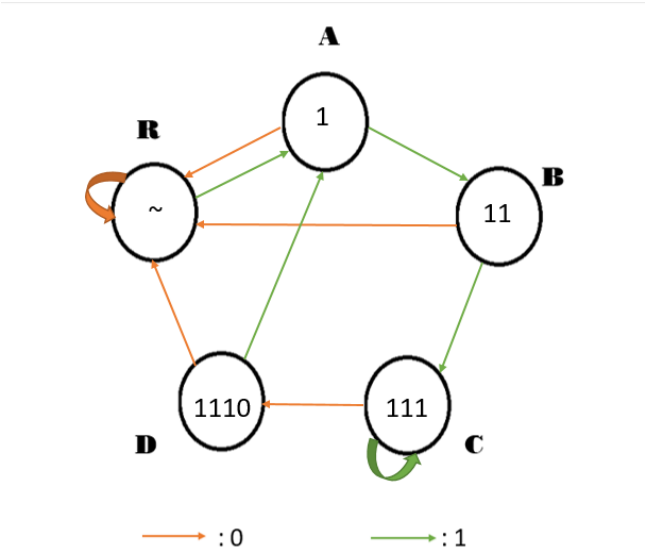
Sơ đồ trạng thái của chuỗi 1100



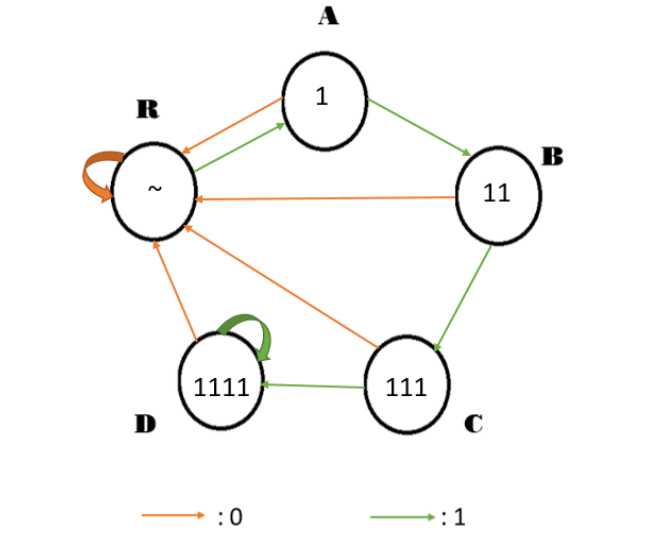
Sơ đồ trạng thái của chuỗi 1101



Sơ đồ trạng thái của chuỗi 1110



Sơ đồ trạng thái của chuỗi 1111



Dưới đây là ví dụ mẫu code Verilog bằng mô hình hành vi.[2]

```
1  module moore0000(seq_in,clock,reset,seq_out);
2
3     input clock;
4     input reset;
5     input seq_in;
6     output reg seq_out;
7     parameter  R=0,
8                A=1,
9                B=2,
10               C=3,
11               D=4;
12    reg [2:0] current_state, next_state;
13    always @(posedge clock, posedge reset)
14       begin
15          if(reset==1)
16             current_state <= R;
17          else
18             current_state <= next_state;
19       end
20
21    always @(current_state,seq_in)
22       begin
23          case(current_state)
24             R:begin
25                if(seq_in==1)
26                   next_state <= R;
27                else
28                   next_state <= A;
29             end
30             A:begin
31                if(seq_in==0)
32                   next_state <= B;
33                else
34                next_state <= R;
35             end
36             B:begin
37                if(seq_in==0)
38                   next_state <= C;
39                else
40                   next_state <= R;
41             end
42             C:begin
43                if(seq_in==0)
44                   next_state <= D;
45                else
46                   next_state <= R;
47             end
48             D:begin
49                if(seq_in==0)
50                   next_state <= D;
51                else
52                   next_state <= R;
53             end
54             default:next_state <= R;
55          endcase
56       end
57
58    always @(current_state)
59       begin
60          case(current_state)
61             R:seq_out <= 0;
62             A:seq_out <= 0;
63             B:seq_out <= 0;
64             C:seq_out <= 0;
65             D:seq_out <= 1;
66             default:  seq_out <= 0;
67          endcase
68       end
69 endmodule
```

3.3 counter 1

Dưới đây là đoạn code verilog, được viết theo mô hình hành vi

```
module counter(input clk, reset, pl, hout, output reg[6:0] counter);
reg [6:0] counter_up;
reg [6:0] preload1;
reg [6:0] preload2;
reg [6:0] preload3;
reg [6:0] preload4;
reg [6:0] preload5;

always @(posedge clk or posedge reset or posedge pl)
   begin
      if (pl) counter_up <= preload2;
      else if (reset)
         counter_up <= 7'd0;
      else if (clk)
         begin
            counter_up <= counter_up + 7'd1;
            preload2 =preload3;
            preload3 =preload4;
            preload4 =preload5;
            preload5 =counter_up;
         end
   end
always @(hout or (reset))
   if (reset) counter<=7'd0;
   else if (hout)
   counter <= counter_up-7'd3;
endmodule
```

3.4 counter 2

Dưới đây là đoạn code verilog, được viết theo mô hình hành vi

```
module counter1(input clk,cp, reset, pl,hout, output reg[6:0] counter);
reg [6:0] counter_up;
reg [6:0] preload2;
reg [6:0] preload3;
reg [6:0] preload4;
reg [6:0] preload5;

always @(posedge clk or posedge reset or posedge pl)
   begin
      if (pl) counter_up = preload2;
      else if (reset)
         counter_up = 7'd0;
      else if(cp)
         begin
            counter_up = counter_up + 7'd1;
         end
      else if (clk)
      begin
         preload2 =preload3;
         preload3 =preload4;
         preload4 =preload5;
         preload5 =counter_up;
      end
   end
always @(hout or reset)
if (hout)
counter = counter_up;
else counter=7'd0;
endmodule
```

3.5 bin to BCD

Chuyển dữ liệu đầu vào là số nhị phân 7 bit sang mã BCD bằng giải thuật DOuble Dabble, tức là dịch dần 7 bit sang trái và so sánh các cột 4 bit với 5. Nếu lớn hơn 5 thì cộng thêm ba vào cho đến khi dịch hết 7 bit.

Đoạn code sau được viết theo mô hình hành vi mô phỏng giải thuật trên.[3]

```
1  module bintoBCD7b(output reg [3:0] tens,
2                    output reg [3:0] ones,
3                    input [6:0] bin);
4   integer i;
5   always @(bin)
6      begin
7         tens=4'd0;
8         ones=4'd0;
9         for (i=6;i>=0;i=i-1)
10        begin
11           if (tens>=5) tens=tens+3;
12           if (ones>=5) ones=ones+3;
13           tens=tens<<1;
14           tens[0]=ones[3];
15           ones=ones<<1;
16           ones[0]=bin[i];
17        end
18     end
19  endmodule
20
```

3.6 Decode 7447

Mô phỏng lại hoạt động của ic 7447, tức là giải mã soos4 bit để đưa vào LED 7 đoạn
Đoạn code sau được viết theo mô hình RTL

```
1   module decode7447(b,A);
2      output [6:0] b;
3      input [3:0] A;
4      assign b[0]=(~A[3]&~A[2]&~A[1]&A[0])|(A[2]&~A[1]&~A[0]);
5      assign b[1]=(A[2]&~A[1]&A[0])|(A[2]&A[1]&~A[0]);
6      assign b[2]=~A[2]&A[1]&~A[0];
7      assign b[3]=(~A[2]&~A[1]&A[0])|(A[2]&~A[1]&~A[0])|(A[2]&A[1]&A[0]);
8      assign b[4]=(A[2]&~A[1])|(~A[3]&A[0])|(~A[2]&A[0]);
9      assign b[5]=(~A[3]&~A[2]&A[0])|(~A[3]&A[1]&A[0])|(~A[2]&A[1]);
10     assign b[6]=(~A[3]&~A[2]&~A[1])|(A[2]&A[1]&A[0]);
11  endmodule
```

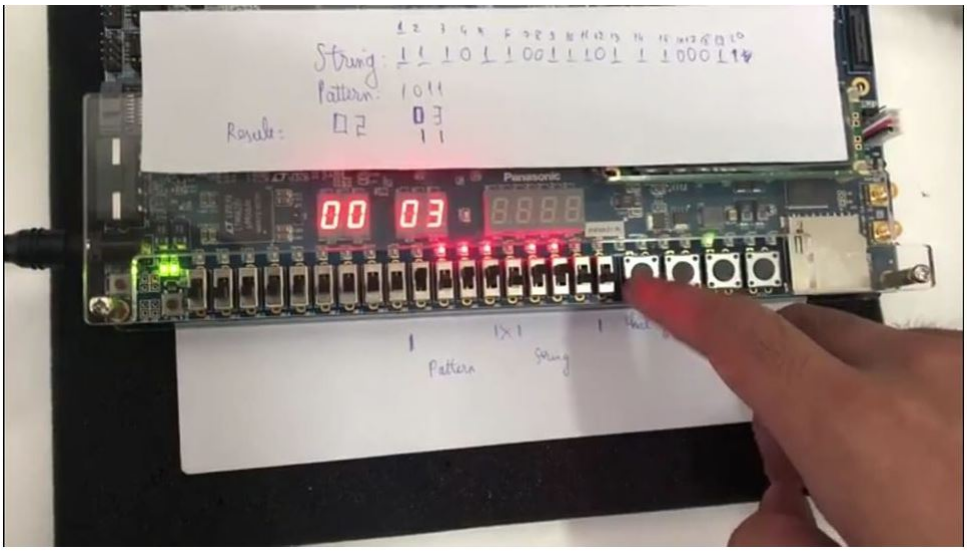
3.7 LEDshift

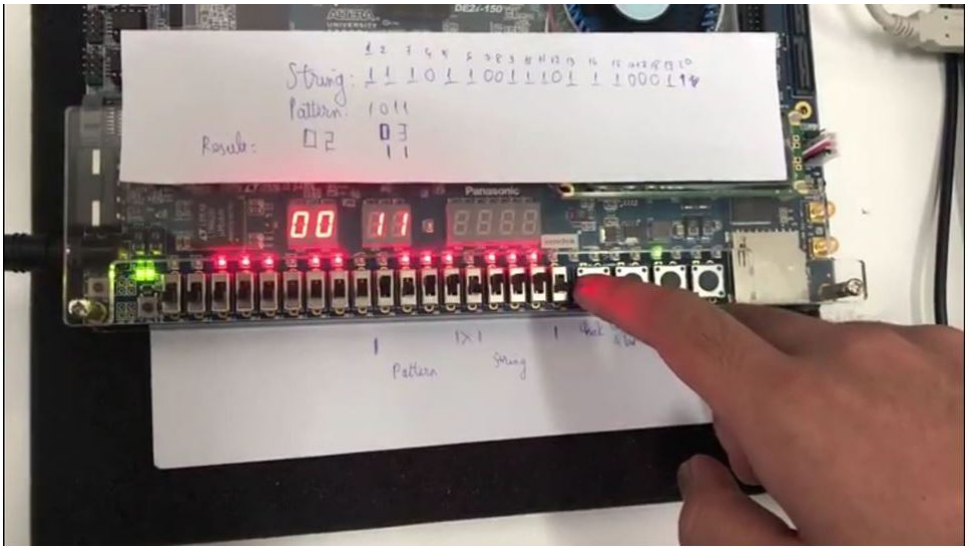
Điều khiển hiển thị cac LED đỏ biểu diễn 18 ký tự gần nhất của văn bản

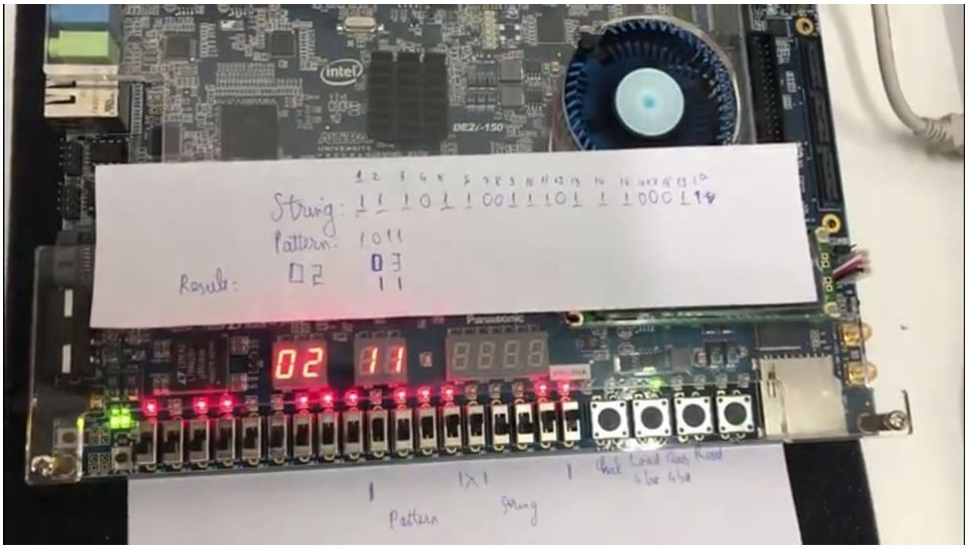
```
1  module LEDshift(A,n,clk1,clk2,reset);
2     output [17:0] A;
3     reg [17:0]A;
4     input n;
5     input clk1;
6     input clk2;
7     input reset;
8     reg [17:0] preload1;
9     reg [17:0] preload2;
10    reg [17:0] preload3;
11    reg [17:0] preload4;
12    reg [17:0] preload5;
13
14    always @ (posedge clk1 or posedge clk2 or posedge reset)
15       if (reset) A[17:0]<=18'b0;
16       else if (clk2) A[17:0]=preload2[17:0];
17       else if(clk1)
18       begin
19          A[17:1]<=A[16:0];
20          A[0]<=n;
21          preload2 =preload3;
22          preload3 =preload4;
23          preload4 =preload5;
24          preload5 =A[17:0];
25       end
26 endmodule
```

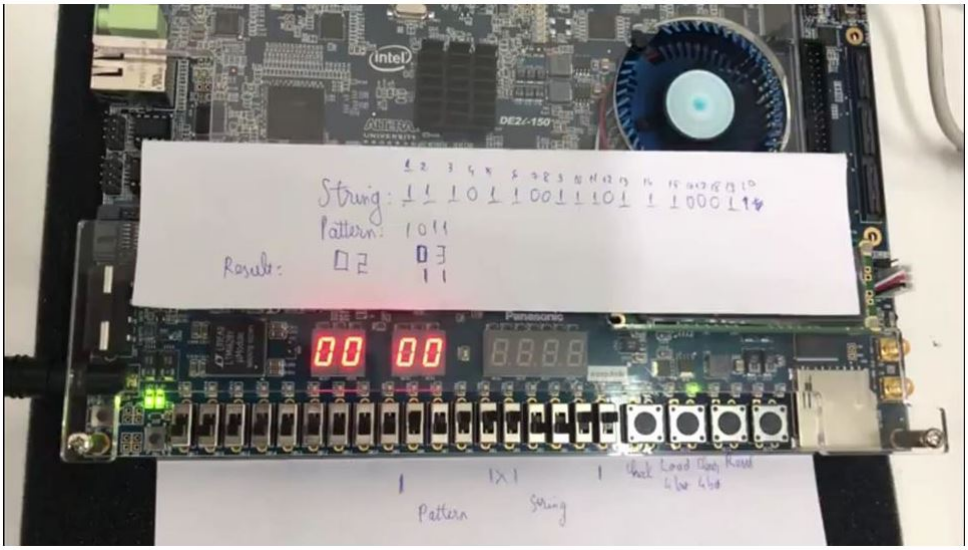
4 Kiểm thử

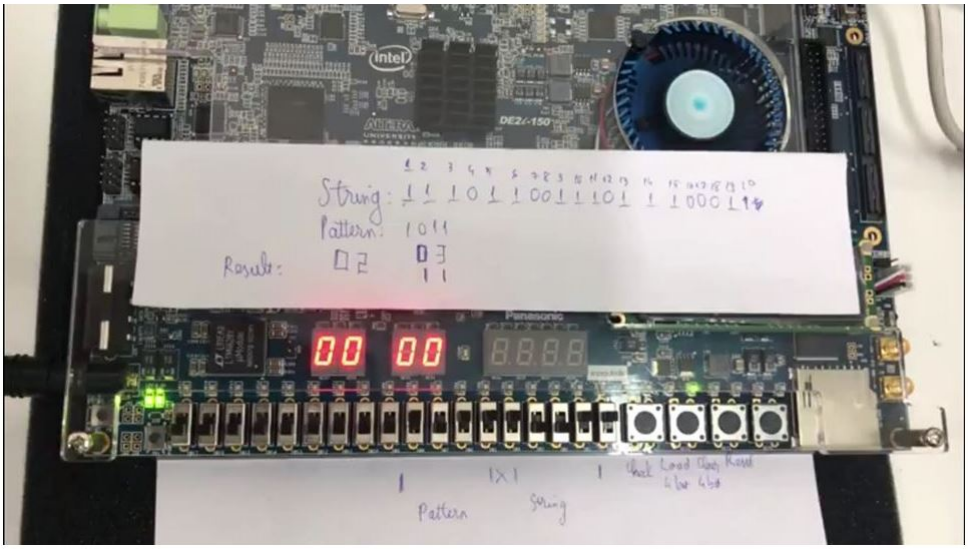
Kiểm thử hệ thống trên board De2i-150:

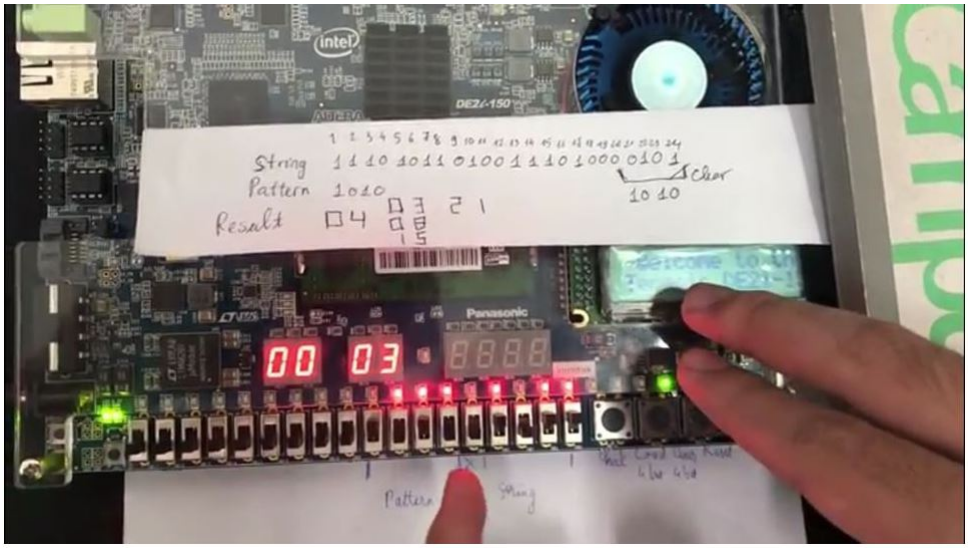


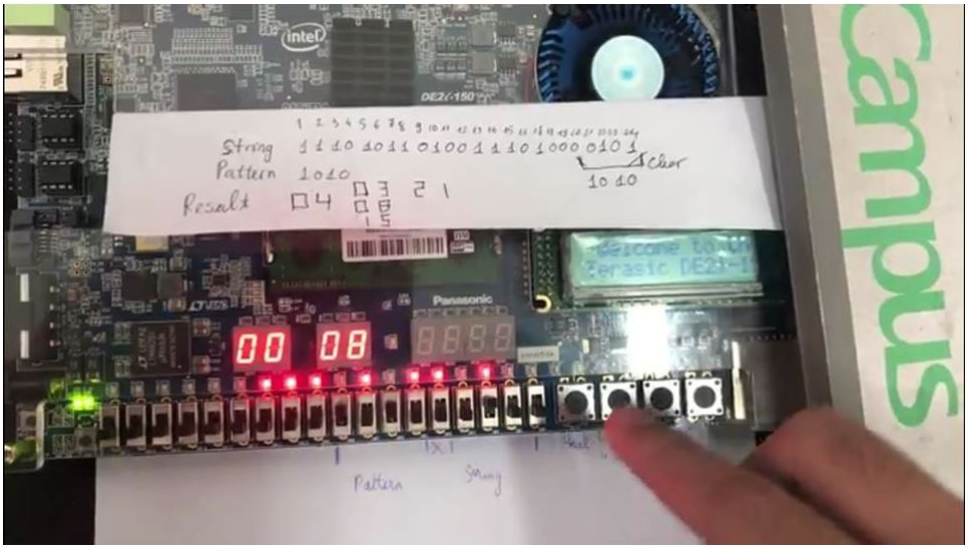

String:
Pattern: 1011
Result:
00
11
Pattern
String

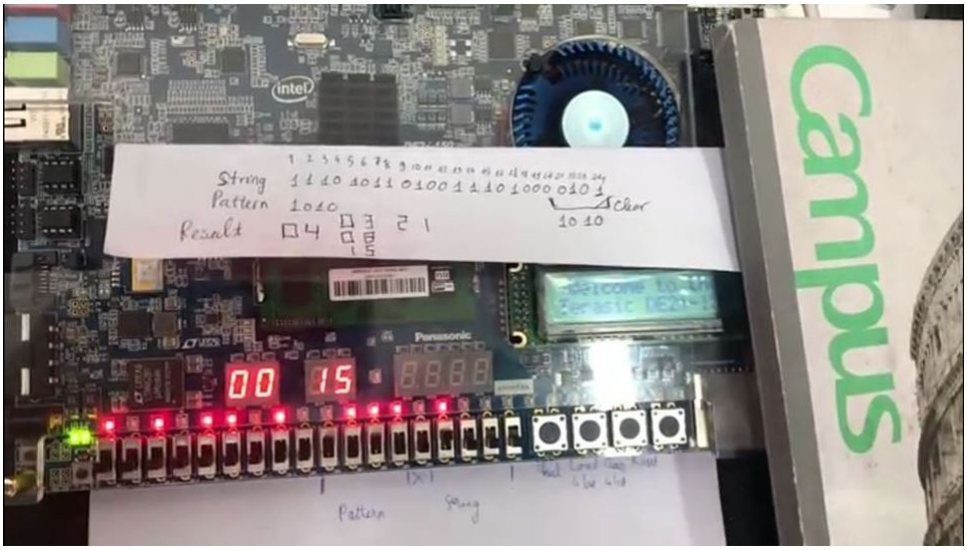

String:
Pattern: 1011
Result:
02
11
Pattern
String

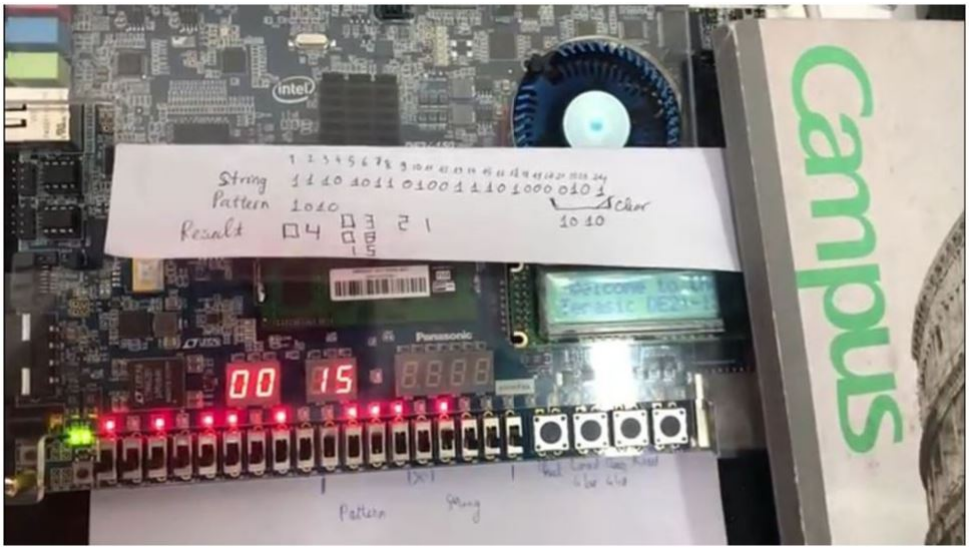

String:
Pattern: 1011
Result:
Pattern
String


String:
Pattern: 1011
Result:
Pattern
String


String
Pattern
Result
Char
00 03
Panasonic
Pattern


String
Pattern
Result
Char
00 08
Panasonic
Pattern





5 Kết luận

Hệ thống phát hiện chuỗi có rất nhiều ứng dụng trong kỹ thuật xử lý tín hiệu. Trên đây nhóm 5 đã trình bày Các bước xây dựng một hệ thông phát hiện chuỗi đơn giản mà nhóm đã tìm hiểu được, đồng thời mô phognr nó trên Board Altera De2i-150.

Qua đó, nhóm đã gặp phải một số khó khăn, chủ yếu ở việc lưu các trạng thái qua mỗi clock để có thể quay lại khi chuỗi bị xoá. Về vấn đề này, nhóm vẫn chưa thể hiện thực hoàn hảo được.

Trong tương lai, nếu có cơ hội, nhóm 5 sẽ cùng nhau thảo luận và đưa ra các giải pháp cải tiến hệ thống, về giao diện với người dùng cũng như các tính năng như xoá các chuỗi con ra khỏi đoạn văn bản hoặc thay đổi cỡ chuỗi mẫu (pattern). Trên hết cả, nhóm sẽ cố gắng hiện thực hoá

hệ thống này, không phải trên Board Altera De2i-150 mà là một hệ thống thực sự với đầy đủ phần cứng

6 Bảng phân công công việc

Nguyễn Trần Quang Minh	Thiết kế các module Moore state machine và LEDshift
Nguyễn Duy Sơn	Thiết kế các module counter1, counter2, bin to BCD, Decode 7447
Võ Ngọc Quý	Thiết kế Shift register PISO và viết báo cáo
Mai Đình Phúc	Hỗ trợ viết báo cáo (Hình ảnh minh hoạ)
Nguyễn Đức Lộc	Slide thuyết trình
Nguyễn Ngọc Lan Anh	Slide thuyết trình

Tài liệu

[1] Tocci, R.J. (2007).*Digital systems: principles and applications*. Prentice Hall PTR, 1988.

[2] FPGA for Students

[3] Wikipedia contributors. (2019, May 12). Double dabble. In Wikipedia, The Free Encyclopedia