CAD3 The Register File Fall 2009

Assignment

Design a Register File (RF) for your microprocessor.

Description

In this CAD assignment, you will design a 16-word, 16-bit RF with 1 write port and two read ports. One of the structures that are central to the datapath is an RF. An RF consists of a set of registers that can be read from or written to. Writing a register requires (i) an address, (ii) the data to be written and (iii) a signal that controls the write timing. Reading an RF can often be controlled by just the address but may include additional logic. The RF for this project is to include an array of flip-flops or latches and properly sized buffers for the associated read/write circuits. Address decode logic is not required in CAD3, but will have to be added to your design when the RF is integrated with the microprocessor control circuitry.

Register Cell

Conventionally, an RF might consist of an array of static RAM cells with read/write circuitry and a sense amplifier. Though this type of implementation yields a fast, compact design, it requires much design effort, especially for the sense amplifiers. In small RFs (few registers), the support circuitry for the SRAMs is used by a small number of registers, making the SRAM-based RF larger than some other types. An alternative approach is to use a latch consistent with your clocking scheme, modified to have two read ports and one write port. One such example is given on the next page. This method makes use of pass transistors at the input and transmission gates at the output of the bit cell. These gates are controlled by decoded signals from the control unit. Alternatively, tri-state buffers could be used to read the RF, and transmission gates to write them. Another way to implement an RF is with multiplexors. While this implementation is conceptually simple, routing all of the mux inputs from the array cells to the muxes will create unnecessary routing bottlenecks.

Drivers and Buffers

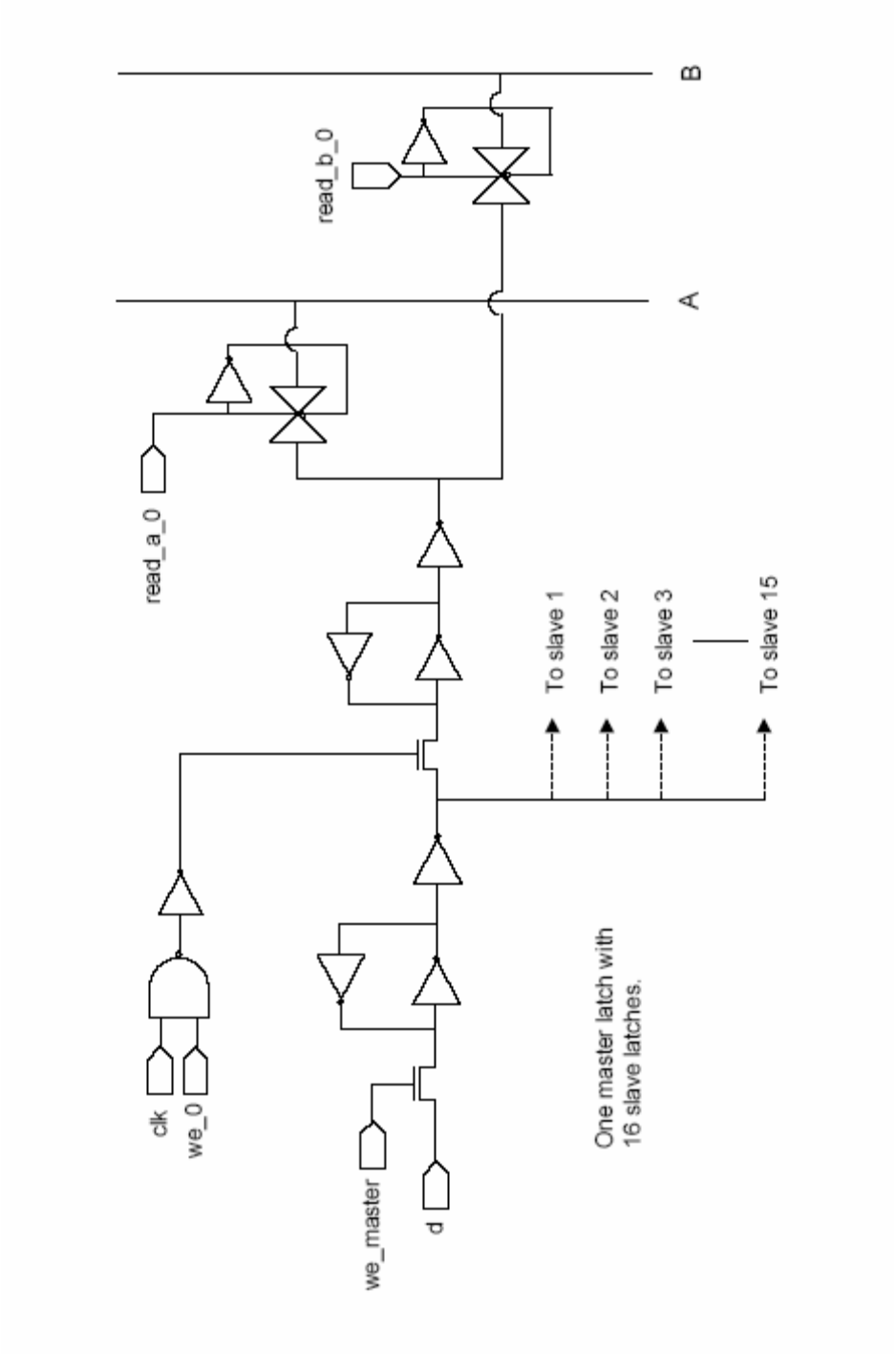
Control signals shared by all 16 bits of a register can have large capacitive loading compared to other signals in your design. The data output busses on the RF may also have large loading associated with them. These signals are candidates for transistor sizing on the gates that drive them or possibly even the insertion of buffers. In either case, these design decisions should be based on HSpice simulations.

Bus Strategy

You can use several methods to drive values onto the microcontroller busses. Two common ones that you can use in this class are multiple busses coming out of the RF with a mux to select the proper word. Another possibility is to have tri-state drivers on the outputs of each component that can drive the bus. This will require only one bus. Note that true tri-state busses require keeper latches if they have active gate inputs connected to them and are allowed to float to intermediate voltages. It is important to decide which strategy you will use. After this assignment, it will be very difficult for you to come back and change the RF. Please see the Document "*Busses in Cadence*" for a description of how to create busses within your schematics.

Metal Layers

Only use the following metal layers to do your routing: metal1, metal2, and metal3. Metals 4 & 5 are being saved for global routing in the last CAD assignment. This means that you must plan your register file carefully so that you only route in the lower layer metals but are still able to keep the layout compact.

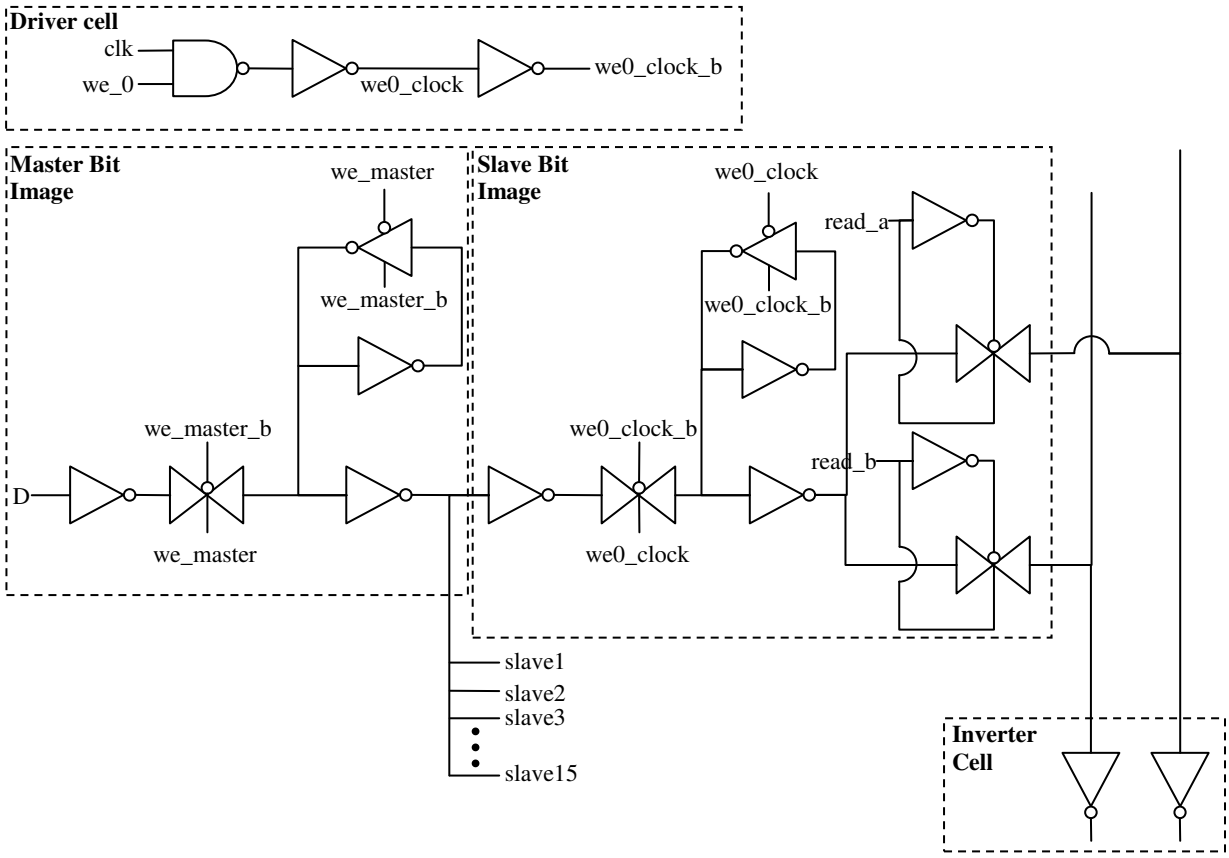


Procedure

Schematic design

Determine your read/write methodology and select (from among your group's CAD 2 cells) a register or latch which will serve as the basis of your register file array cell. Also determine your group's bussing strategy. You may want to design a new register cell. You will not be responsible to design the decoder for this CAD assignment. Remember to use rnfet and rpfet as discussed in discussion to make weak inverters. Then replicate your 16-bit register 16 times to form the array. You will need to design the driver circuits based on the capacitance estimates (see below).

One alternative schematic:



Speed-path Schematic

Create a special schematic for the purpose of sizing devices prior to starting layout. Since it can be difficult to resize devices after layout is complete, it makes sense (particularly in the custom design of array structures like memories or RF's) to enter a simple, representative schematic containing the basic master-slave latches, control signal buffers and estimated parasitic capacitances. For example, let's say your design uses a full-CMOS transmission gate for driving an output bus. One bit of that bus will therefore have 16 full-CMOS transmission gates (one for each register), parasitic routing capacitance as well as any additional loading outside of the register file itself. The source/drain loading due to the transmission gates can be modeled easily with "dummy" transistors -transistors with gate and source tied to vdd (pMOS) or vss (nMOS) and drain tied to the output node. Transistor gate loading can be modeled similarly by tying source/drain to vdd or vss and connecting the gate of the "dummy" device to the node in question. Ballpark capacitance estimates for minimum width routes in this technology are: poly -0.16fF/um, metal1 - 0.25fF/um, metal2 - 0.20fF/um, metal3 - 0.20fF/um, metal4 – 0.20fF/um, metal5 – 0.20fF/um. Once you have an estimate for the routing capacitance, use the capacitor (CAP) symbol from "analogLib" library. You should use this modeling technique for any nodes that look like they will be heavily loaded in terms of routing length and the number of attached transistors. Capacitance of other nodes (e.g. internal latch nodes) will be dominated by transistor loading which is already accounted for by the transistor model. With this schematic in place, you can run HSpice to intelligently size devices in your control signal buffers and devices along your output path. As for target delays, the requirements will vary depending on your group's application. In one clock cycle, your controller will have to decode the instruction word, assert read signals to the RF, the RF will provide operands to the ALU, the ALU will perform an operation and have data ready for writing back to the RF by the end of the cycle. This is the leading candidate for the critical path in these designs. Sometimes memory accesses are the critical path. 100MHz (10ns period) is a relatively easy target frequency to meet with this technology. 400MHz (2.5ns period) is about the maximum.

NCverilog
Run NCVerilog on the RF and verify that it functions as expected by running a few test sequences. For example, write to each location then read from each location.

Layout
Since the RF is one of the key modules of your datapath, care is needed to minimize the parasitic capacitances on critical signal nets, as this directly affects the performance of your microprocessor. Keeping the structure dense and compact while avoiding long interconnects is a prime goal. Mirroring a cell and thereby sharing wells and the supply lines is one of the most frequently used strategies. You should design the floor-plan and routing strategy before you start on your layout. Power lines in the cells should be dimensioned so that they can carry the current required by the row of connected cells. A good ballpark figure for this process is to keep average current densities below 1 milliamp/micron of metal width. Use hierarchy effectively to build up your register file layout. The use of hierarchy would also help you to isolate DRC and LVS failure since DRC and LVS check are done in small increments.

Design Verification
You may end up spending a lot of time on verification if you have errors. The best way to avoid this is to check the design thoroughly as you go along. It is much better to spend 5 minutes in checking the circuit, than to waste 30 minutes debugging the RF macro later. Run DRC, LVS on the entire register file. Extract the parasitics and back-annotate those into your speedpath to get a more accurate read and write time.

Delay Estimation
Now that your layout is complete and PEX has been run, you can add any additional capacitance for loading outside of the RF and derive more realistic delay estimates. There are two ways to do this. You could reuse your speedpath schematic or simply run HSpice on the entire RF. If you reuse your speedpath schematic, you will have to get "real" capacitance information from your backannotation and place or update capacitors in your speedpath. If you run HSpice on the entire RF, force inputs such that only one location is being read. Note that the worst-case in the tri-state bus architecture is when one register is read out onto both output busses simultaneously (e.g. ADD r0,r0 would trigger this path).

Comments

- Since you are working in groups now, you only need to submit one copy of the required register file for grading. We will identify groups by group name. Do your CAD 3 work in that group directory. For example, if you are in group1, your class directory will be:
  /afs/umich.edu/class/eecs427/f09/group1
- Try to think about the routing strategy in advance ---how, and where to use metal1, metal2, metal3, etc. It's ok to occasionally use metal2, metal3 in your leaf cell (lowest-level) layout but limit their use as much as is reasonable and try to stick to preferred routing directions.
- Plan your team's time in advance. Meet early, decide on the basic circuit to implement and decide who will do what and when. Don't wait until a few days before the deadline to start. Get started right away.
- While you are working on CAD 3, think about several other modules on the datapath.
- How are they going to integrate with your RF? How are your busses going to run through your datapath? The more resources you have for routing data signals over the RF, the more flexibility you will have in CAD7 when placing and routing the datapath components together. Making your data outputs and data input easily accessible from top AND bottom of the RF will be helpful.
- Shoot for a compact layout. Success in this area will depend almost entirely on the layout of the array cell which will be replicated 256 times. Remember bit-slice width target of 4.4-6.4um

Requirements

NOTE: **Again, only use metal1, metal2, and metal3 for routing.**

You should have the following files:

- Schematic of the entire register file, including the drivers/buffers
- Schematic of the register file speedpath
- Saved *Analog Environment* states for speedpath read/write rise & fall delay simulations, named "hspiceD_RFspeed_[read|write]_[rise|fall]", which show rise and fall simulations for both the write and read of 1 bit in the full RF speedpath (stimulate any 1 bit, e.g., bit 0). You should submit 4 "RFspeed" states in total (read_rise, read_fall, write_rise, write_fall).
- Wave .png for HSpice showing how you calculated delays (data changing from 1->0 and from 0 ->1).
- NCVerilog .ps files showing reading and writing of a register.
- Layout of the Register File that passes DRC & LVS.
- DRC and LVS reports.
- README file. This file should be a report documenting your work for CAD 3. Please list all the files that your group is submitting. This should discuss the considerations that went into the choice of your RF design and the floor planning of the RF. You should also discuss the development of your speed-path schematic. If your speed-path simulation result is very different from result after extraction, please also include explanation why it is happening. Be sure to include delay analysis and area numbers for your design.
- You need to name your directory **cad3**, and create a "**submit**" directory in your **cad3** directory. Copy all your drc, lvs reports, and simulation waveform files to the "**submit**" directory.

Deadline

You need to turn in CAD3 by **Wedensday, Sept. 30st, 2009, 7:00 pm**.