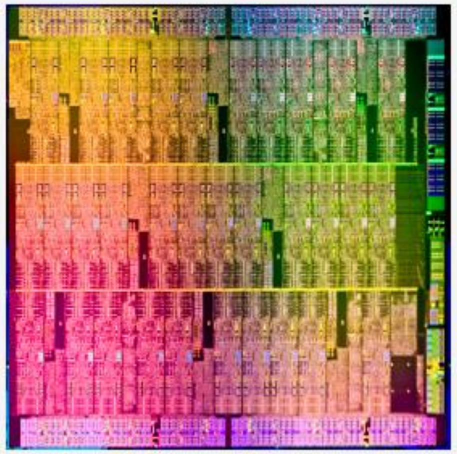
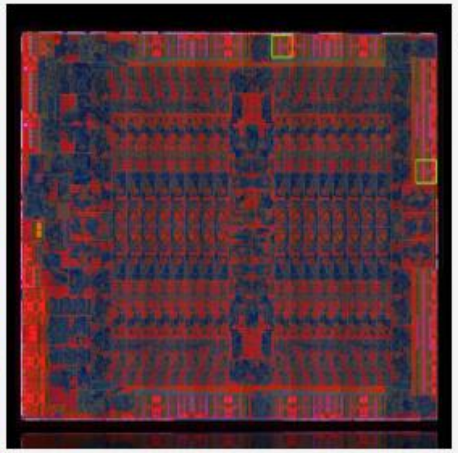
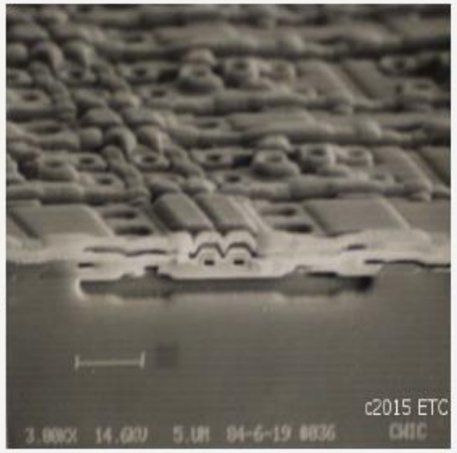
18-344: Computer Systems and the Hardware-Software Interface

Fall 2023



Course Description

Lecture 10: Design Space Exploration

This course covers the design and implementation of computer systems from the perspective of the hardware software interface. The purpose of this course is for students to understand the relationship between the operating system, software, and computer architecture. Students that complete the course will have learned operating system fundamentals, computer architecture fundamentals, compilation to hardware abstractions, and how software actually executes from the perspective of the hardware software/boundary. The course will focus especially on understanding the relationships between software and hardware, and how those relationships influence the design of a computer system's software and hardware. The course will convey these topics through a series of practical, implementation-oriented lab assignments.

Credit: Brandon Lucia

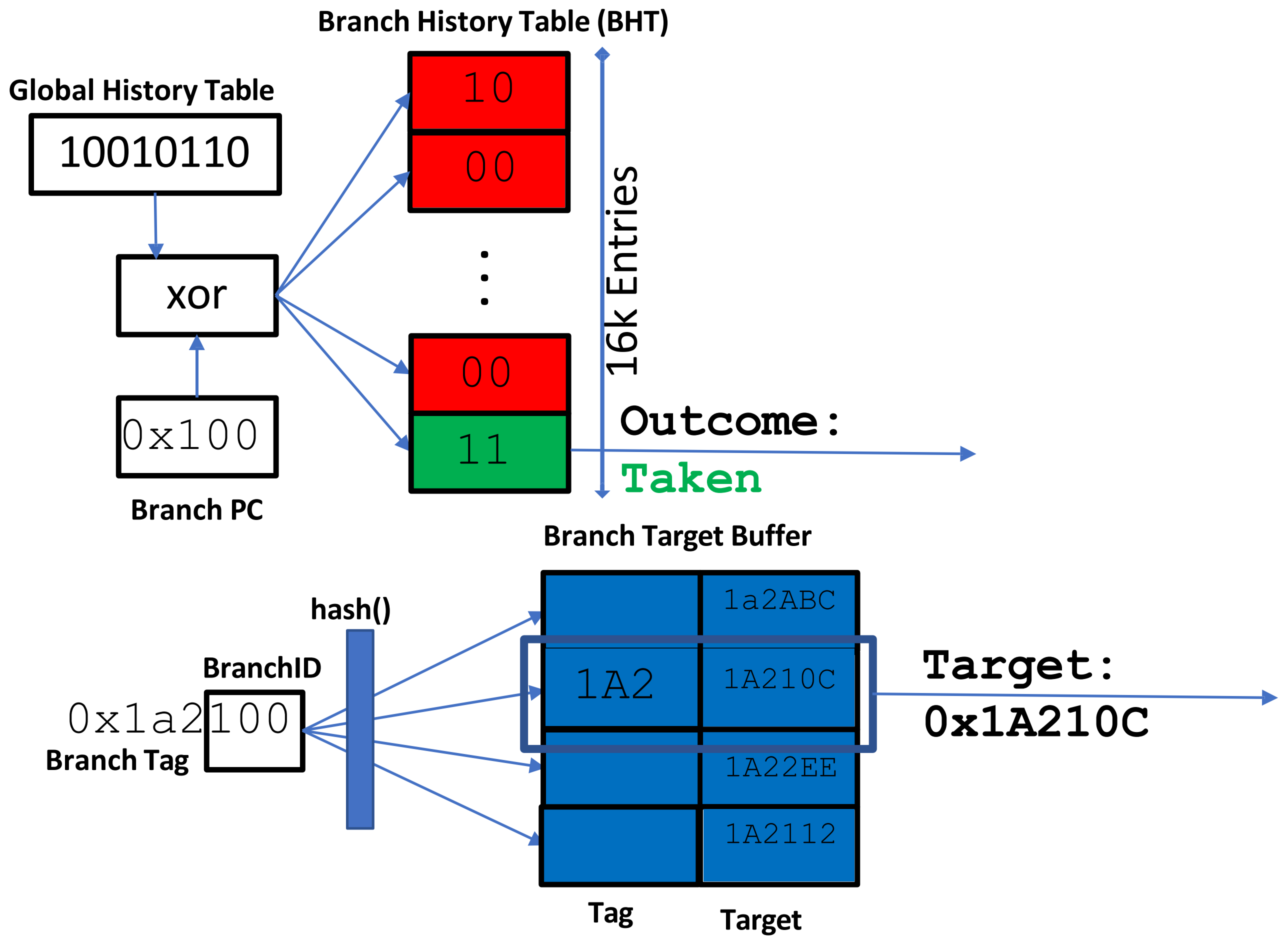
Today: Design Space Exploration

- Defining the design space of a hardware or software system
- Pareto Frontiers and optimizing within a design space
- **Applied** Performance Evaluation
  - Finding the best performing design under constraints

Defining a design space

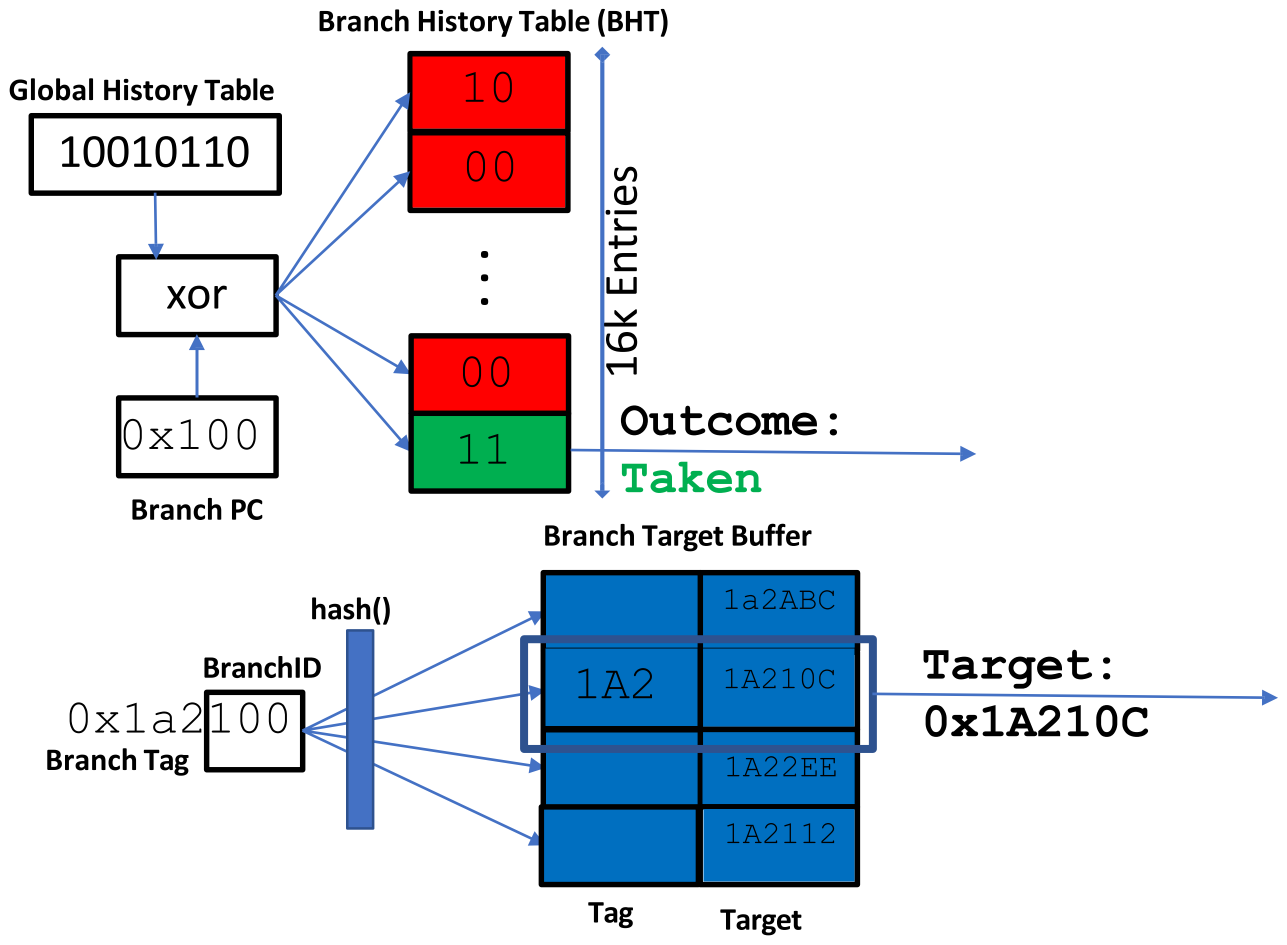
- **A design space is a set of possible incarnations of a system**
- **A design space is defined over a set of parameters**
- **A point in the design space is a concrete system with a concrete value for each of the design space's parameters**
- **Design spaces exist to allow systematic exploration of a collection of possible designs, like architectures.**

Example: Branch Predictor Design Space



What are the parameters / dimensions of this branch predictor's design?

Example: Branch Predictor Design Space



GHT size

BHT # entries

GHT/PC hash func

BHT entry size

BranchID hash

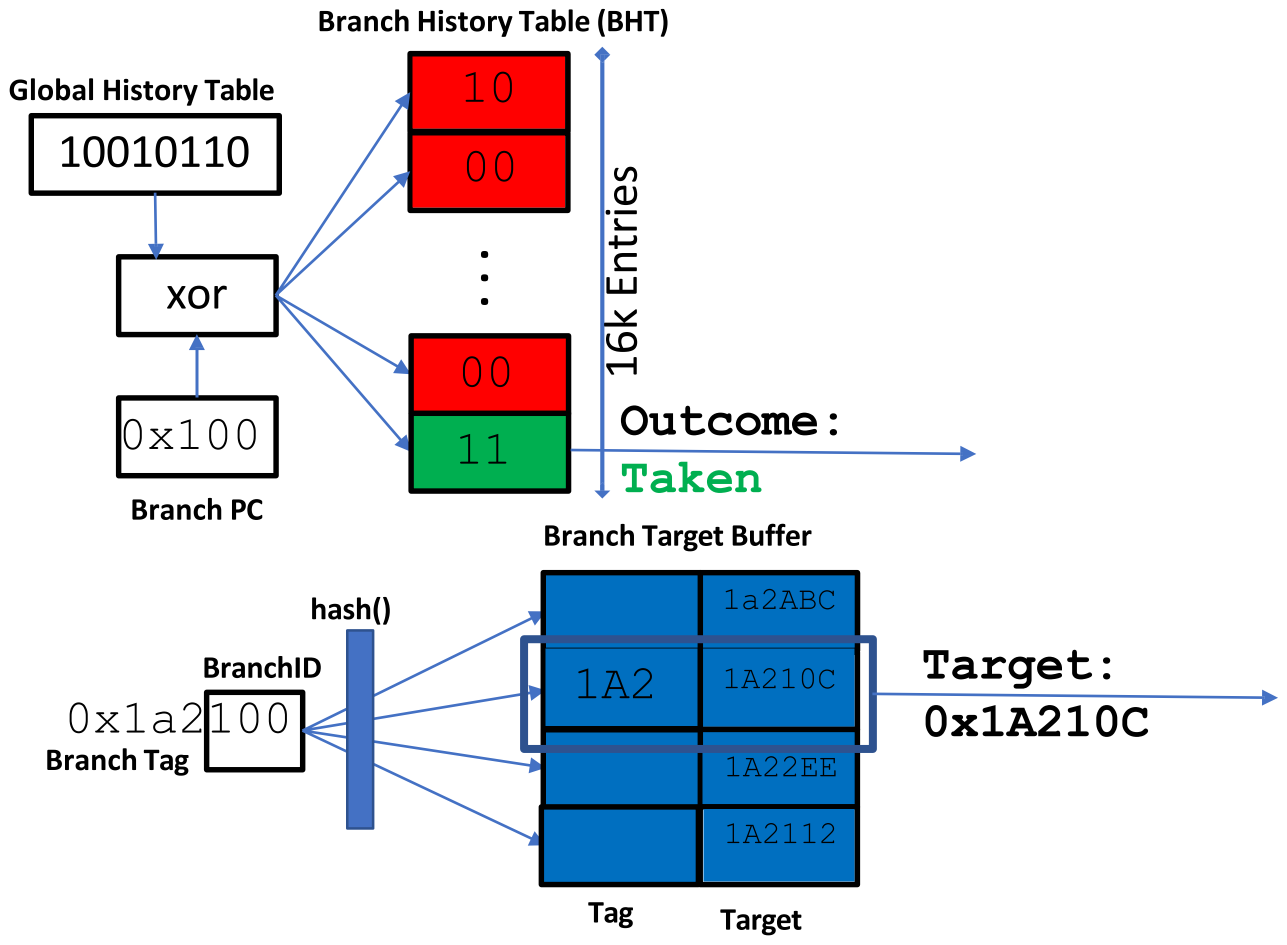
BTB # entries

BTB assoc

These parameters are the dimensions of a design space vector

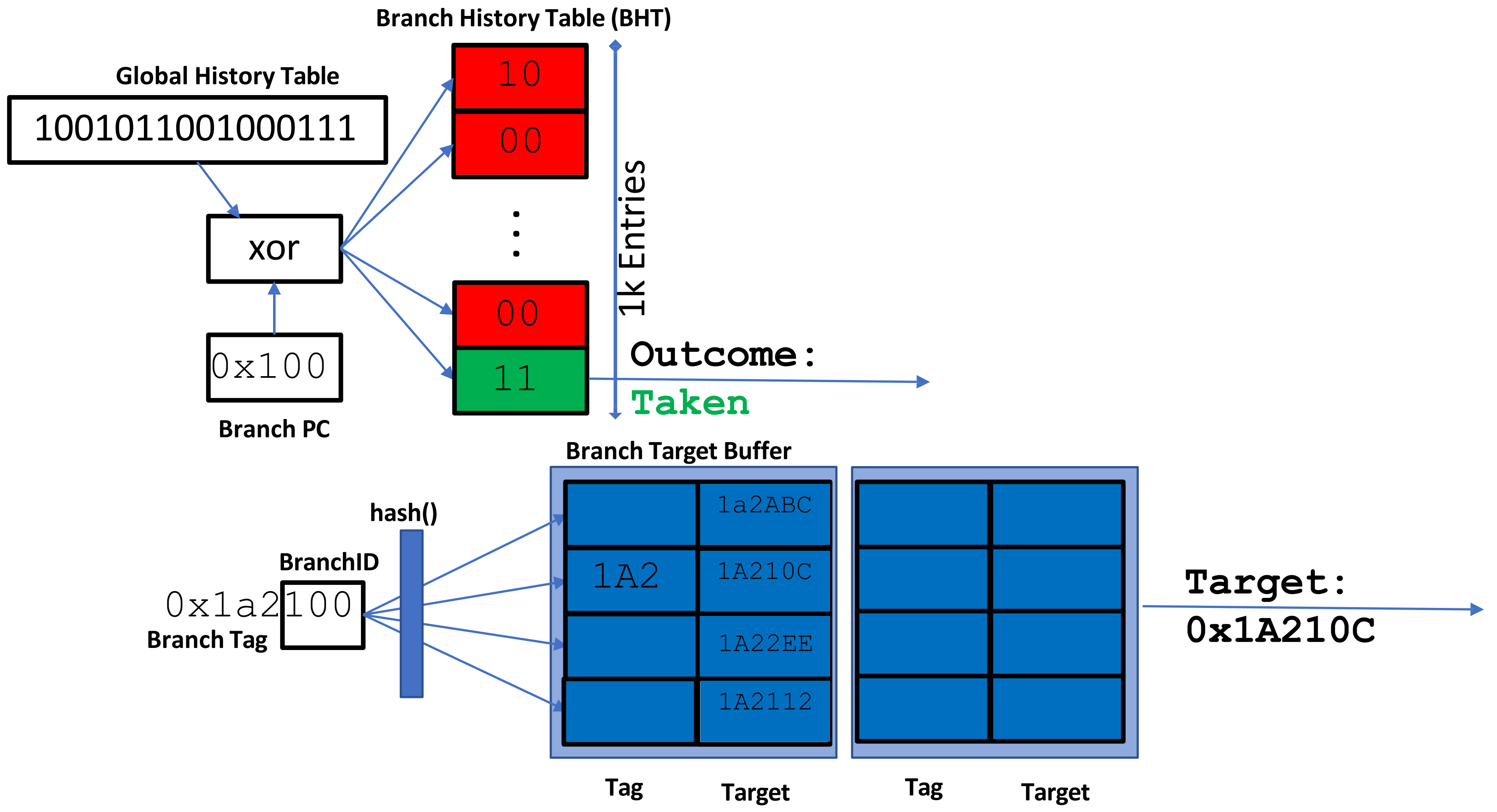
GHT size	Sg
BHT # entries	Nb
GHT/PC hash func	Hp
BHT entry size	Sb
BranchID hash	Hb
BTB # entries	Nt
BTB assoc	At

Example: Branch Predictor Design Space



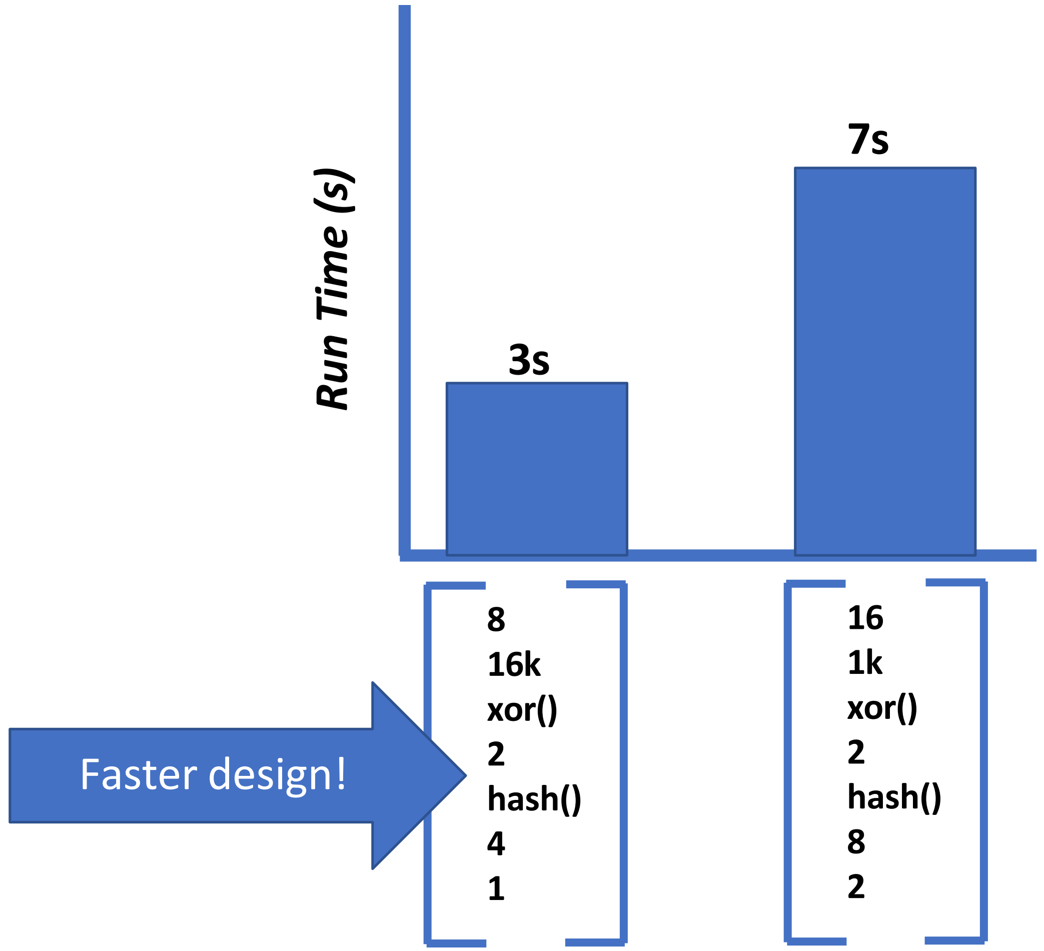
8	Sg
16k	Nb
xor()	Hp
2	Sb
hash()	Hb
4	Nt
1	At

Example: Branch Predictor Design Space

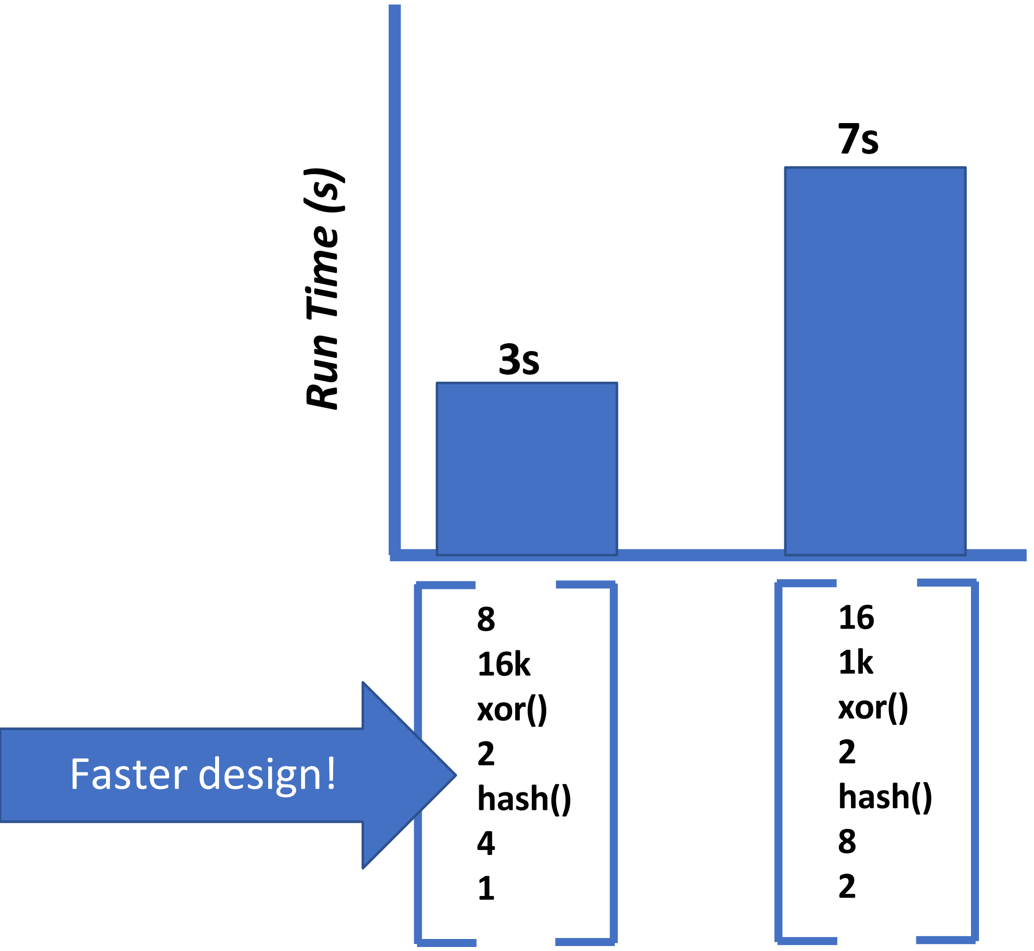


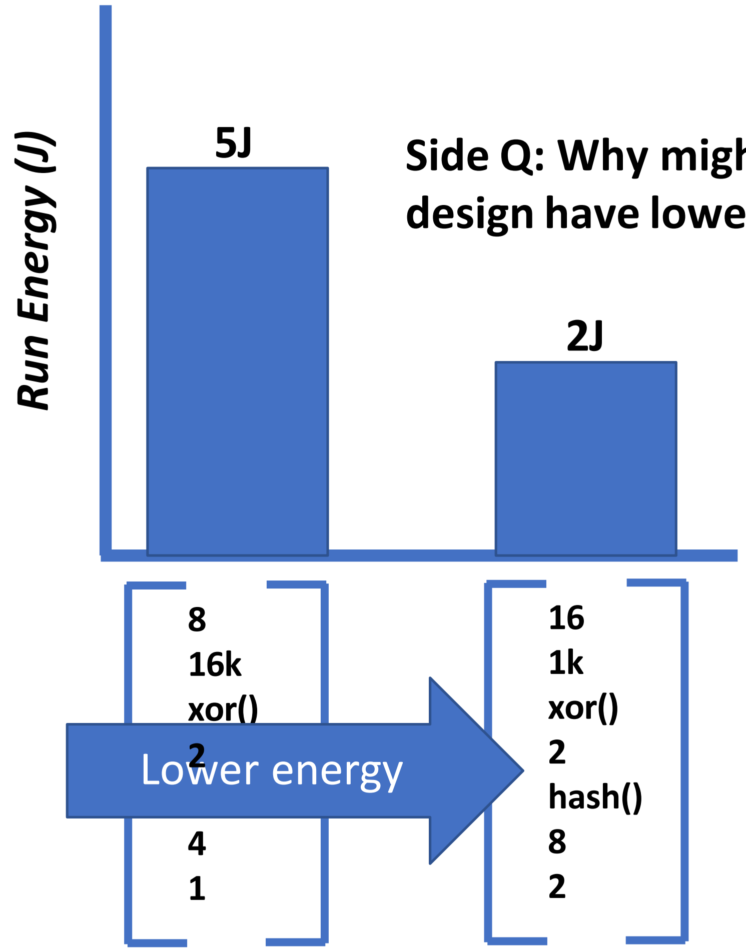
$$\begin{bmatrix} 16 \\ 1k \\ \text{xor}() \\ 2 \\ \text{hash}() \\ 8 \\ 2 \end{bmatrix} \quad \begin{bmatrix} Sg \\ Nb \\ Hp \\ Sb \\ Hb \\ Nt \\ At \end{bmatrix}$$

Can find a good design by measuring points in the design space



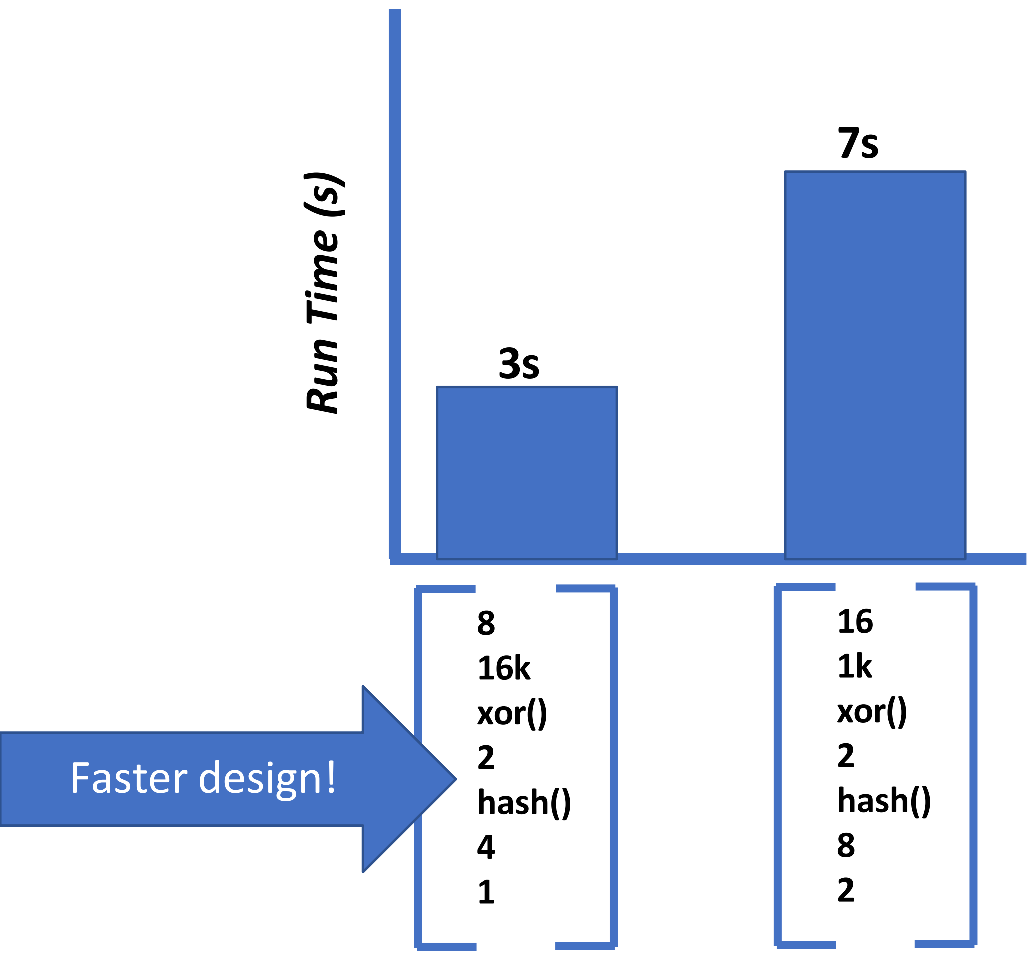
Can find a good design by measuring points in the design space

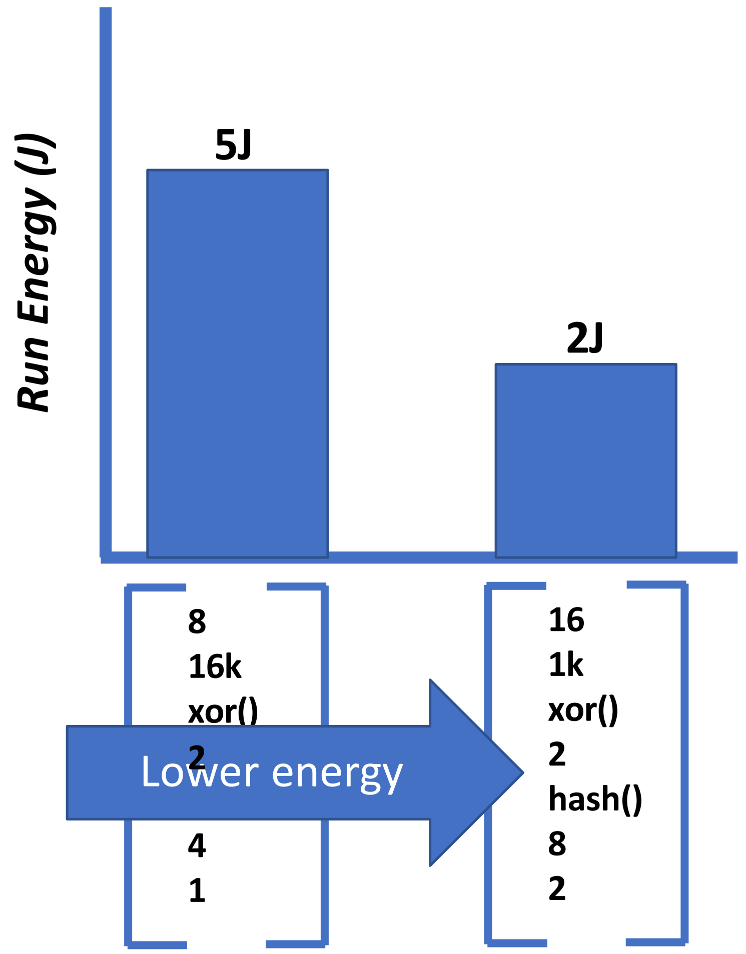




Side Q: Why might this particular design have lower energy?

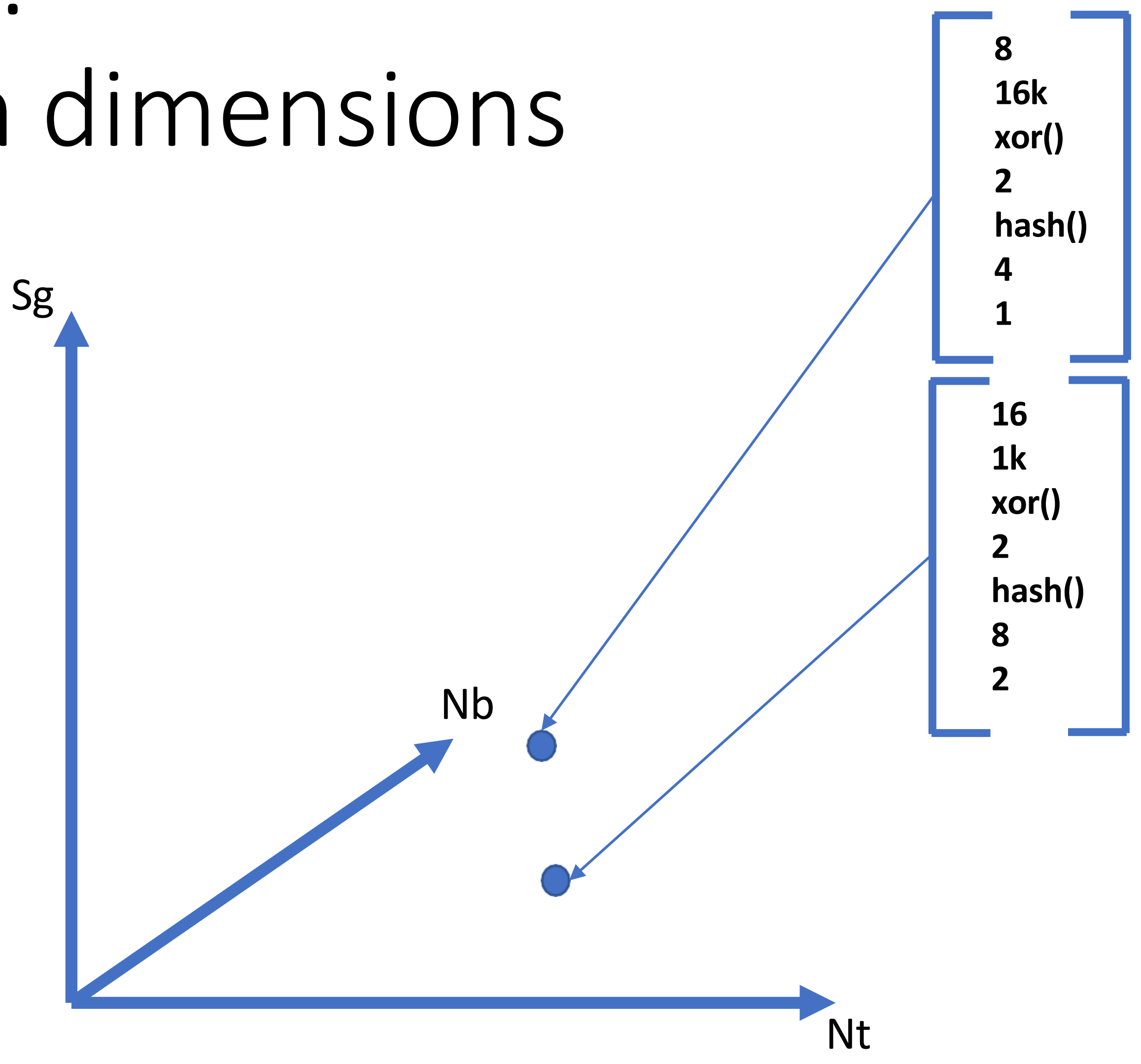
Is one of these better?





Plotting the design space:
Geometric view of design dimensions

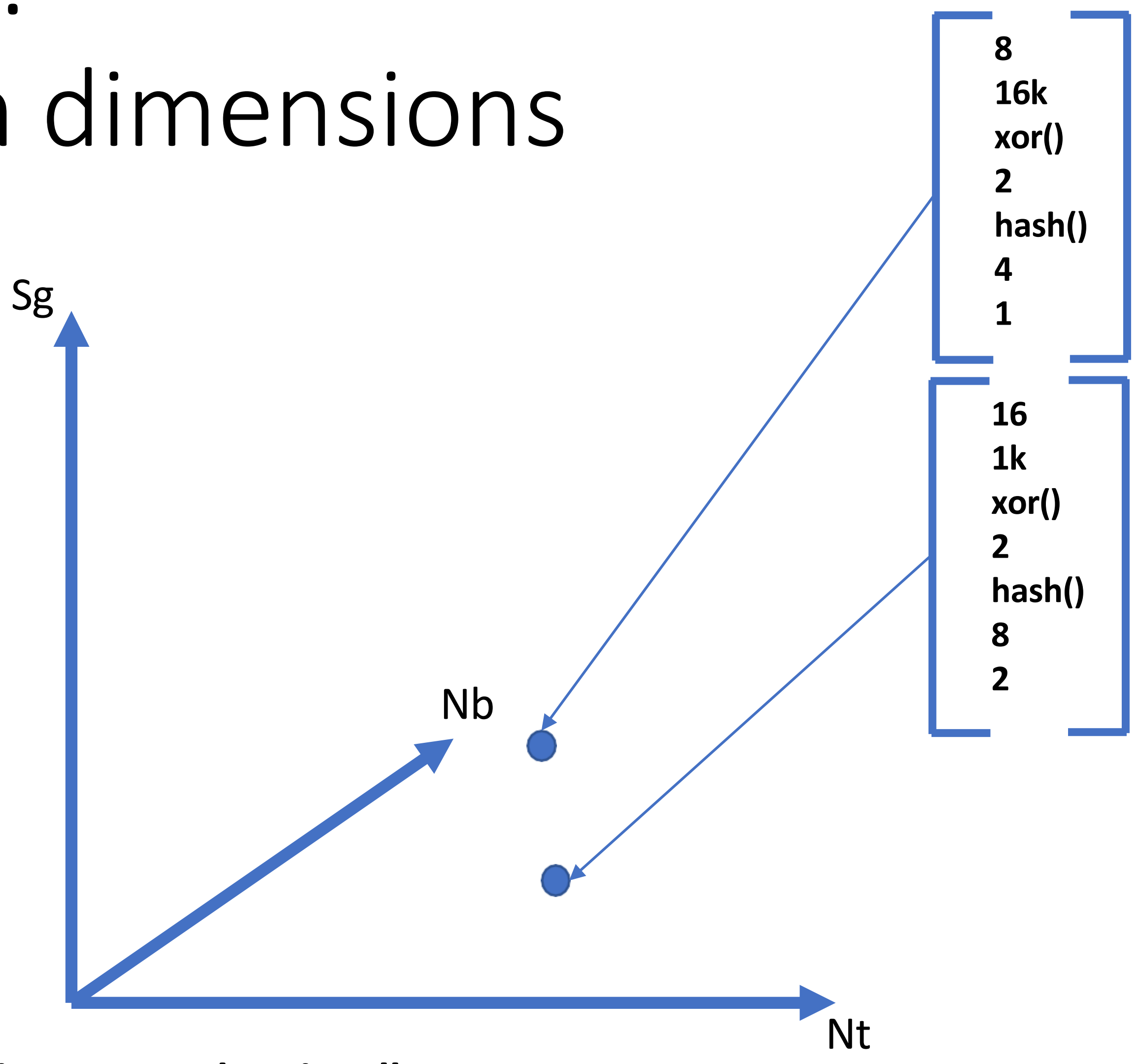
$$\begin{bmatrix} Sg \\ Nb \\ Hp \\ Sb \\ Hb \\ Nt \\ At \end{bmatrix} \Rightarrow$$



$$\begin{bmatrix} 8 \\ 16k \\ xor() \\ 2 \\ hash() \\ 4 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 16 \\ 1k \\ xor() \\ 2 \\ hash() \\ 8 \\ 2 \end{bmatrix}$$

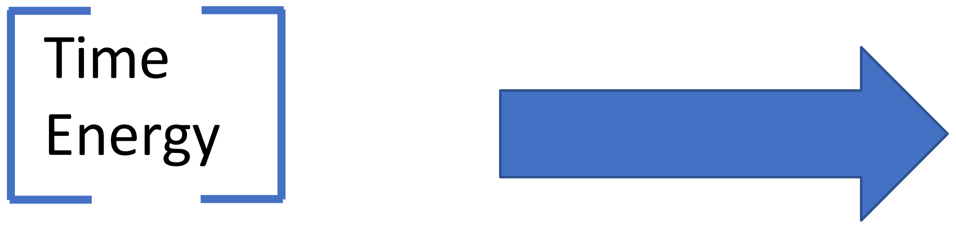
Plotting the design space: Geometric view of design dimensions

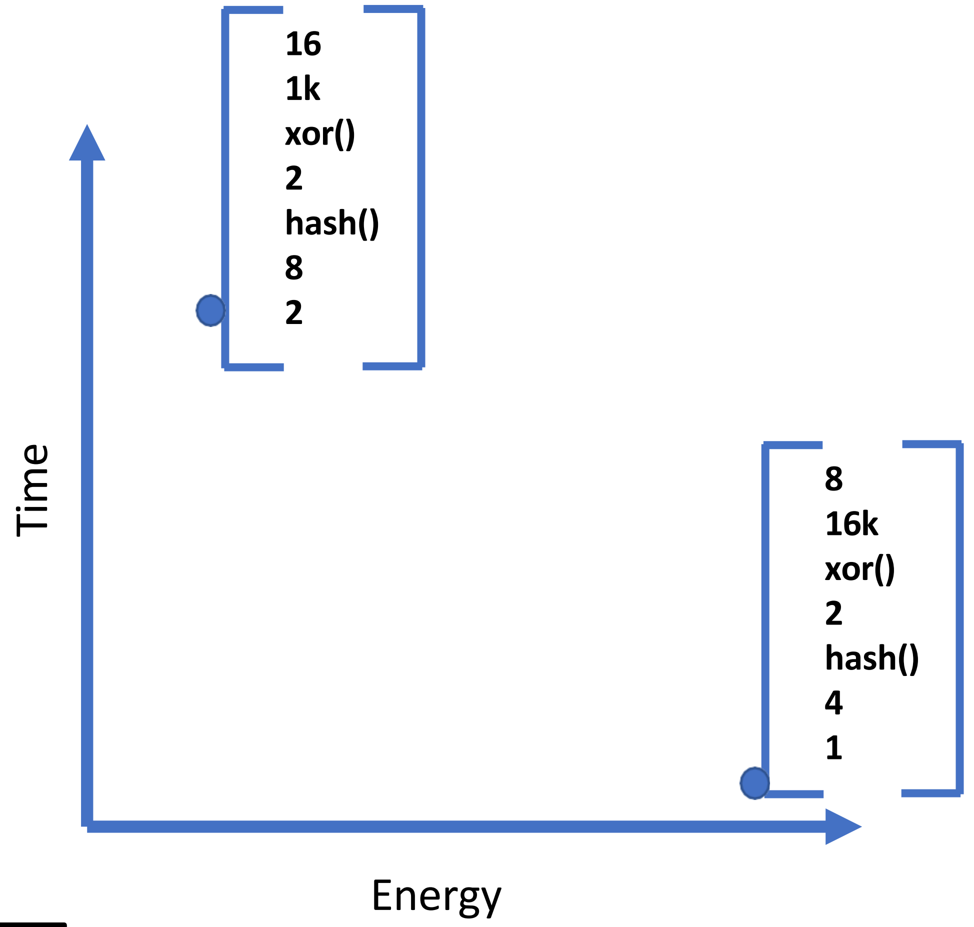
$$\begin{bmatrix} Sg \\ Nb \\ Hp \\ Sb \\ Hb \\ Nt \\ At \end{bmatrix} \Rightarrow$$



Limited medium: too many dimensions to render visually
Limited interpretability: what does position mean?
Can be helpful for clustering designs if non-obvious

Plotting the design space: Geometric view of figures of merit



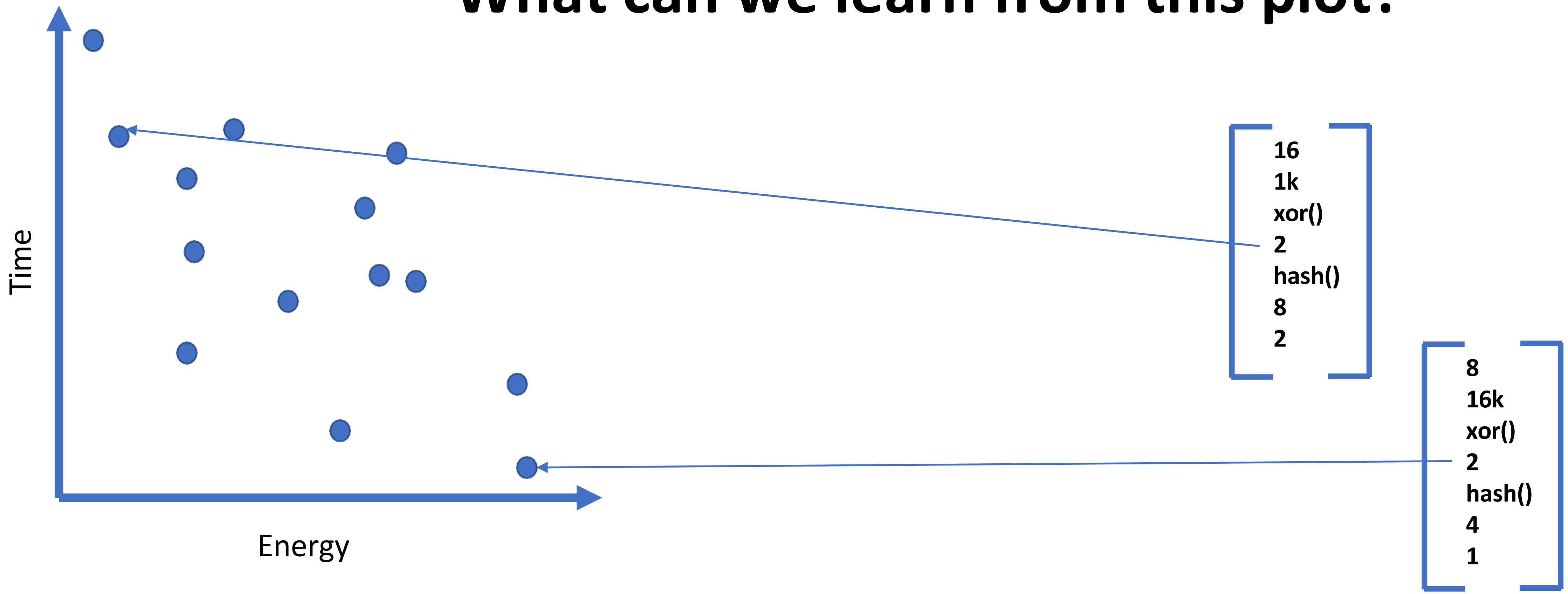


Simple medium: can easily render multiple FoMs & designs
Limited view of designs: points do not show design info
Benefit: allows comparing designs in multiple dimensions

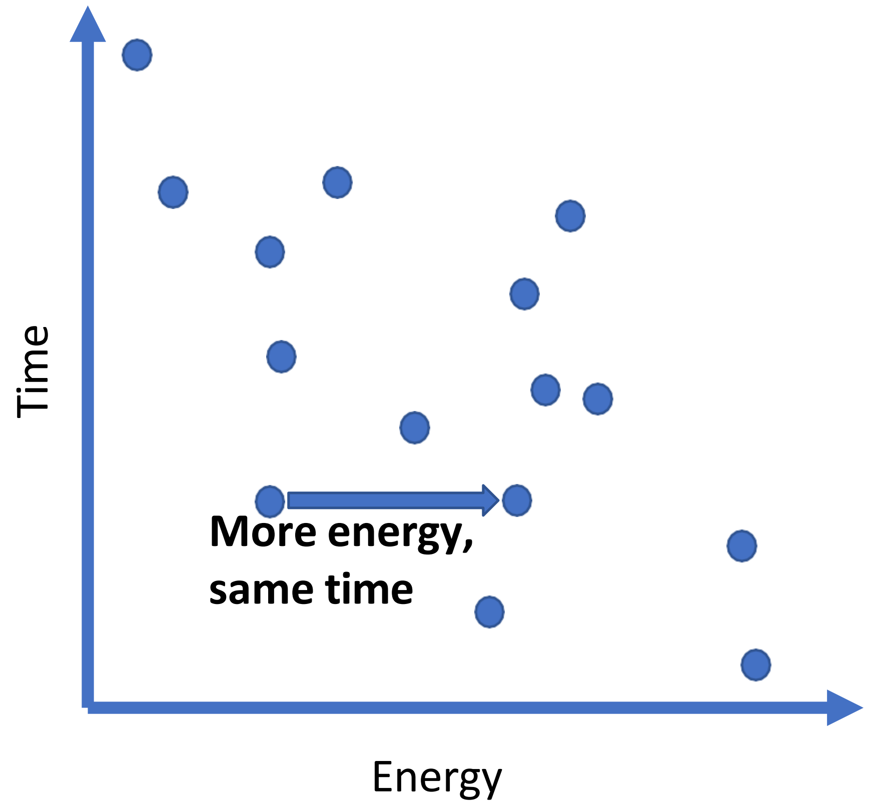
FoM = “Feature of Merit”, i.e. an attribute we care about.

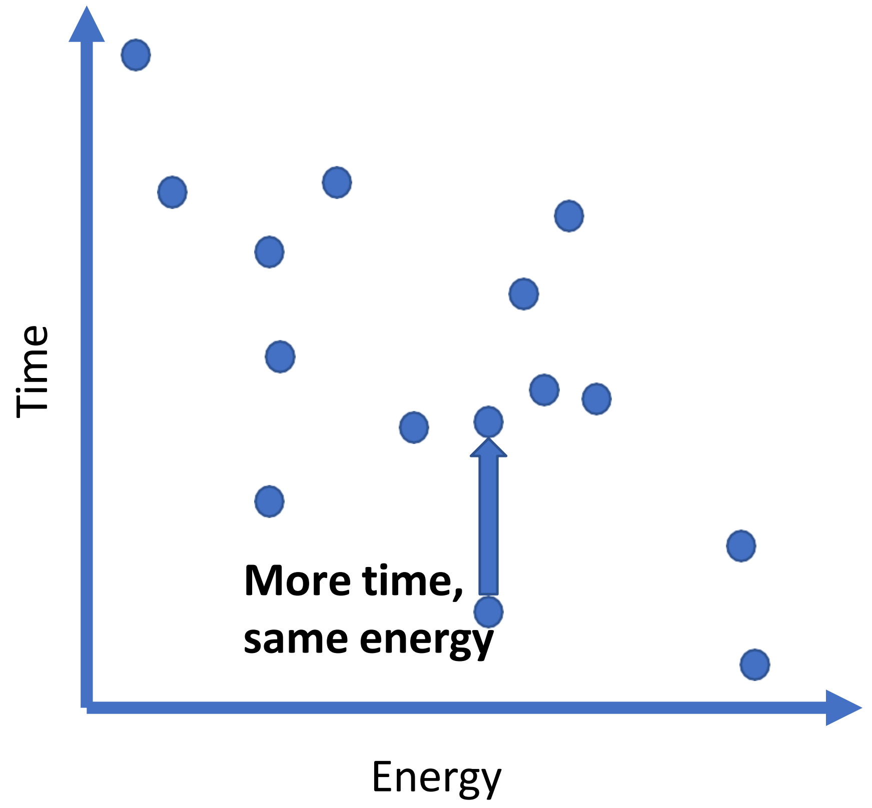
Plotting many designs to study a tradeoff

What can we learn from this plot?

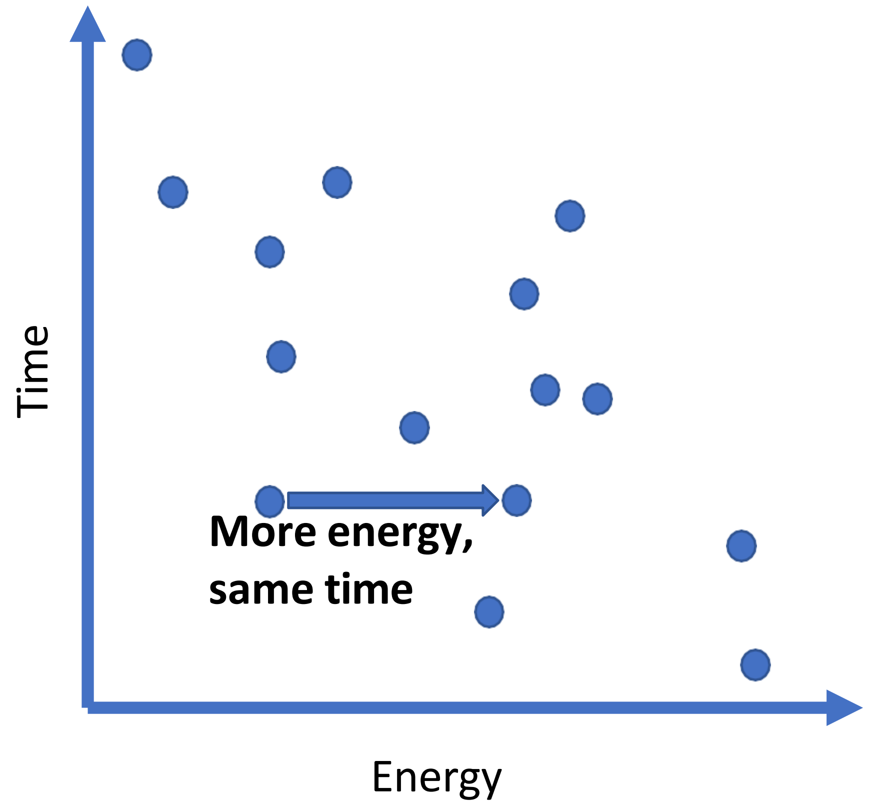


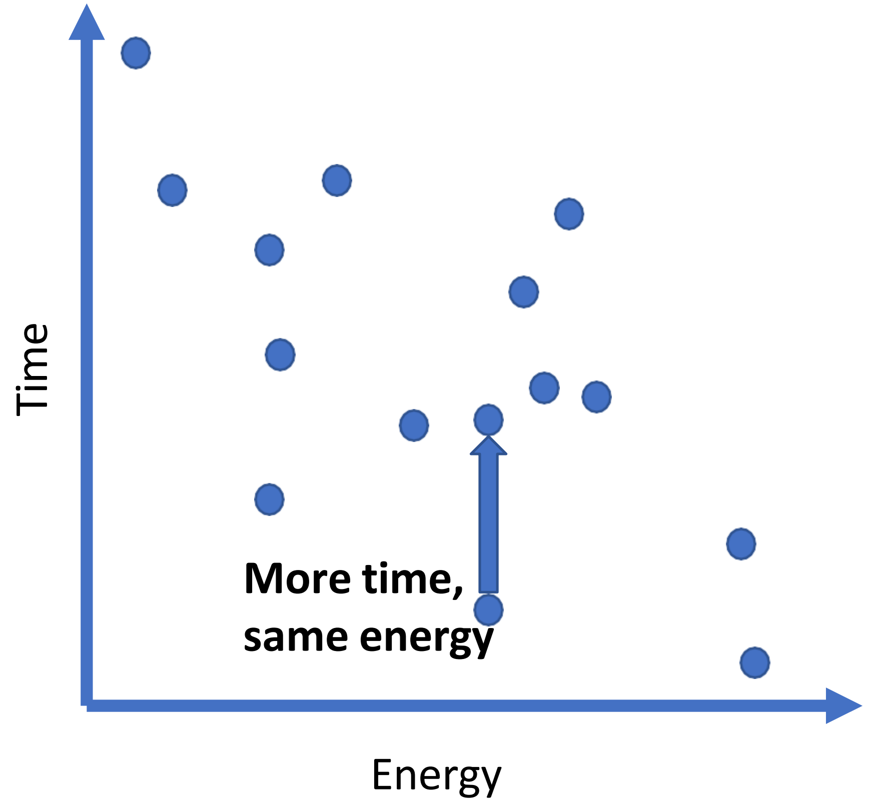
Plotting many designs to study a tradeoff



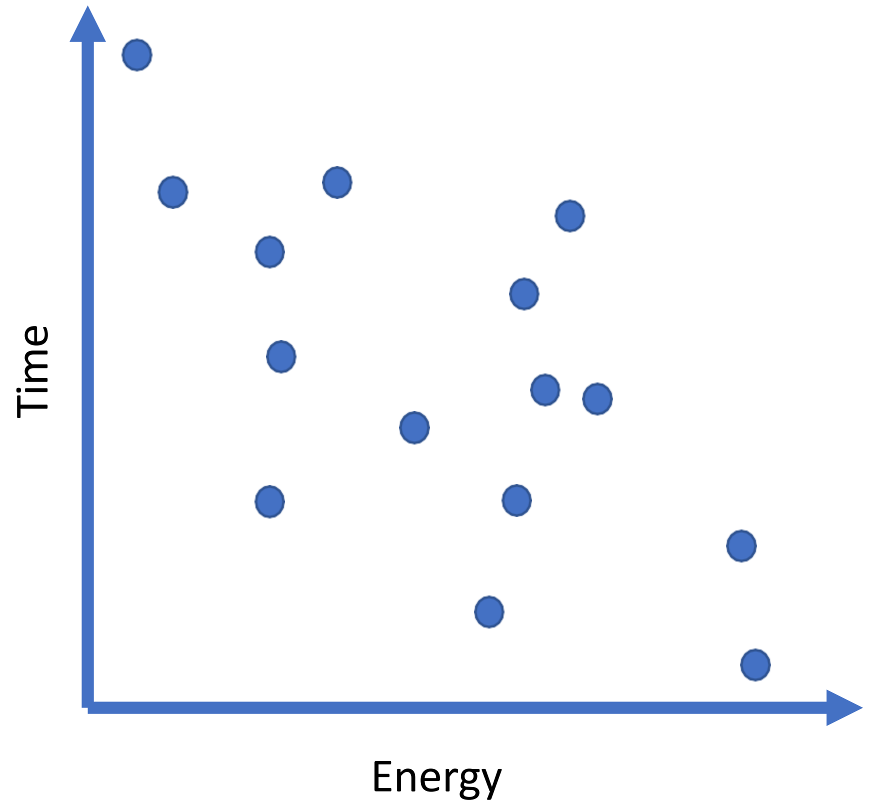


Which points in this plot are optimal?



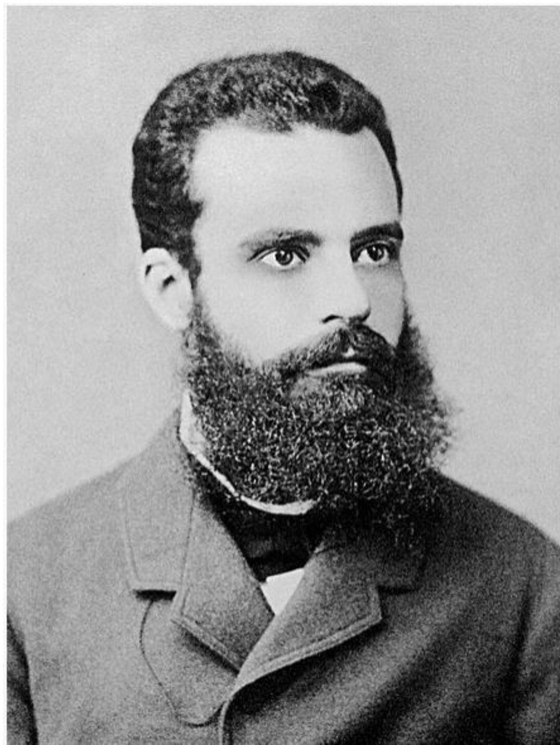


Pareto Optimality of Design Alternatives

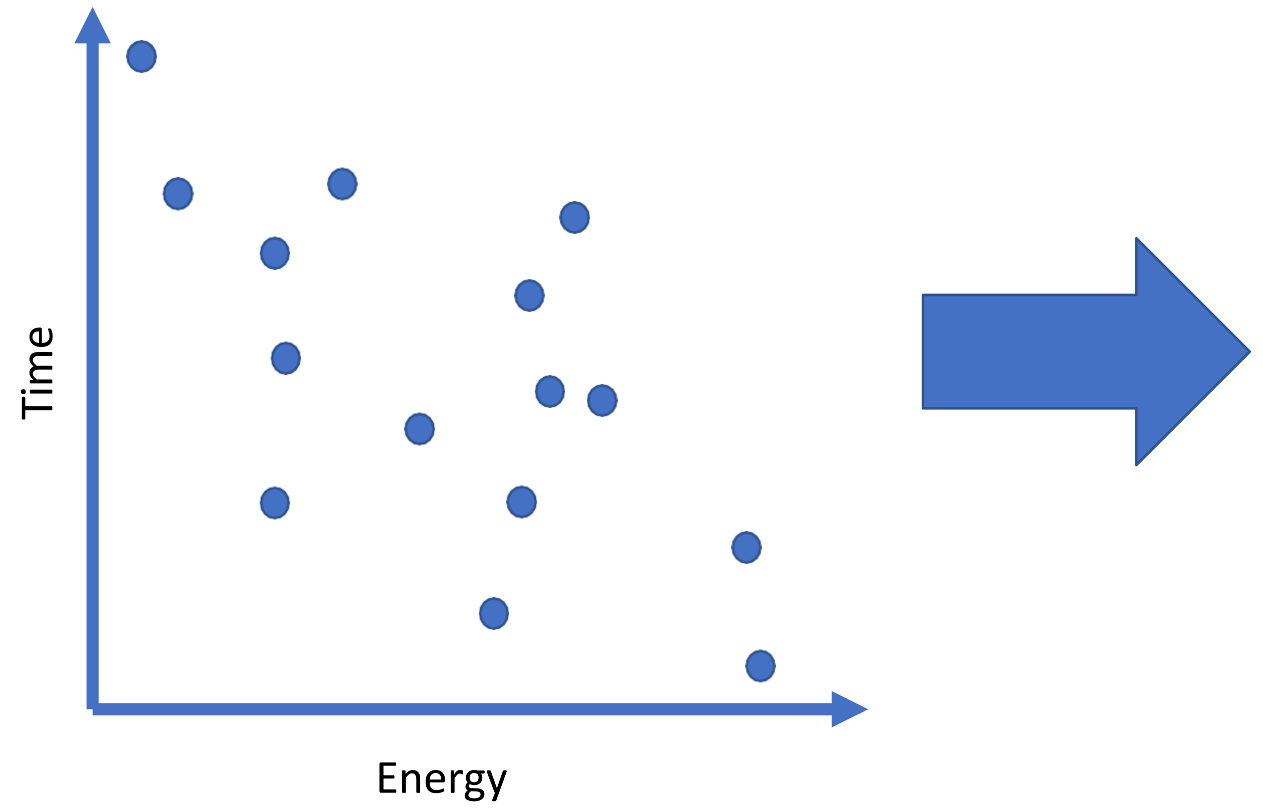


Pareto Optimality:
A design is optimal if no change leads to improvement in one dimension without a loss in at least one other dimension

Vilfredo Pareto



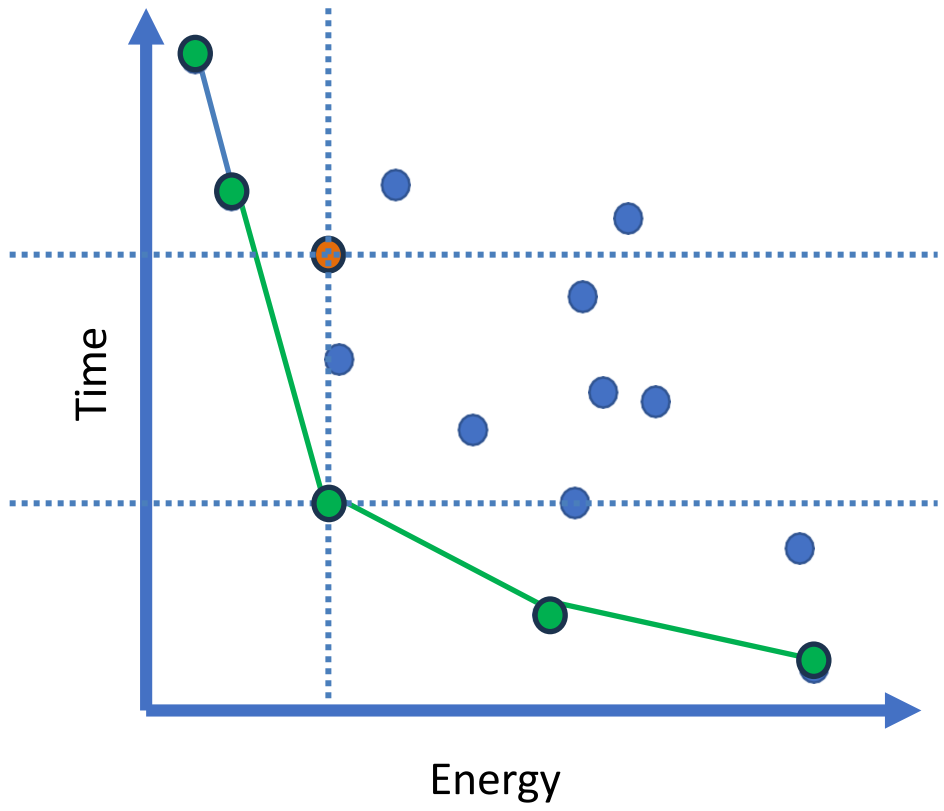
Pareto Optimality of Design Alternatives



All other points offer a level of performance in one dimension which can be had, or more, at a lower cost in the other dimension.

Design Consequence of Pareto Optimality

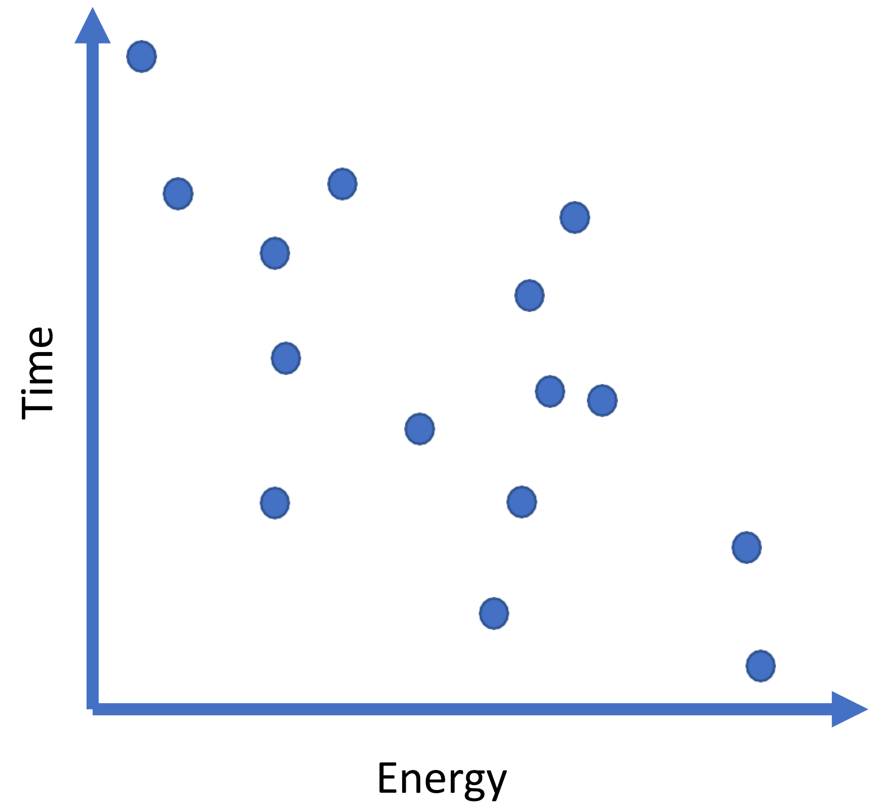
Never select designs other than at the frontier, at least without motivation outside of plot.
Any design anywhere other than at the frontier can achieve the same or better performance at a lower cost w.r.t. the plotted dimensions.

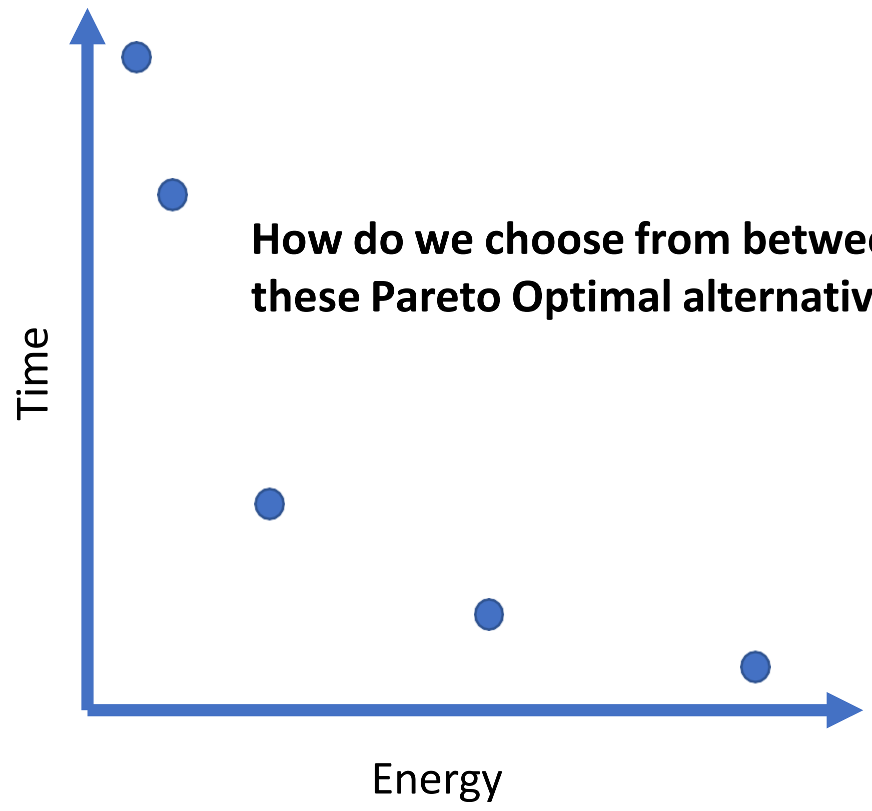


This level of energy efficiency can be had at a lower cost along the frontier

This point is strictly better, right?

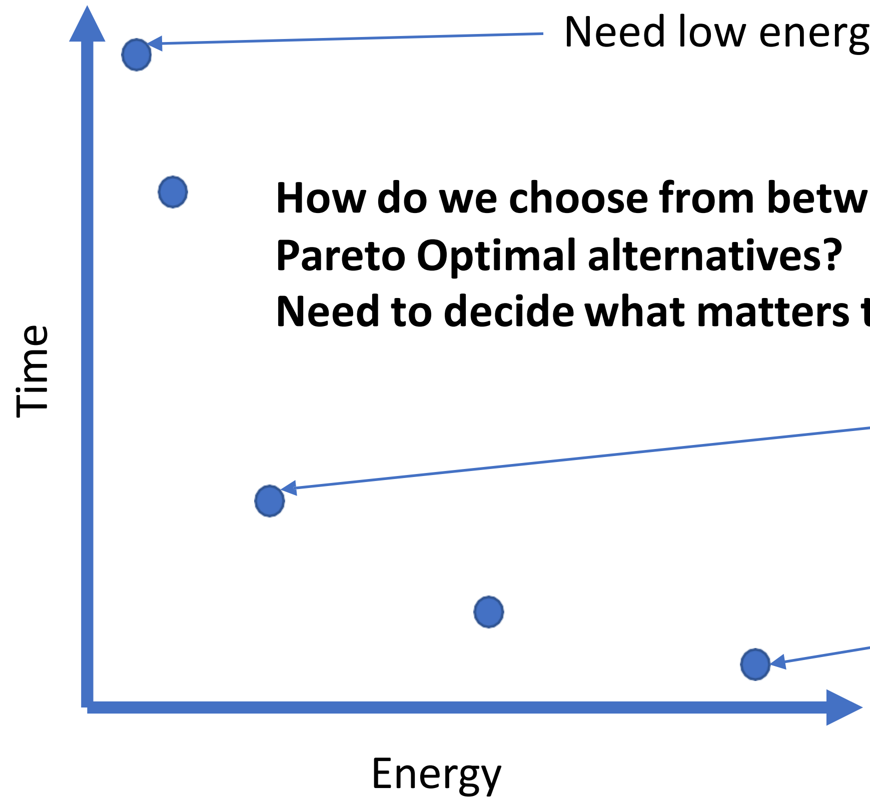
Design Consequence of Pareto Optimality





How do we choose from between these Pareto Optimal alternatives?

Worthwhile Options Are Along the Pareto Frontier



Need low energy (battery powered maybe?)

How do we choose from between these Pareto Optimal alternatives?
Need to decide what matters to you!

Want to strike a balance of performance and energy consumption (not make anyone too mad)

Need high performance (for latency requirement)

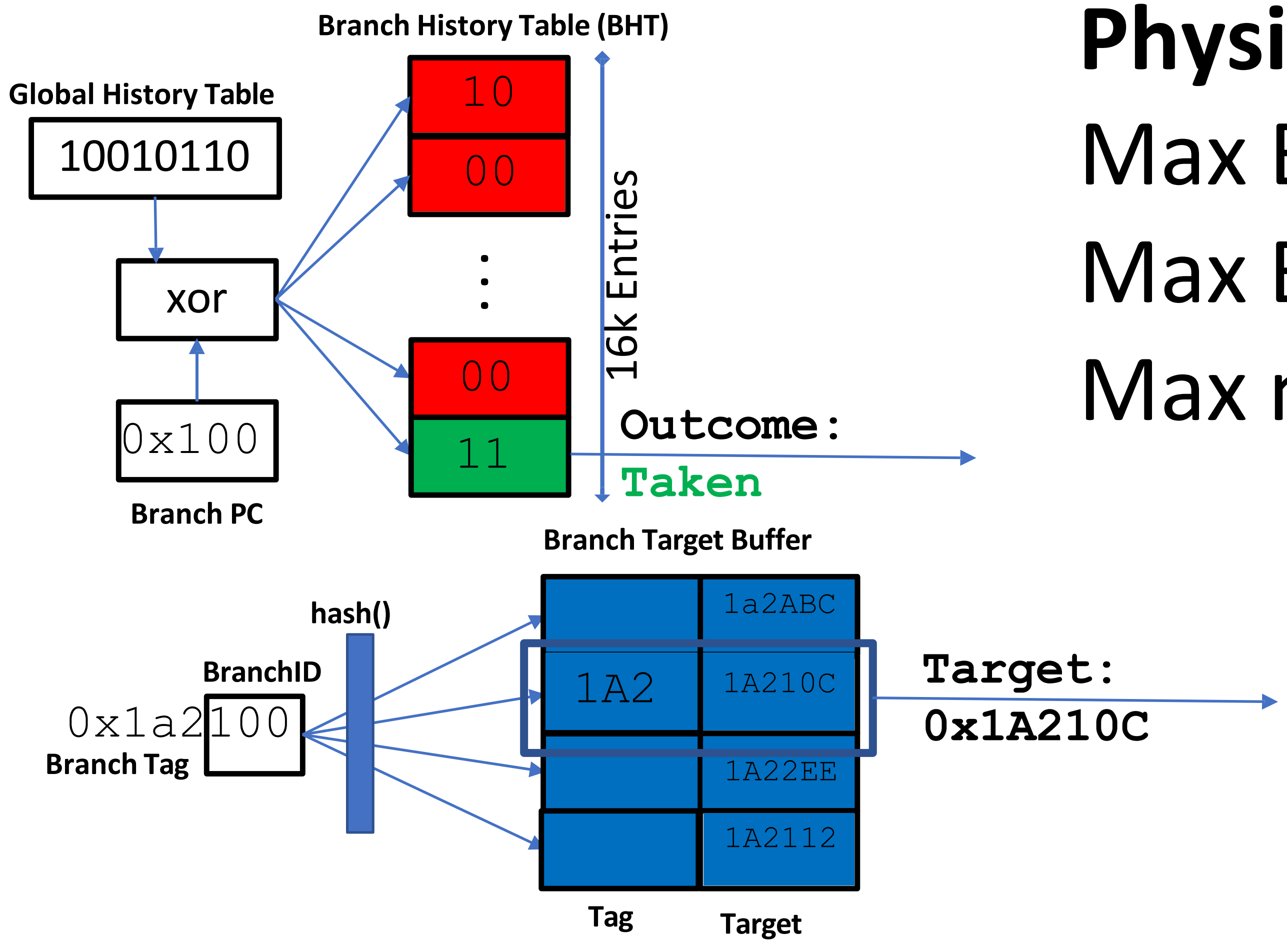
Design Space Exploration

- **Applied** Performance Evaluation to find the best feasible system
  - Define a system's important design parameters
  - Define a system's figure(s) of merit
  - Define a set of constraints on the feasibility of a binding of design parameters
  - Choose a feasible parameter setting and measure its merit
  - Iterate until satisfied:
    - If this system is better than the last one, keep it. If worse, discard it.
    - Choose a parameter and change it

Design Space Exploration

- **Applied** Performance Evaluation to find the best feasible system
  - Define a system’s important design parameters
  - Define a system’s figure(s) of merit
  - Define a set of constraints on the feasibility of a binding of design parameters
  - Choose a feasible parameter setting and measure its merit
  - Iterate until satisfied:
    - If this system is better than the last one, keep it. If worse, discard it.
    - Choose a parameter and change it

Constraining your design space



Physical design constraints

Max BP power = 4mW

Max BTB associativity = 2

Max memory (BTB+BHT) = 20kB

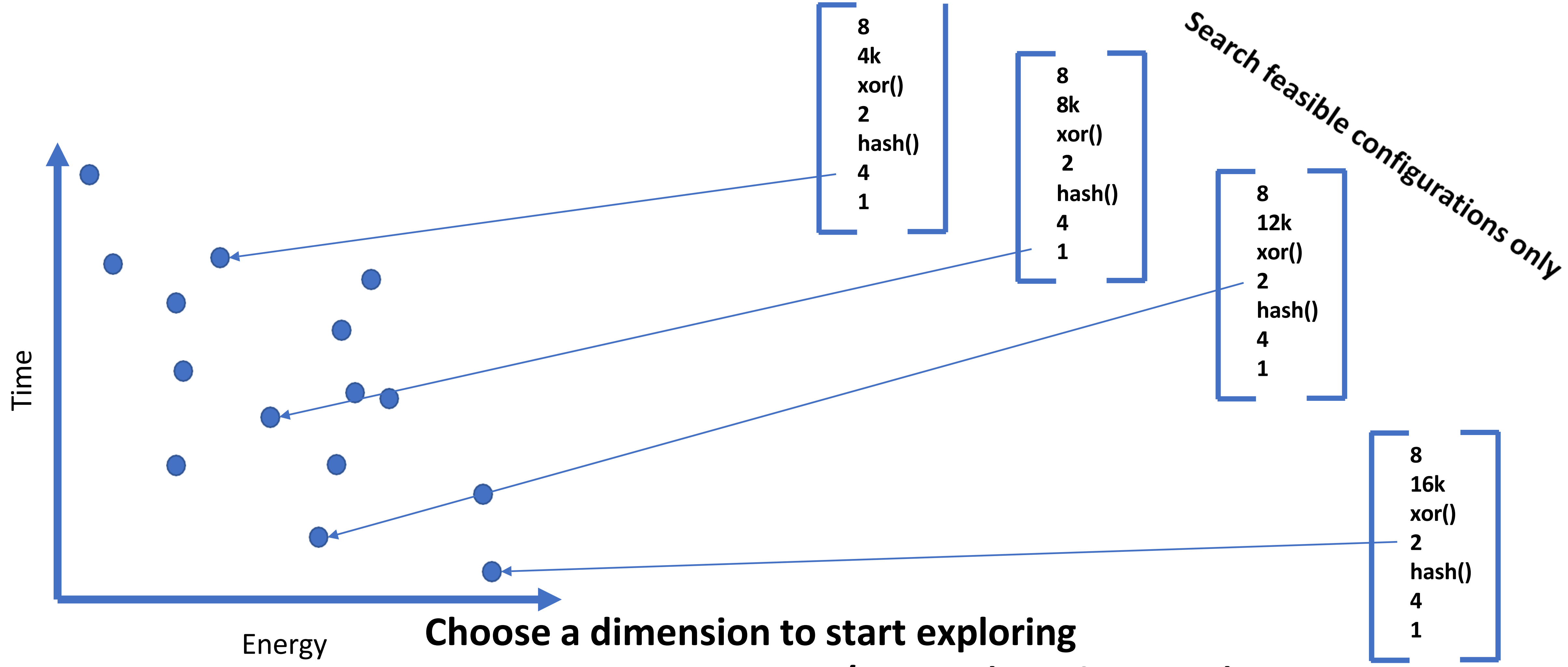
Designs candidates are often described as needing to "Make PPA":

- Power
- Performance
- Area

Design Space Exploration

- **Applied** Performance Evaluation to find the best feasible system
  - Define a system’s important design parameters
  - Define a system’s figure(s) of merit
  - Define a set of constraints on the feasibility of a binding of design parameters
  - Choose a feasible parameter setting and measure its merit
  - Iterate until satisfied:
    - If this system is better than the last one, keep it. If worse, discard it.
    - Choose a parameter and change it
    - Random restarts to search different sub-spaces

Systematically fill out your design space

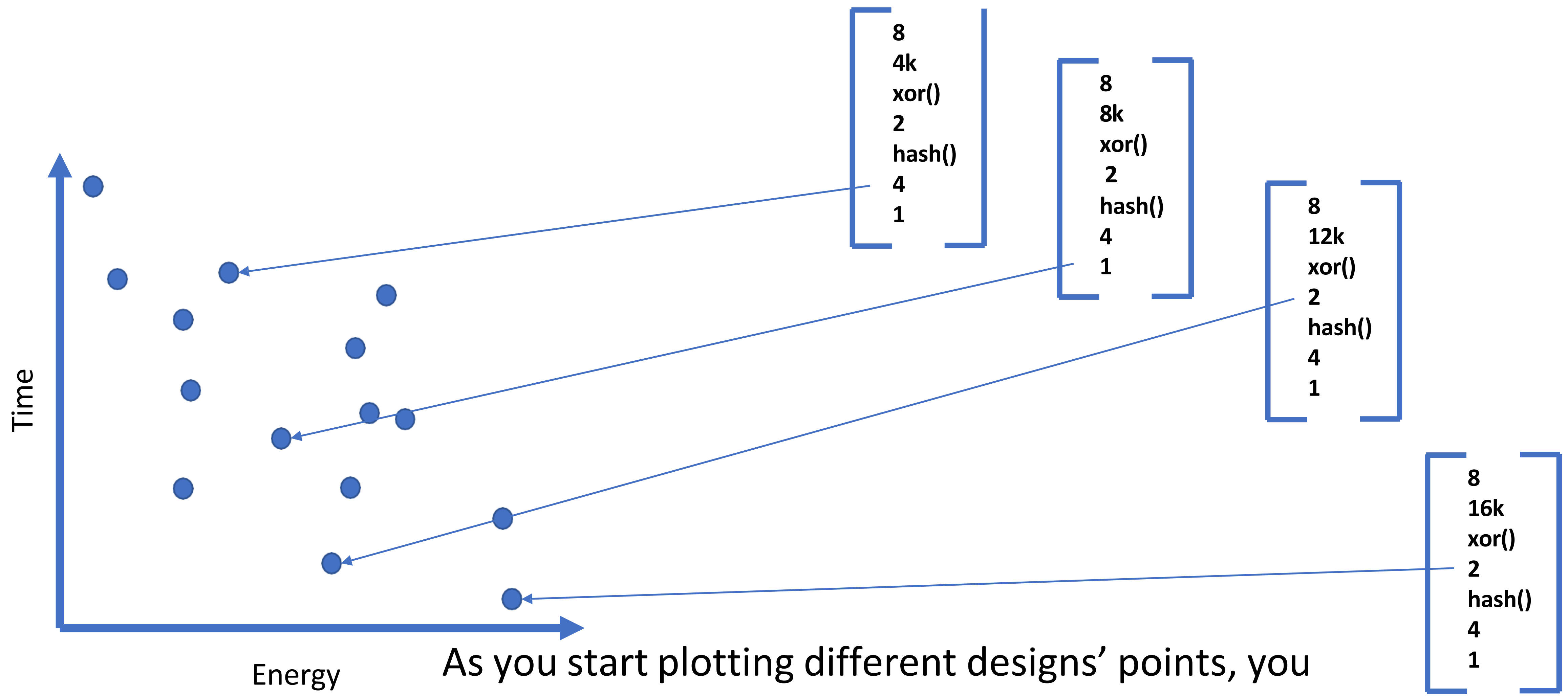


Choose a dimension to start exploring

Experiments to get time/energy (or other FoM)

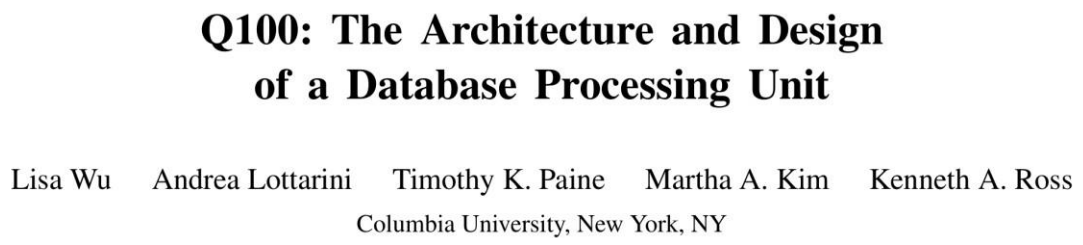
Change dimensions when way off frontier

Systematically fill out your design space



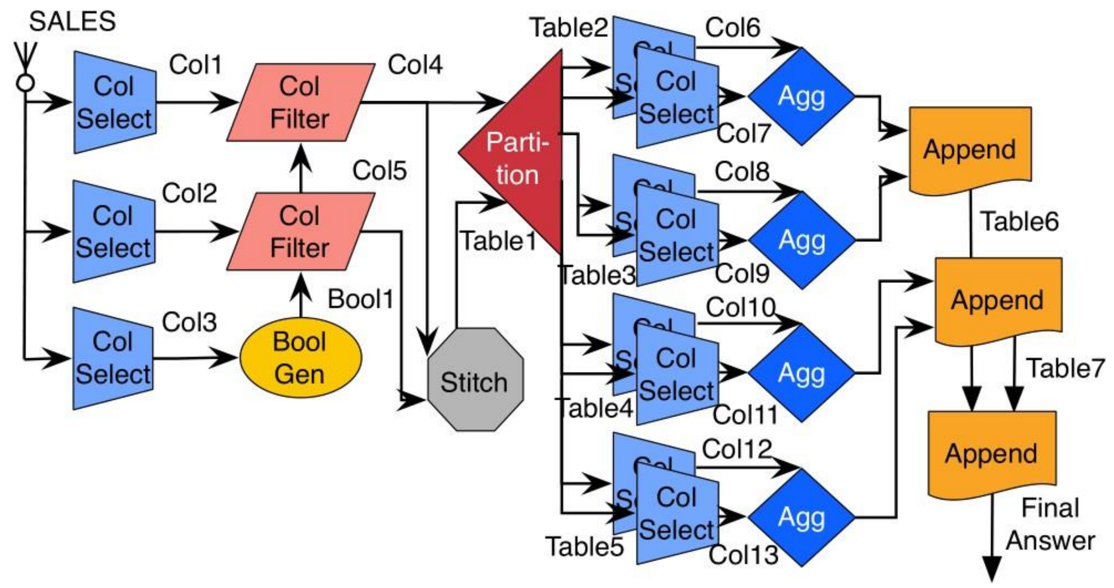
As you start plotting different designs' points, you will discover which dimensions matter to your space **(What do we learn from these labeled points?)**

Example of Design Space Optimization The Q100 Database Acceleration Architecture



Q100: The Architecture and Design of a Database Processing Unit

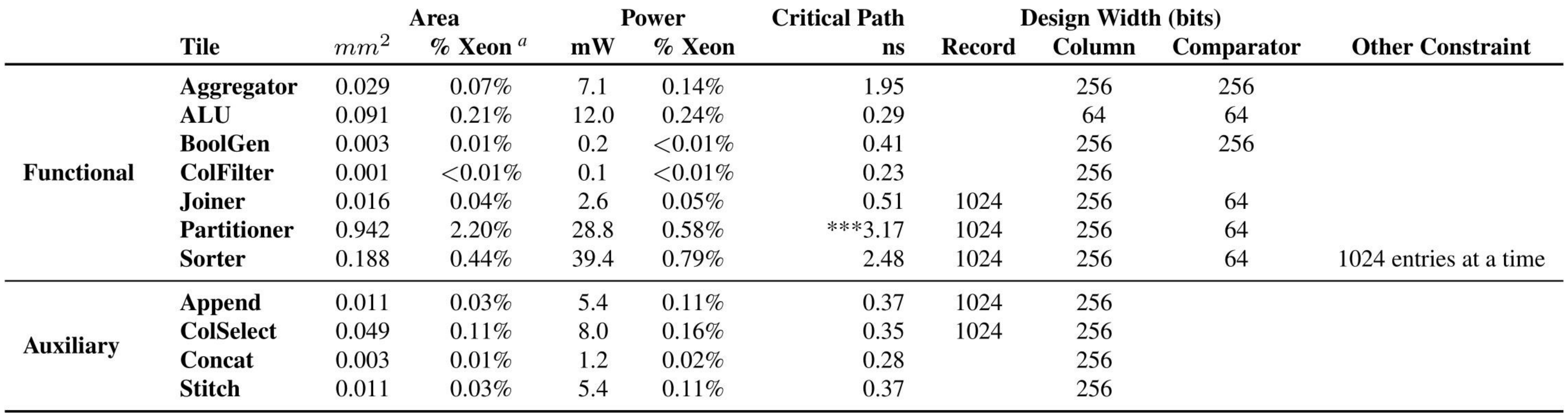
Lisa Wu Andrea Lottarini Timothy K. Paine Martha A. Kim Kenneth A. Ross
Columbia University, New York, NY



Cutting edge database query hardware accelerator

- “GPU for SQL & Database operations”
- Architecture built up of a collection of special computing tiles in hardware
- Each tile runs a particular kind of database operation
- Tiles connected by configurable wires that can be set up to make circuits to do a database query
- (Includes one of the best design space explorations I’ve encountered in a research paper)

Design Space Constraints
The Q100 Database Acceleration Architecture



		Area		Power		Critical Path	Design Width (bits)			
	Tile	mm^2	% Xeon [a]	mW	% Xeon	ns	Record	Column	Comparator	Other Constraint
Functional	Aggregator	0.029	0.07%	7.1	0.14%	1.95		256	256	
	ALU	0.091	0.21%	12.0	0.24%	0.29		64	64	
	BoolGen	0.003	0.01%	0.2	<0.01%	0.41		256	256	
	ColFilter	0.001	<0.01%	0.1	<0.01%	0.23		256		
	Joiner	0.016	0.04%	2.6	0.05%	0.51	1024	256	64	
	Partitioner	0.942	2.20%	28.8	0.58%	***3.17	1024	256	64	
	Sorter	0.188	0.44%	39.4	0.79%	2.48	1024	256	64	1024 entries at a time
Auxiliary	Append	0.011	0.03%	5.4	0.11%	0.37	1024	256		
	ColSelect	0.049	0.11%	8.0	0.16%	0.35	1024	256		
	Concat	0.003	0.01%	1.2	0.02%	0.28		256		
	Stitch	0.011	0.03%	5.4	0.11%	0.37		256		

Design space optimization problem statement:

Choose the right mixture of tiles to have the best performance and power without using too much area or limiting frequency

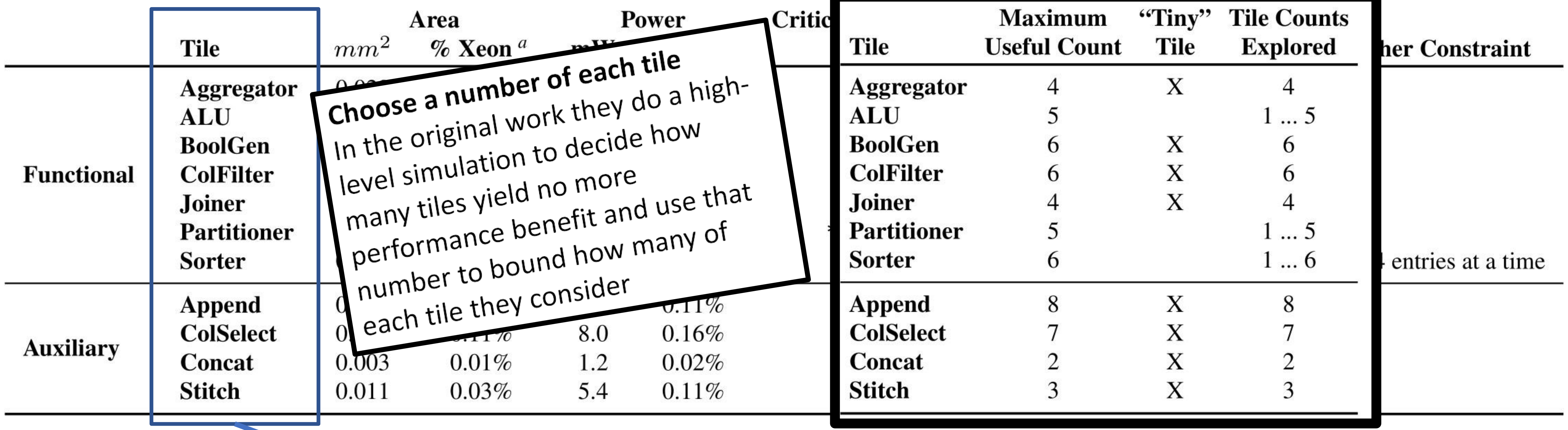
Design Space Constraints
The Q100 Database Acceleration Architecture

	Tile	Area		Power		Critical Path	Design Width (bits)			
		mm^2	% Xeon [a]	mW	[illegible]	ns	Record	Column	Comparator	Other Constraint
Functional	Aggregator	[illegible]	[illegible]	[illegible]	[illegible]	1.95		256	256	
	ALU	[illegible]	[illegible]	[illegible]	[illegible]	0.29		64	64	
	BoolGen	[illegible]	[illegible]	[illegible]	[illegible]	0.41		256	256	
	ColFilter	[illegible]	[illegible]	[illegible]	[illegible]	0.23		256		
	Joiner	[illegible]	[illegible]	[illegible]	[illegible]	0.51	1024	256	64	
	Partitioner	[illegible]	[illegible]	[illegible]	[illegible]	***3.17	1024	256	64	
	Sorter	[illegible]	[illegible]	[illegible]	[illegible]	2.48	1024	256	64	1024 entries at a time
Auxiliary	Append	[illegible]	[illegible]	[illegible]	0.11%	0.37	1024	256		
	ColSelect	[illegible]	[illegible]	8.0	0.16%	0.35	1024	256		
	Concat	0.003	0.01%	1.2	0.02%	0.28		256		
	Stitch	0.011	0.03%	5.4	0.11%	0.37		256		

Choose a number of each tile
In the original work they do a high-level simulation to decide how many tiles yield no more performance benefit and use that number to bound how many of each tile they consider

Design space optimization problem statement:
Choose the right mixture of tiles to have the best performance and power without using too much area or limiting frequency

Design Space Constraints
The Q100 Database Acceleration Architecture



	Tile	Area mm^2	Area % Xeon [a]	Power mW	Power	Critic	her Constraint
Functional	Aggregator						
	ALU						
	BoolGen						
	ColFilter						
	Joiner						
	Partitioner						
	Sorter						entries at a time
Auxiliary	Append	0			0.11%		
	ColSelect	0		8.0	0.16%		
	Concat	0.003	0.01%	1.2	0.02%		
	Stitch	0.011	0.03%	5.4	0.11%		

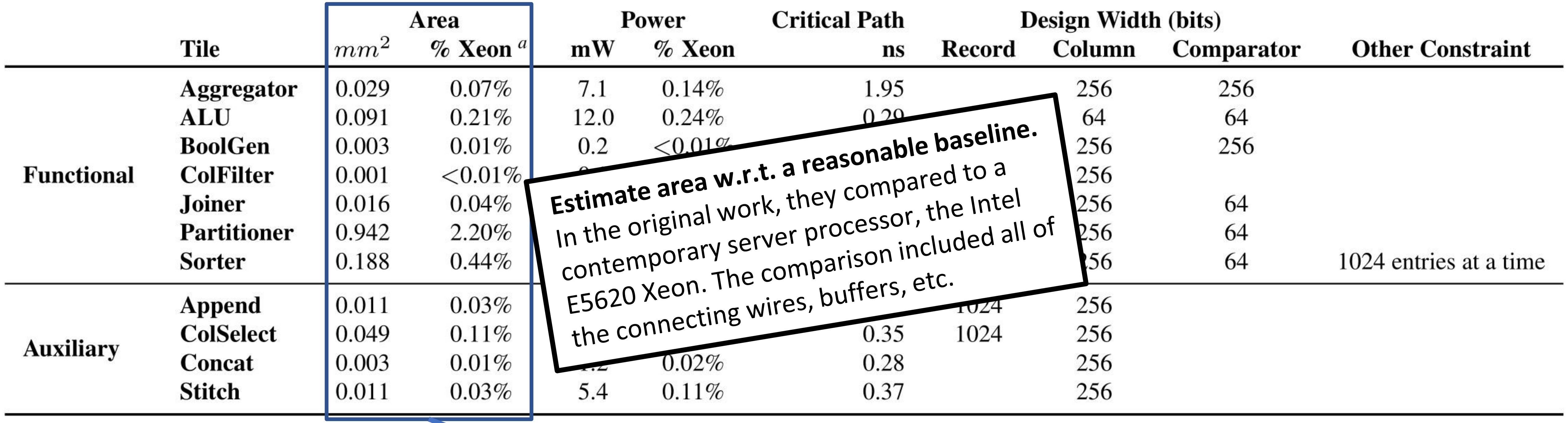
Choose a number of each tile
In the original work they do a high-level simulation to decide how many tiles yield no more performance benefit and use that number to bound how many of each tile they consider

Tile	Maximum Useful Count	"Tiny" Tile	Tile Counts Explored
Aggregator	4	X	4
ALU	5		1 ... 5
BoolGen	6	X	6
ColFilter	6	X	6
Joiner	4	X	4
Partitioner	5		1 ... 5
Sorter	6		1 ... 6
Append	8	X	8
ColSelect	7	X	7
Concat	2	X	2
Stitch	3	X	3

Design space optimization problem statement:

Choose the right mixture of tiles to have the best performance and power without using too much area or limiting frequency

Design Space Constraints
The Q100 Database Acceleration Architecture



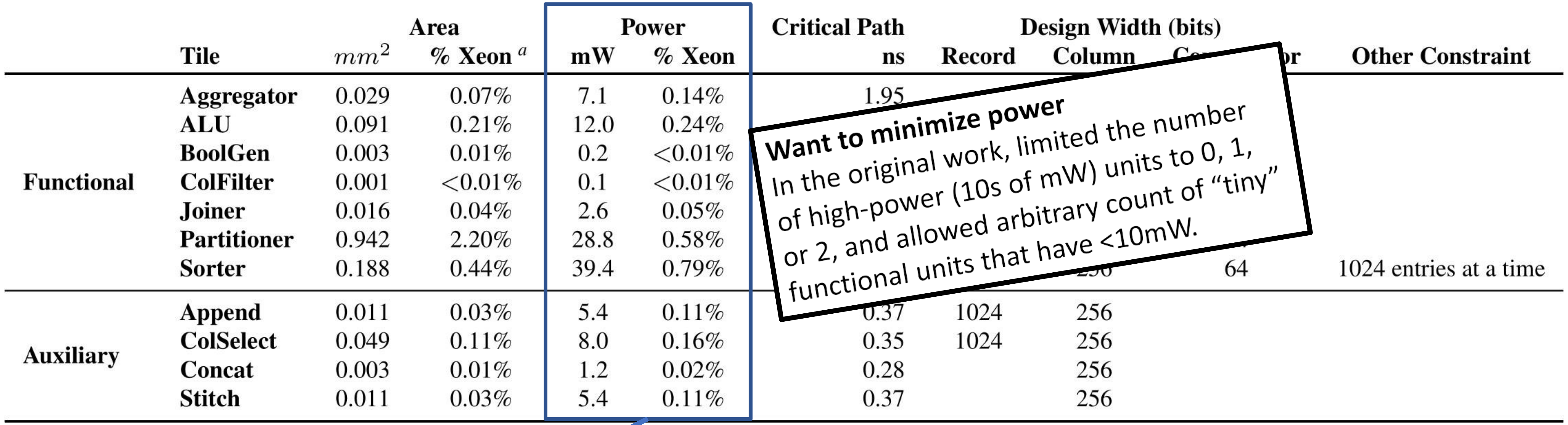
		Area		Power		Critical Path	Design Width (bits)			
	Tile	mm^2	% Xeon [a]	mW	% Xeon	ns	Record	Column	Comparator	Other Constraint
Functional	Aggregator	0.029	0.07%	7.1	0.14%	1.95		256	256	
	ALU	0.091	0.21%	12.0	0.24%	0.29	[illegible]	64	64	
	BoolGen	0.003	0.01%	0.2	<0.01%	[illegible]	[illegible]	256	256	
	ColFilter	0.001	<0.01%	[illegible]	[illegible]	[illegible]	[illegible]	256		
	Joiner	0.016	0.04%	[illegible]	[illegible]	[illegible]	[illegible]	256	64	
	Partitioner	0.942	2.20%	[illegible]	[illegible]	[illegible]	[illegible]	256	64	
	Sorter	0.188	0.44%	[illegible]	[illegible]	[illegible]	[illegible]	256	64	1024 entries at a time
Auxiliary	Append	0.011	0.03%	[illegible]	[illegible]	[illegible]	1024	256		
	ColSelect	0.049	0.11%	[illegible]	[illegible]	0.35	1024	256		
	Concat	0.003	0.01%	1.2	0.02%	0.28		256		
	Stitch	0.011	0.03%	5.4	0.11%	0.37		256		

Estimate area w.r.t. a reasonable baseline.
In the original work, they compared to a contemporary server processor, the Intel E5620 Xeon. The comparison included all of the connecting wires, buffers, etc.

Design space optimization problem statement:

Choose the right mixture of tiles to have the best performance and power without using too much area or limiting frequency

Design Space Constraints
The Q100 Database Acceleration Architecture

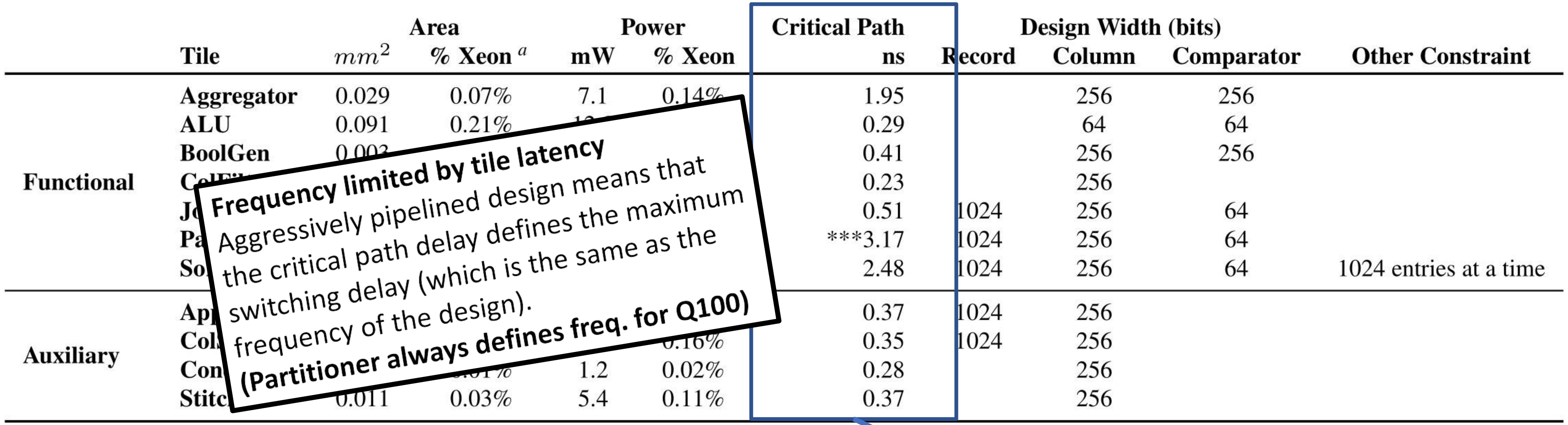


		Area		Power		Critical Path	Design Width (bits)			
	Tile	mm^2	% Xeon [a]	mW	% Xeon	ns	Record	Column	Co[illegible]or	Other Constraint
Functional	Aggregator	0.029	0.07%	7.1	0.14%	1.95				
	ALU	0.091	0.21%	12.0	0.24%	[illegible]				
	BoolGen	0.003	0.01%	0.2	<0.01%	[illegible]				
	ColFilter	0.001	<0.01%	0.1	<0.01%	[illegible]				
	Joiner	0.016	0.04%	2.6	0.05%	[illegible]				
	Partitioner	0.942	2.20%	28.8	0.58%	[illegible]				
	Sorter	0.188	0.44%	39.4	0.79%	[illegible]		256	64	1024 entries at a time
Auxiliary	Append	0.011	0.03%	5.4	0.11%	0.37	1024	256		
	ColSelect	0.049	0.11%	8.0	0.16%	0.35	1024	256		
	Concat	0.003	0.01%	1.2	0.02%	0.28		256		
	Stitch	0.011	0.03%	5.4	0.11%	0.37		256		

Want to minimize power
In the original work, limited the number of high-power (10s of mW) units to 0, 1, or 2, and allowed arbitrary count of "tiny" functional units that have <10mW.

Design space optimization problem statement:
Choose the right mixture of tiles to have the best performance and power without using too much area or limiting frequency

Design Space Constraints
The Q100 Database Acceleration Architecture

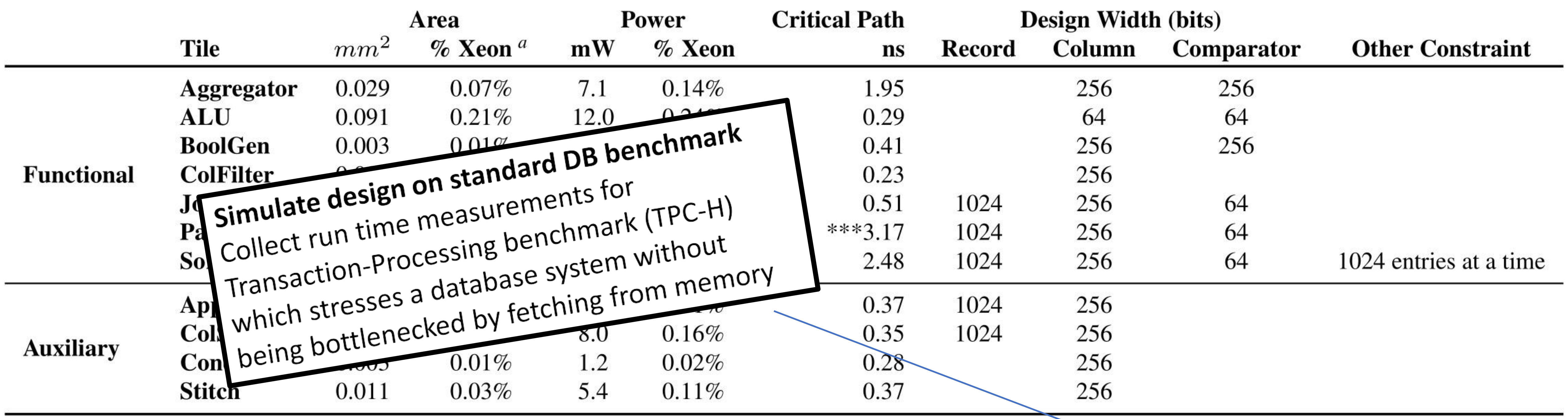


		Area		Power		Critical Path	Design Width (bits)			
	Tile	mm^2	% Xeon [a]	mW	% Xeon	ns	Record	Column	Comparator	Other Constraint
Functional	Aggregator	0.029	0.07%	7.1	0.14%	1.95		256	256	
	ALU	0.091	0.21%	[illegible]	[illegible]	0.29		64	64	
	BoolGen	0.003	[illegible]	[illegible]	[illegible]	0.41		256	256	
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.23		256		
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.51	1024	256	64	
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	***3.17	1024	256	64	
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2.48	1024	256	64	1024 entries at a time
Auxiliary	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.37	1024	256		
	[illegible]	[illegible]	[illegible]	[illegible]	0.16%	0.35	1024	256		
	[illegible]	[illegible]	[illegible]	1.2	0.02%	0.28		256		
	[illegible]	0.011	0.03%	5.4	0.11%	0.37		256		

Frequency limited by tile latency
Aggressively pipelined design means that the critical path delay defines the maximum switching delay (which is the same as the frequency of the design).
(Partitioner always defines freq. for Q100)

Design space optimization problem statement:
Choose the right mixture of tiles to have the best performance and power without using too much area or limiting frequency

Design Space Constraints
The Q100 Database Acceleration Architecture

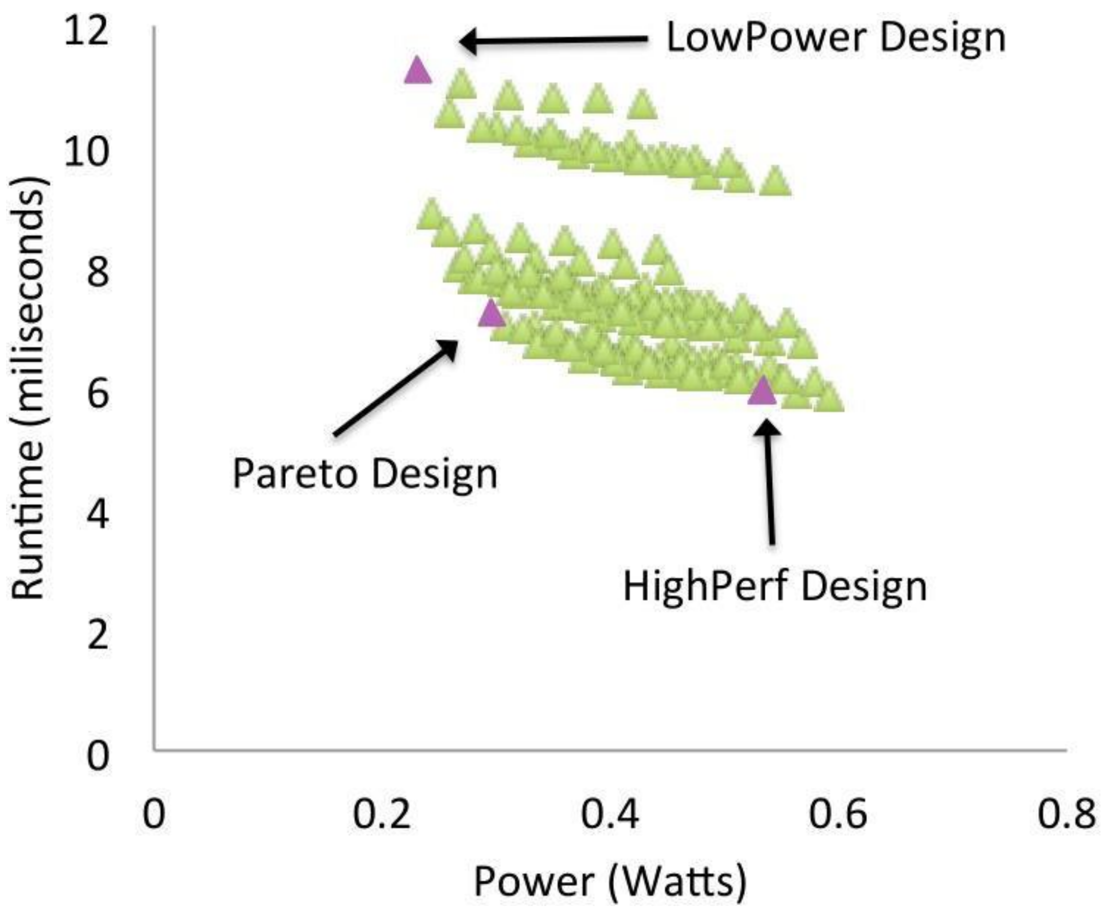


	Tile	Area		Power		Critical Path	Design Width (bits)			Other Constraint
		mm^2	% Xeon [a]	mW	% Xeon	ns	Record	Column	Comparator	
Functional	Aggregator	0.029	0.07%	7.1	0.14%	1.95		256	256	
	ALU	0.091	0.21%	12.0	[illegible]	0.29		64	64	
	BoolGen	0.003	[illegible]	[illegible]	[illegible]	0.41		256	256	
	ColFilter	[illegible]	[illegible]	[illegible]	[illegible]	0.23		256		
	Jo	[illegible]	[illegible]	[illegible]	[illegible]	0.51	1024	256	64	
	Pa	[illegible]	[illegible]	[illegible]	[illegible]	***3.17	1024	256	64	
	So	[illegible]	[illegible]	[illegible]	[illegible]	2.48	1024	256	64	1024 entries at a time
Auxiliary	Ap	[illegible]	[illegible]	[illegible]	[illegible]	0.37	1024	256		
	Col	[illegible]	[illegible]	8.0	0.16%	0.35	1024	256		
	Con	[illegible]	0.01%	1.2	0.02%	0.28		256		
	Stitch	0.011	0.03%	5.4	0.11%	0.37		256		

Simulate design on standard DB benchmark
Collect run time measurements for Transaction-Processing benchmark (TPC-H) which stresses a database system without being bottlenecked by fetching from memory

Design space optimization problem statement:
Choose the right mixture of tiles to have the best performance and power without using too much area or limiting frequency

Q100 Pareto Frontier

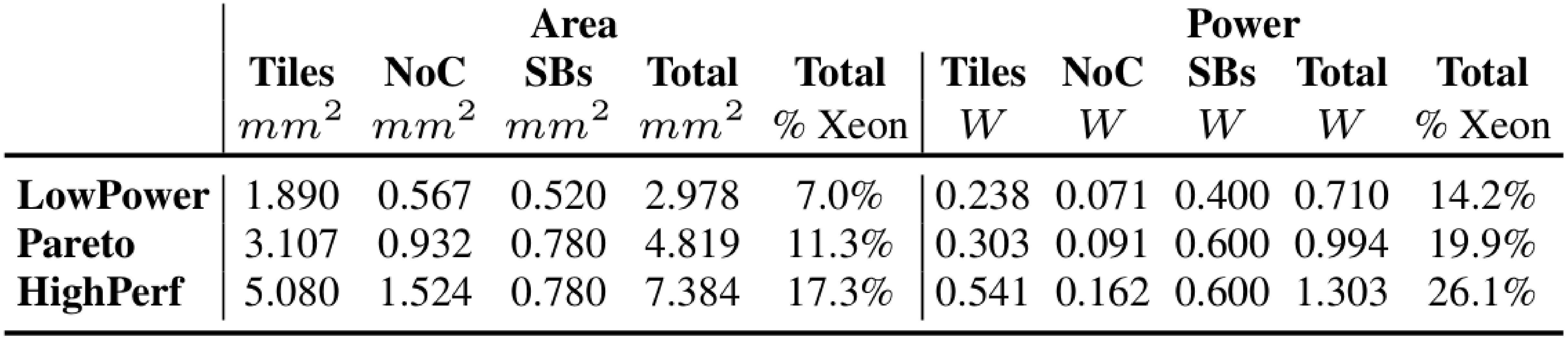


Pareto plot from a research paper on the Q100 Database accelerator by Wu et al, ASPLOS 2014

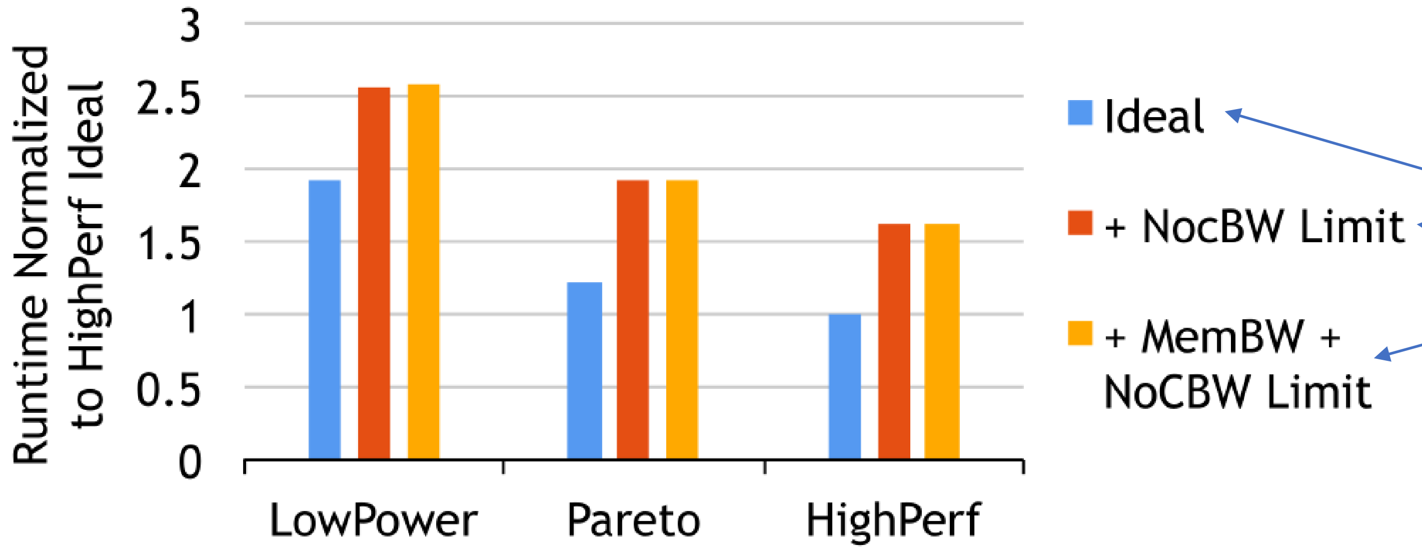
- How did they select magenta points?
- What other points might they have selected?
- What is the value in seeing all these points?

- "Pareto Design" as used in the paper means the design that maximizes (runtime) performance per watt.
- Although there were designs with nominally better runtime, the goal of the paper was to select three options for further study. The two options with a nominally better runtime were only negligibly better but at a much higher cost in terms of energy, rendering them less interesting to the authors.

Results of Design Space Exploration



	Area					Power				
	Tiles mm^2	**NoC** mm^2	**SBs** mm^2	**Total** mm^2	**Total** % Xeon	**Tiles** W	**NoC** W	**SBs** W	**Total** W	**Total** % Xeon
LowPower	1.890	0.567	0.520	2.978	7.0%	0.238	0.071	0.400	0.710	14.2%
Pareto	3.107	0.932	0.780	4.819	11.3%	0.303	0.091	0.600	0.994	19.9%
HighPerf	5.080	1.524	0.780	7.384	17.3%	0.541	0.162	0.600	1.303	26.1%



Final results show idealized design and results that include adding in costs related to the on-chip network and memory access bandwidth

Heat Plots Can Be Used to Explore 2D Space

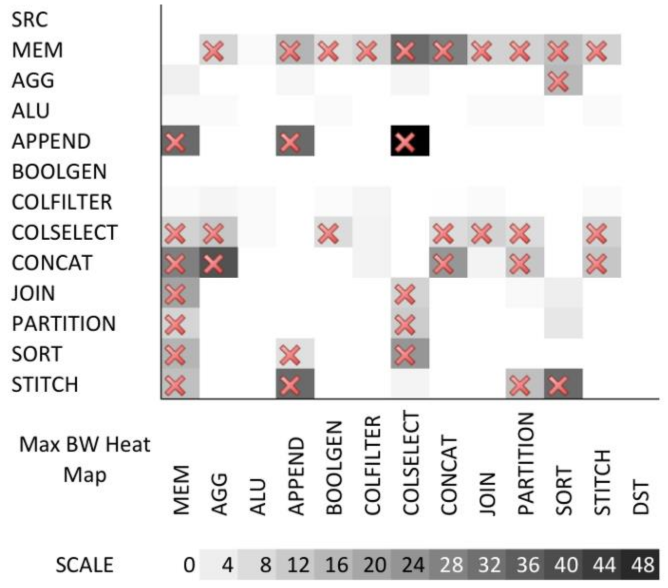


Figure 10. Even with a LowPower design, the communication bandwidth for most connections exceed the provisioned 6.3 GB/s NoC bandwidth, marked as X's in the figures.

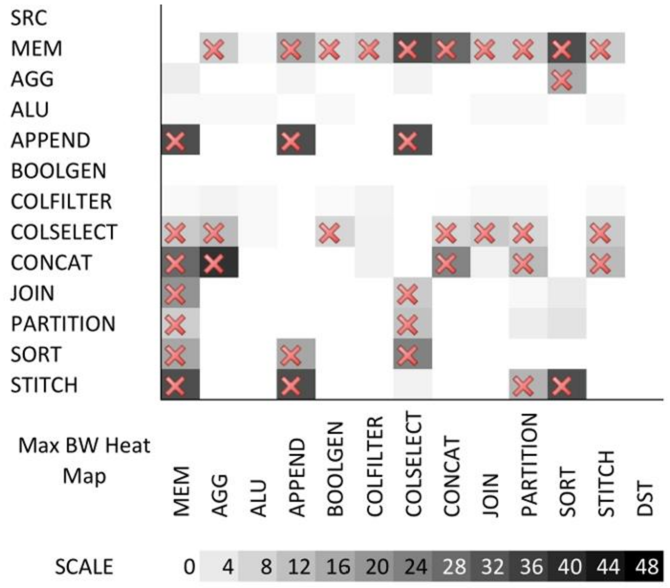


Figure 11. Similar to connection count heat map, Pareto design maximum intra-connection bandwidth exhibit almost identical behavior as HighPerf design.

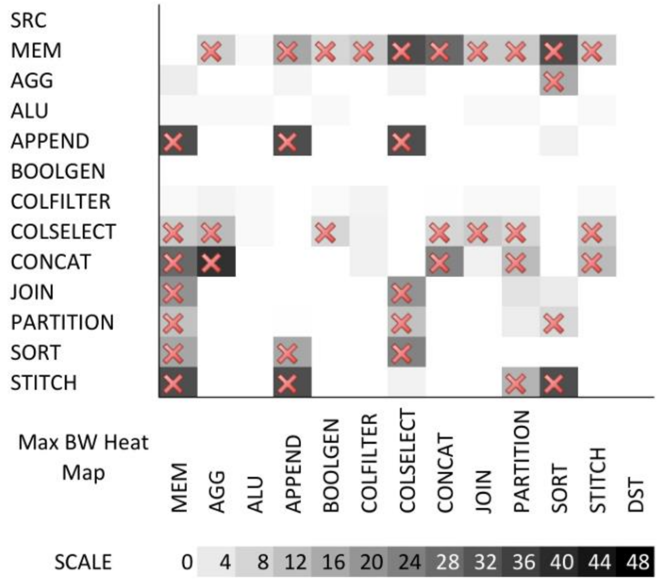


Figure 12. Heat map of HighPerf design max bandwidth per connection.

Here heat plots are used to show the communication bandwidth needed between tiles and which design elements exceed a reference threshold.

Q100 Takeaways / What did we just learn

- Practical application of design space exploration
- Defined design space based on tiles and connections between tiles
- Defined constraints and optimization goals based on power, area, frequency
- Runs experiments to produce Pareto Frontier with performance and power as main design dimension
- Final designs come from Pareto Frontier – fast, balanced, low-power
- Compare design to characteristics of known baseline (Xeon)

What to think about next?

- Miscellaneous (micro)architectural tricks & optimizations (future)
  - Super-scalar Out-of-Order
  - VLIW
  - Vector processors / SIMD
  - SIMT/GPU