```
// Creates an instance of the ALU control unit (see the module defined in Figure C.5.16 on page C-38

   // Input ALUOp is control-unit set and used to describe the instruction class as in Chapter 4
   // Input IR[5:0] is the function code field for an ALU instruction
   // Output ALUCtl are the actual ALU control bits as in Chapter 4

ALUControl alucontroller (ALUOp,IR[5:0],ALUCtl); //ALU control unit

// Creates a 3-to-1 multiplexor used to select the source of the next PC

   // Inputs are ALUResultOut (the incremented PC) , ALUOut (the branch address), the jump target address
   // PCSource is the selector input and PCValue is the multiplexor output

Mult3to1 PCdatasrc (ALUResultOut,ALUOut,JumpAddr, PCSource , PCValue);

 // Creates a 4-to-1 multiplexor used to select the B input of the ALU

   // Inputs are register B,constant 4, sign-extended lower half of IR, sign-extended lower half of IR << 2
   // ALUSrcB is the selector input
   // ALUBin is the multiplexor output

Mult4to1 ALUBinput (B,32'd4,SignExtendOffset,PCOffset,ALUSrcB,ALUBin);


 // Creates a MIPS ALU

   // Inputs are ALUCtl (the ALU control), ALU value inputs (ALUAin, ALUBin)
   // Outputs are ALUResultOut (the 32-bit output) and Zero (zero detection output)

MIPSALU ALU (ALUCtl, ALUAin, ALUBin, ALUResultOut,Zero); //the ALU

// Creates a MIPS register file

   // Inputs are
   // the rs and rt fields of the IR used to specify which registers to read,
   // Writereg (the write register number), Writedata (the data to be written), RegWrite (indicates a
   write), the clock
   // Outputs are A and B, the registers read
   registerfile regs (IR[25:21],IR[20:16],Writereg,Writedata,RegWrite,A,B,clock); //Register file

// The clock-triggered actions of the datapath

always @(posedge clock) begin   if (MemWrite) Memory[ALUOut>>2] <= B; // Write memory--must be a store

   ALUOut <= ALUResultOut; //Save the ALU result for use on a later clock cycle

   if (IRWrite) IR <= MemOut; // Write the IR if an instruction fetch

   MDR <= MemOut; // Always save the memory read value

   // The PC is written both conditionally (controlled by PCWrite) and unconditionally
      if (PCWrite || (PCWriteCond & Zero)) PC <=PCValue;

end
endmodule
```

FIGURE e4.14.6 A Verilog version of the multicycle MIPS datapath that is appropriate for synthesis. (*Continued*)