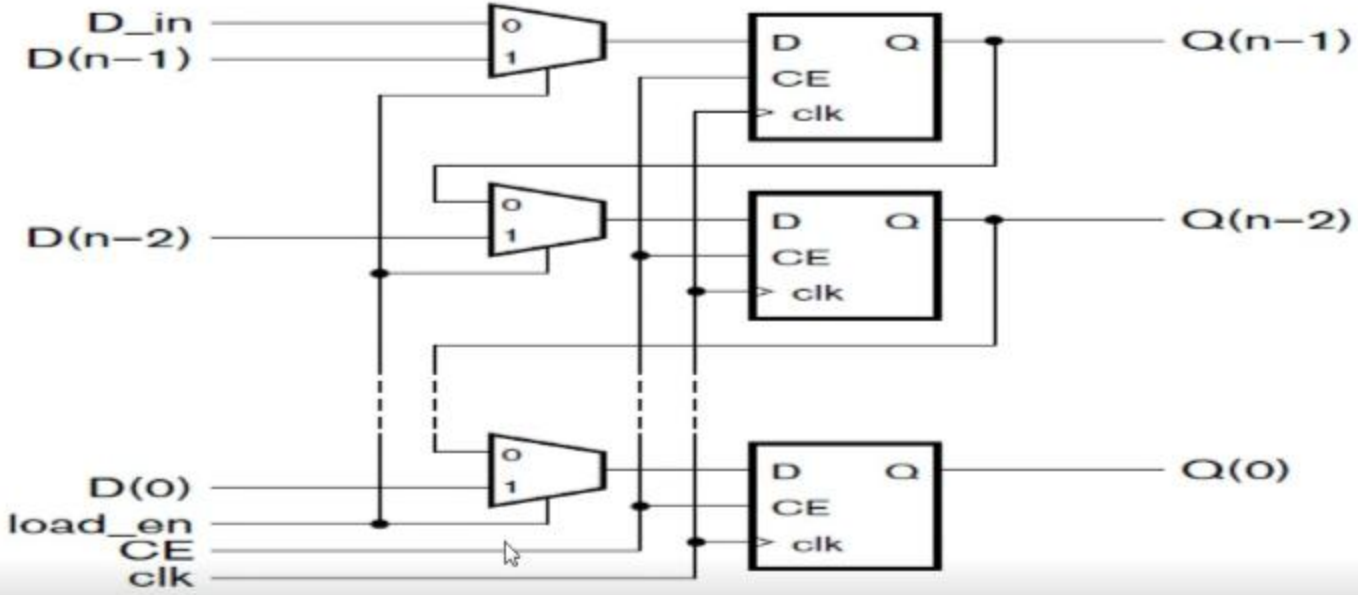
EXAMPLE 4.1 Develop a Verilog model for a pipelined circuit that computes the average of corresponding values in three streams of input values, a, b and c. The pipeline consists of three stages: the first stage sums values of a and b and saves the value of c; the second stage adds on the saved value of c; and the third stage divides by three. The inputs and output are all signed fixed-point numbers indexed from 5 down to −8.

```
module average_pipeline ( output reg signed [5:-8] avg,
                          input      signed [5:-8] a, b, c,
                          input                    clk );

  wire signed [5:-8] a_plus_b, sum, sum_div_3;
  reg  signed [5:-8] saved_a_plus_b, saved_c, saved_sum;

  assign a_plus_b = a + b;

  always @(posedge clk) begin  // Pipeline register 1
    saved_a_plus_b <= a_plus_b;
    saved_c        <= c;
  end

  assign sum = saved_a_plus_b + saved_c;

  always @(posedge clk)  // Pipeline register 2
    saved_sum <= sum;

  assign sum_div_3 = saved_sum * 14'b00000001010101;

  always @(posedge clk)  // Pipeline register 3
```

Shift Registers

- . A *shift register*, on the other hand, can perform a shift operation on the stored data.
- shift operation has the effect of scaling a numeric value by a power of 2.
- shift operations are also used to implement serial transfer of data, that is, transfer one bit at a time over a single wire, instead of using separate wires for each of the bits of data
- and combine arithmetic scaling with storage functions.

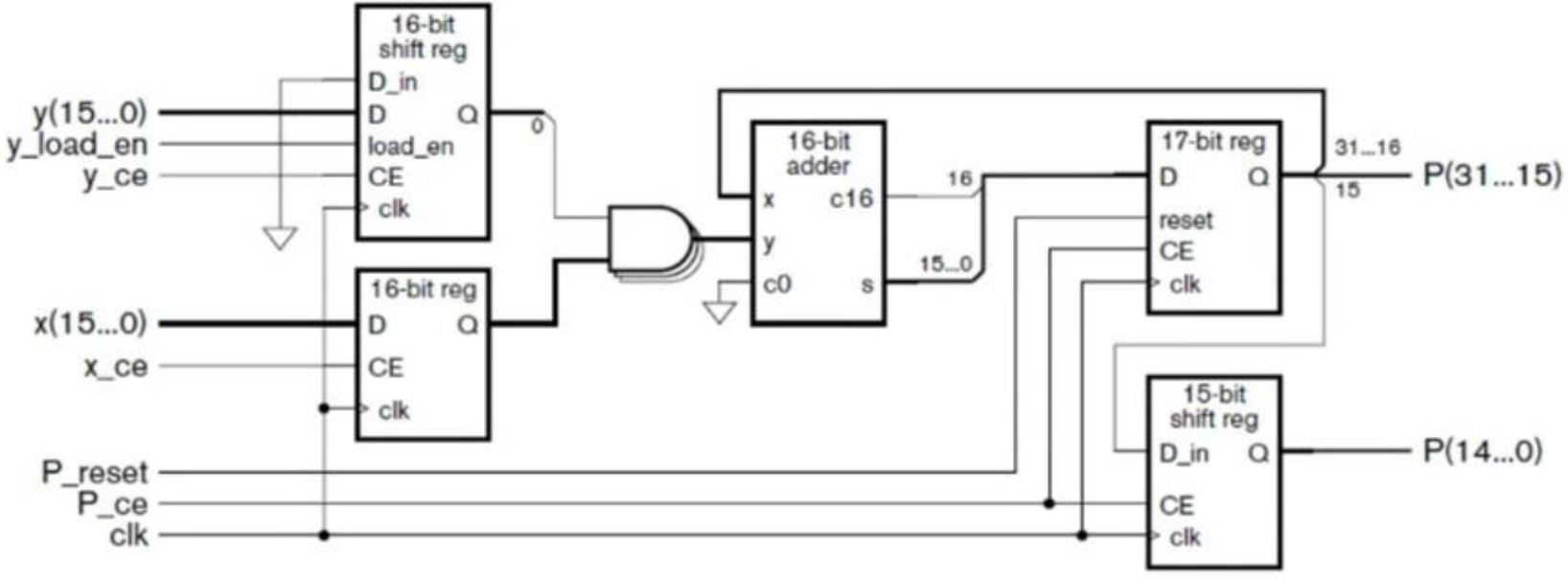

D_in
D
load_en
CE
clk
Q


D_in
D(n−1)
D(n−2)
D(0)
load_en
CE
clk
0
1
D
CE
clk
Q
Q(n−1)
Q(n−2)
Q(0)

- shows how it can be implemented with D flip-flops and multiplexers.
- The shift register is updated on a rising clock edge when CE is 1. In that case, when the load_en signal is 1, the multiplexers select new data on the D(n–1) through D(0) inputs for updating the register.
- Alternatively, when CE is 1 and load_en is 0, the multiplexers select the existing data, shifted right by one place. The least significant bit is discarded, and the most significant bit is updated with the value of the D_in signal.
- If we tie D_in to 0, the shift register performs a logical shift right operation on the stored data. Alternatively, if we connect the most significant output bit back to D_in, the shift register performs an arithmetic shift right operation.

- shows how it can be implemented with D flip-flops and multiplexers.
- The shift register is updated on a rising clock edge when CE is 1. In that case, when the load_en signal is 1, the multiplexers select new data on the D(n–1) through D(0) inputs for updating the register.
- Alternatively, when CE is 1 and load_en is 0, the multiplexers select the existing data, shifted right by one place. The least significant bit is discarded, and the most significant bit is updated with the value of the D_in signal.
- If we tie D_in to 0, the shift register performs a logical shift right operation on the stored data. Alternatively, if we connect the most significant output bit back to D_in, the shift register performs an arithmetic shift right operation.

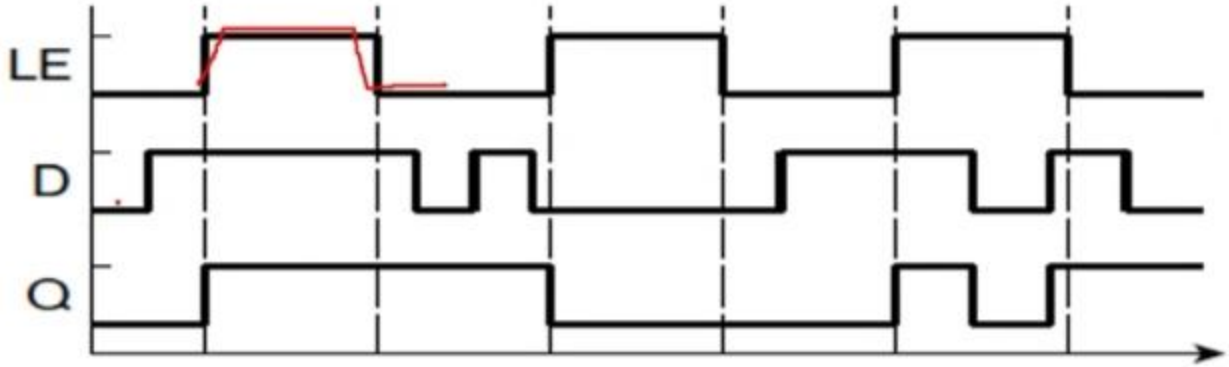
How to perform multiplication of unsigned integers by addition of partial products. Construct a multiplier for two 16-bit operands containing just one adder that adds successive partial products over successive clock cycles. The final product is 32 bits.



- SOLUTION In order to perform the operation over multiple cycles, we need a number of registers to hold intermediate results, as shown in Figure
- The x operand is stored in an ordinary register whose output connects to an array of 16 AND gates that form a partial product. The y operand is stored in a shift register whose least significant bit, Q(0), controls the AND gates. The y operandis shifted on successive cycles, thus giving the 16 successive partial products. The sum of the partial products are accumulated in a 17-bit ordinary register and a 15-bit shift register. Since the shift register is never required to load data other than through the D_in connection, the data and load_en inputs are absent. On each clock cycle, the least significant bit of the ordinary register is shifted into the shift register, and the remaining bits of the ordinary register are added with the next partial product. By shifting the accumulated sum in this way, partial products are added at successively more significant positions of the result.

L ATC H E S

- As we have seen, a flip-flop is a basic sequential circuit element that stores one bit. Most digital circuits use edge-triggered flip-flops that store a new data value when the clock signal changes from 0 to 1. No further values are stored while the clock remains at 1, nor when the clock returns to 0.
- Some systems, however, use sequential elements called latches, with slightly different timing for storage of values. Figure shows a symbol for a latch, and Figure shows the timing behavior


LE
D
Q

EXAMPLE 4.5 The following always block is intended to model multi- plexer circuitry that selects between a number of inputs to assign to outputs z1 and z2. Identify the error in the block and describe the behavior that results.

```
always @*
if (~sel) begin
z1 <= a1; z2 <= b1;
End
 else begin
z1 <= a2; z3 <= b2;
end
```

SOLUTION The assignment to z3 in the "else" part of the if statement should assign to z2. As a consequence, z2 is not updated on that execution path and z3 is not updated on the execution path in which sel is 0. Thus, the block implies transparent latches for z2 and z3. The latch for z2 is transparent when sel is 0 and stores a value when sel is 1. The latch for z3 is transparent when sel is 1 and stores a value when sel is 0. This unintended behavior can be corrected simply by changing the target of the assignment from z3 to z2, as it should be.