```
always @(posedge clk, negedge rst) begin
  if (~rst) begin
    cnt<=6'h0   ;
  end else begin
    cnt<=nxt_cnt;
  end
end
endmodule
```

```
module circuit6 (input             rst      ,
                 input             clk      ,
                 input             data_in  ,
                 output            data_out ,
                 output reg [4:0]  data_rev);

assign data_out = data_rev[4];
always @ (negedge rst, posedge clk) begin
  if (~rst) begin
    data_rev  <= 5'h0                      ;
  end else begin
    data_rev  <= {data_rev[3:0], data_in};
  end
end
endmodule
```

PBL 10: Four-Bit Full-Adder with Tri-State Buffer

Design a 4-bit adder that is connected to a tri-state buffer, as depicted in the block diagram shown in Figure 6.18. The input signals, "a" and "b", are both 4 bits wide, resulting in a 5-bit output signal, "c". To control the functionality of the tri-state buffer, an enable signal labeled as "en" is required. When the enable signal evaluates to FALSE (or binary zero), the output of the tri-state buffer will be in a high-impedance state, denoted as high-Z, indicating an open circuit. When the enable signal evaluates to TURE (or binary one), the output will be assigned as the adder's output.

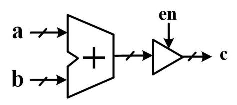


FIGURE 6.18
Block Diagram of 4-bit Adder with Tri-State Buffer

PBL 11: Four-Bit Full-Adder with Latch

Design a 4-bit adder that is connected to a latch, as depicted in the block diagram shown in Figure 6.19. The input signals, "a" and "b", are both 4 bits wide, resulting in a 5-bit output signal, "c". The latch requires two control signals: an enable signal, denoted as "en" to activate the latch, and an asynchronous reset signal, denoted as "rst" to reset the latch output to hexadecimal 5'h0.

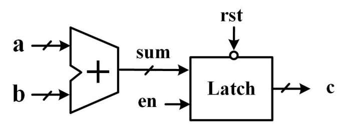


FIGURE 6.19
Block Diagram of 4-bit Adder with Latch

PBL 12: Four-Bit Full-Adder with Register

Design a 4-bit adder that is connected to a register, as depicted in the block diagram shown in Figure 6.20. The input signals, "a" and "b", are both 4 bits wide, resulting in a 5-bit output signal, "c". The register requires two control signals: a clock signal, denoted as "clk", and an asynchronous reset signal, denoted as "rst" to reset the register output to hexadecimal 5'h0.

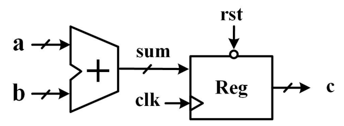
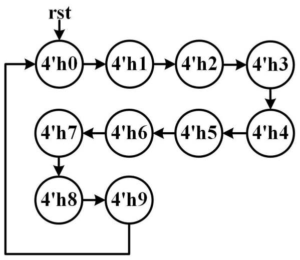


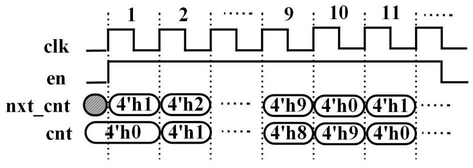
FIGURE 6.20
Block Diagram of 4-bit Adder with Register

PBL 13: Counter 0-9

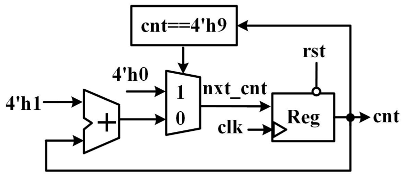
Design a counter that counts clock cycles from the hexadecimal value 4'h0 to 4'h9, as specified in the design specification shown in Figure 6.21(a). The counter has the following input/output connections, as summarized in Table 6.3. It operates based on an asynchronous reset and is triggered by the rising clock edge. When the counter reaches the maximum value of hexadecimal 4'h9, it restarts from 4'h0 in the next counting loop. The output of the counter is a 4-bit value, accommodating hexadecimal values ranging from 4'h0 to 4'h9.



(a) Design Specification of Counter for 10 Cycles.



(b) Timing Diagram of Counter for 10 Cycles.



(c) Block Diagram of Counter for 10 Cycles.

FIGURE 6.21
Design Specification of Counter for 10 Cycles

TABLE 6.3
Counter 0-9 IOs Description

Name	Direction	Bit Width	Description
clk	Input	1	Clock, rising edge trigger, 100 MHz
rst	Input	1	Asynchronous reset, 0 valid
en	Input	1	Enable signal, 1 valid
cnt	Output	4	Counter output

Figure 6.21(b) illustrates the timing diagram of the counter, showcasing its behavior from hexadecimal 4'h0 to 4'h9. To handle the initialization of the counter when it reaches hexadecimal 4'h9, the utilization of a comparator and a multiplexer is required.

Figure 6.21(c) presents a block diagram depicting these components. The comparator compares the counter output with the hexadecimal value 4'h9. When the counter reaches 4'h9, the comparator output evaluates to TRUE. This triggers the multiplexer to switch its channels. If the comparator output is TRUE, the multiplexer selects the input signal representing 4'h0, causing the counter to reset back to hexadecimal 4'h0. On the other hand, if the comparator output is FALSE, the multiplexer selects the input signal representing the current counter value plus hexadecimal 4'h1, incrementing the counter by one during each clock cycle.

PBL 14: Timer 0-9

Design a timer that counts from the hexadecimal value 4'h0 to 4'h9 with a unit of 0.1 microseconds (us). The design's input/output connections, summarized in Table 6.4, define the necessary interface for the timer design. The sequential circuit utilizes an asynchronous reset and is triggered by the rising clock edge.

Hint: It's essential to emphasize that with a clock frequency of 100 MHz, each clock cycle has a duration of 10 nanoseconds (ns). Therefore, to achieve a unit of 0.1 us, you would need to utilize 10 times the number of clock cycles.

TABLE 6.4
Timer 0-9 IOs Description

Name	**Direction**	**Bit Width**	**Description**
clk	Input	1	Clock, rising edge trigger, 100 MHz
rst	Input	1	Asynchronous reset, 0 valid
en	Input	1	Enable the timer from 0 to 9
tim	Output	4	The timer output

The design specification, presented in Figure 6.22(a), provides an overview of the design details for the timer. It illustrates a 2-level design structure for the timer implementation. The first-level counter is responsible for generating every 10-cycle unit, where each cycle corresponds to a duration of 0.01 us or 10 ns. The second-level timer counts the number of 10-cycle units and increments the "tim" output until it reaches the hexadecimal value 4'h9.

To gain a better understanding of the timer's behavior, a timing diagram in Figure 6.22(b) illustrates the output signal "tim" incrementing by hexadecimal value 4'h1 for every 10 clock cycles or when the counter reaches the value of 4'h9. After the 100th clock cycle, both the "cnt" and "tim" signals restart from the hexadecimal value 4'h0.

The block diagram in Figure 6.22(c) visually represents the components

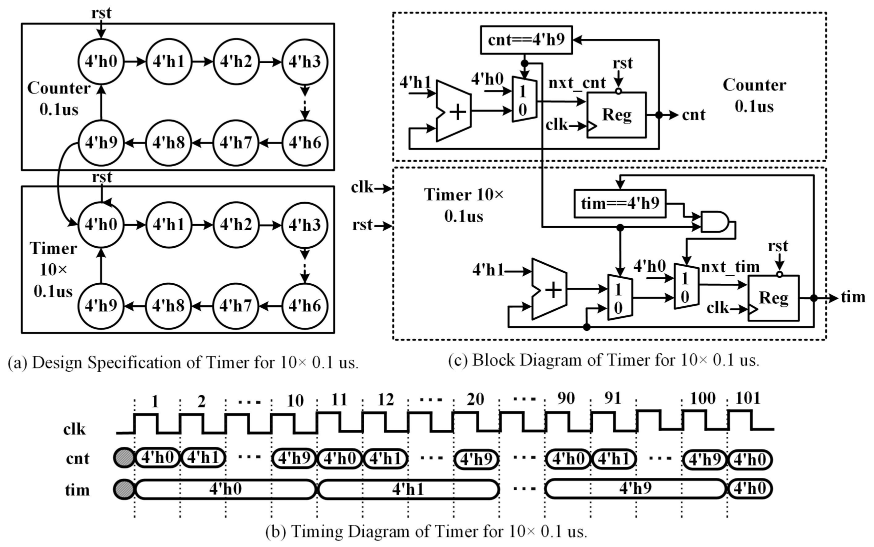


FIGURE 6.22
Design Specification of Timer for 10× 0.1 us

involved in the timer design. It showcases the counter and timer designs described earlier and emphasizes the logic necessary to reset both the counter and timer output. This reset condition is detected by an AND gate, which examines the outputs from the two comparators to ensure that both the counter and timer have reached the hexadecimal value 4'h9 before restarting the counting process.

PBL 15: Timing Interface

Design a timing interface as depicted in the timing diagram shown in Figure 6.23. The output signal, named "sig", is initially set to binary zero. It transitions to binary one within the time intervals of 0-0.3 us and 0.6-0.9 us. Between 0.4 us and 0.5 us, it resets back to binary zero. After 0.9 us, the signal is initialized to zero again. The IOs of the timing interface is summarized in Table 6.5.

According to the specification, the block diagram is depicted in Figure 6.24. The timer design remains the same as the design block shown in Figure 6.22. The output of the timer is compared with hexadecimal values 4'h3, 4'h4, and 4'h9. When the condition that the "tim" signal is "greater than 4'h4 AND less than 4'h9" is TRUE, the "sig" output is set to binary 1'b1. Another condition

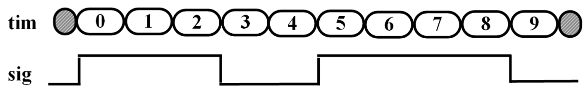


FIGURE 6.23
Timing Diagram of Timing Interface

TABLE 6.5
IOs Description of Timing Interface

Name	Direction	Bit Width	Description
clk	Input	1	Clock, 100 MHz
rst	Input	1	Asynchronous reset, 0 valid
en	Input	1	Enable signal
sig	Output	1	Signal output

to set the "sig" output as binary 1'b1 is when the "tim" signal is less than hexadecimal 4'h3. Therefore, an AND gate and an OR gate are required to implement the circuit for the "sig" output.

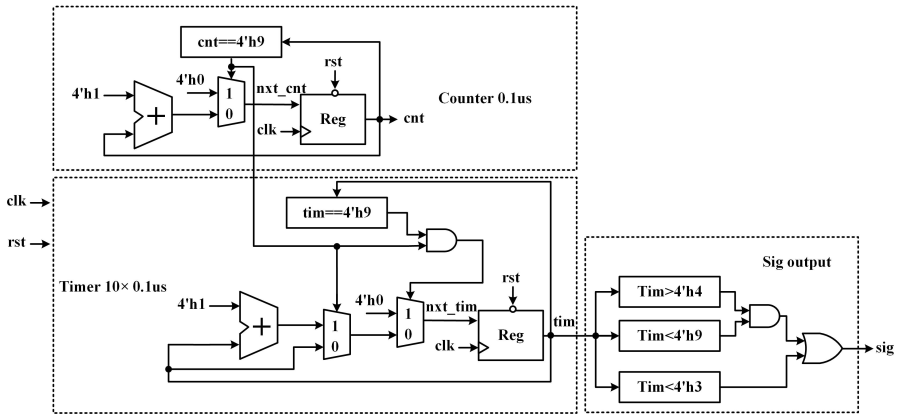


FIGURE 6.24
Block Diagram of Timing Interface

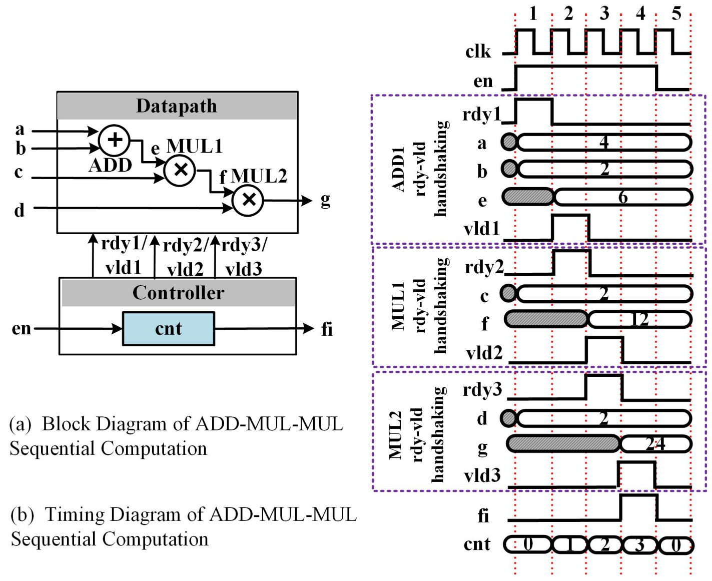
PBL 16: Timing Controller

Design a circuit composed of a datapath and a timing controller, as specified in the block diagram shown in Figure 6.25(a). The datapath consists of three

8-bit binary design components: an adder followed by two multipliers in series. Assuming that each binary component takes one clock cycle from its input to output, the final output “g” of the entire datapath can be produced after three clock cycles once the inputs “a-d” are provided.

To coordinate the binary operations within the datapath, a timing controller can be implemented using a counter. The enable input “en” activates the counter and the indicator output “fi” signals the completion of the data processing. All the inputs and outputs are summarized in Table 6.6.

Hint: The binary adder can be straightforwardly described in Verilog using the “+” operator, while the binary multiplier can be described using the “*” operator. In this case, we assume that there are no overflow situations for both the adder and multiplier designs. Between the ADD-MUL1-MUL2 components, three 8-bit registers are needed to isolate the datapath. These registers serve to store and propagate the intermediate values within the datapath, ensuring proper synchronization, timing, and speed for the data processing.



(a) Block Diagram of ADD-MUL-MUL Sequential Computation

(b) Timing Diagram of ADD-MUL-MUL Sequential Computation

FIGURE 6.25
Design Specification of Timing Controller

Between the datapath and the timing controller, the **ready-valid** mechanism is utilized to indicate the readiness of inputs and the validation of outputs for each component. Specifically, the signals “rdy1-vld1” are used for the adder (ADD), “rdy2-vld2” for the first multiplier (MUL1), and “rdy3-vld3” for the second multiplier (MUL2). These signals facilitate communication and

TABLE 6.6
IOs Description of Timing Controller

Name	Direction	Bit Width	Description
clk	Input	1	Clock, 100 MHz
rst	Input	1	Asynchronous reset, 0 valid
en	Input	1	Enable signal
a-d	Input	8	Input data
g	Output	8	Output data
fi	Output	1	Finish indicator

synchronization between the components, ensuring that inputs are available and outputs are valid on the bus interfaces.

Specifically, the ready-valid protocol is primarily discussed in Figure 6.25(b). In the first clock cycle, the inputs "a-b" are ready on the buses, indicated by the asserted signal "rdy1". In the subsequent clock cycle, the output "e" from the adder becomes valid on the bus, indicated by the asserted signal "vld1". Similarly, in the second clock cycle, the inputs "e" and "c" for the second multiplier are ready, indicated by "rdy2". Then, in the third clock cycle, the output "f" from the second multiplier becomes valid, indicated by "vld2". For the final data operation, the inputs "f" and "d" are ready in the third clock cycle, indicated by "rdy3". Finally, in the fourth clock cycle, the output "g" becomes valid, indicated by "vld3". It is important to note that the completion signal "fi" is asserted along with the "vld3" signal due to the last clock cycle for the entire data processing.

To follow the timing control, a 2-bit counter can be utilized. It progresses from decimal value 2'd0 to 2'd3 and then restarts to 2'b0. Once the data processing is enabled by the input "en", all ready-valid signals are generated in sequence, and the completion of the data processing is signaled by the output indicator "fi". The enable signal "en" should be set to a low state within the clock cycle when the "fi" output is asserted. The "enable-finish" handshaking guarantees the completion of data processing in the final clock cycle.