7.2 (Registers) Design a register with two load signals, that enable the loading of data from two different sources.

<table>
<thead>
<tr>
<th>Load A</th>
<th>Load B</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Load A</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Load B</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>unused</td>
</tr>
</tbody>
</table>

7.3 (Registers) Explain the difference between resetting the register and loading an all-0’s input into the register.

Resetting the register is an asynchronous operation, while loading 0’s into it is synchronous.
7.5 (Shift-Registers) Using a 4-bit shift register, construct a 4-bit register that can rotate its content one position to the left or right.

By connecting the left-most and right-most output lines to the $IR$ and $IL$ input lines, respectively, we implement the left/right rotator without any additional logic.

Logic schematic
7.8 (Counters) Design a binary counter that counts up only in:

(a) even numbers (0, 2, 4, 6, 8, \ldots)
(b) odd numbers (1, 3, 5, 7, 9, \ldots)

\[ \begin{array}{cccc|c}
Q_3 & Q_2 & Q_1 & Q_0 & \text{Next-state} \\
\hline
0 & 0 & 0 & 0 & 0 0 1 0 \\
0 & 0 & 1 & 0 & 0 1 0 0 \\
0 & 1 & 0 & 0 & 0 1 1 0 \\
1 & 1 & 1 & 0 & 1 0 0 0 \\
1 & 0 & 0 & 0 & 1 0 1 0 \\
1 & 0 & 1 & 0 & 1 1 0 0 \\
1 & 1 & 0 & 0 & 1 1 1 0 \\
1 & 1 & 1 & 0 & 0 0 0 0 \\
\end{array} \]

We obtain the equations:

\[ D_3 = Q_3Q_1' + Q_3Q_2' + Q_3'Q_2Q_1 \]
\[ D_2 = Q_2 Q'_1 + Q'_2 Q_1 \]
\[ D_1 = Q'_1 \]
\[ D_0 = 0 \]
7.13 (Asynchronous counters) Construct a 4-bit asynchronous counter using

(a) D flip-flops
(b) J-K flip-flops

(a) D-type flip-flop implementation
(b) JK-type flip-flop implementation
7.15 (Register files) Design an $8 \times 4$ register file with

(a) 1 write and 2 read ports
(b) 2 write and 1 read ports
(c) 2 write and 2 read ports
Register-file cell

(a) 1 write and 2 read ports
7.16 (Memories) Design

(a) $256K \times 8$ RAM using $256K \times 1$ RAM chips
(b) $64K \times 32$ RAM using $64K \times 8$ RAM chips
(c) $1M \times 1$ RAM using $256K \times 1$ RAM chips
(d) $256K \times 8$ RAM using $64K \times 8$ RAM chips
7.19 (Datapaths) Design a simple datapath that can compute the expression:

(a) $\sum_{i=1}^{n} a_i x_i$
(b) $\sum_{i=1}^{n} a_i x_i + b_i$
(c) $\sum_{i=1}^{n} x_i^2 + x_i + c_i$
(b) $\sum_{i=1}^{n} a_i x_i + b_i$
7.23 (Datapaths) Define and implement the controller for the datapath that could execute the algorithm developed in Problem 7.22.

From the state diagram and table, the next-state equations are:

\[
\begin{align*}
D_2 &= Q_1Q_0 + Q_2Q_1' \\
D_1 &= Q_2Q_1' + Q_1Q_0' + Q_2Q_1Q_0' \\
D_0 &= Q_2'Q_0' + \text{Carry}'Q_2Q_1
\end{align*}
\]

The control table would then consist of:

| \(Q_2Q_1Q_0\) | \(I_E\) | \(WA_2WA_1WA_0WE\) | \(RAA_2RAA_1RAA_0RAE\) | \(RAB_2RAB_1RAB_0RAE\) | \(MS_1S_0\) | \(OE\) |
|----------------|
| 000 | 0 | ******0 | ******0 | ******0 | ******0 | 0 |
| 001 | 1 | 0011 | ******0 | ******0 | ******0 | 0 |
| 010 | 1 | 0101 | ******0 | ******0 | ******0 | 0 |
| 011 | 0 | ******0 | 0111 | 0101 | 101 | 1 |
| 100 | 0 | 0111 | ******0 | ******0 | ******0 | 101 | 0 |
| 101 | 0 | 0111 | ******0 | ******0 | ******0 | 101 | 0 |
| 110 | 0 | ******0 | 0111 | ******0 | ******0 | 101 | 1 |
\[
\begin{align*}
IE &= Q_2'Q_1Q_0' + Q_2'Q_1'Q_0 \\
WA_2 &= 0 \\
WA_1 &= Q_2 + Q_1 \\
WA_0 &= Q_2 + Q_1' + Q_0 \\
WE &= Q_2'(Q_1'Q_0 + Q_1Q_0') + Q_2Q_1' \\
RAA_2 &= 0 \\
RAA_1 &= Q_2 \\
RAA_0 &= 1 \\
REA &= Q_2'Q_1Q_0' + Q_2Q_1Q_0' \\
RAB_2 &= 0 \\
RAB_1 &= 1 \\
RAB_0 &= 0 \\
REB &= Q_2'Q_1Q_0 \\
M &= 1 \\
S_1 &= Q_2Q_1Q_0' \\
S_0 &= 1 \\
\text{Shifter} &= 0000 \\
OE &= Q_2'Q_1Q_0 + Q_2Q_1Q_0'
\end{align*}
\]