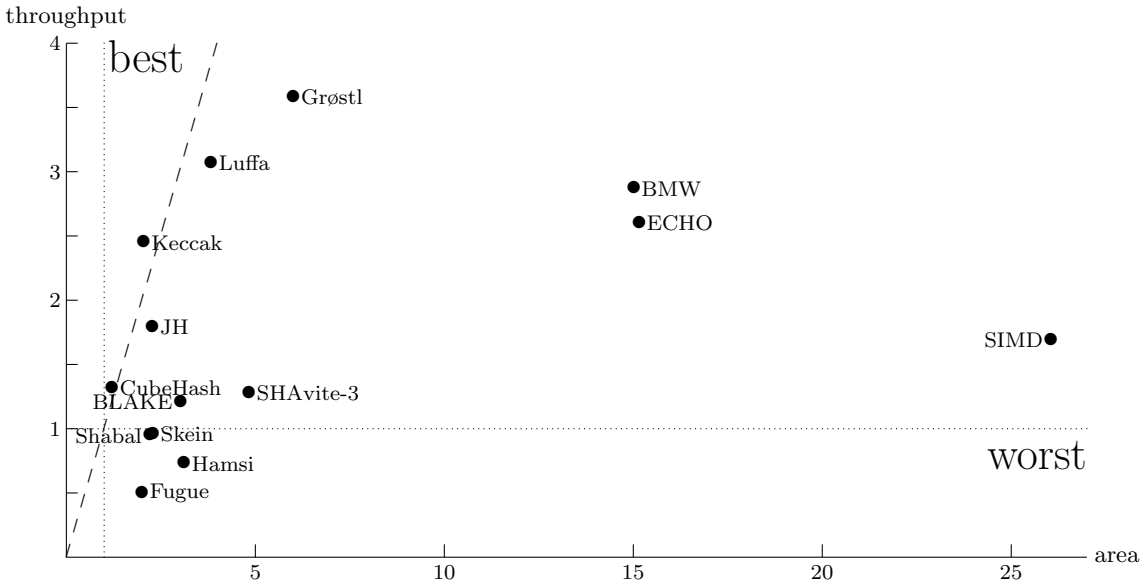
Visualizing area-time tradeoffs for SHA-3

Daniel J. Bernstein *

Department of Computer Science (MC 152)
The University of Illinois at Chicago
Chicago, IL 60607–7053
djb@cr.yp.to

The following diagram was presented in an FPGA-benchmarking talk at the Second SHA-3 Candidate Conference two weeks ago:



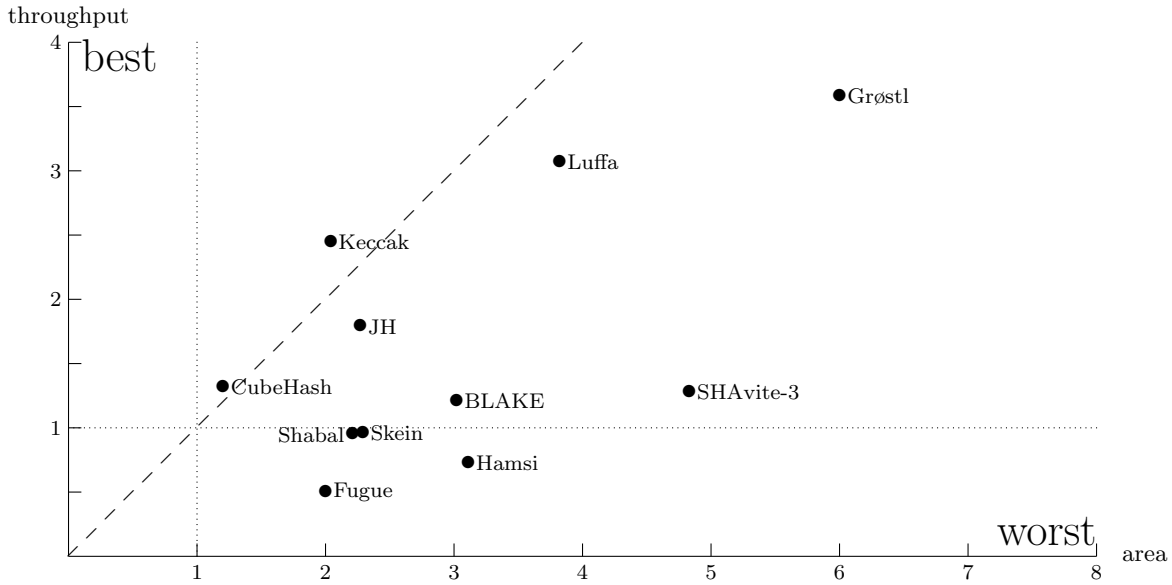
The points in the diagram are labelled by various SHA-3 candidates with high-security 512-bit output. The horizontal axis is area: FPGA slices, divided by FPGA slices for SHA-512. The vertical axis is throughput: gigabits per second, divided by gigabits per second for SHA-512. "Best" appears in the top-left corner: minimum area and maximum throughput.

The presenter pointed out the candidates closest to the word "best" in this diagram. The top five candidates are obviously (1) Luffa, (2) Grøstl, (3) Keccak, (4) JH, and (5) CubeHash.

Is this a meaningful comparison? Are these candidates actually the best? Or are we being fooled by the diagram?

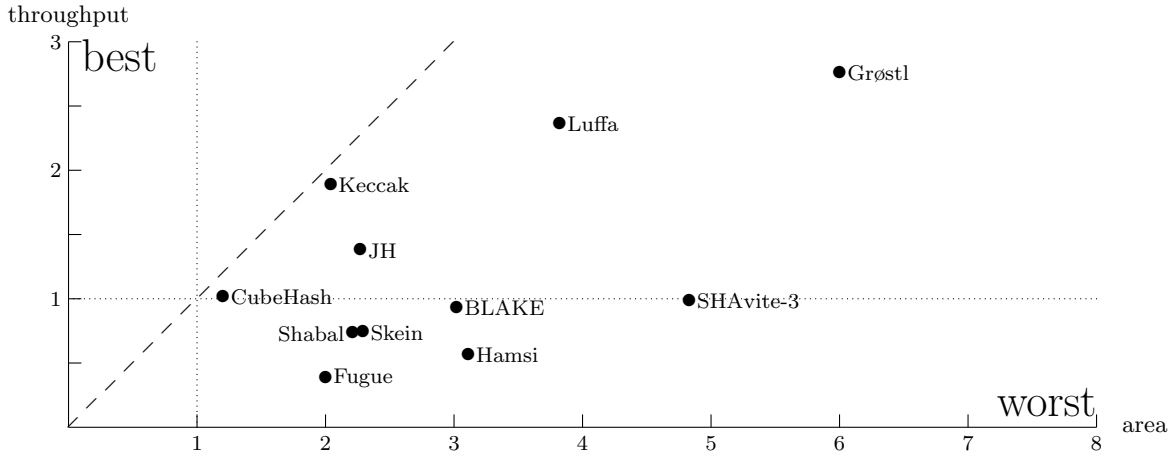
Problem 1: How are the axes scaled? The axes in this diagram were scaled separately: the distance from the origin to 1 on the vertical axis is much larger than the distance from the origin to 1 on the horizontal axis. This scaling was chosen to fit the complete diagram of current SHA-3 candidates within a 1024×768 projector screen—hardly an objective scientific criterion. Let's plot exactly the same data on "square" axes:

* Permanent ID of this document: 1acfb913bd21cdf616afd004e254a55c. Date of this document: 2010.09.06. This work was supported by the National Institute of Standards and Technology under grant 60NANB10D004.



The data hasn't changed—but the list of candidates closest to "best" in the diagram has changed dramatically: instead of (1) Luffa, (2) Grøstl, (3) Keccak, (4) JH, and (5) CubeHash we have (1) Keccak, (2) CubeHash, (3) JH, and then a tough fight between Shabal, Skein, Fugue, Luffa, and BLAKE.

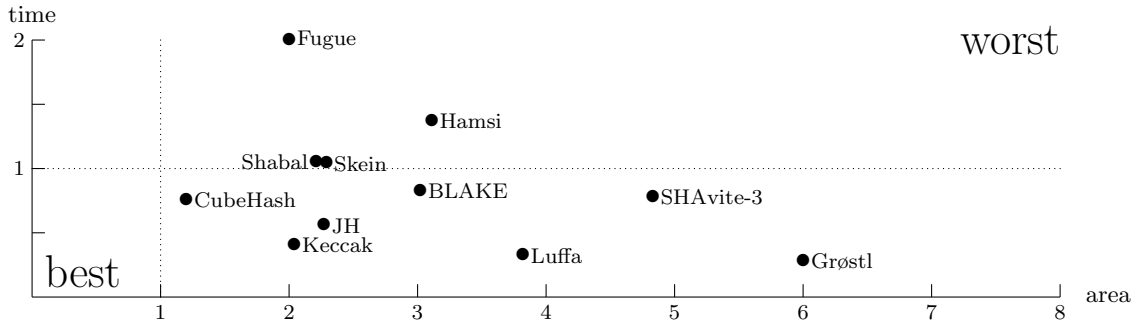
Problem 2: How was the data scaled? The "square" choice of axes in the second diagram might seem fair and balanced and uncontroversial. Suppose, however, that a clever implementor figures out how to make SHA-512 run 1.3× faster within the same number of FPGA slices. "Throughput" in the diagram is relative to SHA-3, so it is divided by 1.3:



In this diagram, CubeHash is as close to "best" as Keccak is, and Luffa is clearly behind Shabal, Skein, Fugue, and BLAKE. This makes no sense: how can a speedup in SHA-512 change the list of best SHA-3 candiates?

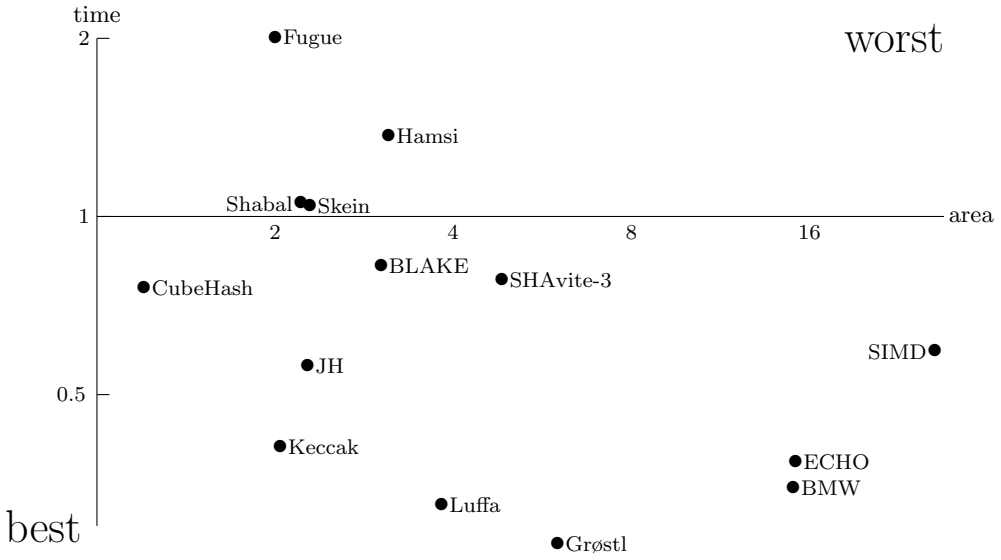
Problem 3: Cost, or inverse cost? All of these diagrams have a further asymmetry in the choice of axes: area is a cost, while throughput is an *inverted* cost. Let's eliminate the

inversion and plot time instead of throughput. Here is the original data (no hypothetical SHA-512 speedup) with the new axes:



Many of the previous comparisons between SHA-3 candidates have now been solidly reversed—again, from exactly the same data! CubeHash is clearly closest to "best," for example, and Shabal and Skein are clearly much closer to "best" than Fugue, BLAKE, and Luffa.

Problem 4: Linear, or logarithmic? Quite a few readers have been screaming "use a logarithmic scale!" since the first page. Here's the same data once again, with area and time both plotted logarithmically (and with BMW, ECHO, and SIMD again visible):



A logarithmic scale has many well-known advantages over a linear scale. Inversion no longer matters—it simply mirrors the diagram from top to bottom, without changing relative distances. The SHA-512 scaling no longer matters—it simply shifts the diagram, again without changing relative distances. A factor of 2 speedup, or a factor of 2 area reduction, always moves the same distance on this logarithmic diagram, while its visibility in a linear diagram depends on the starting point—a linear diagram squeezes small numbers close together, hiding important differences in those numbers.

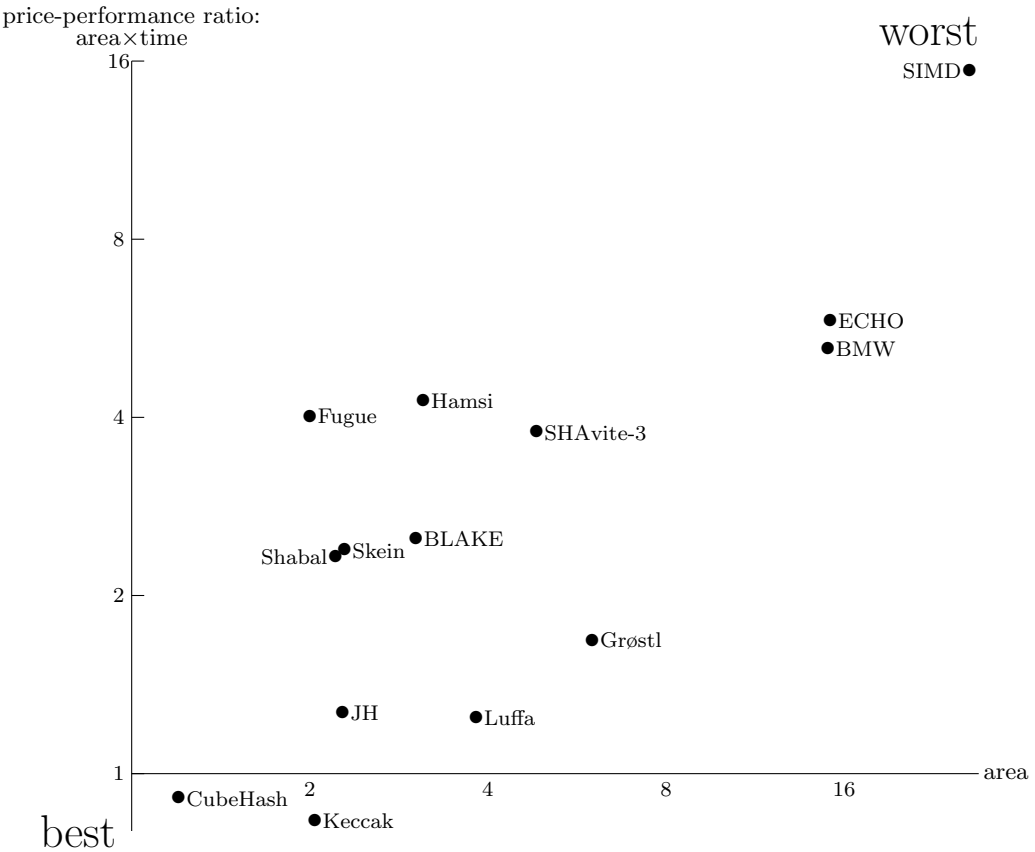
In this diagram Keccak and CubeHash are closest to "best," followed by JH, then Luffa, and then a fight between Shabal, Skein, BLAKE, and Grøstl.

Problem 5: Performance, or price-performance ratio? Throughput per se is a meaningless number, because it can always be arbitrarily increased by parallel repetition. This Skein implementation, for example, consumes 1520 CLB slices on a Virtex 5 and hashes 2.8 gigabits per second; 100 copies of the same implementation, spread across 100 chips, consume a total of 152000 CLB slices and hash 280 gigabits per second.

Of course, this parallel performance requires having 100 messages to hash at the same time. There are some applications with far fewer messages to hash—but if hashing is so rare then why would anyone care about its performance? If an application is bottlenecked by hashing a small number of *extremely long* messages, why can't the application switch to Skein's parallelizable tree-hash mode? This isn't specific to Skein: there's a generic parallelizable SHA-3-Tree construction that's as secure as SHA-3 and almost as fast as SHA-3.

Let's assume that the user has a total of G gigabits per second to hash, and let's assume that the user can find enough parallelism in this data to productively use $\lceil G/2.8 \rceil$ copies of the Skein implementation. The cost of handling this data—the number of CLB slices spent on those Skein implementations—is then $1520\lceil G/2.8 \rceil \approx (1520/2.8)G$. The factor in front of G, the quotient $1520/2.8$, is a meaningful ratio with a standard name: "price-performance ratio." This ratio isn't improved by parallel repetition or by any other trivial technique; it shows the overall cost of Skein for hashing heavy volumes of data.

The FPGA-benchmarking talk said that price-performance ratio was its top optimization target. Let's highlight this ratio in the diagram by using it as the vertical axis, replacing the meaningless performance axis:



This diagram clearly shows several vertical levels—several levels of price-performance ratio, exactly the main feature of interest. Keccak and CubeHash are best (and better than SHA-512), with Keccak slightly ahead; JH and Luffa are next; Grøstl is next, only about a factor of 2 behind Keccak and CubeHash; Shabal, Skein, and BLAKE are clearly behind Grøstl; etc.

The diagram also shows a wide range of horizontal positions (areas) at each vertical level. Even if price-performance ratio is the top criterion, small area has several obvious advantages: for example, $15200\lceil G/28\rceil$ is similar to $1520\lceil G/2.8\rceil$ for very large G, but is much larger for many interesting values of G.

Disclaimers. I don't mean to criticize the author of the first diagram. The diagram isn't bad, especially by comparison to typical benchmarking papers! I'm simply pointing out that the same data could be graphed in a more informative way.

I also don't mean to suggest that the implementations featured in all of these diagrams are optimal. For example, it's already known that a different implementation strategy for Shabal produces better price-performance ratios on many FPGAs. The diagrams should be updated whenever necessary to reflect the best implementations.

Finally, I don't mean to suggest that these diagrams are complete. Suppose, for example, that the user's top priority is actually area, and that price-performance ratio is secondary. These diagrams suggest that CubeHash has by far the smallest area, followed by Fugue, Keccak, Shabal, JH, and Skein—but these are areas of implementations that were optimized for something else! There is no reason to think that this data has anything to do with the areas of implementations that are optimized for area, or that balance area with other criteria. A more thorough diagram would have a series of several points for each SHA-3 candidate, showing the best price-performance ratio achievable for each area.