Final project
2nd milestone report

Team names
Shehab Bahaa Naga
201700313
Section: 1
instructor: Dr. Amr Helmy

seg7.sv

```
1  module sevenseg(
2                  input logic [3:0] data,
3                  output logic [6:0] segements
4                  );
5
6  always_comb
7  case(data)
8  //                          abc_defg
9  0:        segements = 7'b111_1110 ;
10 1:        segements = 7'b011_0000 ;
11 2:        segements = 7'b110_1100 ;
12 3:        segements = 7'b111_1001 ;
13 4:        segements = 7'b011_0011 ;
14 5:        segements = 7'b101_1011 ;
15 6:        segements = 7'b101_1111 ;
16 7:        segements = 7'b111_0000 ;
17 8:        segements = 7'b111_1111 ;
18 9:        segements = 7'b111_0011 ;
19 10:       segements = 7'b111_0111 ;
20 11:       segements = 7'b001_1111 ;
21 12:       segements = 7'b000_1101 ;
22 13:       segements = 7'b011_1101 ;
23 14:       segements = 7'b100_1111 ;
24 15:       segements = 7'b100_0111 ;
25 default: segements = 7'b000_0000 ;
26 endcase
27 endmodule
28
```

project.tb

```
// the idea of the whole structure that will be under testing
/*module project(input logic clk,
               input logic reset,
               output logic [6:0] segments
               );
logic [63:0] x;
logic [3:0] data;

graycode g_raycode(x[63:0]);
counter  C_ounter(clk, reset, x[63:0], data[3:0]);
sevenseg s_evenseg(data[3:0], segments[6:0]);
endmodule*/

module testbench3();
logic clk;
logic reset;
logic [63:0] x;
logic [3:0] data;
logic [6:0] segments;

// instance
graycode g_raycode(x[63:0]);
counter  C_ounter(clk, reset, x[63:0], data[3:0]);
sevenseg s_evenseg(data[3:0], segments[6:0]);

always
begin
clk=1; #50;
clk=~clk; #50;
end

initial
begin
reset=1; #25;
reset=0;
end

always
begin
#10;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h0 & segments[6:0] == 7'b111_1110 ) $display ("SUCCESS");  else $error("0 failed");
#65;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h0 & segments[6:0] == 7'b111_1110 ) $display ("SUCCESS"); else $error("0 failed");
#50;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h1 & segments[6:0] == 7'b011_0000) $display ("SUCCESS"); else $error("1 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h3 & segments[6:0] == 7'b111_1001) $display ("SUCCESS"); else $error("3 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h2 & segments[6:0] == 7'b110_1100) $display ("SUCCESS"); else $error("2 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h6 & segments[6:0] == 7'b101_1111) $display ("SUCCESS"); else $error("6 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h7 & segments[6:0] == 7'b111_0000) $display ("SUCCESS"); else $error("7 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h5 & segments[6:0] == 7'b101_1011) $display ("SUCCESS"); else $error("5 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h4 & segments[6:0] == 7'b011_0011) $display ("SUCCESS"); else $error("4 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'hc & segments[6:0] == 7'b000_1101) $display ("SUCCESS"); else $error("c failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'hd & segments[6:0] == 7'b011_1101) $display ("SUCCESS"); else $error("d failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'hf & segments[6:0] == 7'b100_0111) $display ("SUCCESS"); else $error("f failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'he & segments[6:0] == 7'b100_1111) $display ("SUCCESS"); else $error("e failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'ha & segments[6:0] == 7'b111_0111) $display ("SUCCESS"); else $error("a failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'hb & segments[6:0] == 7'b001_1111) $display ("SUCCESS"); else $error("b failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h9 & segments[6:0] == 7'b111_0011) $display ("SUCCESS"); else $error("9 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h8 & segments[6:0] == 7'b111_1111) $display ("SUCCESS"); else $error("8 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h0 & segments[6:0] == 7'b111_1110) $display ("SUCCESS"); else $error("0 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h1 & segments[6:0] == 7'b011_0000) $display ("SUCCESS"); else $error("1 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h3 & segments[6:0] == 7'b111_1001) $display ("SUCCESS"); else $error("3 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h2 & segments[6:0] == 7'b110_1100) $display ("SUCCESS"); else $error("2 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h6 & segments[6:0] == 7'b101_1111) $display ("SUCCESS"); else $error("6 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h7 & segments[6:0] == 7'b111_0000) $display ("SUCCESS"); else $error("7 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h5 & segments[6:0] == 7'b101_1011) $display ("SUCCESS"); else $error("5 failed");
#100;
assert (x[63:0]==64'h01326754cdfeab98 & data[3:0]== 4'h4 & segments[6:0] == 7'b011_0011) $display ("SUCCESS"); else $error("4 failed");
#75;
end

endmodule
```

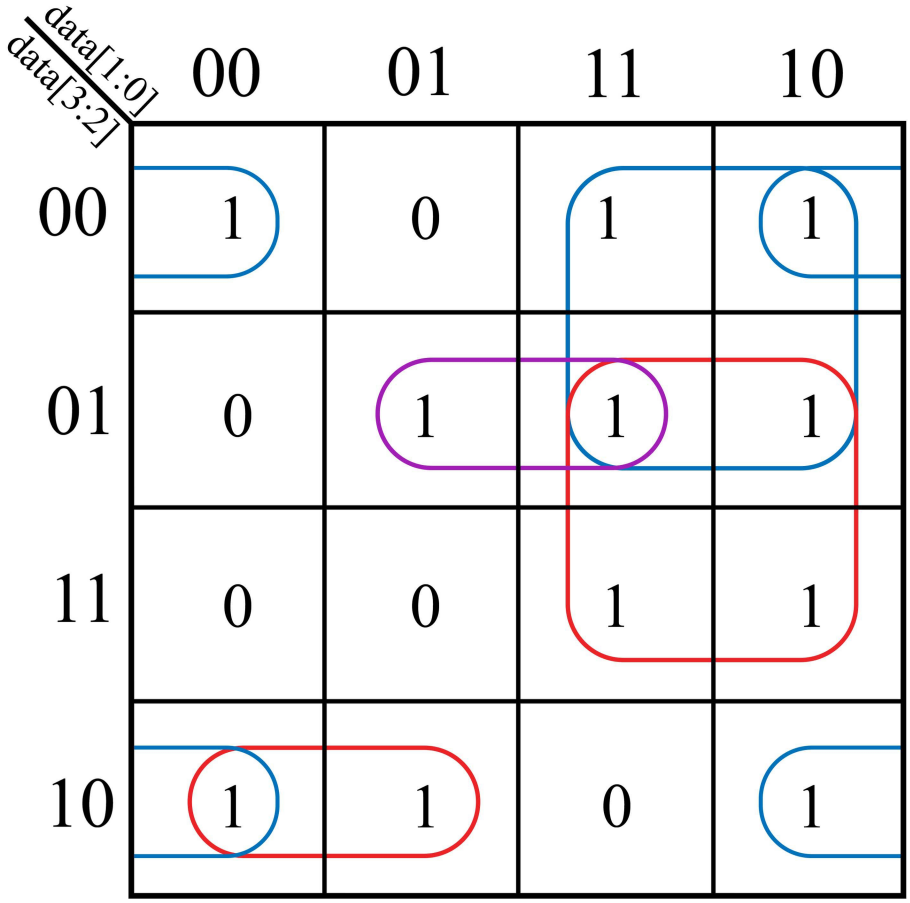
Truth table

data[3:0]	a	b	c	d	e	f	g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
1010	1	1	1	0	1	1	1
1011	0	0	1	1	1	1	1
1100	0	0	0	1	1	0	1
1101	0	1	1	1	1	0	1
1110	1	0	0	1	1	1	1
1111	1	0	0	0	1	1	1

K-maps and boolean equations

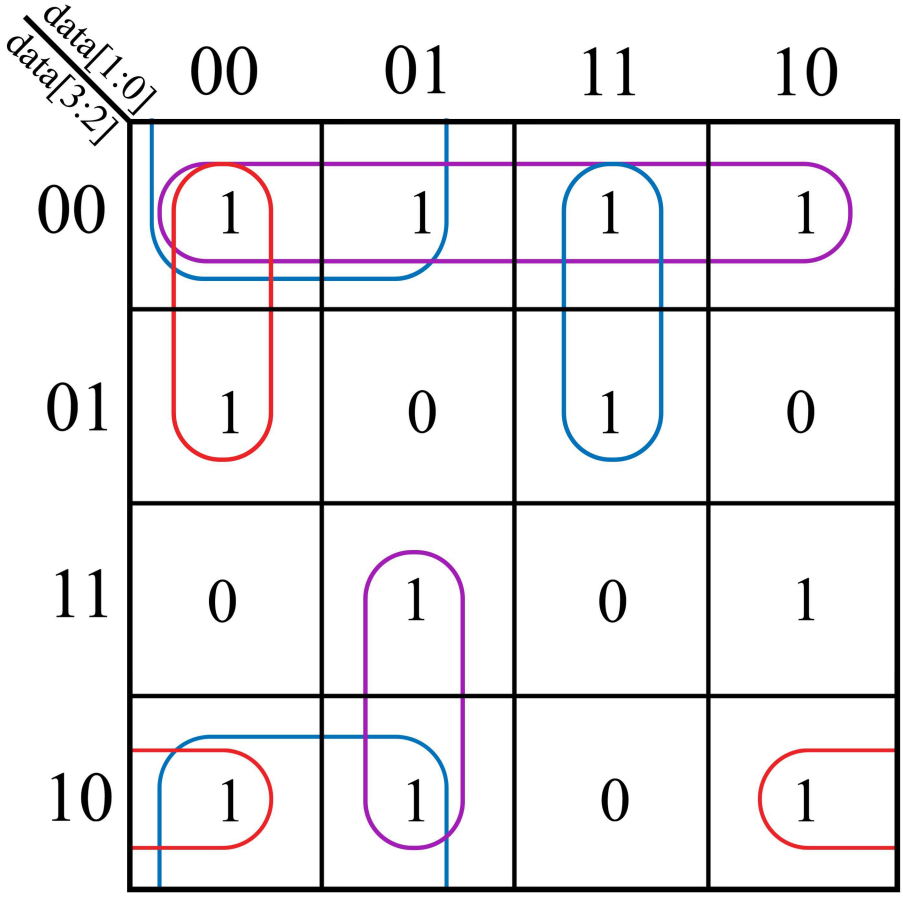
NOTE: the different colors used in the k-maps are just for clarifying each different group for each implicant

a



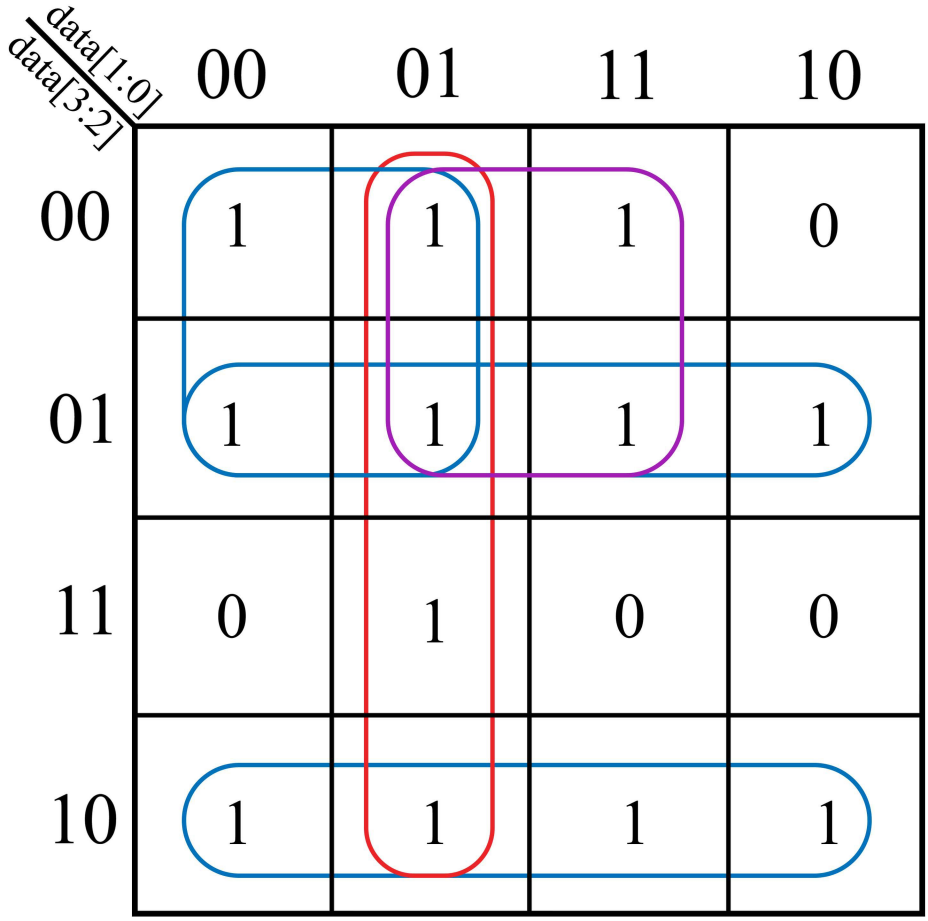
$$a = \overline{data\ [3]}.data\ [1] + data\ [2].data\ [1] + \overline{data\ [3]}.data\ [2].data\ [0] + data\ [3].\overline{data\ [2]}.\overline{data\ [0]} + \overline{data\ [3]}.\overline{data\ [2]}.\overline{data\ [0]} + data\ [3].\overline{data\ [2]}.data\ [1]$$

b



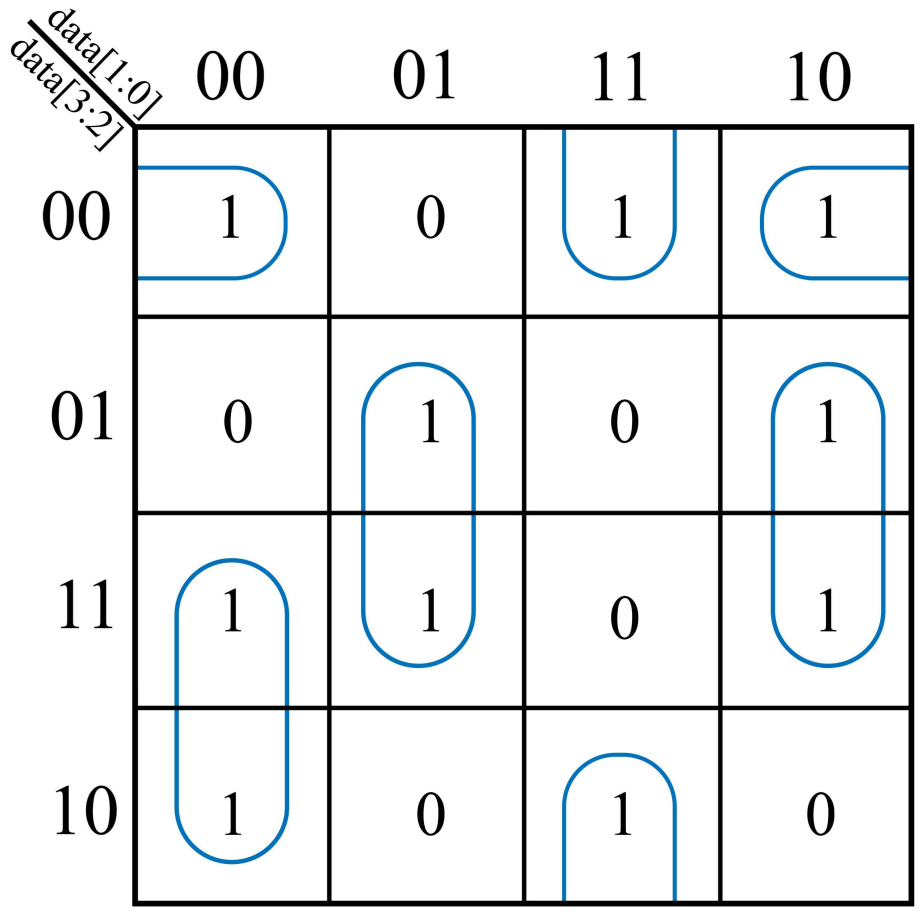
$$b = \overline{data\ [3]}.\overline{data\ [2]} + \overline{data\ [2]}.\overline{data\ [1]} + \overline{data\ [3]}.\overline{data\ [1]}.\overline{data\ [0]} + \overline{data\ [3]}.data\ [1].data\ [0] + data\ [3].\overline{data\ [1]}.data\ [0] + data\ [3].\overline{data\ [2]}.\overline{data\ [0]}$$

c



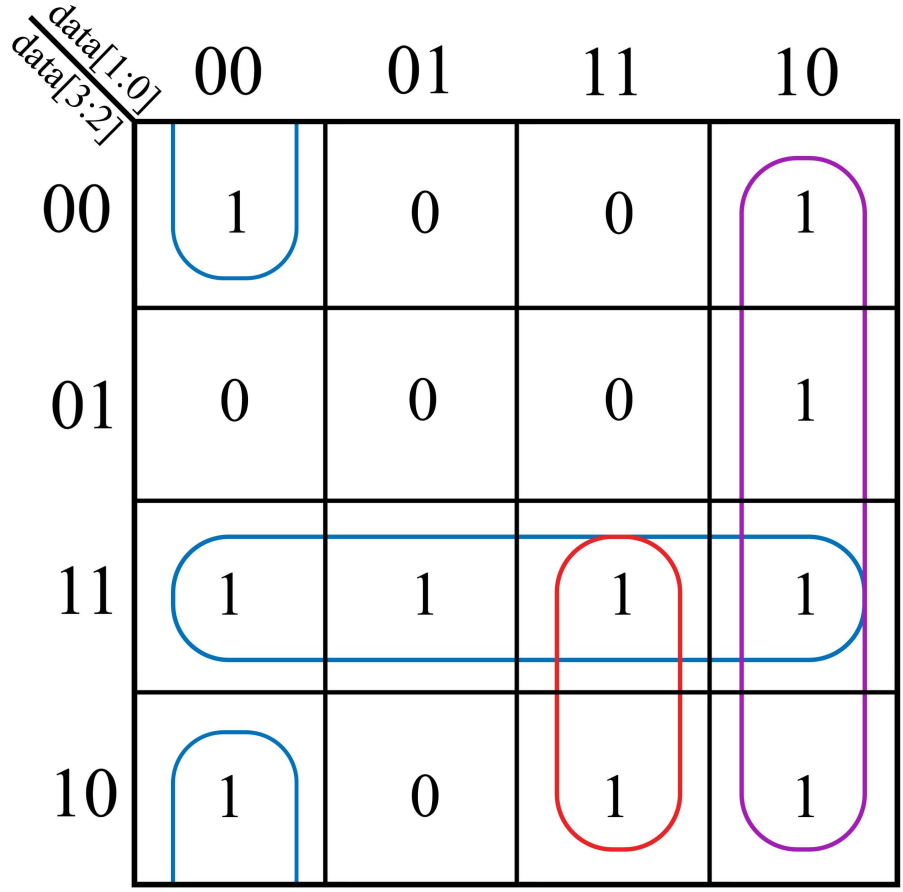
$$c = \overline{data\,[3]}.data\,[2] + data\,[3].\overline{data\,[2]} + \overline{data\,[1]}.data\,[0] + \overline{data\,[3]}.\overline{data\,[1]} + \overline{data\,[3]}.data\,[1]$$

d



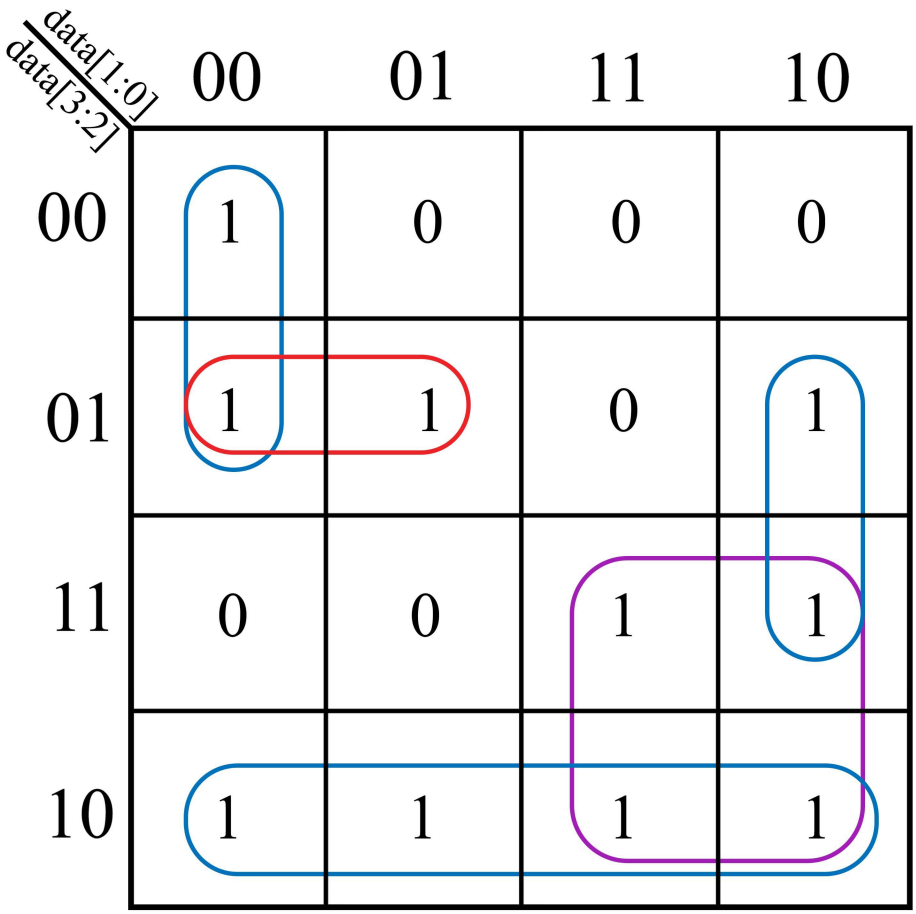
$$d = data\,[3].\overline{data\,[1]}\,.\overline{data\,[0]} + data\,[2].\overline{data\,[1]}.data\,[0] + data\,[2].data\,[1].\overline{data\,[0]} + \overline{data\,[2]}.data\,[1].data\,[0] + \overline{data\,[3]}.\overline{data\,[2]}.\overline{data\,[0]}$$

e



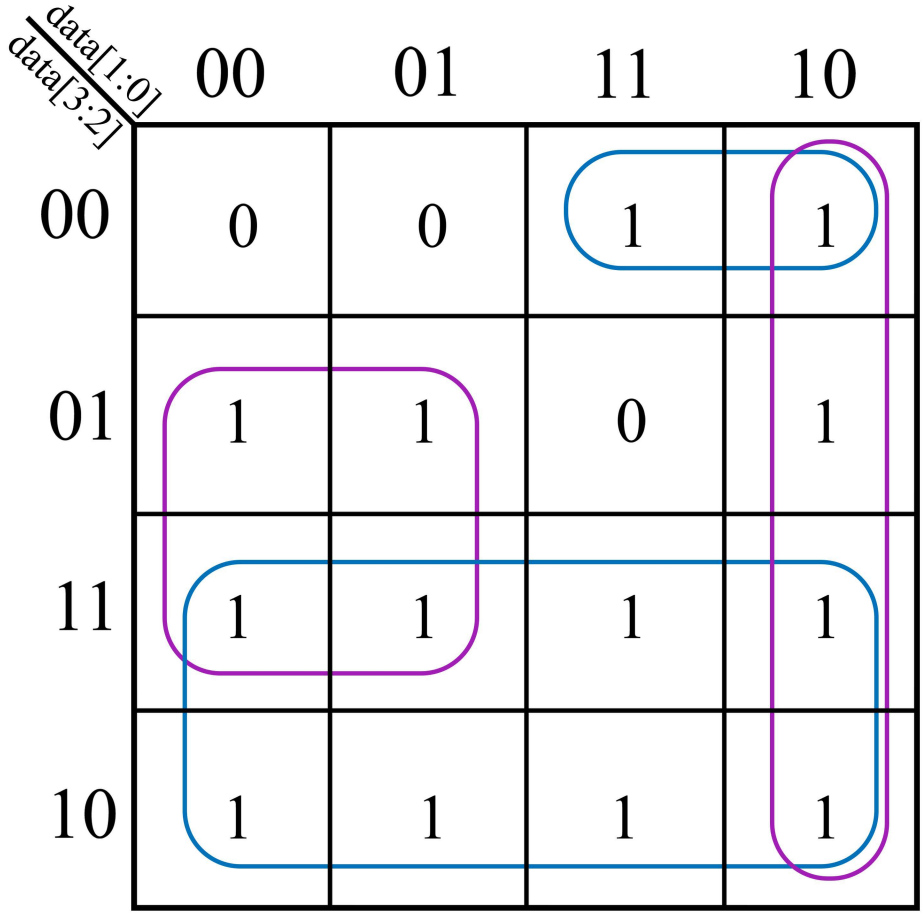
$$e = data[3].data[2] + data[1].\overline{data[0]} + data[3].data[1].data[0] + \overline{data[2]}.\overline{data[1]}.\overline{data[0]}$$

f



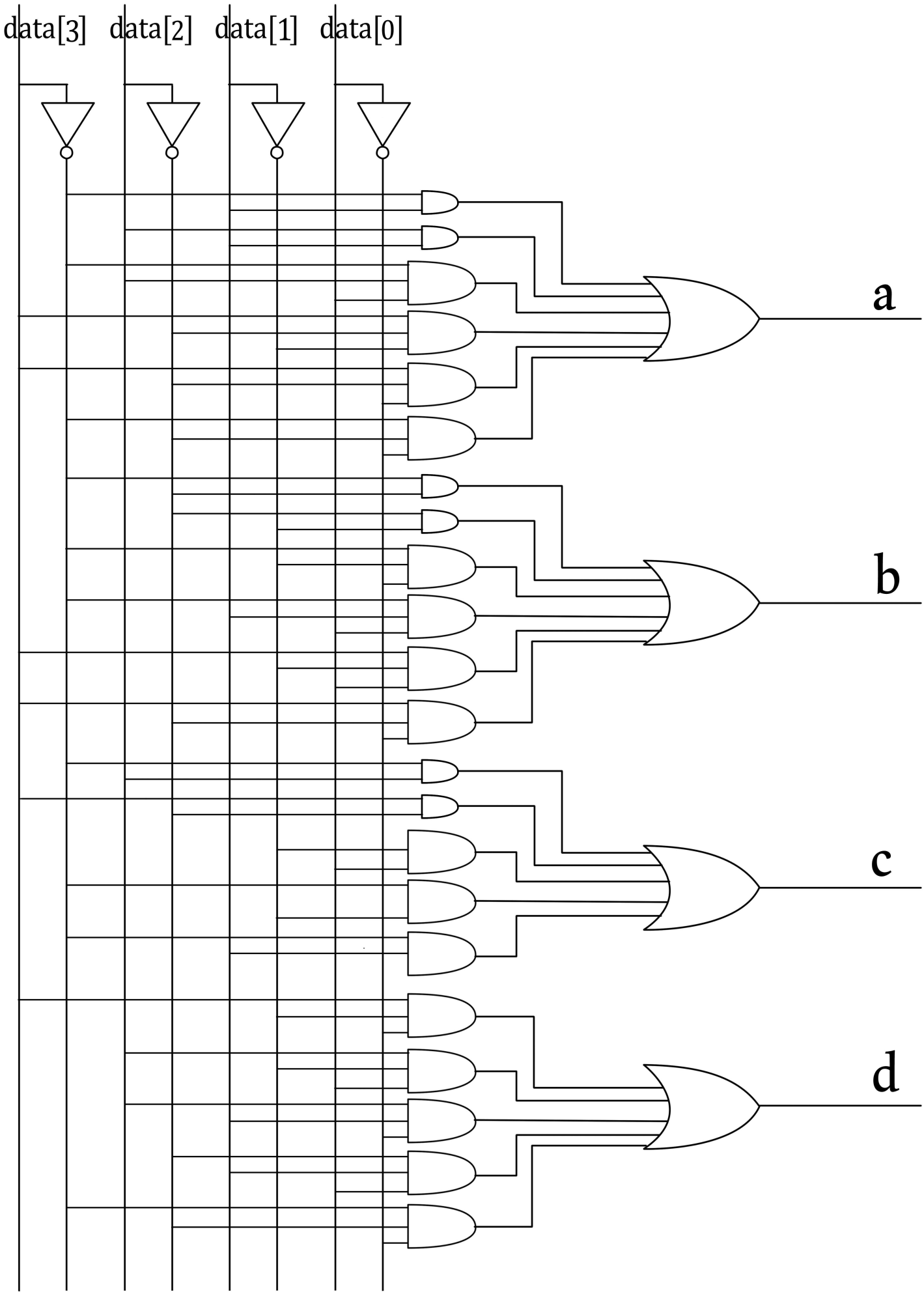
$$f = data[3].\overline{data[2]} + data[3].data[1] + data[2].data[1].\overline{data[0]} + \overline{data[3]}.\overline{data[1]}.\overline{data[0]} + \overline{data[3]}.data[2].\overline{data[1]}$$

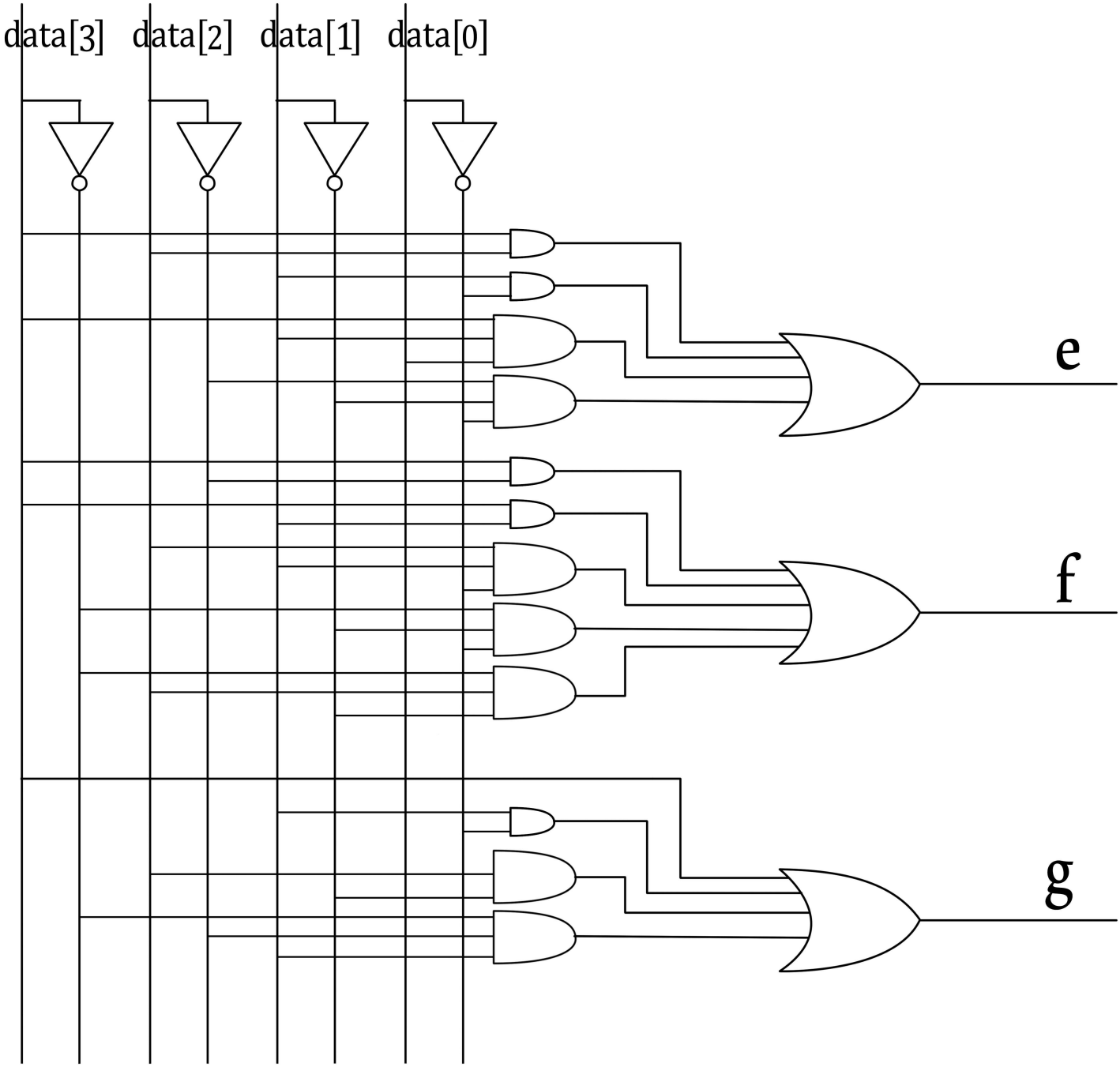
g



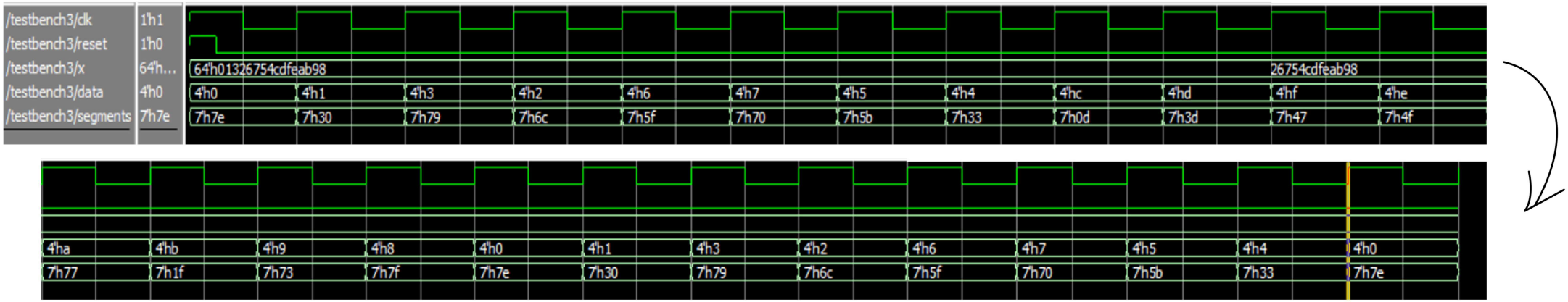
$$g = data[3] + data[1].\overline{data[0]} + data[2].\overline{data[1]} + \overline{data[3]}.\overline{data[2]}.data[1]$$

logic gates schematic


data[3]
data[2]
data[1]
data[0]
a
b
c
d


data[3]
data[2]
data[1]
data[0]
e
f
g

waveform
My specific x = 23



NOTE: the second image is continueous with the first one; they are separated here because of the limited length of the report

NOTE: the waveform of my specified X ends just before the yellow line in the second image
after the yellow line is the beginning of the new flow of my specified X

THE SIGNALS

- x is the output of the garycode and the input of the counter
- data is the output of the counter and the input of the seven-segment decoder
- segments is the output of the seven-segment decoder

And this waveform shows the data flow integration between the three modules