sum of 1000 (8) and a CARRY of 0 into the MSD adder. The MSD adder will add 0010 (2) and 0101 (5) for a sum of 0111 (7) and no CARRY OUT. Thus, at the sum outputs we have

$$[\Sigma] = 0111 \quad 1000 \quad 0101$$

and there is a CARRY output of 0 from the MSD adder. This result is, of course, the BCD representation of the decimal sum 785_{10}.

Review Questions

1. What are the three basic parts of a BCD adder circuit?
2. Describe how the BCD adder circuit detects the need for a correction and executes it.

6-17 ALU INTEGRATED CIRCUITS

There are several integrated circuits available that are called arithmetic/logic units (ALUs) even though they do not have the full capabilities of a computer's arithmetic/logic unit. These ALU chips are capable of performing several different arithmetic and logic operations on binary data inputs. The specific operation that an ALU IC is to perform is determined by a specific binary code applied to its function-select inputs. Some of the ALU ICs are fairly complex, and it would require a great amount of time and space to explain and illustrate their operation. In this section we will use a relatively simple, yet useful, ALU chip to show the basic concepts behind all ALU chips. The ideas presented here can then be extended to the more complex devices.

The 74LS382/HC382 ALU

Figure 6-16(a) shows the block symbol for an ALU that is available as a 74LS382 (TTL) and as a 74HC382 (CMOS). This 20-pin IC operates on two four-bit input numbers, $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$, to produce a four-bit output result $F_3F_2F_1F_0$. This ALU can perform *eight* different operations. At any given time, the operation that it is performing depends on the input code applied to the function-select inputs $S_2S_1S_0$. The table in Figure 6-16(b) shows the eight available operations. We will now describe each of these operations.

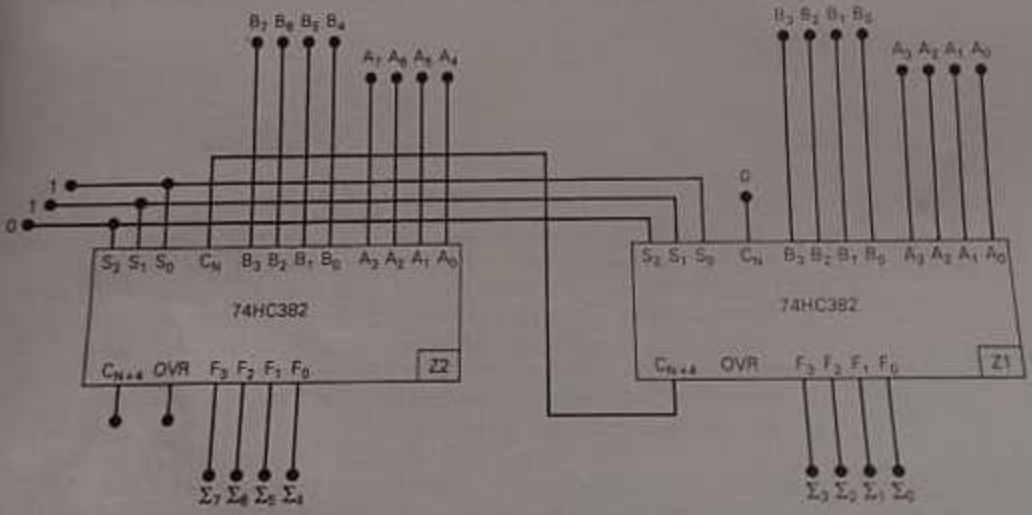
CLEAR OPERATION With $S_2S_1S_0 = 000$, the ALU will *clear* all of the bits of the F output so that $F_3F_2F_1F_0 = 0000$.

ADD OPERATION With $S_2S_1S_0 = 011$, the ALU will add $A_3A_2A_1A_0$ to $B_3B_2B_1B_0$ to produce their sum at $F_3F_2F_1F_0$. For this operation, C_N is the carry into the LSB position, and it must be made a 0. C_{N+4} is the carry output from the MSB position. OVR is the overflow indicator output; it detects overflow when signed numbers are being used. OVR will be a 1 when an add or a subtract operation produces a result that is too large to fit into four bits (including the sign bit).

2. The sum appears at the F outputs of Z1 and Z2. The lower-order bits appear at Z1, and the higher-order bits at Z2.
3. The C_N input of Z1 is the carry into the LSB position. For addition, it is made a 0.
4. The carry output $[C_{N+4}]$ of Z1 is connected to the carry input $[C_N]$ of Z2.
5. The *OVR* output of Z2 is the overflow indicator when signed eight-bit numbers are being used.
6. The corresponding select inputs of the two chips are connected together so that Z1 and Z2 are always performing the same operation. For addition, the select inputs are shown as 011.

EXAMPLE 6-13

How would the arrangement of Figure 6-17 have to be changed in order to perform the subtraction $(B - A)$?



Notes: Z1 adds lower-order bits.
Z2 adds higher-order bits.
Σ_7-Σ_0 = 8-bit sum.
OVR of Z2 is 8-bit overflow indicator.

RE 6-17 Two 74HC382 ALU chips connected as an eight-bit adder.

Solution

The select input code [see table in Figure 6-16(b)] must be changed to 001, and the C_N input of Z1 must be made a 1.

Other ALUs

The 74LS181/HC181 is another four-bit ALU. It has four select inputs which can select any of 16 different operations. It also has a mode input bit that can switch between logic operations and arithmetic operations (add and subtract). This ALU has an $A = B$ output that is used to compare the magnitudes of the A and B inputs. When the two input numbers are exactly equal, the $A = B$ output will be a 1; otherwise, it is a 0.

The 74LS881/HC881 is similar to the 181 chip, but it has the capability of performing some additional logic operations.

Review Questions

1. Apply the following inputs to the ALU of Figure 6-16, and determine the outputs: $S_2S_1S_0 = 001$, $A_3A_2A_1A_0 = 1110$, $B_3B_2B_1B_0 = 1001$, $C_N = 1$.
2. Change the select code to 011 and C_N to 0, and repeat question 1.
3. Change the select code to 110, and repeat question 1.
4. Apply the following inputs to the circuit of Figure 6-17, and determine the outputs: $B = 01010011$, $A = 00011000$.
5. Change the select code to 111, and repeat question 4.
6. How many 74HC382s are needed to add two 32-bit numbers?

6-18 IEEE/ANSI SYMBOLS

Figure 6-18(a) shows the IEEE/ANSI symbol for a one-bit adder (full adder). Note that the symbol Σ is used to indicate the addition operation. Figure 6-18(b) is the IEEE/ANSI symbol for the 7483/74283 four-bit parallel adder. Note that the letters P and Q are used to represent the two four-bit inputs, and Σ is used for the four-bit output sum. The P, Q, and Σ are specified by the IEEE/ANSI standard and must be

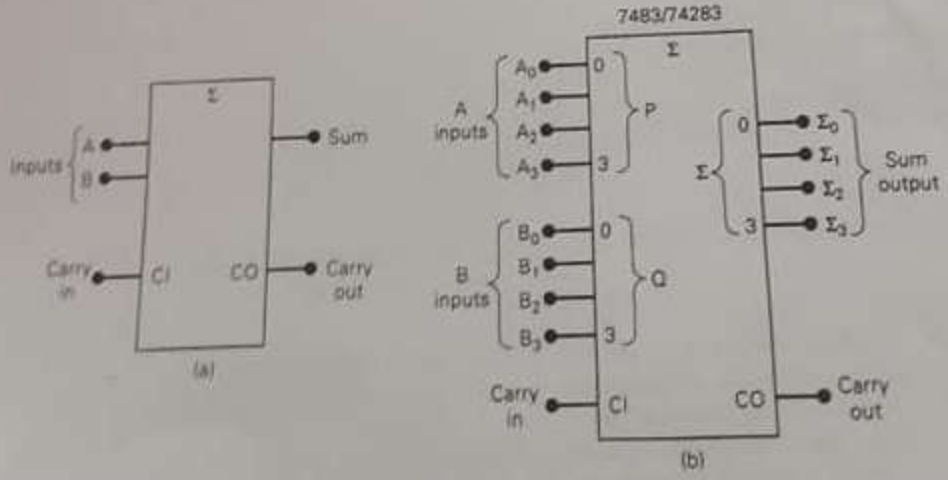


FIGURE 6-18 IEEE/ANSI symbols for (a) a full adder and (b) a four-bit parallel adder IC (7483/74283).

AND OPERATION With $S_2S_1S_0 = 110$, the ALU will perform a bit-by-bit AND operation on the A and B inputs. For example, with $A_3A_2A_1A_0 = 0110$ and $B_3B_2B_1B_0 = 1100$, the ALU will generate a result of $F_3F_2F_1F_0 = 0100$.

PRESET OPERATIONS With $S_2S_1S_0 = 111$, the ALU will set all of the bits of the output so that $F_3F_2F_1F_0 = 1111$.

EXAMPLE 6-12

(a) Determine the 74HC382 outputs for the following inputs: $S_2S_1S_0 = 010$, $A_3A_2A_1A_0 = 0100$, $B_3B_2B_1B_0 = 0001$, and $C_{in} = 1$.

(b) Change the select code to 011 and repeat.

Solution

(a) From the function table in Figure 6-16(b), 010 selects the $(A - B)$ operation. The ALU will perform the 2's-complement subtraction by complementing B and adding it to A and C_{in}. Note that $C_{in} = 1$ is needed to complete the 2's complement of B effectively.

$$\begin{array}{rl} & 1 \leftarrow C_{in} \\ & 0100 \leftarrow A \\ + & \underline{1110} \leftarrow \overline{B} \\ & 10011 \\ C_{in+4} & \uparrow \quad \uparrow\text{—}F_3F_2F_1F_0 \end{array}$$

As always in 2's-complement subtraction, the CARRY OUT of the MSB is discarded. The correct result of the $(A - B)$ operation appears at the F outputs.

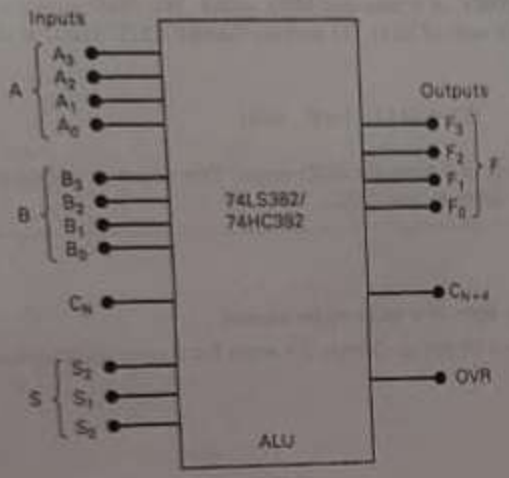
The OVR output is determined by considering the input numbers to be signed numbers. Thus, we have $A_3A_2A_1A_0 = 0100 = +4_{10}$ and $B_3B_2B_1B_0 = 0001 = +1_{10}$. The result of the subtract operation is $F_3F_2F_1F_0 = 0011 = +3_{10}$, which is correct. Therefore, no overflow has occurred, and $OVR = 0$. If the result had been negative, it would have been in 2's-complement form.

(b) A select code of 011 will produce the sum of the A and B inputs. However, since $C_{in} = 1$, there will be a carry of 1 added into the LSB position. This will produce a result of $F_3F_2F_1F_0 = 0110$, which is 1 greater than $(A + B)$. The C_{in+4} and OVR outputs will both be 0. For the correct sum to appear at F, the C_{in} input must be at 0.

Expanding the ALU

A single 74LS382 or 74HC382 operates on four-bit numbers. Two or more of these chips can be connected together to operate on larger numbers. Figure 6-17 shows how two four-bit ALUs can be combined to add two eight-bit numbers, $B_7B_6B_5B_4B_3B_2B_1B_0$ and $A_7A_6A_5A_4A_3A_2A_1A_0$, to produce the output sum $\Sigma_7\Sigma_6\Sigma_5\Sigma_4\Sigma_3\Sigma_2\Sigma_1\Sigma_0$. Study the circuit diagram and note the following points:

1. Chip Z1 operates on the four lower-order bits of the two input numbers. Chip Z2 operates on the four higher-order bits.



A = 4-bit input number
B = 4-bit input number
C_N = carry into LSB position
S = 3-bit operation select inputs
F = 4-bit output number
C_{N+4} = carry out of MSB position
OVR = overflow indicator

(a)

Function Table

S_2	S_1	S_0	Operation	Comments
0	0	0	CLEAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B minus A	Needs $C_N = 1$
0	1	0	A minus B	Needs $C_N = 1$
0	1	1	A plus B	Needs $C_N = 0$
1	0	0	A ⊕ B	Exclusive-OR
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

Notes: S inputs select operation
OVR = 1 for signed-number overflow

(b)

FIGURE 6-16 (a) Block symbol for 74LS382/HC382 ALU chip; (b) function table showing how select inputs (S) determine what operation is to be performed on A and B inputs.

SUBTRACT OPERATIONS With $S_2S_1S_0 = 001$, the ALU will subtract the A input number from the B input number. With $S_2S_1S_0 = 010$, the ALU will subtract B from A. In either case the difference appears at $F_3F_2F_1F_0$. Note that the subtract operations require that the C_N input be a 1.

XOR OPERATION With $S_2S_1S_0 = 100$, the ALU will perform a bit-by-bit XOR operation on the A and B inputs. This is illustrated below for $A_3A_2A_1A_0 = 0110$ and $B_3B_2B_1B_0 = 1100$.

$$A_3 \oplus B_3 = 0 \oplus 1 = 1 = F_3$$
$$A_2 \oplus B_2 = 1 \oplus 1 = 0 = F_2$$
$$A_1 \oplus B_1 = 1 \oplus 0 = 1 = F_1$$
$$A_0 \oplus B_0 = 0 \oplus 0 = 0 = F_0$$

The result is $F_3F_2F_1F_0 = 1010$.

OR OPERATION With $S_2S_1S_0 = 101$, the ALU will perform a bit-by-bit OR operation on the A and B inputs. For example, with $A_3A_2A_1A_0 = 0110$ and $B_3B_2B_1B_0 = 1100$, the ALU will generate a result of $F_3F_2F_1F_0 = 1110$.