**California Institute of Technology**
**Department of Computer Science**
**Computer Architecture**

CS184a, Fall 2000

# Final Exercise

Monday, November 27

---

**Due:** Friday, Dec. 8, 5:00PM

You have just taken a job at Hgrep Inc., an internet hardware startup looking to be the world leader in custom processing to accelerate search tasks. HI's flagship product is a chip for analyzing data streams at high data rates and marking all occurrences of a programmed regular expression. Two Internet search giants and several government TLA's have sampled the HI1; they find it intriguing but have a few concerns about its capabilities and future directions. Your job is to help develop the roadmap for the HI2 and HI3 in order to secure HI's position as a leader in this emerging niche market.

HI1 was developed by Otto Mata, a hot shot graduate from Atlantic Tech. Otto, remembering his elementary automata theory recalled that a regular expression could be recognized by a finite automata. He knew any regular expression could easily be converted into an NFA, and any NFA could, in turn, be reduced to a DFA. From his basic circuits class, he knew he could implement a finite state machine in silicon to physically implement the automata.

Otto knew that programmability was an important feature here. AmIA needs to search for the topic of the day in newsfeeds from around the world. WideComm wants to find all Internet references to itself or its products. Consequently, Otto decided to build a programmable FSM. To handle the high-bandwidth traffic, he set as a goal consuming one ASCII character on every cycle. Otto knew that he could build a programmable FSM out of a memory. Consequently, he hit on the idea of simply building a large memory that took in an 8b ASCII character on every cycle, combined that value with an 8b state value, then looked up a result in the large memory. The result is an 8b next state value and a 1b value indicating when the programmed state bit is found. Otto targeted the HI1 at an 8ns memory cycle rate, giving it the ability to handle Gbit data streams (e.g. Gigabit ethernet).

Where necessary, you should be able to estimate the sizes in terms of memory bits, programmable switches, registers, and wires. Assume wires have a $10\lambda$ wire pitch (trace and space). The following table provides a set of building block areas you can assume for your analysis.

| **Component** | Area in $\lambda^2$ |
|---:|:---|
| Memory Cell | 1,000 |
| Programmable Switch (includes memory bit) | 2,500 |
| Register | 4,000 |
| FPGA Logic Block (4-LUT, FF, interconnect) | $10^6$ |
| Plentium (500MHz, 0.18$\mu$m) | $10^{10}$ |

Your CTO, Ned Tono, needs your analysis on his desk by 5pm next Friday so he can incorporate them into his briefing over the weekend in preparation for a talk to second round funders the following Monday.

1. Otto's design is obviously quite simple. However, Otto wasn't big on documentation. Diagram Otto's design from the 125MHz, 8b input data stream to 1b output stream showing memory, registers, datapath, and anything else necessary to understand the operation. Estimate the area which will be required for Otto's design. Otto's design was in a $0.35\mu$m process. About how large was the HI1 die?
2. Potential customers wonder why the HI1 can outperform their Plentium processors and wonder if HI is taking a loss on these components (which sell for much less than the Plentium) in order to build market share. Explain for your fellow HI engineers the intrinsic advantages of HI1 over implementing the same computation on a commodity Plentium (describe nature of area and speed benefits; quantify where feasible).
3. Noting that you should now be looking at a $0.18\mu$m process, Ned wonders where to go with the HI2. Several customers have wondered if the HI1 could search for multiple regular expressions simultaneously. This sounds like easy parallelism to you. You hit on the idea of making the HI1 FSM into a tile which you can array onto a larger die in order to deal with multiple regular expressions. Show the modified tile design and how these tiles are composed into a chip. You should show the input and output stream connections, and will need to describe how the output is encoded. Ned says he's very comfortable giving you roughly $100\text{mm}^2$ of core die area. How many FSMs can you get on such a chip?
4. One of the key problems raised by No Such Agency is that the regular expression must be expressible with only 256 states. For some of their expressions that's really too small. They wonder if you can make your device handle larger expressions (they'd really like to not see such limits). Otto had sent an old email to Ned saying that increasing the memory was an easy thing to do. After doing the multi-FSM HI2 sketch above, you're not comfortable with the idea of simply making the FSM state bits, and hence memory, larger since that would decrease the number of FSMs you can get on the chip. You're also starting to be concerned about scaling the clock rate with such a large memory. Instead, you offer to show how you can modify the tile (slightly) so that you can use multiple tiles to solve a large problem. Show your modified tile design and explain its operation.
5. Another problem plaguing the AmIA is that foreign languages don't fit into the ASCII character set. They wonder if a future HI chip could support unicode (16b character sets). What would be the impact on FSM size if you extended Otto's scheme to use 16b characters instead of 8b? You examine some of their typical queries and see that they seldom use more than 100 different character in any search. How would you exploit this property to use the HI2 tiled architecture (above) to solve their problem instead. You will have to make another slight modification to the HI2 tile; show your modified tile design and describe the impact using this functionality will have on HI2 capabilities.

6. Marketing has been pushing for a larger speed improvement than what your technology is giving you. How would you process two characters at a time? As a starting point, assume that you are working with a design which only needs to distinguish 16 different symbols for actions. Taking this the next step, how would the tile change to accommodate this and the more general case in which we programmably divide the 16b's of input to the FSM memory among symbols and states?

7. Seeing your juicy market, Xgrep is now seeking venture funding to compete against you. Their idea is to use commercial FPGAs to solve the same problem. On the "[Gg]ore" and "[Bb]ush" benchmarks they are claiming $10\times$ your density. Is this a credible claim? Provide a detailed explanation.

8. If we consider only simple string matches (no Kleene star's or complicated disjuncts, maybe single character wildcards; *e.g.* we're searching for something like "now is the time for all good men...”), for what size strings will the Xgrep solution be less dense than the Hgrep solution? (if you didn't develop it for the previous one, I'm expecting you to develop a straightforward mapping of this match problem into FPGA Logic Blocks so you know how resources grow with the string size.)

9. How can you increase your density on short, "simple" expressions?

10. You've now added several non-memory feature to your tile including programmable interconnect and programmable addressing. In light of this challenge from Xgrep, you'd like to make HI3 be a lean programmable device. You're willing to sacrifice your performance on unicode, but know you need to keep the ability to perform multiple searches, chain FSM blocks to handle larger regexps, and use encoding to reduce the space of characters seen by the FSM. What is the right memory depth for each tile? How do you stack up against Xgrep?

11. **Optional, Open Ended** Ned didn't ask, but you do want to impress him with your technical prowess. Besides, you do want to make sure your 100,000 stock options in HI will be worth something some day. If you think Hgrep is following the right track, what else should they be considering to make the architecture more competitive? If you think both Xgrep and Hgrep are off track, what alternate approach would you advocate? You know that using rough quantification of area and benefits as you've done in the earlier part of this assignment is likely to give your proposal the weight to get Ned's attention.