**Module: R3: DLD + DSD**

**Section:** Sequential Circuits **Task:** Design Problem

**Design Problem**

**Sequential Circuits**

* **Question: Design a stopwatch using counters:**

1. **Design:**

I have used four counters to make this stopwatch. The stopwatch can show up to 9 minutes, 59 seconds, and 9 tenths of a seconds. So, three counters go up to 9, and one counter goes up to 5.

First, we count from 0 to 9 (for tenths of a second). When we reach 9, it sends a signal to the next counter (for seconds). This next counter also counts from 0 to 9. When it reaches 9, it sends a signal to the third counter

This counter goes up to 5. When it reaches 5, it sends a signal to the last counter (for minutes), which counts up to 9.

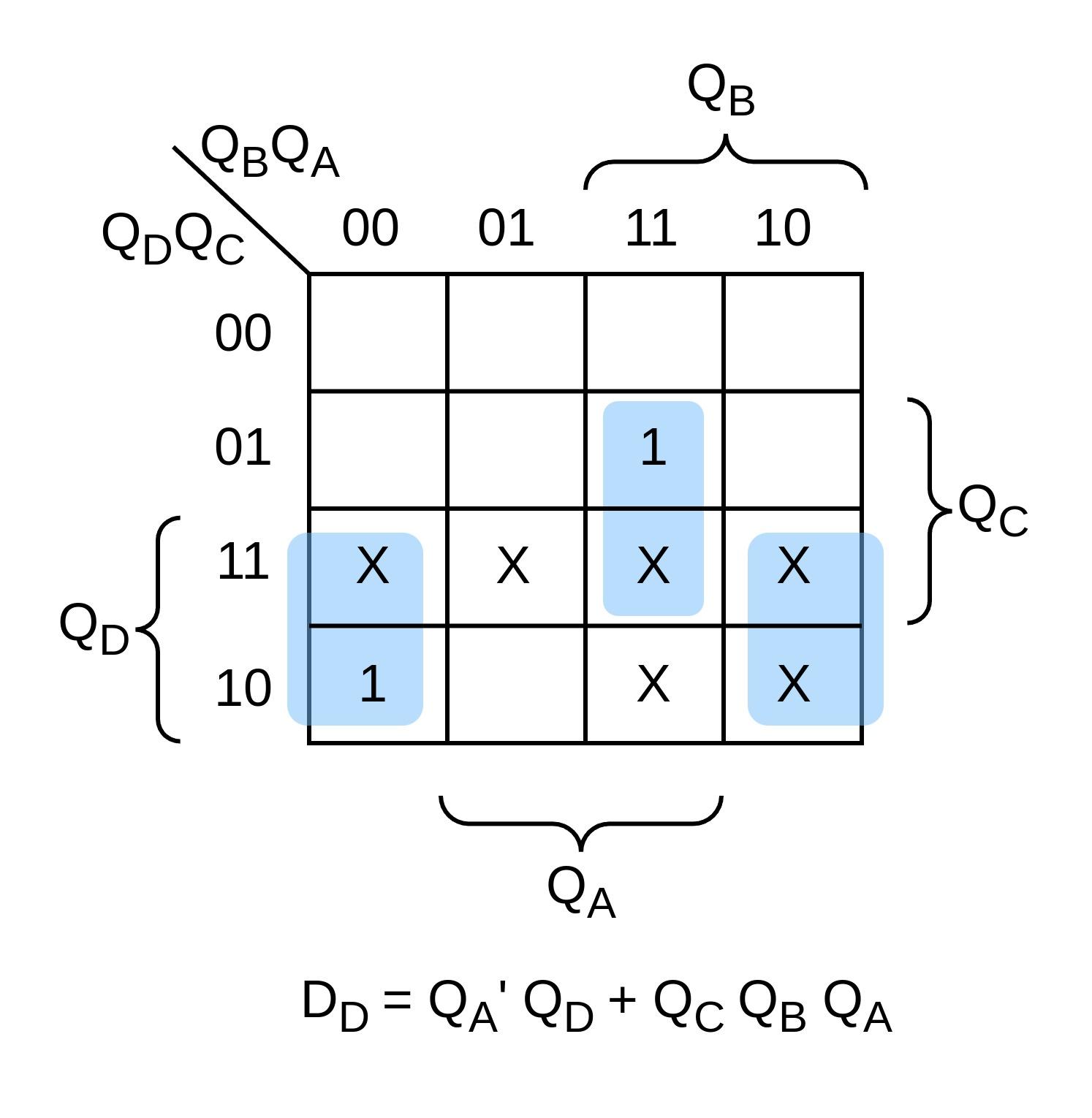
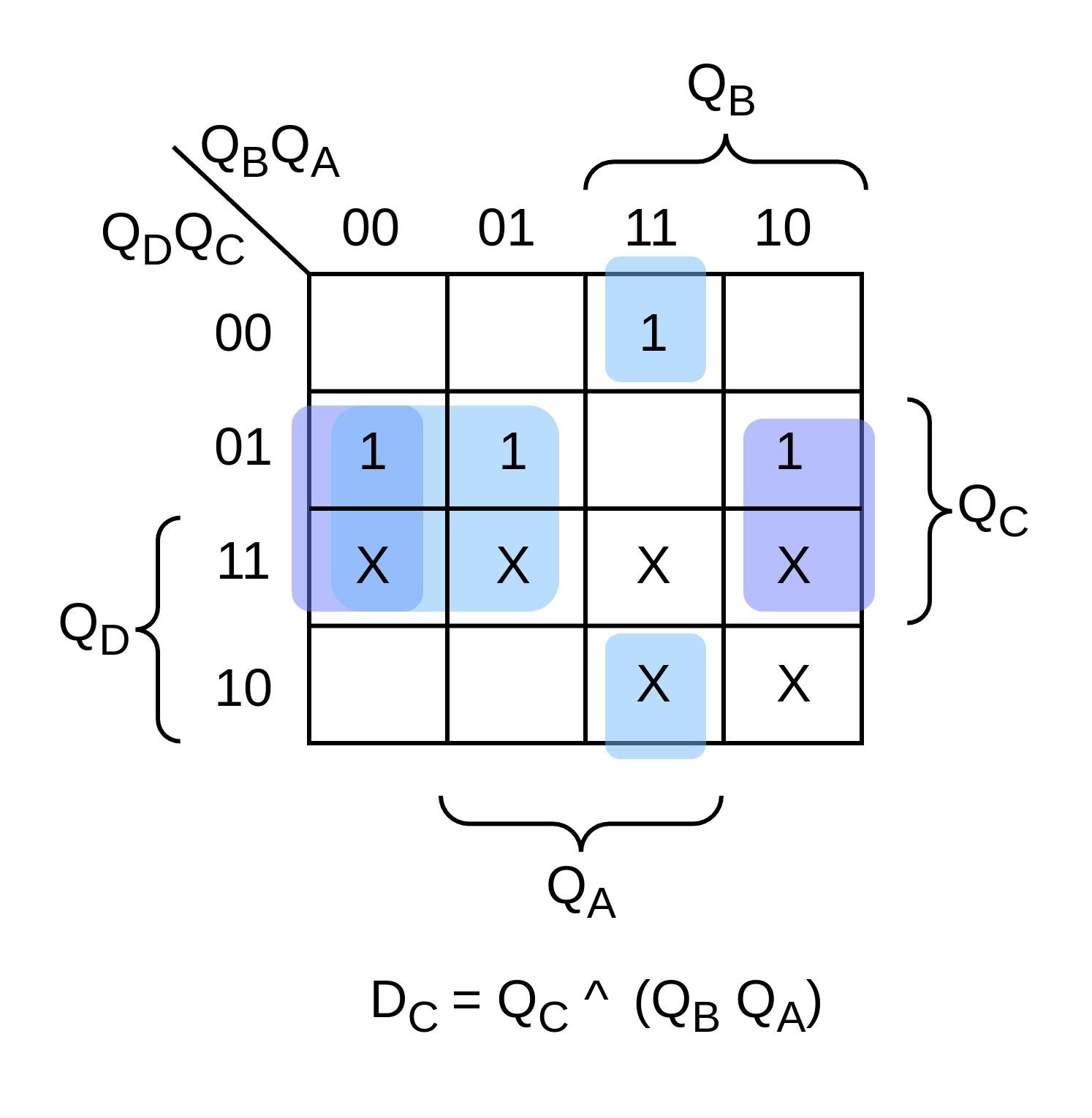
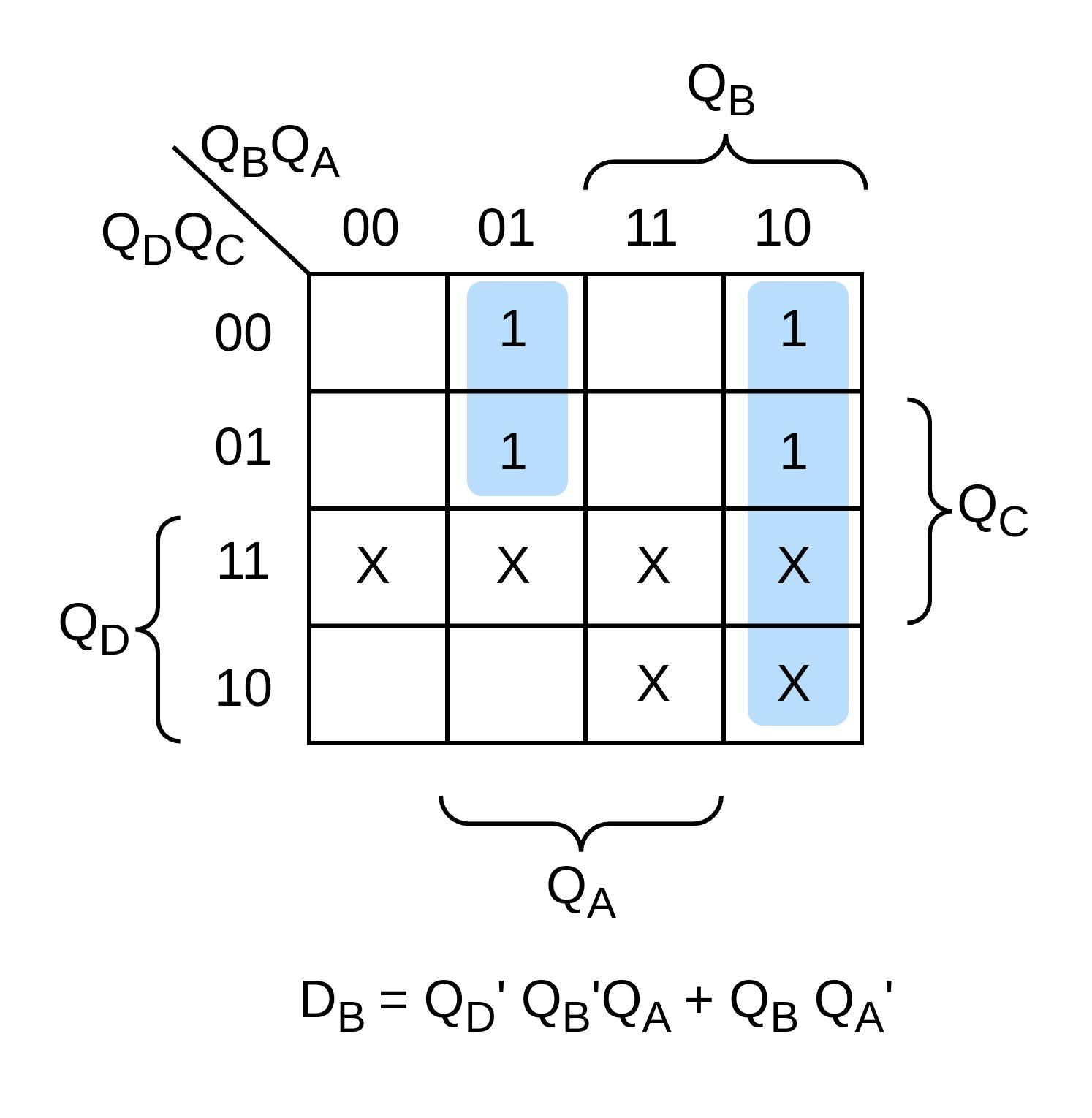
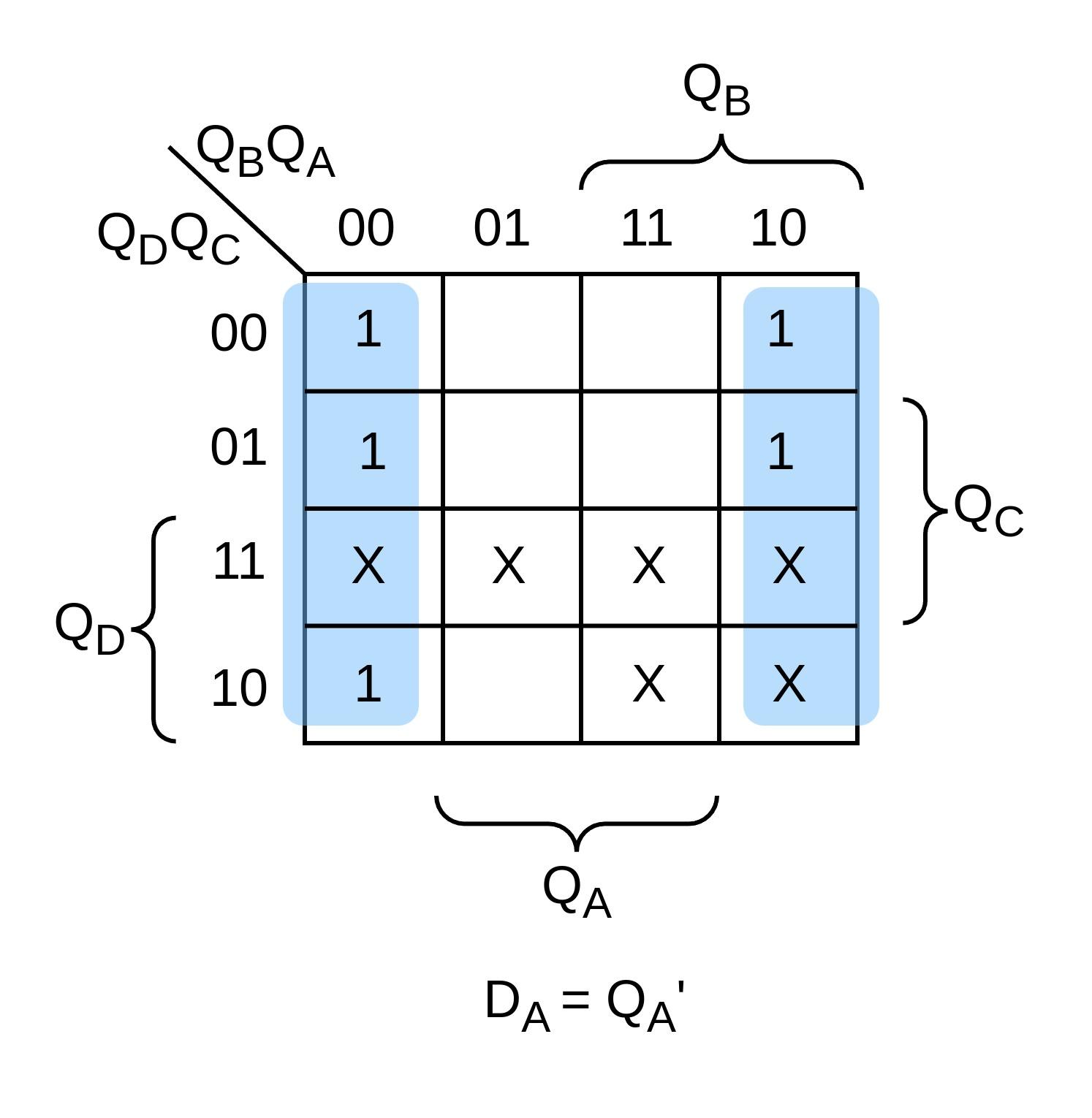
So, each counter sends a signal to the next one when it reaches its maximum count. This way, we keep track of time accurately.

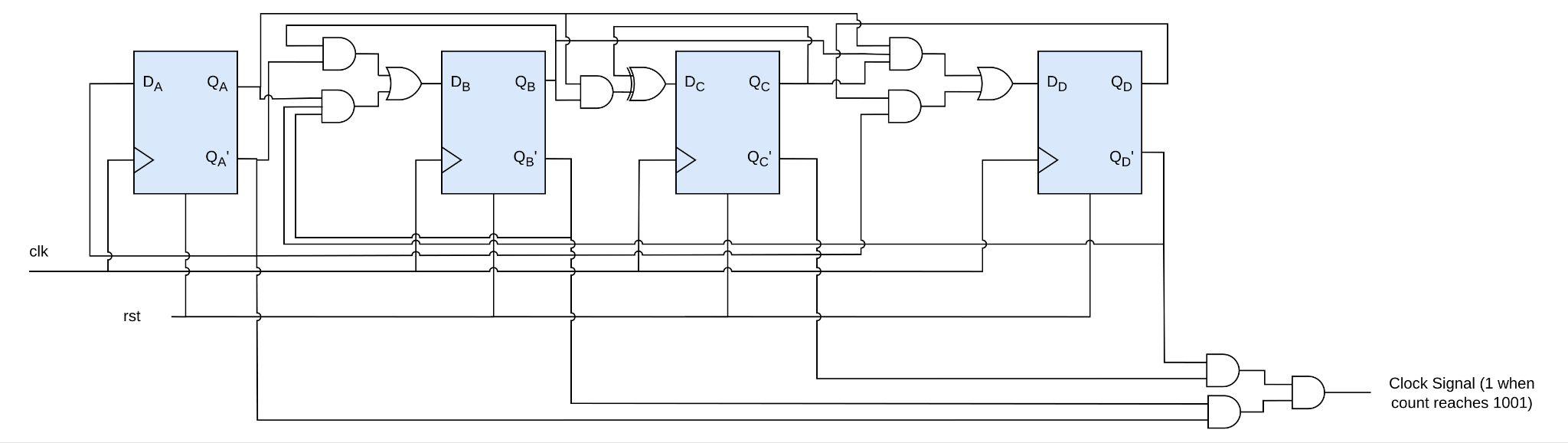
The truth tables will be as follows:

1. **Truth Table:**
   * **Counter 9**

| Present State | | | | Next State | | | | FF Inputs | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QD | QC | QB | QA | QD (t+1) | QC (t+1) | QB (t+1) | QA (t+1) | DD | DC | DB | DA |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Using K-Maps:

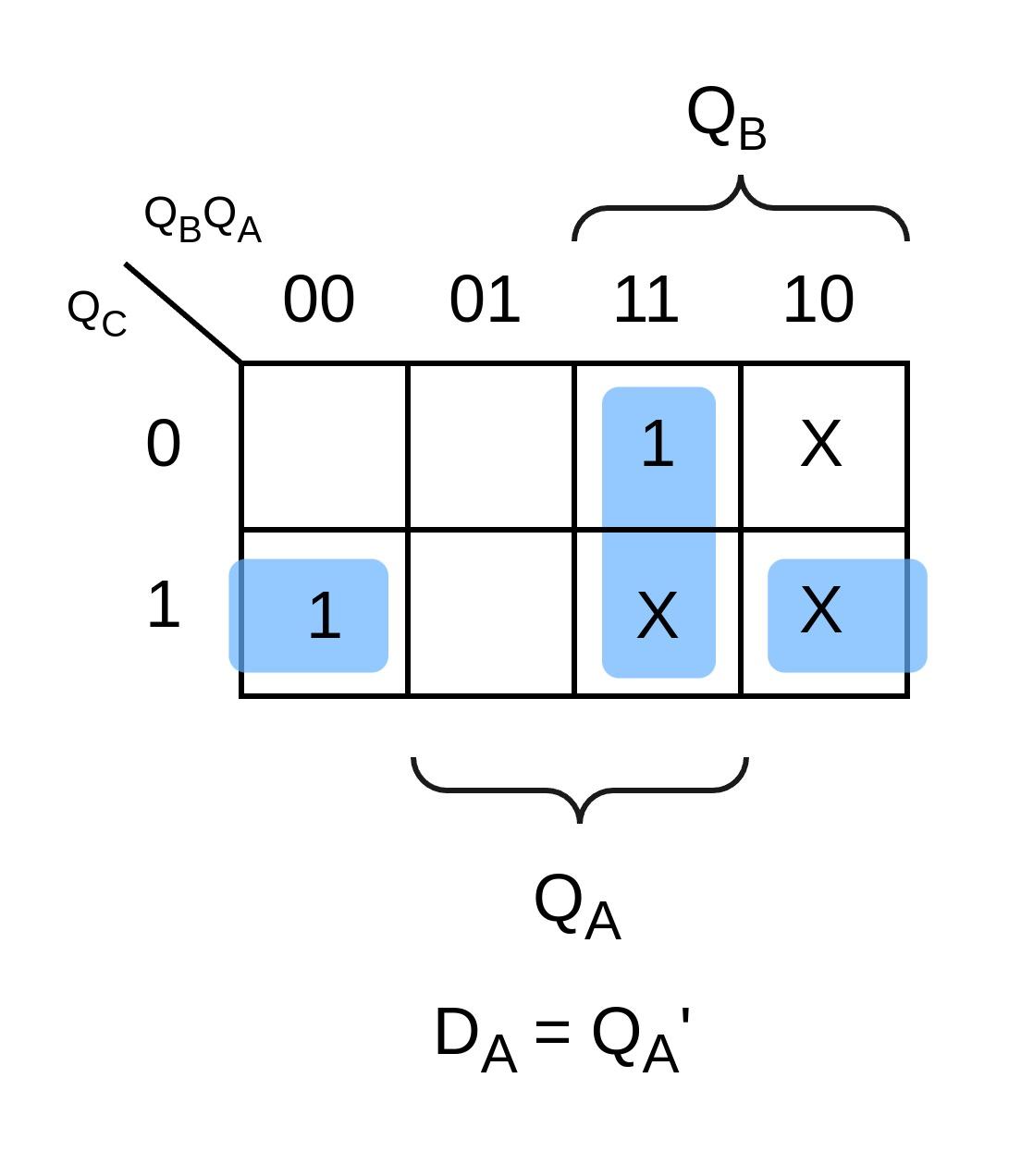
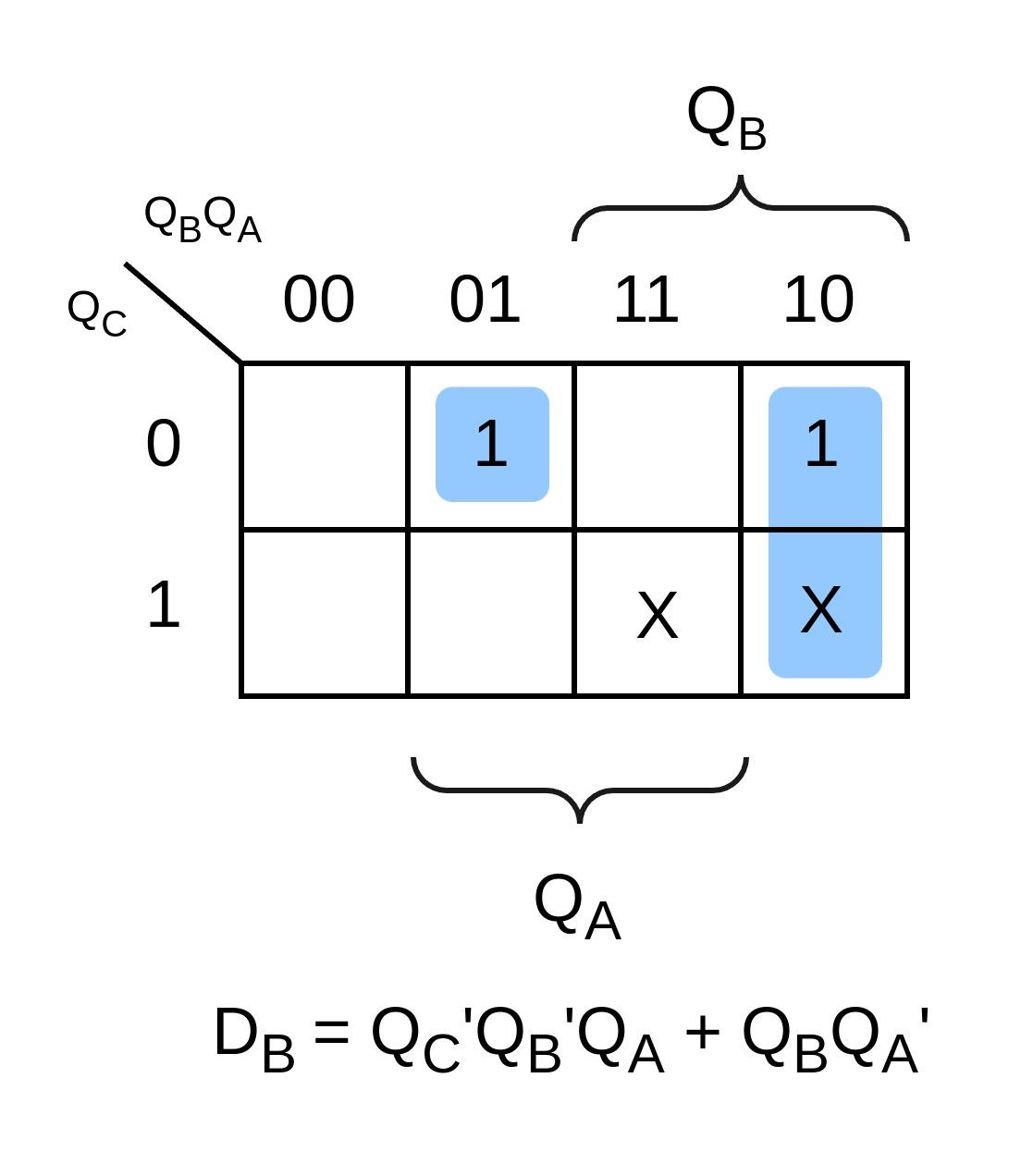
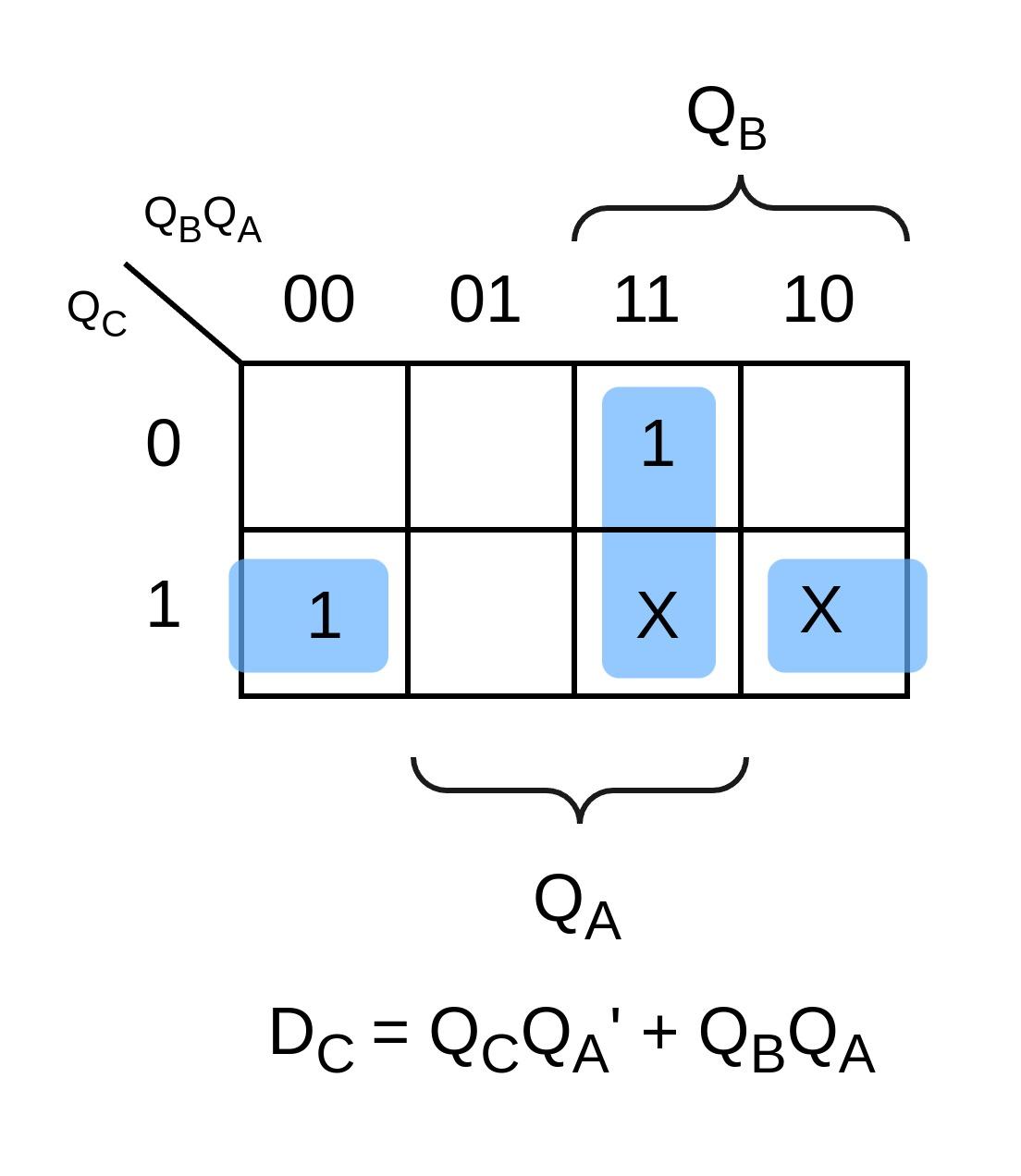
   

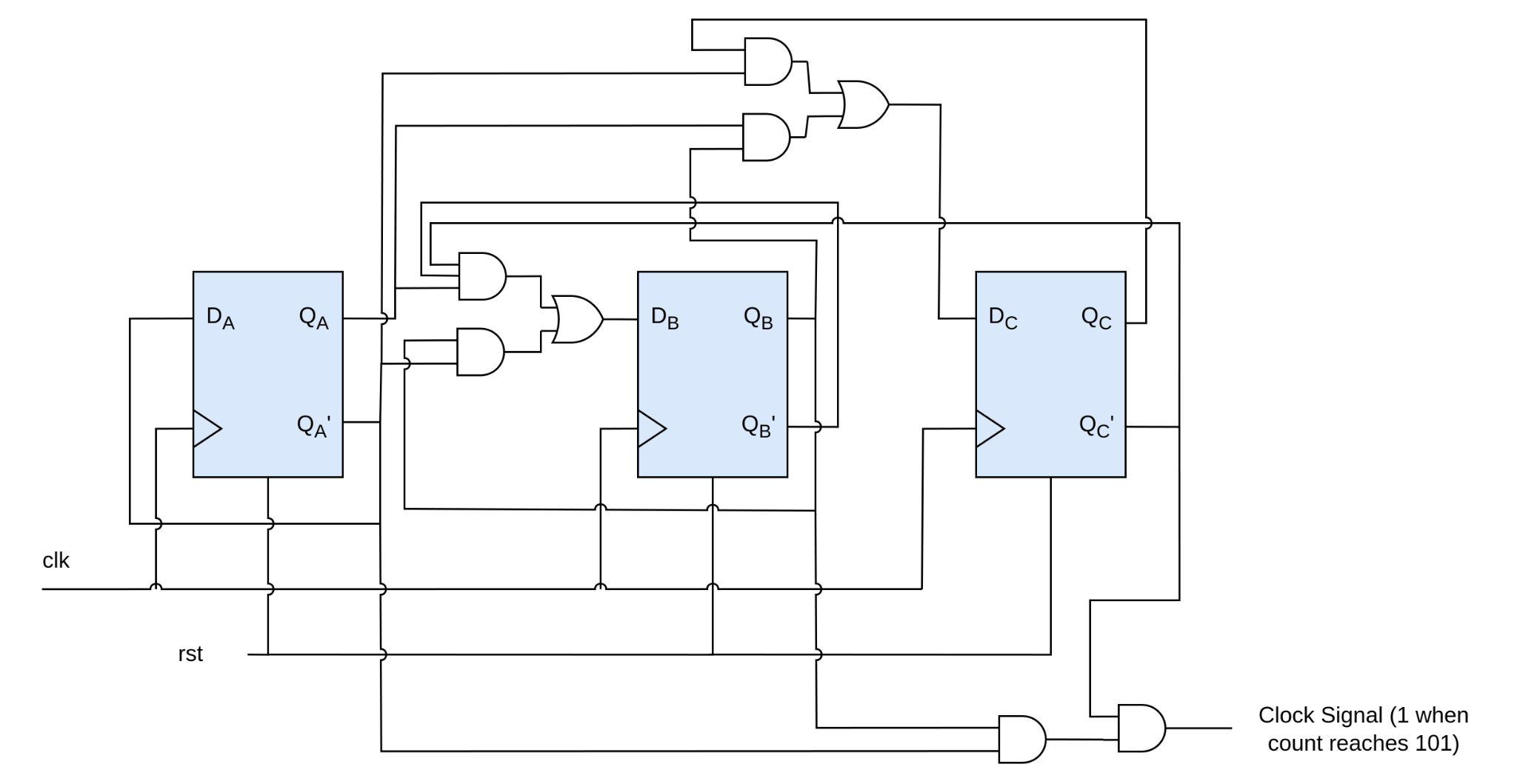


* + **Counter 5**

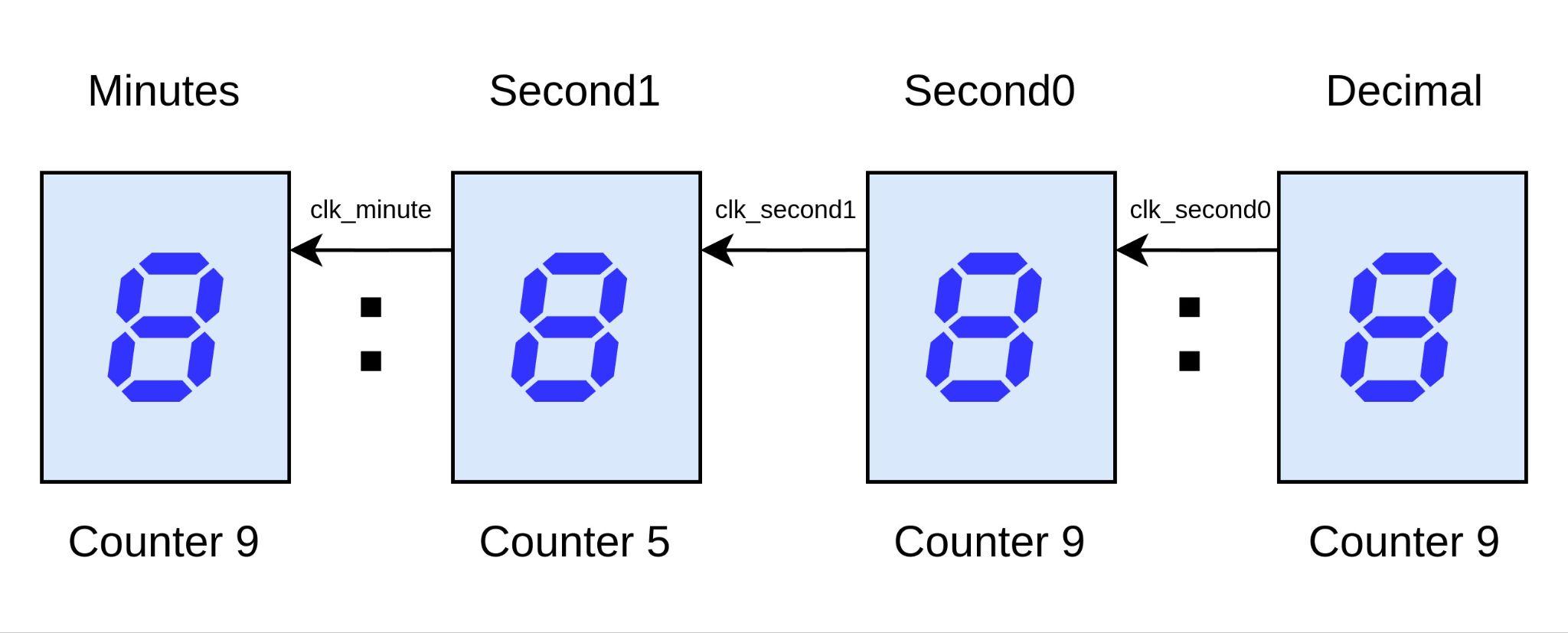
| Present State | | | Next State | | | FF Inputs | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QC | QB | QA | QC (t+1) | QB (t+1) | QA (t+1) | DC | DB | DA |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Using K-Maps:





Final circuit will look like following:



1. **Verilog Code:**

module stopwatch (input clk, clr, output [3:0] min, sec0, deci, output [2:0] sec1);

//Deci

counter9 decimal (.clk(clk), .rst(clr), .out(deci));

//Second\_0 (LSB)

wire clk\_sec0;

assign clk\_sec0 = (~deci[3] & ~deci[2] & ~deci[1] & ~deci[0]);

counter9 second\_0 (.clk(clk\_sec0), .rst(clr), .out(sec0));

//Second\_1 (MSB)

wire clk\_sec1;

assign clk\_sec1 = (~sec0[3] & ~sec0[2] & ~sec0[1] & ~sec0[0]);

counter5 second\_1 (.clk(clk\_sec1), .rst(clr), .out(sec1));

//Minutes

wire clk\_minute;

assign clk\_minute = (~sec1[2] & ~sec1[1] & ~sec1[0]);

counter9 minute (.clk(clk\_minute), .rst(clr), .out(min));

endmodule

module counter9 (input clk, rst, output [3:0] out);

wire D\_3, D\_2, D\_1, D\_0;

// D\_d = D\_3 = Q0' Q3 + Q2 Q1 Q0

assign D\_3 = (~out[0] & out[3]) | (out[2] & out[1] & out[0]);

// D\_c = D\_2 = Q2 ^ (Q1 Q0)

assign D\_2 = out[2] ^ (out[1] & out[0]);

// D\_b = D\_1 = (Q3' Q1' Q0) + (Q1 Q0')

assign D\_1 = (~out[3] & ~out[1] & out[0]) | (out[1] & ~out[0]);

// D\_a = D\_0 = Q0'

assign D\_0 = ~out[0];

d\_ff D (.d(D\_3), .clk(clk), .rst(rst), .q(out[3]));

d\_ff C (.d(D\_2), .clk(clk), .rst(rst), .q(out[2]));

d\_ff B (.d(D\_1), .clk(clk), .rst(rst), .q(out[1]));

d\_ff A (.d(D\_0), .clk(clk), .rst(rst), .q(out[0]));

endmodule

module counter5 (input clk, rst, output [2:0] out);

wire D\_2, D\_1, D\_0;

// D\_c = D\_2 = Q1 Q0 + Q2 Q0'

assign D\_2 = (out[1] & out[0]) | (out[2] & ~out[0]);

// D\_b = D\_1 = Q1 Q0' + Q2' Q1' Q0

assign D\_1 = (out[1] & ~out[0]) | (~out[2] & ~out[1] & out[0]);

// D\_a = D\_0 = Q0'

assign D\_0 = ~out[0];

d\_ff C (.d(D\_2), .clk(clk), .rst(rst), .q(out[2]));

d\_ff B (.d(D\_1), .clk(clk), .rst(rst), .q(out[1]));

d\_ff A (.d(D\_0), .clk(clk), .rst(rst), .q(out[0]));

endmodule

module d\_ff (input d, clk, rst, output reg q);

always @(posedge clk or posedge rst) begin

if (rst)

q <= 1'b0;

else

q <= d;

end

endmodule

1. **Testbench:**

`timescale 1ms/100us

module tb\_stopwatch;

//input clk, clr, output reg [3:0] min, sec0, deci, output reg [2:0] sec1

reg clk, clr;

wire [3:0] min, sec0, deci;

wire [2:0] sec1;

stopwatch dut (.clk(clk), .clr(clr), . min(min), .sec0(sec0), .deci(deci), .sec1(sec1));

always #50 clk = ~clk;

initial begin

$dumpvars;

clk = 0;

clr = 0;

#20;

clr = 1;

#300;

clr = 0;

#650000;

$finish;

end

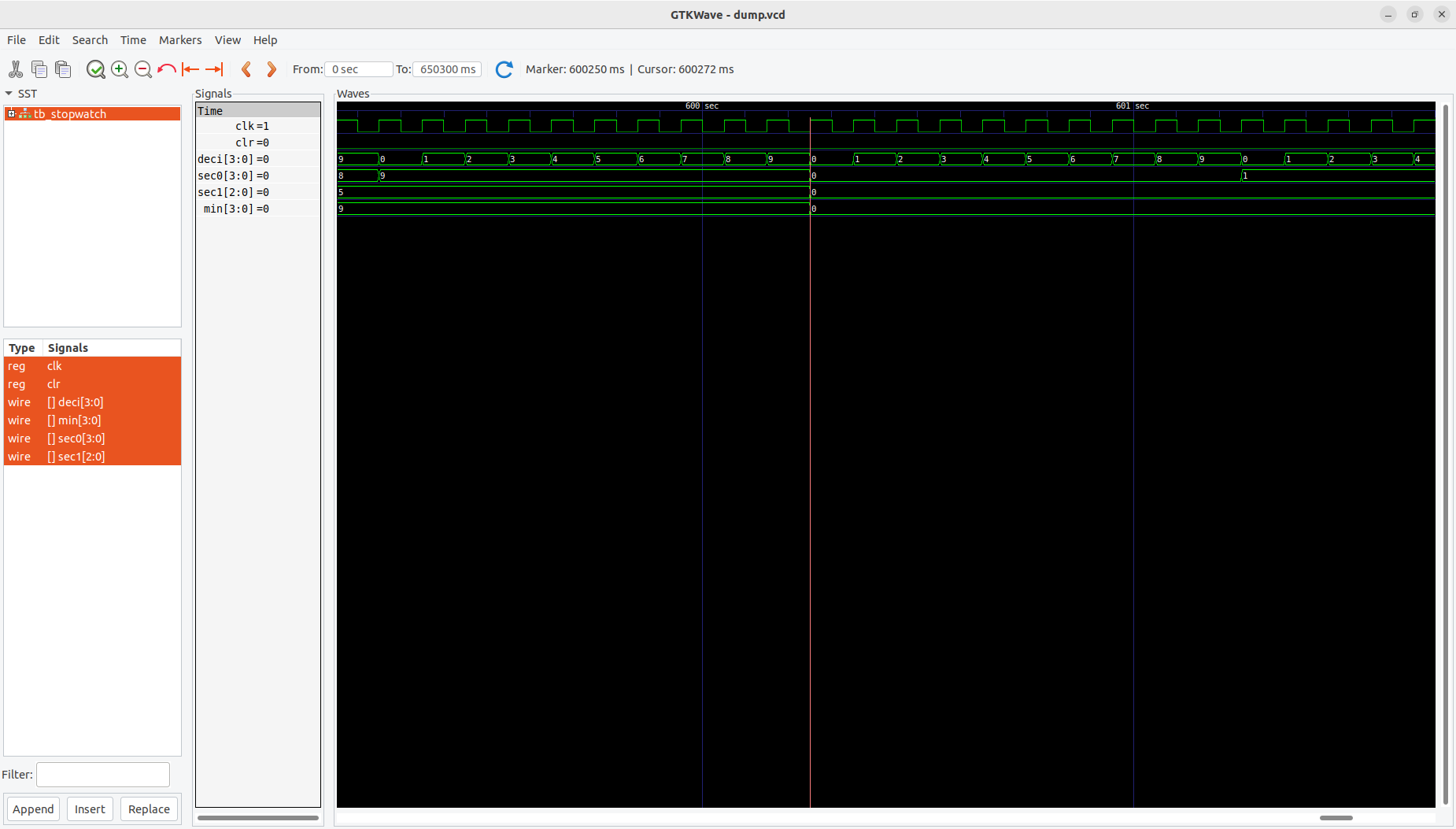
initial begin

$monitor("%d : %d%d : %d",min,sec1,sec0,deci);

end

endmodule

1. **Output:**

****

1. **Duty Cycle and Frequency:**
   1. **Pulse-width:** From the waveform, it is evident that the pulse-width is 50 ms. Since we are toggling the clock every 50 time units in our code and our timescale is in ms (1ms = time unit).
   2. **Duty Cycle:**

Duty Cycle = (Time the signal is high/Total time period) \* 100

Duty Cycle = (50 ms/100 ms) \* 100 = 50%

* 1. **Frequency:**

Frequency = 1/time period

Frequency = 1/100 ms = 10 Hz