4.2 Trace the behavior of an 8-bit parallel-load register with 8-bit input I, 8-bit output Q, load control input `ld`, and synchronous clear input `clr` by completing the timing diagram in Figure 4.96.

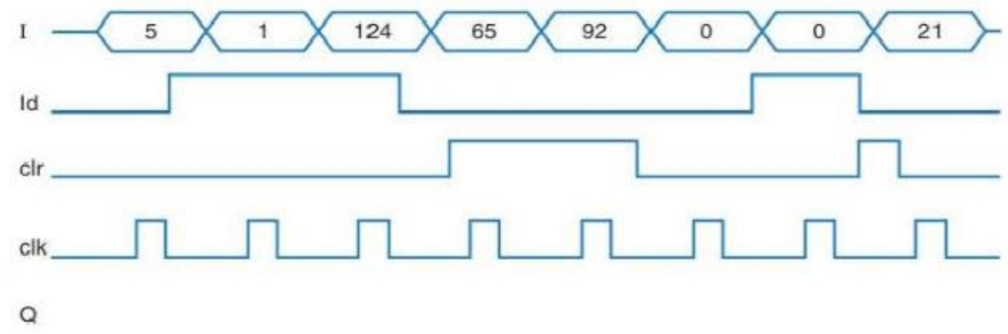

Figure 4.96 Timing diagram.

4.3 Design a 4-bit register with 2 control inputs `s1` and `s0`; 4 data inputs `I3`, `I2`, `I1`, and `I0`; and 4 data outputs `Q3`, `Q2`, `Q1`, and `Q0`. When `s1s0=00`, the register maintains its value. When `s1s0=01`, the register loads `I3`...`I0`. When `s1s0=10`, the register clears itself to `0000`. When `s1s0=11`, the register complements itself, so for example, `0000` would become `1111`, and `1010` would become `0101`. (*Component design problem.*)

4.10 Assuming AND gates have a delay of 2 ns, OR gates have a delay of 1 ns, and XOR gates have a delay of 3 ns, compute the longest time required to add two numbers using an 8-bit carry-ripple adder.

4.13 Design an adder that computes the sum of four 8-bit numbers, using 8-bit carry-ripple adders. *(Component use problem.)*

4.21 Use magnitude comparators and logic to design a circuit that computes the maximum of two 16-bit numbers. *(Component use problem.)*

4.30 Convert the following two's complement binary numbers to decimal numbers:

(a) 11100000
(b) 01111111
(c) 11110000
(d) 11000000
(e) 11100000

4.33 Convert the following decimal numbers to 8-bit two's complement binary form:

(a) 29
(b) 100
(c) 125
(d) –29
(e) –100
(f) –125
(g) –2

4.40 Design an ALU with two 8-bit inputs A and B, and control inputs x, y, and z. The ALU should support the operations described in Table 4.3. Use an 8-bit adder and an arithmetic/logic extender. *(Component design problem.)*

TABLE 4.3 Desired ALU operations.

Inputs			Operation
x	y	z	
0	0	0	S = A – B
0	0	1	S = A + B
0	1	0	S = A * 8
0	1	1	S = A / 8
1	0	0	S = A NAND B (bitwise NAND)
1	0	1	S = A XOR B (bitwise XOR)
1	1	0	S = Reverse A (bit reversal)
1	1	1	S = NOT A (bitwise complement)

4.62 Design a 4x4 three-port (2 read, 1 write) register file. *(Component design problem.)*

4.64 A 4x4 register file's four registers initially each contain 0101.

(a) Show the input values necessary to read register 3 and to simultaneously write register 3 with the value 1110.

(b) With these values, show the register file's register values and output values before the next rising clock edge, and after the next rising clock edge.