EECS 427

CAD5 The Shifter

Assignment

To design a 16-bit shifter for your microprocessor.

Description

The shifter is an essential element for many microprocessor operations. It may be used to align or scale data, manipulate bits and bytes, or in an automatic or program-controlled shift-and-add multiply function. In the baseline machine, you are required to implement only a Logical Shift, which shifts data in a register (dest) as specified by a signed count operand (twos complement). A positive count specifies a left shift; a negative count specifies a right shift. You must support a shift amount contained in a register or the immediate field of the instruction. All bits shifted out of the destination register are lost. All destination bits not mapped from the original operand are filled with zeros.

Other common shift functions, which you may want to add to your processor, depending upon your application, are arithmetic shifts, arithmetic shift including the carry bit, byte swap, and rotate. In right arithmetic shifts, the high-order bits are filled with the original sign bit.

To shift the contents of a register one bit per clock cycle, an ordinary shift register, which can be assembled from cascaded D-latches, could be used. However if there is a need to shift data by an arbitrary number of bit positions within one clock cycle, a dedicated, programmable shifter is required.

Two frequently used approaches are (i) Barrel Shifters, and (ii) Logarithmic Shifters. While the barrel shifter implements the whole shifter as an array of pass transistors or multiplexers, the logarithmic shifter uses a staged approach. In general, barrel shifters are appropriate for small shift width and logarithmic shifters are more suitable for shifters of large width. Both types of shifter can be implemented in full CMOS transmission gates or nmos pass gates. Nmos pass gate designs can benefit from level restorers to avoid problems associated with Vt drops.. Shifters should be disabled in low-power designs when they are not in use.

Barrel Shifter

A simple 4x4, transmission-gate barrel shifter is shown in the Rabaey text (Figure 11-37, page 595). It consists of an array of transmission gates, with the number of rows equal to the bit width of the data words, and there is one column for each shift possibility. In this case, both are set equal to four. The input to the shifter is the value to be shifted, a literal <6:0> and the shift amount <3:0>. Weste and Eshraghian's Table 8.7 shows how different types of shifter can be implemented by appropriate choice of the literals. To do left shifts or rotates, the shift control inputs must be swapped. The control wires are routed diagonally through the array. A major advantage of this shifter is that a signal has to pass through at most one transmission gate, regardless of the size of the shifter. The shifter delay does, nevertheless, grow linearly with the size of the shifter, as the capacitance at the buffer inputs rises with additional columns.

The layout size of this shifter is dominated by the number and pitch of wires, rather than by transistor area. The size of the shifter grows linearly with the number of shift positions allowed. A decoder is required to select one of the shift lines. This decoder is more easily implemented in the controller (cad8) should you choose to implement a barrel shifter.

Logarithmic Shifters

In logarithmic shifters, the total shift value is decomposed into shifts over powers of two. A shifter with a maximum shift width of M consists of a $\log_2$ M stages, where the i-th stage either shifts over 2^i or passes the data unchanged. Rabaey (Figure11-38, page 597) shows a multiplexor-based logarithmic shifter. One could also use tristate buffer multiplexors or logic gate multiplexors instead of transmission gates. A circuit like this with many transmission gates in series would benefit from having buffers inserted along the path - perhaps every three stages or so.. This example is hard wired to give arithmetic right and left shifts (by sign extension for right shift, and making the least significant bits zero for left shift.). Minor changes will convert it to a logical shift, or a small amount of logic can replace the hard wiring to allow several kinds of shifts.

Logarithmic shifters have intrinsic decode, as the shift bits are used directly to control the muxes or transmission gates. Small barrel shifters, however, may be more compact than logarithmic shifters. Logarithmic shifters are always better when the word size is large and there are many shift possibilities.

Procedure

- **Schematic Design**
  You may implement any shifter which achieves the required functionality - both left and right logical shifts from 0 to 15 bits. If you are designing the pass transistor-based shifter, keep in mind that you will also have to design the decoder, but it is sufficient in this CAD assignment to design just the shifter array itself. (The decoder can be done as part of the control.). Keep in mind that the shift amount is a 2's complement amount - not signed magnitude! Some of the shifters presented in class or in the text assume a signed magnitude shift amount. Provide buffering for any array-wide control signals.
- **Modelsim**
  Run Modelsim on the 16-bit shifter and verify that it functions as expected by running a few test sequences (shift left and right by up to 15 bits).
- **Pre-layout Eldo**
  As with earlier designs, build a schematic for device sizing purposes prior to starting layout. In both shifter types you will have some significant routing capacitance to deal with. Bumping up device widths will help to some extent. You may find, especially if you've implemented a ripple-carry adder in cad4, that your shifter is much faster than your alu. You might therefore be tempted to ignore speed optimization for the shifter. However, keep in mind that the control input has worst-case delay when the control amount is encoded in a register in the register file and that you may incur additional delays if the control amount is manipulated in the controller prior to being sent to the shifter.
- **Layout**
  You have to make sure the shifter is bitslice width-matched with rest of your datapath structures. You may find the layout of the shifter to be metal intensive, especially if you are implementing the pass gate implementation, so think first of how to run metal lines. There is a tendency to want to make the shifter bitslice width smaller than the register file and alu. This really doesn't provide any benefit, though.
- **Design Verification**
  Run DRC, LVS and PEX on the entire shifter. Extract the parasitics, and back-annotate those to your eldonet viewpoint.
- **Analog Simulation**
  Find the delays through your shifter using Eldo.
- **Add delays to your schematic.**
- **Resimulate in Modelsim to demonstrate proper simulation.**

Requirements

You should have the following in your group's **cad5** directory.

- Schematic of the entire shifter, including the drivers/buffers (but not necessarily the decoder).
- Eldo config file and wave .jpgs showing how you calculated delays (with details about your critical path in the README file).
- Modelsim (with delays added) .wlf and .do showing both left shift, right shift by at least two different values, (and no shift). Please make sure your .wlf and .do files can be properly loaded in Modelsim prior to submitting your assignment.
- Layout of the shifter. This should include hierarchical layout of the drivers and buffers.

You must also have the following error-free report files in your groups **cad5** directory.

- DRC report from ICrules.
- LVS report from ICtrace(M).
- PEX reports from ICextract(M).
- README file. This file should be a report documenting your work for CAD5. This should discuss the considerations that went into the choice of your shifter design and the floorplanning. You should also discuss the rise and fall delays and the critical path. Any other comments that you feel are relevant should be included.