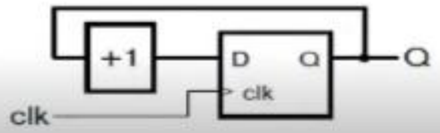
Counters

A counter is a sequential component that increments or decrements a stored value. Counters occur in many digital circuit applications. For example, if an application requires a given operation to be performed on

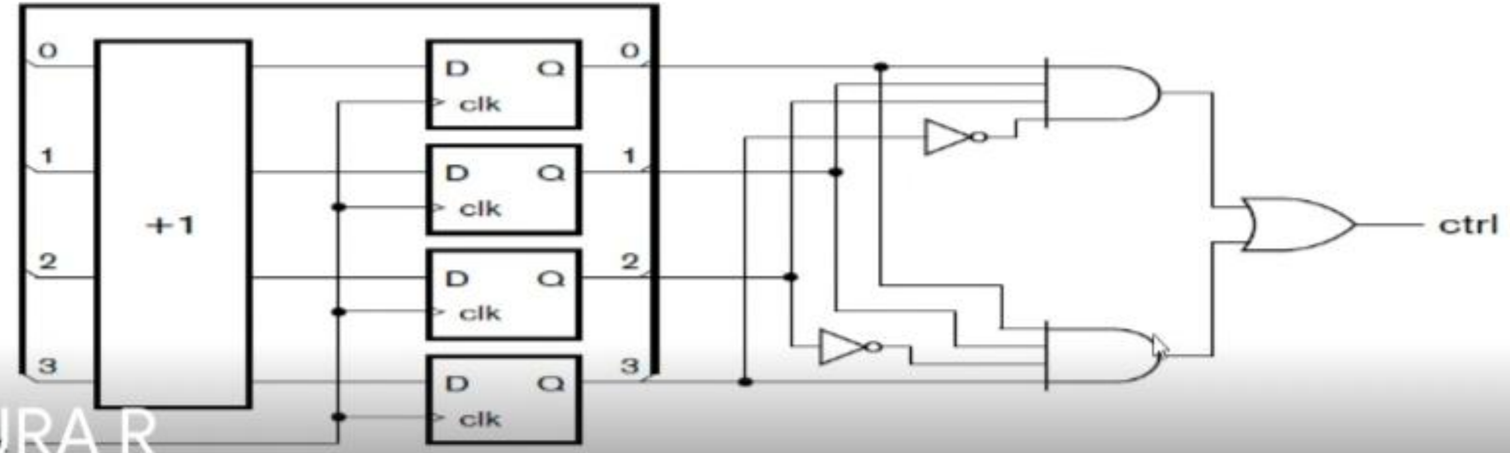
- a number of items of data or to be repeated a number of times, a counter can be used to keep track of how many items have been processed or how many times the operation has been performed. Counters are also used as timers, by counting the number of intervals of a fixed duration that have passed.
- A simple form of counter is composed of an edge-triggered register counter composed of a register and an inc and an incrementer, as shown in Figure .



- The value stored in the reg ister is interpreted as an unsigned binary integer. The incrementer can be implemented using the circuit we described for an unsigned incrementer. The counter increments the stored value on every clock edge. When the stored count value reaches its maximum value ($2n — 1$, for an n-bit counter), the incrementer yields a result of all zeros,with the carry out being ignored. This result value is stored on the next clock edge.
- Thus, the counter acts like the odometer in a car, rolling over to zeros after reaching its maximum value. Mathematically speaking, the counter increments modulo $2n$. The counter goes through all $2n$ unsigned binary integer values in order every $2n$ clock cycles. One use for such a counter is in conjunction with a decoder to produce periodic control signals.

Design a circuit that counts 16 clock cycles and produces a control signal, ctrl, that is 1 during every eighth and twelfth cycle.

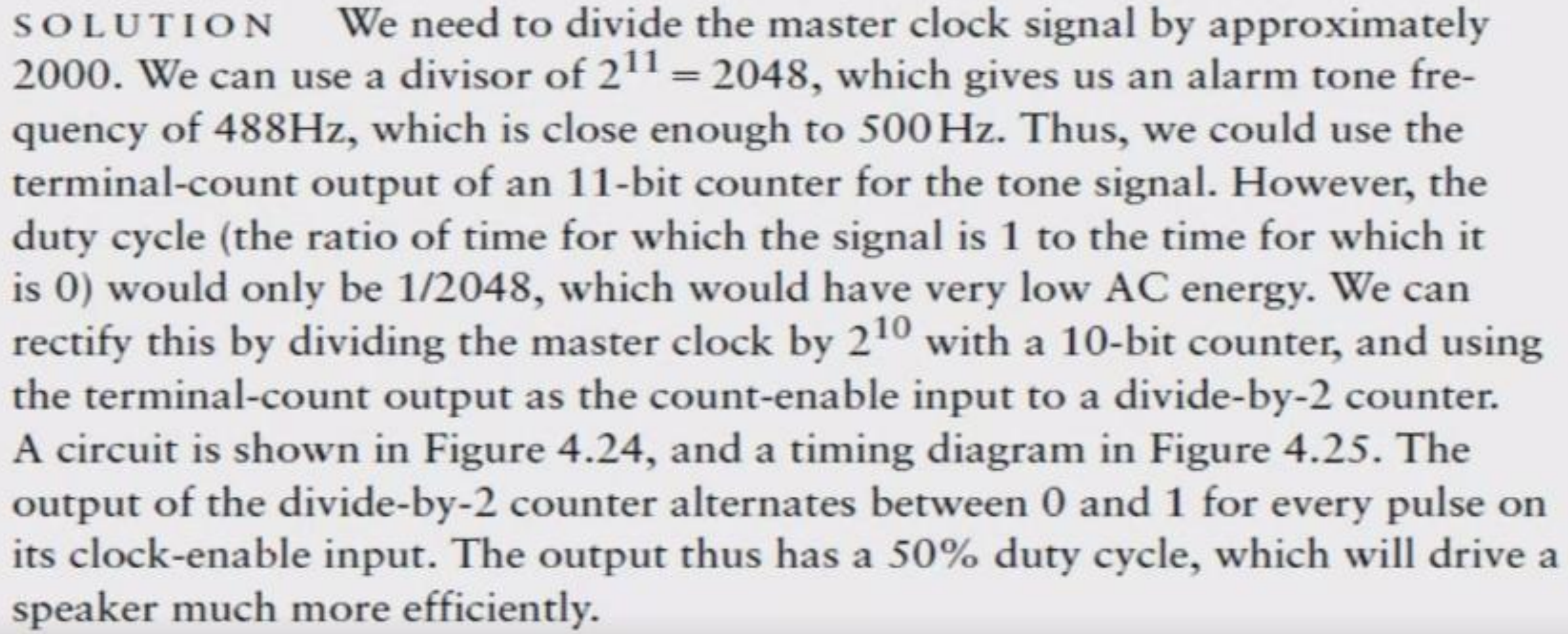
SOLUTION We need a 4-bit counter, since $16 = 2^4$. The counter counts from 0 to 15 and then wraps back to 0. During the eighth cycle, the counter value is 7 (0111_2), and during the twelfth cycle, the counter value is 11 (1011_2). We can generate the control signal by decoding the two required counter values and forming the logical OR of the decoded signals. The required circuit is shown in Figure 4.21.



Develop a Verilog model of the circuit for previous example

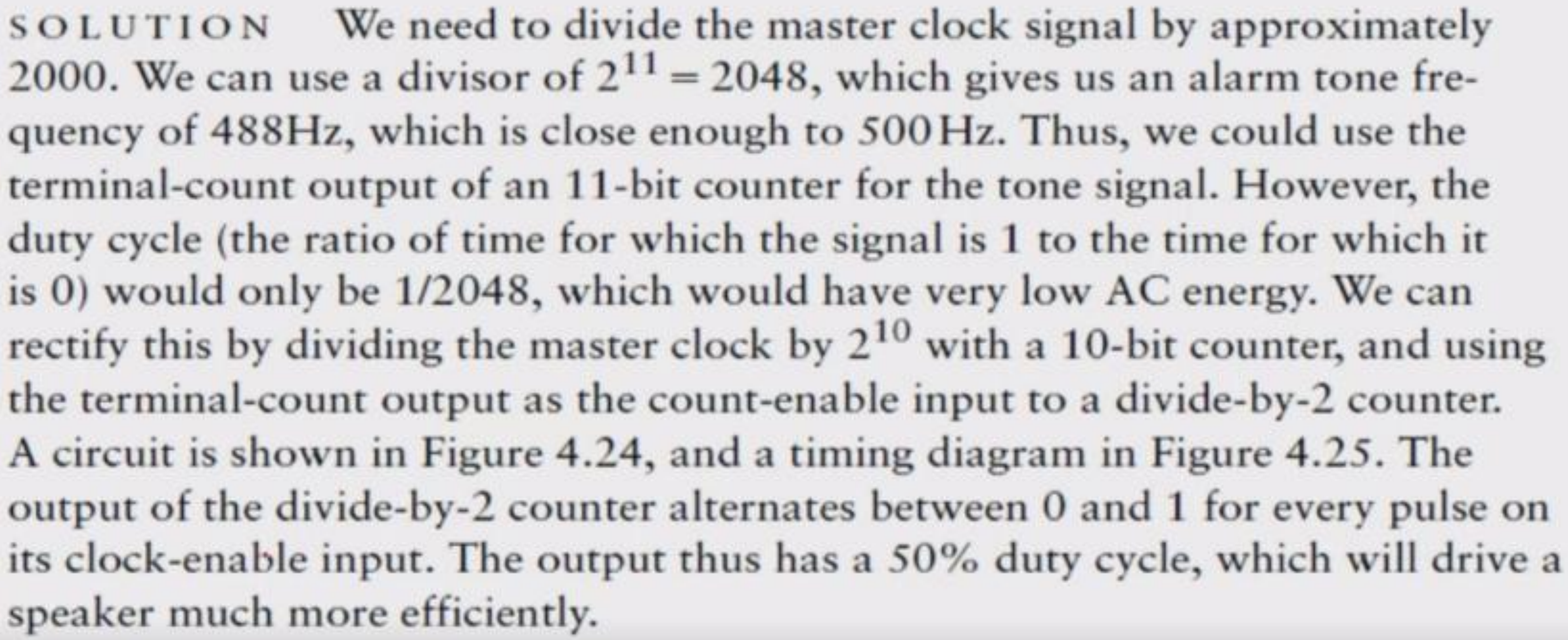
```
module decoded_counter ( output ctrl,
                         input  clk );

  reg [3:0] count_value;

  always @(posedge clk)
    count_value <= count_value + 1;

  assign ctrl = count_value == 4'b0111 ||
                count_value == 4'b1011;

endmodule
```

A digital alarm clock needs to generate a periodic signal at a frequency of approximately 500Hz to drive the speaker for the alarm tone. Use a counter to divide the system’s master clock signal, with a frequency of
1 MHz, to derive the alarm tone.

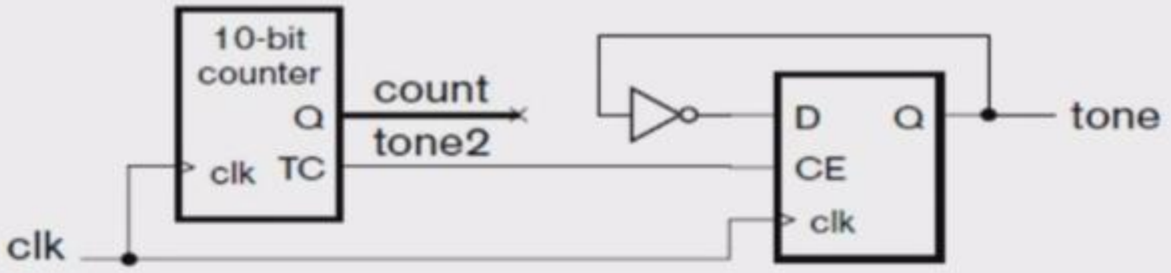


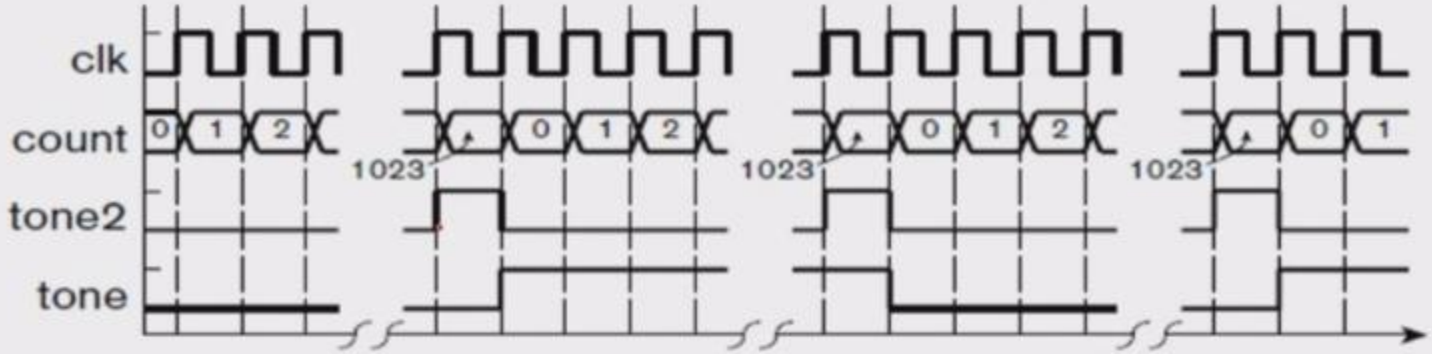
SOLUTION We need to divide the master clock signal by approximately 2000. We can use a divisor of $2^{11} = 2048$, which gives us an alarm tone frequency of 488Hz, which is close enough to 500 Hz. Thus, we could use the terminal-count output of an 11-bit counter for the tone signal. However, the duty cycle (the ratio of time for which the signal is 1 to the time for which it is 0) would only be 1/2048, which would have very low AC energy. We can rectify this by dividing the master clock by 2^{10} with a 10-bit counter, and using the terminal-count output as the count-enable input to a divide-by-2 counter. A circuit is shown in Figure 4.24, and a timing diagram in Figure 4.25. The output of the divide-by-2 counter alternates between 0 and 1 for every pulse on its clock-enable input. The output thus has a 50% duty cycle, which will drive a speaker much more efficiently.

A digital alarm clock needs to generate a periodic signal at a frequency of approximately 500Hz to drive the speaker for the alarm tone. Use a counter to divide the system's master clock signal, with a frequency of
1 MHz, to derive the alarm tone.



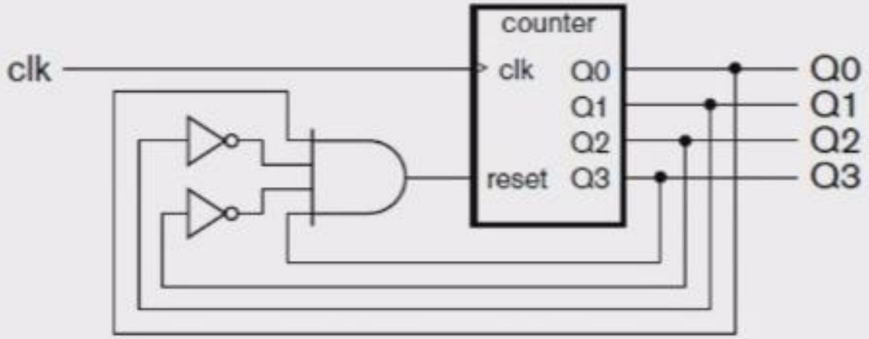
SOLUTION We need to divide the master clock signal by approximately 2000. We can use a divisor of $2^{11} = 2048$, which gives us an alarm tone frequency of 488Hz, which is close enough to 500 Hz. Thus, we could use the terminal-count output of an 11-bit counter for the tone signal. However, the duty cycle (the ratio of time for which the signal is 1 to the time for which it is 0) would only be 1/2048, which would have very low AC energy. We can rectify this by dividing the master clock by 2^{10} with a 10-bit counter, and using the terminal-count output as the count-enable input to a divide-by-2 counter. A circuit is shown in Figure 4.24, and a timing diagram in Figure 4.25. The output of the divide-by-2 counter alternates between 0 and 1 for every pulse on its clock-enable input. The output thus has a 50% duty cycle, which will drive a speaker much more efficiently.


10-bit counter
Q
clk
TC
count
tone2
clk
D
Q
CE
clk
tone


clk
count
tone2
tone
0
1
2
1023
0
1
2
1023
0
1
2
1023
0
1

Design a circuit for a modulo 10 counter, otherwise known as a *decade counter*.

SOLUTION The maximum count value is 9, so we need 4 bits for the counter. The unsigned binary code word for 9 is 1001_2. We can decode this value and use it to reset to counter to 0 on the next clock cycle. The circuit is shown in Figure 4.26.



MADHURA R