California Institute of Technology
Department of Computer Science
Computer Architecture

CS184a, Fall 2000 — Assignment 2: Arithmetic and Pipelining — Monday, October 2

Due: Monday, October 9, 10:30AM

You may do sections (A and B) or (B and C). C is primarily intended as a more challenging (interesting) alternative for students who have already had considerable experience with computer arithmetic and basic logic design.

You may use hierarchical schematics. Where appropriate quantities are 4b, unsigned numbers. Use a schematic drawing program for circuits.

A: Pipelining and Tradeoffs

1. Review your spatial sort from Logic.A.4
   - What is the latency? (assuming each NAND2 has unit delay)
   - Pipeline so that each comparison operation is its own pipe state.
   - What is the latency and throughput of the pipelined design?
   - How much further can you pipeline this design?
2. Design a sequential sorter:
   - Use a single comparator
   - Use a small memory (4 memory slots is sufficient; you may use up to 16).
   - Use a minimum amount of control.
   - Don't get clever–a simple O(n^2) solution is fine.
   - Describe the operation.
   - Assuming the data "magically" appears in the memory before starting your logic, and "magically" is removed later, what is the throughput and latency of your implementation?

   optional Describe what we might do with the data memory so the data appeared and disappeared like this as far as this unit controller is concerned?
3. Using gates with at most 4-inputs, design the bit-slice for an ALU which can perform four operations:
   - addition

- subtraction
- bitwise NAND,
- reducing or – "reducing or" is an OR on the entire datapath; if any bit in the A-input to the datapath is one, the LSB of the output should be one. Otherwise (all bits are zero), the LSB should be zero.[1] The non-LSB bits are don't care for this operation.

B: Multipliers

We saw in lecture how to built various adders. In this problem, I'm asking you to review various techniques for building multipliers.

1. Show a spatial multiplier build out of simple, ripple-carry adders.
    - What is the latency?
    - What options exist for pipelining this design? (describe the space of designs available and the register cost-throughput tradeoff)
2. Show a shift-and-add multiplier design.
    - Take as input two numbers and a control signal that tells it to start.
    - Produces as output the $2n$-bit product of the two numbers, along with a signal indicating that the multiplier is busy and a signal indicating when the output result is valid.
    - Uses a single adder and $O(n)$ state registers.
    - Report on latency and throughput.
3. Consider using a redundant representation for values inside the multiplier. In particular, consider that you represent a number as two binary words which, when added produce the value you are representing.
    - How does this change the latency of a single addition?
    - Show the resulting multiplier which starts with numbers in standard form, but uses this representation internally. What is its resulting latency, area, and pipelineability?
    - What do you need to do on the output of the multiplier to convert the result back into normal form? What is the overall latency and area of the multiplier including this final return to standard form?
    - How would you apply this idea to the shift-and-add multiplier, and how would it change your latency and throughput results?

[1]typo in original – andre

C: Advanced Problems

1. How can we do better (lower latency) than the multiplier in B.3?
2. Design a bit-serial multiplier that operates at the input rate of the data. That is, it takes in operands one bit per cycle and produces results one bit per cycle. There is no bound specified on the number of cycles of latency. The multiplication may be $n \times n \rightarrow n$, where you discard the low n bits of the result.
3. Show the design for a 5-tap median filter, using a single comparator and a small memory.
    - you get a sequence of words as input; one word appears at a time; the timing between words is determined by your implementation.
    - you produce as output a sequence of words, one per input; the output value is the median of the last 5 values received at the time of the output.
    - use a single comparator.
    - use a small (<16 element) memory.
    - show your control logic and any additional datapath and state elements you need for your implementation.
    - what is the throughput rate of your design?