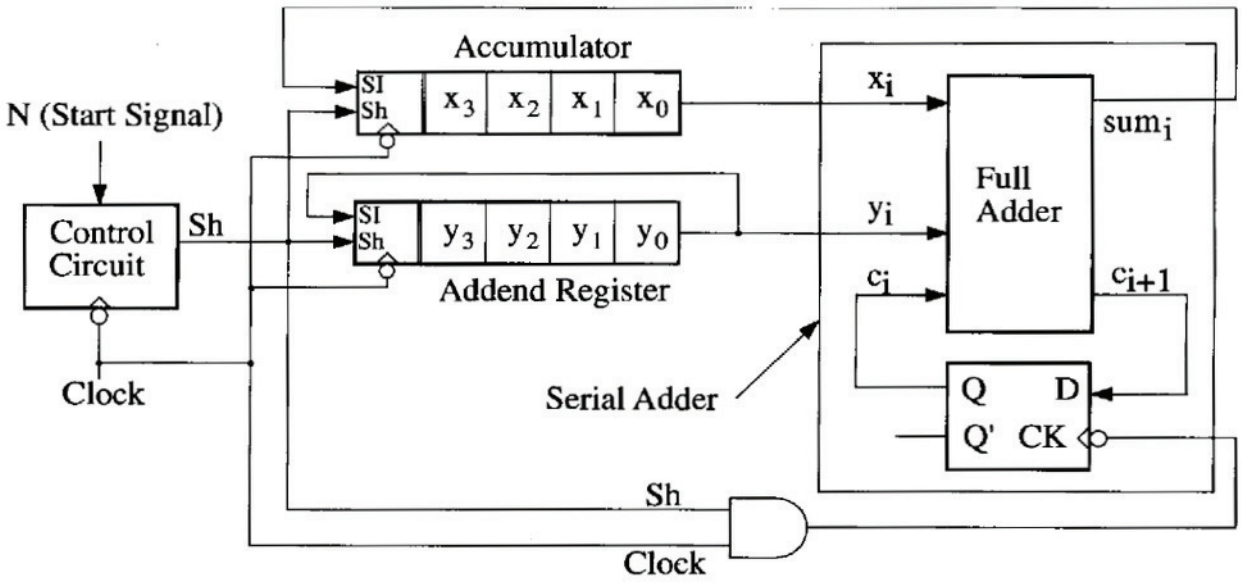
This is an example of a serial adder from a book *

DESIGN OF A SERIAL ADDER WITH ACCUMULATOR

A control circuit for a digital system is a sequential network that outputs a sequence of control signals. These signals cause operations such as addition and shifting to take place at appropriate times. In this section, we illustrate the design of a control circuit for a serial adder with accumulator. Figure 4-1 shows the block diagram for the adder. Two shift registers are used to hold the 4-bit numbers to be added, X and Y. The box at the left end of each shift register shows the inputs: *Sh* (shift), *SI* (serial input), and *Clock*. When $Sh = 1$ and the clock is pulsed, *SI* is entered into x_3 (or y_3) as the contents of the register are shifted right one position. The X-register serves as the accumulator, and after four shifts, the number X is replaced with the sum of X and Y. The addend register is connected as a cyclic shift register, so after four shifts it is back to its original state and the number Y is not lost. The serial adder consists of a full adder and a carry flip-flop. At each clock time, one pair of bits is added. When $Sh = 1$, the falling edge of the clock shifts the sum bit into the

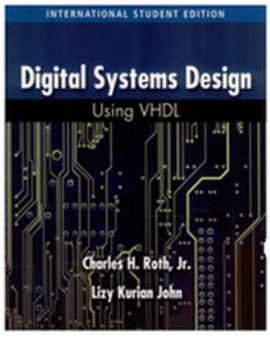
Figure 4-1 Serial Adder with Accumulator



*

Digital System Design using VHDL (2nd edition)

Charles H. Roth, Jr., Lizy Kurian John, **Thomson**, International Student Edition, 2008, ISBN 10: 0-495-24470-8, ISBN 13: 978-0-495-24470-7).



accumulator, stores the carry bit in the carry flip-flop, and causes the addend register to rotate right. Additional connections needed for initially loading the X and Y registers and clearing the carry flip-flop are not shown in the block diagram.

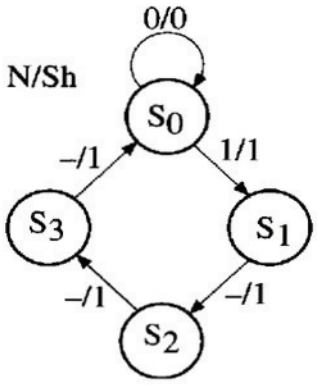
Table 4-1 illustrates the operation of the serial adder. In this table, t_0 is the time before the first clock, t_1 is the time after the first clock, t_2 is the time after the second clock, etc. Initially, at time t_0 the accumulator contains X, the addend register contains Y, and the carry flip-flop is clear. Since the full adder is a combinational network, $x_0 = 1$, $y_0 = 1$, and $c_0 = 0$ are added after a short propagation delay to give 10, so $sum_0 = 0$ and carry $c_1 = 1$. When the first clock occurs, sum_0 is shifted into the accumulator, and the remaining accumulator digits are shifted right one position. The same shift pulse stores c_1 in the carry flip-flop and cycles the addend register right one position. The next pair of bits, $x_1 = 0$ and $y_1 = 1$, are now at the full adder input, and the adder generates the sum and carry, $sum_1 = 0$ and $c_2 = 1$. The second clock pulse shifts sum_1 into the accumulator, stores c_2 in the carry flip-flop and cycles the addend register right. Bits x_2 and y_2 are now at the adder input, and the process continues until all bit pairs have been added. After four clocks (time t_4), the sum of X and Y is in the accumulator, and the addend register is back to its original state.

Table 4-1 Operation of Serial Adder

	X	Y	c_i	sum_i	c_{i+1}
t_0	0101	0111	0	0	1
t_1	0010	1011	1	0	1
t_2	0001	1101	1	1	1
t_3	1000	1110	1	1	0
t_4	1100	0111	0	(1)	(0)

The control circuit for the adder must now be designed so that after receiving a start signal, the control circuit will output $Sh = 1$ for four clocks and then stop. Figure 4-2 shows the state graph and table for the control circuit. The network remains in S_0 until a start signal (N) is received, at which time the network outputs $Sh = 1$ and goes to S_1. Then $Sh = 1$ for three more clock times, and the network returns to S_0. It will be assumed that the start signal is terminated before the network returns to state S_0, so no further action occurs until another start signal is received. Dashes (don't cares) on the graph indicate that once S_1 is reached, the network operation continues regardless of the value of N.

Figure 4-2 Control State Graph and Table for Serial Adder



Present State	Next State N=0	Next State N=1	Present Output (Sh) N=0	Present Output (Sh) N=1
S_0	S_0	S_1	0	1
S_1	S_2	S_2	1	1
S_2	S_3	S_3	1	1
S_3	S_0	S_0	1	1

So, take this as an example design because we are going to adapt it to our "know-how"