installed into the CS1_ slot. A jumper can then be installed that causes RomSel to be asserted. When the system is turned on, RomSel=1 causes the ROM module to become the boot ROM, and new software can be loaded into CS0_ ROM.

The remainder of the address space is sparsely populated. Occupied memory regions are spread out to reduce the complexity of the decoding logic by virtue of requiring fewer address bits. If the UART were located immediately after the SRAM, the logic would have to consider the state of A[23:16] rather than just A[23:20]. The fifth and final used memory region is reserved for internal control and status registers. This decoding logic can be written in Verilog as shown in Fig. 10.5.

The address decoding logic is written here in behavioral form with a *case* construct. Case statements enable actions to be associated with individual states of a causal variable. Note that the chip select outputs are declared as regs even though they are not flops, because they are assigned in an always block instead of in a continuous assignment. Prior to the case statement, all of the always

```
module GlueLogic (
  Addr,
  RomSel,
  CS0_,
  CS1_,
  CS2_,
  CS3_
);

input  [23:20] Addr;
input          RomSel;
output         CS0_, CS1_, CS2_, CS3_;

reg            CS0_, CS1_, CS2_, CS3_;
reg            IntSel;

always @(Addr or RomSel)
begin
  CS0_   = 1'b1;  // establish default values to simplify case
  CS1_   = 1'b1;  // statement and prevent formation of latches
  CS2_   = 1'b1;
  CS3_   = 1'b1;
  IntSel = 1'b0;

  case (Addr[23:20])
    4'b0000 : begin
                CS0_ = RomSel;
                CS1_ = !RomSel;
              end
    4'b0001 : begin
                CS0_ = !RomSel;
                CS1_ = RomSel;
              End
    4'b0010 : CS2_   = 1'b0;
    4'b0011 : CS3_   = 1'b0;
    4'b0100 : IntSel = 1'b1;
  endcase
end

endmodule
```

FIGURE 10.5 Address decoding logic.

block's outputs are assigned to default inactive states. This is necessary to prevent the synthesis software from inferring an unwanted latch instead of simple combinatorial logic. If a combinatorial always block does not assign a reg value for all combinations of inputs, the synthesis tool determines that the reg should hold its previous state, thereby creating a latch. Latches are prevented by either exhaustively listing all combinations of inputs or by assigning a default value to all variables somewhere in the always block.

Unintended latches are the bane of HDL design. There are valid instances when a latch is desired, but latches are often inferred mistakenly, because the RTL code does not properly handle default cases wherein the variable is not assigned. Combinatorial logic must always assign values to variables regardless of the logic path. Otherwise, *statefullness* is implied. Verilog's blocking assignments enable multiple values to be assigned to a single variable in an always block, and the last assigned variable is the one that takes effect. Therefore, latches are avoided by assigning default values up front.

An active-high signal, IntSel, is decoded but not yet used for the purposes of selecting internal control and status registers that will be discussed shortly.

When expanding a CPU bus, an engineer must be careful that too many devices are not placed onto the bus, because output pins are rated only for certain drive strengths. As the lengths of the interconnecting wires increase and the number of loads increase, it may become necessary to extend the CPU bus using bidirectional buffers as discussed earlier. Figure 10.6 shows how our hypothetical system might use such buffers to isolate the plug-in ROM module so that the electrical impact of connectors and a separate module are minimized. For clarity, control signals are not shown. A unidirectional buffer isolates the address bus, and a bidirectional buffer isolates the data bus. No control is necessary for the address buffer, because it can be configured to always pass the address bus to the ROM module socket. However, the data buffers require control, because they must direct data out to the module for writes and in from the module for reads. Therefore, the tri-state control of the data buffer must be operated according to the address decode and read/write status.

The existing address decoding logic can be augmented to provide the necessary functionality. One additional input is required: the CPU's active-low read enable signal. An additional output is required to operate the data buffer's direction select signal. When high, the buffers will pass data from the CPU side to the ROM and, when low, the buffers will drive the CPU data bus with data presented by the ROM. A second always block can be added as shown in Fig. 10.7. New port and variable declarations are assumed.

Another common function of support logic is providing general I/O signals that the CPU can use to interact with its environment. Such interaction can include detecting an opening door and turning

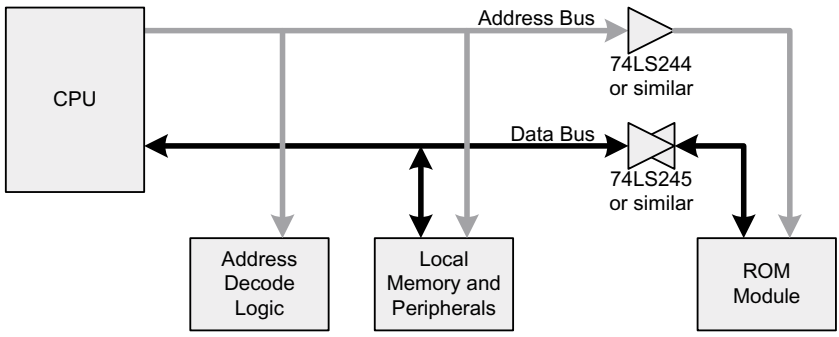

FIGURE 10.6 Bus extension buffers.