```
// synchronous reset

always @(posedge CLK)
begin
  if (RESET) // RESET evaluated only at CLK rising edge
    Q <= 1'b0;
  else
    Q <= D;
end

// asynchronous reset

always @(posedge CLK or posedge RESET)
begin
  if (RESET) // RESET evaluated whenever it goes active
    Q <= 1'b0;
  else
    Q <= D;
end
```

FIGURE 10.3 Verilog RTL flip-flop inference.

the non-blocking assignment does not take effect until after the current simulation time unit. This is analogous to the behavior of a real flop wherein the output does not transition until a finite time has elapsed from its triggering event. Under certain circumstances, either type of assignment will yield the same result in both simulation and synthesis. In other situations, the results will differ, as illustrated in Fig. 10.4.

In the first case, regs Q1 and Q2 are tracked at two different instants in time. First, their current states are maintained as they were just prior to the clock edge for the purpose of using their values in subsequent assignments. Second, their new states are assigned as dictated by the RTL. When Q2 is assigned, it takes the previous value of Q1, not the new value of Q1, which is D. Two flops are inferred.

In the second case, variables Q1 and Q2 are tracked at a single instant in time. Q1 is assigned the value of variable D, and then Q2 is assigned the new value of variable Q1. Q1 has become a temporary placeholder and has no real effect on its own. Therefore, only a single flop, Q2, is inferred.

Utilizing HDL to design logic requires software tools more complex than just pencil and paper. However, the benefits quickly accumulate for designs of even moderate complexity. The digital

```
// Non-blocking assignments: two flops inferred

always @(posedge CLK)
begin
  Q1 <= D;
  Q2 <= Q1;
end

// Blocking assignments: one flop inferred

always @(posedge CLK)
begin
  Q1 = D;
  Q2 = Q1;
end
```

FIGURE 10.4 Verilog blocking vs. non-blocking assignment.

functions and techniques discussed in the remainder of this chapter show how practical HDL design can be.

10.2 CPU SUPPORT LOGIC

Most digital systems require some quantity of miscellaneous glue logic to help tie a CPU to its memory and I/O peripherals. Some of the most common support functions are address decoding, basic I/O signals, interrupt control, and timers. Another common function is interface conversion whereby the CPU needs to talk with a peripheral that has an interface that is incompatible with that of the CPU. Interface conversion can range from simple control signal polarity adjustments to complex buffering schemes that cross clock domains with FIFOs.

Address decoding is usually a combinatorial implementation, because many CPU interfaces are nonpipelined. When performing address decoding and other bus control functions for a pipelined CPU bus, a more complex synchronous circuit is called for that can track the various pipeline stages and take the necessary actions during each stage. Basic combinatorial address decoding consists of mapping ranges of addresses to chip selects. Chip select signals are usually active-low by convention and are numbered upward from 0. For the sake of discussion, consider the 24-bit memory map in Table 10.1 to design an address decoder.

TABLE 10.1 Example Memory Map

Address Range	Qualifier	Chip Select	Function
0x000000–0x0FFFFF	RomSel=0	CS0_	1-MB default boot ROM
0x100000–0x1FFFFF	RomSel=1		
0x100000–0x1FFFFF	RomSel=0	CS1_	1-MB ROM module
0x000000–0x0FFFFF	RomSel=1		
0x200000–0x21FFFF	N/A	CS2_	128-kB SRAM
0x220000–0x2FFFFF	N/A	None	Unused
0x300000–0x30000F	N/A	CS3_	UART
0x300010–0x3FFFFF	N/A	None	Unused
0x400000–0x4FFFFF	N/A	Internal	Control/status registers
0x500000–0xFFFFFF	N/A	None	Unused

Four external chip selects are called out. Instead of using the asterisk to denote active-low signals, the underscore is used, because an asterisk is not a valid character for use in a Verilog identifier. The first two chip selects are used for ROM (e.g., flash or EPROM) and their memory ranges are swappable according to the RomSel signal. It is sometimes useful to provide an alternate boot ROM that can be installed at a later date for various purposes such as a software upgrade. When boot ROM is implemented in flash, the CPU is able to load new data into its ROM. If there is no other way to send a new software image to the system, the image can be loaded onto a ROM module that is temporarily