ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process

Mark D. Aagaard, and Nuša Zidarič
Department of Electrical and Computer Engineering
University of Waterloo, Ontario, Canada
{maagaard,nzidaric}@uwaterloo.ca
February 22, 2021

This report presents area, throughput, and energy results for synthesizing the NIST Lightweight Cryptography Round 2 candidates on five ASIC cell libraries using two different synthesis tool suites.

Contents

List of Figures

This work was supported in part by the Canadian Natural Science and Engineering Research Council (NSERC) and the Canadian Microelectronics Corp (CMC).
The authors are among the designers of the ciphers ACE, Spix, SpoC, and WAGE. We have tried to analyze the data objectively and present the information transparently.

List of Tables

1 Introduction

This report presents area, throughput, and energy results for synthesizing the NIST Lightweight Cryptography Round 2 candidates on five ASIC cell libraries using two different synthesis tool suites. This report is the ASIC complement to the data in the FPGA benchmarking report published by the Cryptographic Engineering Research Group (CERG) at George Mason University (GMU) [5].

First and foremost, we would like to thank Kris Gaj and the rest of CERG for integrating the implementer's source code and LWC Development Package and sharing their knowledge and many insights gained from their LWC benchmarking efforts. Next we would like to thank the cipher implementers for their LWC Hardware API-compliant [4, 1] implementations and consent to be included in this report. The source code and LWC Development Package [4] are identical to that used in the Phase 4 version of the FPGA benchmarking report from February 2021 [5].

For unique names and features of the ciphers and cipher instances, please refer to Table 1 in the FPGA benchmarking report [5]. Figure 1.1 shows the marker symbols and colours used throughout this report to distinguish the different ciphers and instances of each cipher. Each cipher has its own marker (shape). Colour is used to distinguish the different instances of each cipher. In the plots, some names have been shortened from the full name shown in Figure 1.1.

This report includes 89 instances (variants) of 30 different hardware packages of 23 ciphers from the NIST LWC competition. Two instances of AESGCM by GMU CERG are included as reference comparisons. We began with 110 instances, 38 hardware packages, and 27 ciphers collected by the FPGA benchmarking group and included all instances that synthesized and simulated correctly with the ASIC synthesis tools. We contacted the implementation groups of the unsynthesizable instances and most groups submitted updated code that synthesized correctly. Some cipher instances were synthesizable, but the synthesized netlists had different behaviour than the original VHDL or Verilog code. We contacted the implementers, but have not yet able to resolve these problems, because we are unfamiliar with the implementation details of the ciphers and the implementers would require access to the proprietary simulation libraries for the ASIC cell libraries to reproduce the behaviour that we witnessed.

Section 2 describes the methodology used to conduct the experiments and analyse the data. The primary metrics that we consider are throughput (as measured in bits per clock cycle), area, energy, and area$\times$energy; see Section 3 for details.

All area results are measured after physical synthesis (place-and-route). We used five different cell libraries and two different tool suites. To obtain a unique datapoint for each cipher instance for a given metric, we present our findings with relative plots that are averaged over the cell libraries and tool-suites. This allows us to move beyond the characteristics of a single technology. For a detailed description of data presentation, see Section 2.6. The relative area vs throughput results are shown in Section 4, relative energy vs throughput in Section 5, and relative area$\times$energy vs throughput in Section 6.

Each instance has many different possible areas and clock speeds. Because these ciphers are intended for lightweight usage, we set the target clock speed for synthesis at a speed that all instances could achieve without incurring an area penalty. Section 7 examines tradeoffs between clock speed and area, and their impact on the choice of clock speed for several cipher instances. All results are based on synthesizing the ciphers for minimum area. The base plots, *i.e.*, the plots for individual cell library-tool suite combinations, showing the area vs throughput, energy vs throughput, and area$\times$energy vs throughput are in Appendix A, Appendix B, and Appendix C, respectively. Appendix D presents a table that summarizes the average scaled data and rankings.

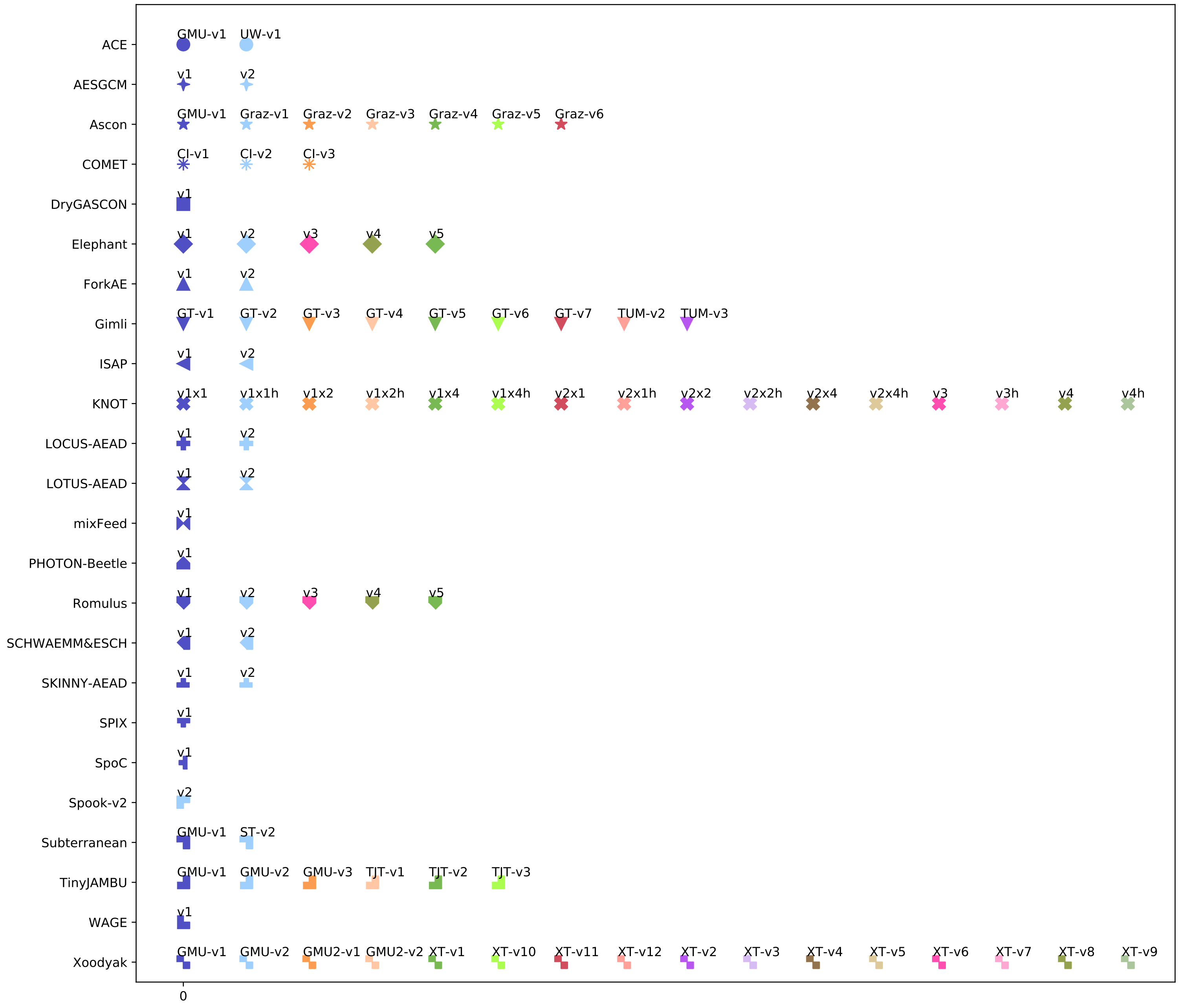

Figure 1.1: Legend of marker symbols and colours

2 Methodology

2.1 Cell Libraries and Tools

Table 1: ASIC cell libraries

Company	Size	Name	VDD
ST Micro	65 nm	CORE65LPLVT	1.25 V
TSMC	65 nm	tpfn65gpgv2od3 200c and tcbn65gplus 200a	1.0V
ST Micro	90 nm	CORE90GPLVT and CORX90GPLVT	1.0 V
TSMC 90	90 nm	tcbn90ghp 210a	1.0 V
ARM/IBM	130 nm	CMRF8SF LPVT with SAGE-X v2.0 cells	1.2,V

Table 2: Synthesis tools

Logic	Physical
Synopsys Design Compiler vP-2019.03	Cadence Encounter v14.13
Cadence Genus v18.10	Cadence Innovus v18.10

We present the results for the five ASIC cell libraries obtained with two tool suites, listed in Tables 1 and 2. Due to intellectual property concerns, the 10 different combinations of cell library and tool suite are anonymized into a letter and number. Each cell library is denoted by a letter A, B, C, D, or E. Each tool suite is denoted by a 1 or 2.

2.2 VHDL Compatibility

Some cipher instances were unsynthesizable with Design Compiler and/or Genus. We contacted the implementers of these ciphers and in most cases received updated source code that was synthesizable. Some cipher instances synthesized into netlists that had different behaviour than the original VHDL or Verilog source code.

Summary of compatibility issues:

1. The most common reason that code was unsynthesizable was that Design Compiler and Genus have restrictive rules for how signals may be used as array indices. In short, a signal may *not* be used in an array index expression on the right-hand-side of an assignment. On the left-hand-side of an assignment, signals may be used only in very simple range expressions, such as:

```
to_integer(sig) + const downto to_integer(sig)
```

2. If a signal is used as an array index on the right side of an assignment, the entire right side expression must be very simple. Source code of the form:

```
z <= a(to_integer(i)+c downto to_integer(i))
                       xor b;
```

often needed to be decomposed into two assignments:

```
tmp <= a(to_integer(i) + c downto to_integer(i));
z <= tmp xor b;
```

3. In VHDL, a common mistake is to forget to include a signal in the sensitivity list of a process that is intended to be combinational. Because of how easy it is to make this mistake, Design Compiler ignores the sensitivity list in the source code and assumes that the designer intended the process to be sensitive to all signals that it reads. Genus obeys the sensitivity list and generates hardware with latches or unusually clocked flip flops. Incomplete sensitivity lists result in different behaviours

between simulation and synthesis, and between different synthesis tools. A good solution is to use the VHDL-2008 `all` keyword as the sensitivity list.

4. Some cipher instances synthesized into netlists that had different behaviour than the original VHDL or Verilog source code. It would be very difficult for the implementers to reproduce the simulation results, because they would need access to the proprietary ASIC simulation libraries. We are continuing to investigate this issue.

5. Design Compiler does not support reading ports of mode `out`, even though this is a feature of VHDL 2008 and Design Compiler has a VHDL-2008 compatibility mode.

2.3 Synthesis Scripts

All synthesis runs for a specific tool were done with the same script. The scripts are designed to guide the synthesis tools to minimize area while achieving the target clock speed.

Because these ciphers are intended for lightweight usage, we set the target clock frequency at a speed that all instances could achieve without incurring an area penalty. In other words, the target clock period is longer than the actual clock period of the slowest instance when optimized for minimum area without any clock period constraint. Through experimentation, we found that a clock speed of 50 MHz (20 ns clock period) was sufficient for all instances. Section 7 analyzes tradeoffs between clock speed and area.

Synthesis was done with clock gating enabled and all synthesis optimizations set to maximum effort. Design Compiler has a variety of flags that can be used to fine tune the synthesis algorithms. We set these flags to minimize area, based on our experience with synthesizing lightweight cryptographic hardware for this set of cell libraries. The flag settings had a relatively small impact: 3–5% in the instances we checked.

2.4 Simulation

Simulation was performed with Mentor Graphics Questasim 10.7c. Testvectors for functional verification were created by CERG and generously shared with us.

Some netlists had Xs in the simulation, probably due to using initial values in signal declarations. To prevent these Xs, our simulation script does an initial run with just asserting reset and driving the clock, then collects a list of all registers whose value is `'X'` or `'U'`. We then initialize these registers to `'0'` in the simulation script and run the complete simulation.

2.5 Aggregating Data

To make it easier to identify overall trends and characteristics of the different ciphers across the 10 different combinations of cell libraries and tools, we aggregate the data. As described in Section 3, we present three primary metrics: Area, Energy and Area×Energy. For each of these metrics, we aggregate the data from all of the cell-library and tool combinations into a single number that characterizes *on average* how a particular cipher instance compares relative to all of the cipher instances when evaluated on the same cell-library and tool. In the following, we describe this computation in detail for area.

Let A denote area and let k run over all combinations of cell library and tool suite, i.e., $k =$`A1,A2,B1,...,E2`. For each k, we compute the average area A'_k as the geometric mean over the areas of all cipher instances.

For each cipher instance i and (cell-library,tool) configuration k, we compute the *scaled area* $\bar{A}_k(i) = \frac{A_k(i)}{A'_k}$. To reduce the effect of outlier results, we drop the highest and lowest k for each instance i, and compute the *average scaled area* of the instance i as geometric mean over the scaled area of that cipher instance for the remaining cell library-tool suite combinations (*i.e.*, the remaining $A_k(i)$ for a fixed i). We use the average scaled area as a unique area value for the instance i, and present this data in the plots named "average scaled area", "average scaled energy" and "average scaled area× energy".

2.6 Presentation of Data

The cipher instances that were unsynthesizable or that generated netlists whose behaviour differed from the original source code are not included in the analysis. In the energy analysis, some cipher instances synthesized correctly for some combinations of cell library and synthesis tool and incorrectly for others. Only the cases that simulated correctly are included in the energy analysis. Thus, when paging through the plots in Appendix B and Appendix C, when a particular cipher instance is not shown in one of the plots, it is because that particular configuration resulted in an incorrect netlist for that instance.

In the plots, all of the instances of a cipher use the same marker and are connected by a thin coloured line, to make it easier to detect trends and patterns between the different instances of an individual cipher. Colour is used to distinguish the different instances of each cipher.

We present the data graphically as area vs throughput, energy vs throughput, and area×energy vs throughput. Because throughput is a primary design decision in implementing a cipher instance, we put throughput on the x-axis as the controlling variable in the plots. The other metrics are dependent on the throughput, and so are shown on the y-axis. The axes are shown in logarithmic scale.

The plots use grey "contour" lines to show design tradeoffs that are equally good for the data being plotted. For example, in Figure 4.1: Area-vs-throughput, all of the points on a line have the same throughput/area ratio. Better, or more efficient, instances have higher throughput and lower area, or more simply, are located in the lower right corner of the plot.

3 Metrics

The primary metrics that we consider are throughput (as measured in bits per clock cycle), area, energy, and area×energy.

Throughput: For throughput, we use the steady-state (long message) throughput for encryption as measured from simulation by the George Mason Cryptographic Engineering Group [5]. The steady-state throughput does not include loading, associated data, or hashing. The FPGA report shows that most instances have the same throughput for both encryption and decryption. A more thorough analysis that includes short messages and hashing would be possible with additional resources.

Clock speed: For embedded systems implemented on ASICs, clock speed is usually limited by power consumption, rather than the delay through the circuit. Although the clock speed is generally considered as a *bigger-is-better* metric, design tradeoffs with clock speeds much lower than the maximum achievable are preferred. Section 7 illustrates the tradeoffs between area and clock speed for several ciphers. With ASICs, increasing the clock speed beyond the speed of the minimal area circuit, comes with an area penalty. Hence, for lightweight ciphers implemented on ASICs, throughput, as measured in terms of bits per clock cycle, is a better measure of performance than clock speed.

Area: We report the circuit area in terms of gate equivalents (GE) for a given cell library. All area results are measured after physical synthesis (place-and-route). We used a density of 95%; that is, the total area of the core is computed as the area of the gates (cells) divided by 0.95. Through extensive experimentation, we found that this is the highest density that works consistently without incurring significant overhead in delay due to wiring congestion or design rule violations. A few cipher instances encountered design rule violations on a few cell libraries and choice of synthesis tool. These combinations of (instance,library,tool) are not included in the reported data.

Energy: We measured power consumption with timing simulation of the physical netlist using a sequence of 1000 clock cycles of encrypting a long message. Through experiments, we found that this length of simulation was sufficient to give precise results, in that running longer messages had less than a 1% effect on the power consumption. The sequence is taken from the middle of processing a message: it does not include loading the key and nonce, initialization, associate data, or tag generation. The goal was to measure "steady state" energy consumption for encryption to provide a baseline measurement of energy for each cipher instance. We did not have time to evaluate the full range of behaviours. As with the performance analysis, the data on power and energy from the FPGA report can be used to roughly extrapolate the data presented here to short messages, decryption, and hashing.

One of the cell libraries was less reliable in generating netlists that simulated correctly. The energy analysis includes the eight configurations (4 cell libraries and both synthesis tools) that reliably generated netlists that simulated correctly.

We used the throughput to convert power consumption as reported by the tools into energy per bit:

$$\text{Energy}(J/bit) = \frac{\text{Power}(J/s)}{\text{ClockSpeed}(cyc/s) \times \text{Throughput}(bit/cyc)}$$

Power is composed of two parts: dynamic power consumption and static power consumption. Dynamic power is linearly dependent on clock frequency and static power is independent of clock frequency. Figure 3.1 shows the distribution of the percentage of power that is leakage power for the four cell libraries at the clock speeds of 100 MHz, 50 MHz, and 5 MHz. Library B has the lowest percentage of leakage power. Even at 5 MHz, the vast majority of cipher instances have less than 10% leakage power. In contrast, library D has a much higher percentage of leakage power: at 5 MHz, most cipher instances have 65%–85% of their total power consumed by leakage.

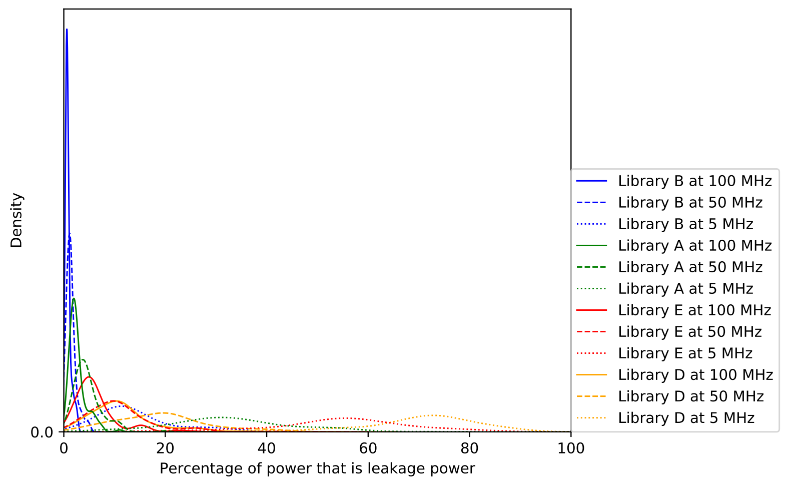

Figure 3.1Distribution of leakage power

The energy analysis was done for both synthesis tool suites and four out of the five cell libraries because we were unable to get reliable energy results for one of the libraries. We ran the analysis at clock speeds of 5 MHz, 20 MHz, 50 MHz, and 100 MHz. Appendix B shows the results for the eight configurations of cell library and tool at 50 MHz. The relative positions of the cipher instances were quite similar across the different clock speeds.

Area×Energy: Area and energy are both a *smaller-is-better* metric, and of special interest when targeting lightweight applications. The area-energy product is a derived metric, used to combine the two to simplify high-level comparisons between ciphers.

4 Area

Figure 4.1 shows the graph of area vs. throughput and Figure 4.2 shows an expanded version of the densest part of the graph.

As throughput increases, area increases. The most common way to increase throughput is to process more bits in parallel, up to the block width of the cipher, then unroll the rounds. Both of these throughput optimizations increase the combinational area of the CryptoCore portion circuit but do not change the registers. Increasing the number of bit processed per cycle while leaving the input/output port widths constant will increase throughput while leaving the area of the input/output circuitry (*e.g.*, PreProcesssor and PostProcessor) unchanged. Thus, doubling the throughput causes the area to go up by less than a factor of 2.

All of the LWC cipher instances are smaller than the two AESGCM instances at equivalent throughputs. Xoodyak and Gimli each have one instance that is larger than the AESGCM instances, but these Xoodyak and Gimli instances are for much higher throughputs than the AESGCM instances.

The ciphers with many instances (Xoodyak (16), KNOT (16), Gimli (10), Ascon (6), TinyJAMBU (6), Elephant (5), and Romulus (5)) generally exhibit the expected curve of increasing area with throughput. KNOT and Xoodyak have somewhat zig-zag patterns that illustrate multiple choices for key sizes, cryptographic primitives, inclusion of hashing, and implementation styles.

TinyJAMBU is the smallest cipher and its highest throughput instance has a relatively high throughput/area ratio. Subterranean is both quite small and is the most efficient in throughput/area. Romulus has small instances for throughputs of 4 bpc and below. Ciphers that have highly efficient instances include: Xoodyak, Ascon, and KNOT. At higher throughputs (16 bpc and higher), Gimli has some highly efficient instances.

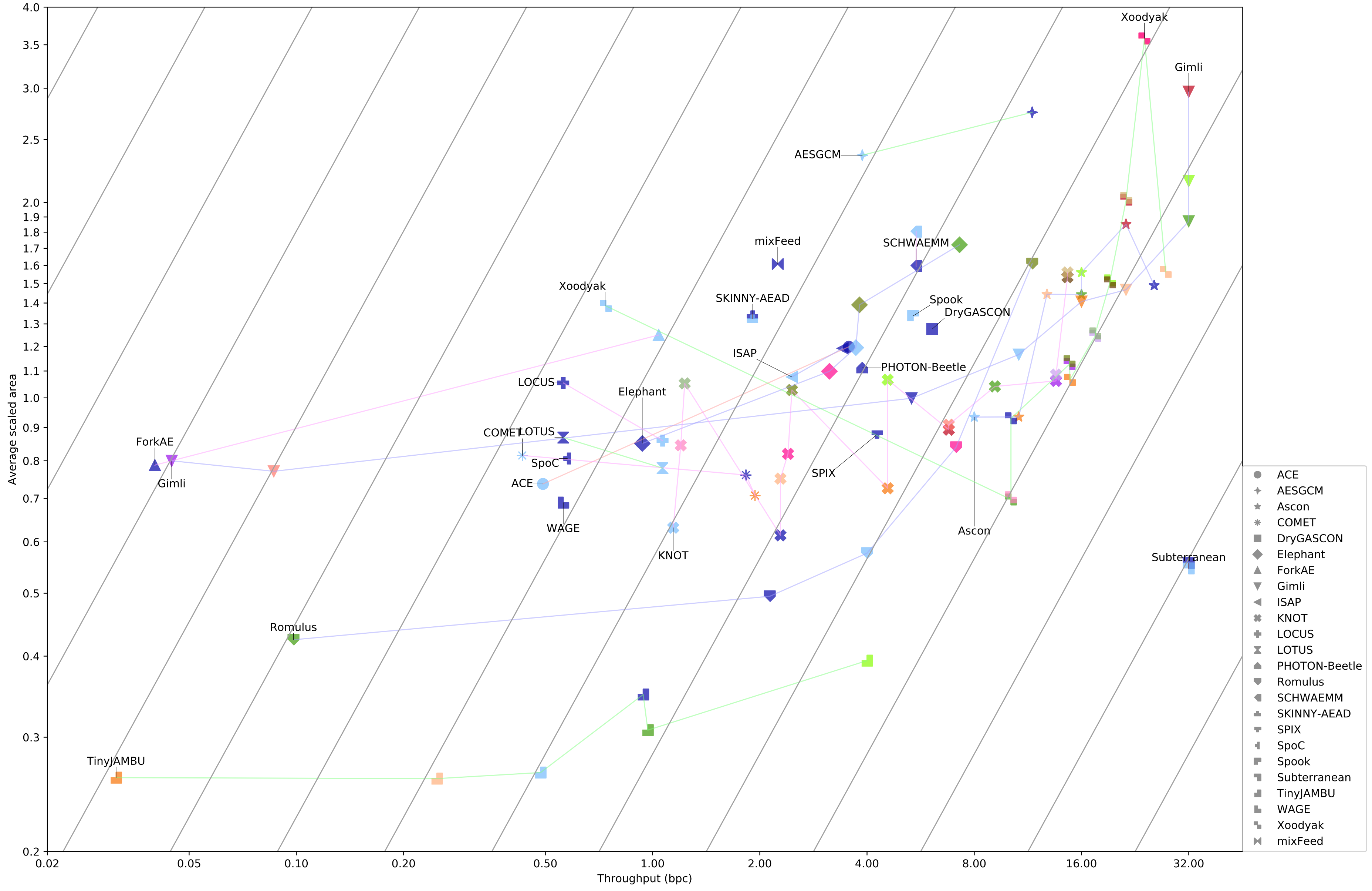

Figure 4.1: Average scaled area vs throughput

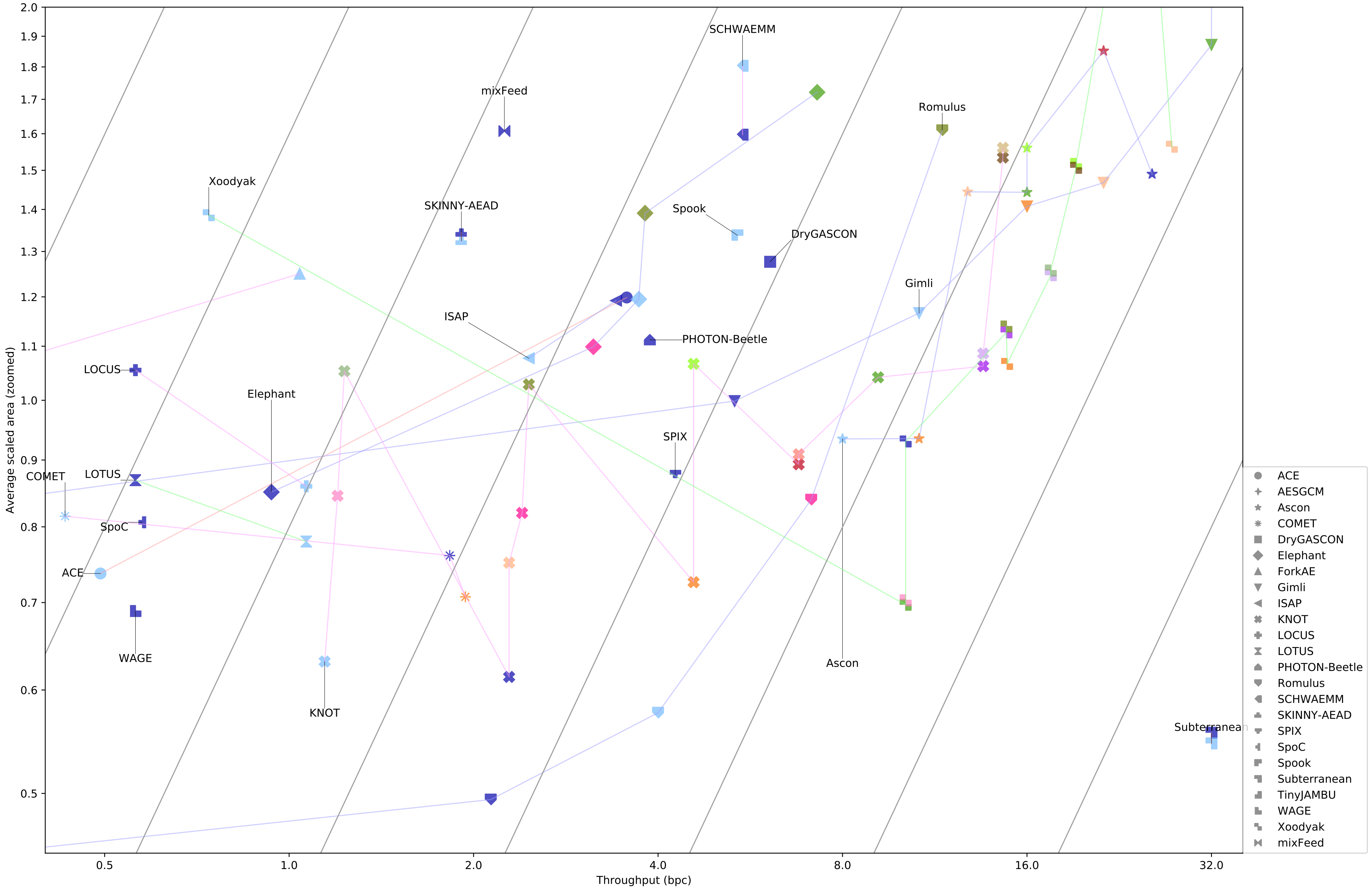

Figure 4.2: Average scaled area vs throughput (zoom)

5 Energy

Subterranean stands out for its extremely low energy consumption. Other ciphers with excellent energy efficiency include high-throughput instances of Gimli, Xoodyak, and Ascon. At lower throughputs, TinyJAMBU and COMET are the most energy efficient.

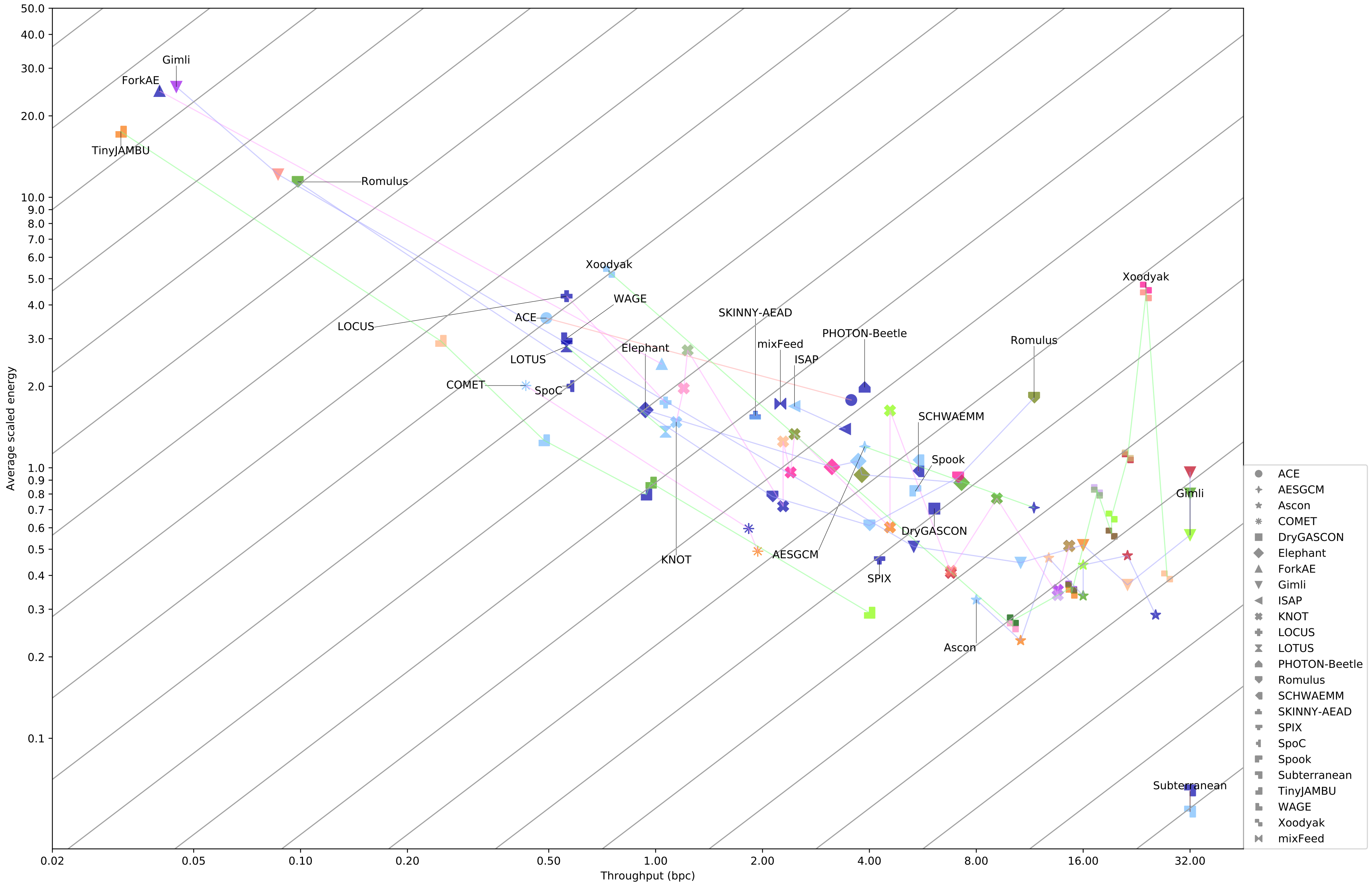

Figure 5.1: Average scaled energy vs throughput

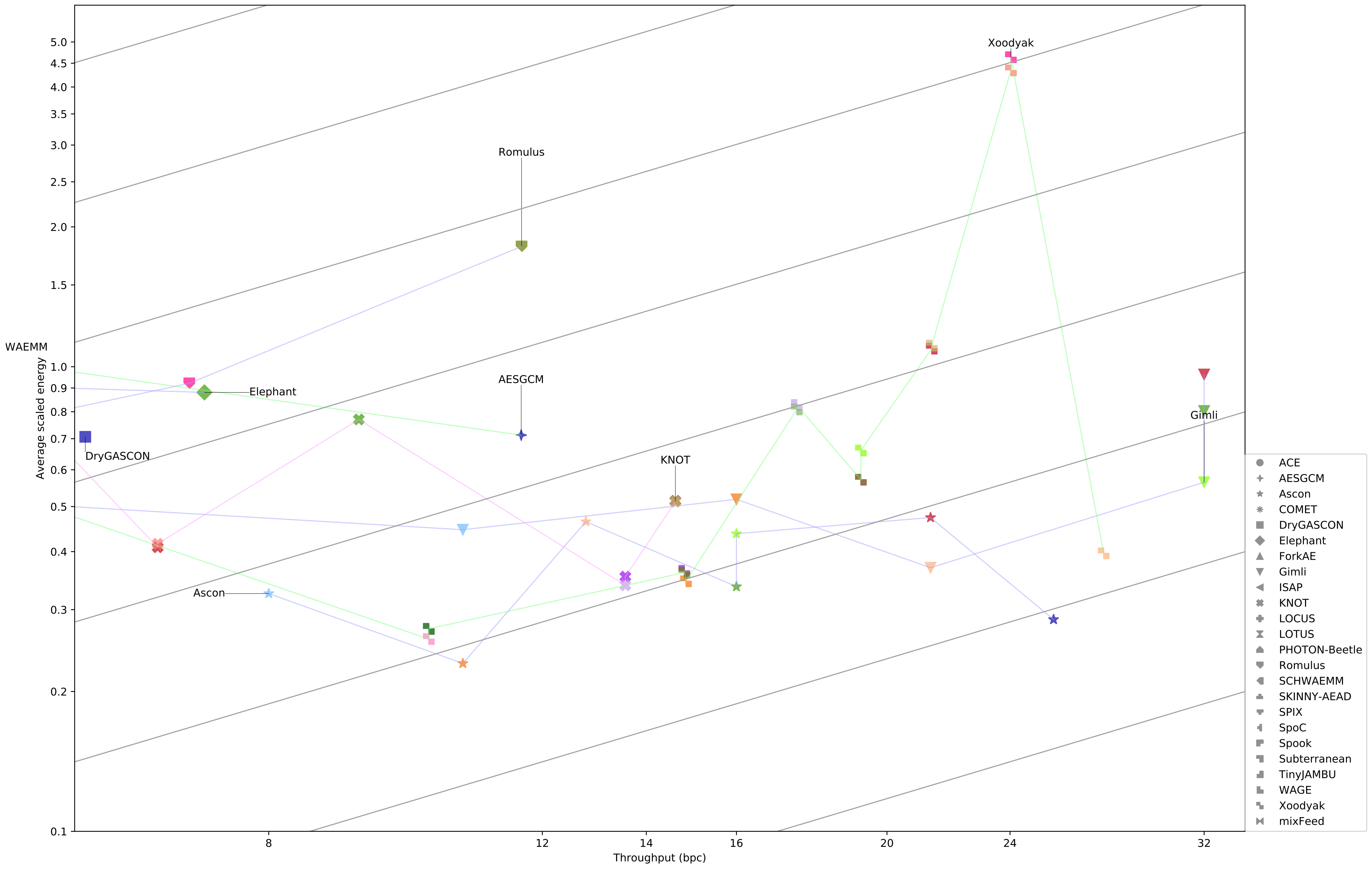

Figure 5.2: Averaged scaled energy vs throughput (zoomed to show high throughput instances)

6 Area and Energy

As throughput increases, area increases and energy decreases. The plots for area vs throughput and energy vs throughput show that the decrease in energy is more dramatic than the increase in area. For example, TinyJAMBU has a 58× reduction in energy and a 1.14× increase in area; Gimli has 2.20× and 97×. Therefore, as throughput increases, area×energy generally decreases.

Again, Subterranean stands out in efficiency. TinyJAMBU also does extremely well on this metric. After these two, there are a number of ciphers with high efficiency. Ciphers that achieve low area and energy across a range of higher throughputs include Xoodyak, Ascon, KNOT, Romulus, and COMET. Gimli-v1x2h does notably well compared to other ciphers in the 16-32 bpc throughput range. Gimli also offers an extremely wide range of possible throughputs: from approximately 0.05 bpc to 32 bpc. Romulus and COMET have instances that do well at throughputs of 4 bpc and below.

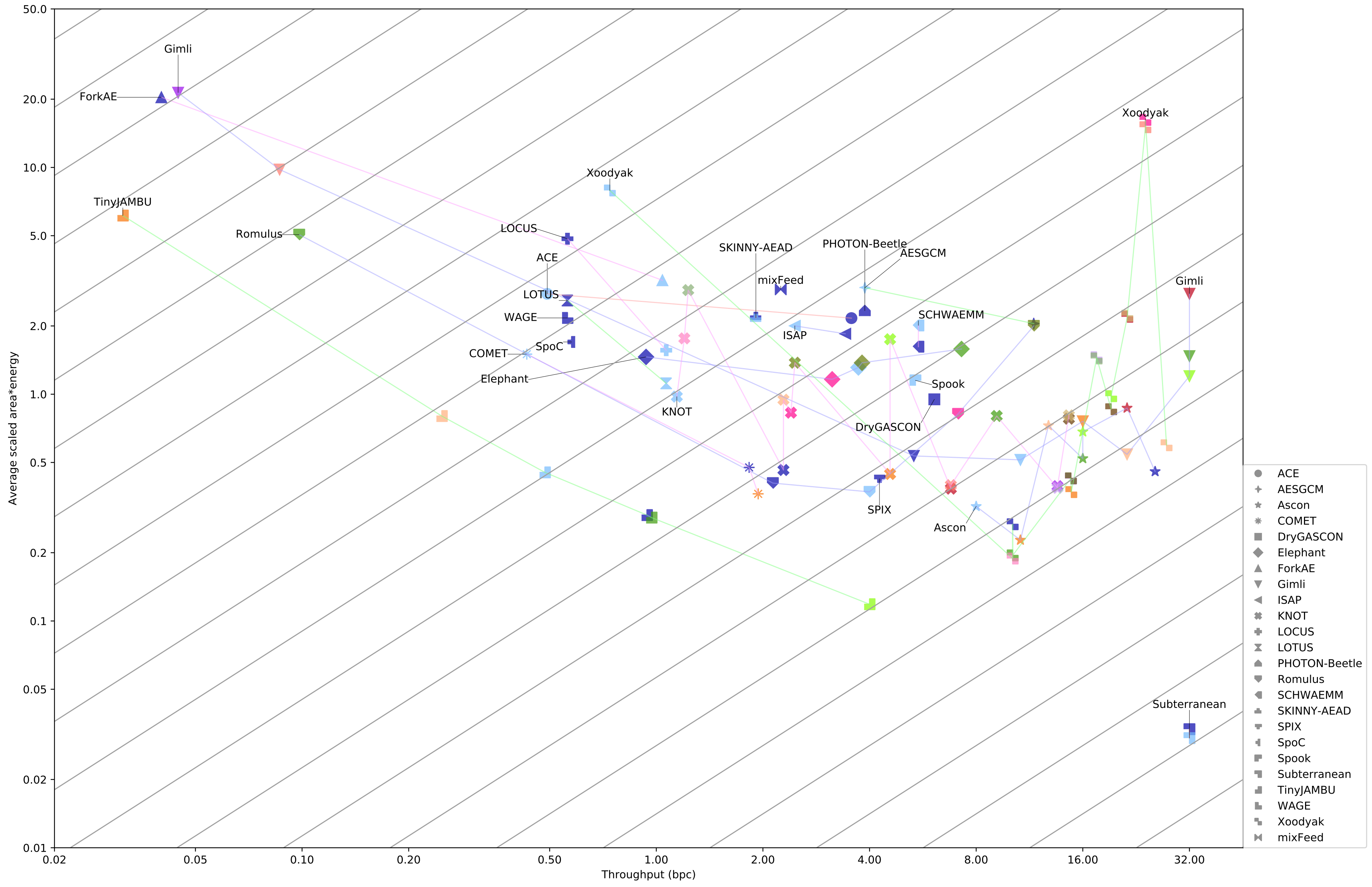

Figure 6.1: Average scaled area × energy vs throughput

7 Clock Speed

Figure 7.1 shows how actual clock speed and area vary with the target clock speed given to the synthesis tool. This plot is for a randomly chosen cipher instance. While the details of the plot vary with each circuit, the general shapes of the curves are relatively consistent across multiple circuits, cell libraries, and synthesis tools.

For low target clock speeds, the actual clock speed is higher than the target. Once the target clock period approaches the actual delay of the circuit (approximately 2.5 ns or 500 MHz for this circuit), a tradeoff between clock speed and area emerges.

As the target clock speed increases, the synthesis tool uses performance optimizations that increase circuit area and stops using area optimizations that increase delay. For example, common subexpression elimination can be very effective at both the Boolean and algebraic level in reducing area, but the intermediate signals can lead to "deeper" expressions that will have more delay. At the circuit level, to increase performance, synthesis tools will use larger gates that have higher drive strength.

As the target clock speed further increases, the incremental cost in increased area rises dramatically. Eventually, the synthesis tool is no longer able to further increase the clock speed.

The minimum area of the circuit (less than 15 kGE) is less than half of the area of the circuit when synthesized for maximum clock speed (more than 30 kGE). The red line shows the ratio of actual clock speed to area and the green line shows actual clock speed to area squared. These two metrics lead to choices of optimal clock speeds that balance area and clock speed.

For the example circuit, if optimizing for minimum area, the maximum clock speed that can be used without sacrificing area is 500 MHz. Using speed/area^2 the optimal clock speed for this circuit is approximately 900 MHz and using speed/area the optimal clock speed is approximately 1 GHz. If optimizing for maximum clock speed independent of area, the maximum speed is approximately 1.3 GHz.

Figure 7.2 shows the rough plots of area vs actual clock speed for several randomly chosen cipher instances. The leftmost point of each curve indicates the area and clock speed of the minimum area implementation of the cipher instance. The rightmost point of each curve indicates the area and clock speed of the maximum clock speed implementation of the circuit. The different implementations were synthesized by changing the target clock speed — the original source code and synthesis scripts are not changed.

The table below shows how the circuits rank in speed for their minimal area implementation and their maximum speed implementation. There is some rough correlation, for example green and red make up 2 out of the top 3 instances in both rankings. But, other the ranking for some instances changes dramatically, such as purple being the 6th fastest in the minimal area ranking but 3rd fastest in the max speed ranking. The view of speed and area would become more complex with the inclusion of additional metrics such as speed/area and speed/area^2.

The overall lesson for the cipher instances is that analyzing clock speed requires choosing the metric that is being optimized for, and then performing many synthesis runs (similar to the Athena project [2]) to find the optimal choice for the given metric. In addition, Figures 7.1 and 7.2 show logic synthesis results. Physical synthesis more than doubles the synthesis run times and has the additional complexity of adding density as an additional parameter that needs to be evaluated.

	Speed rank	
Cipher instance	Min area	Max speed
Green	1	1
Orange	2	4
Red	3	2
Dark blue	4	7
Brown	5	5
Purple	6	3
Blue	7	6

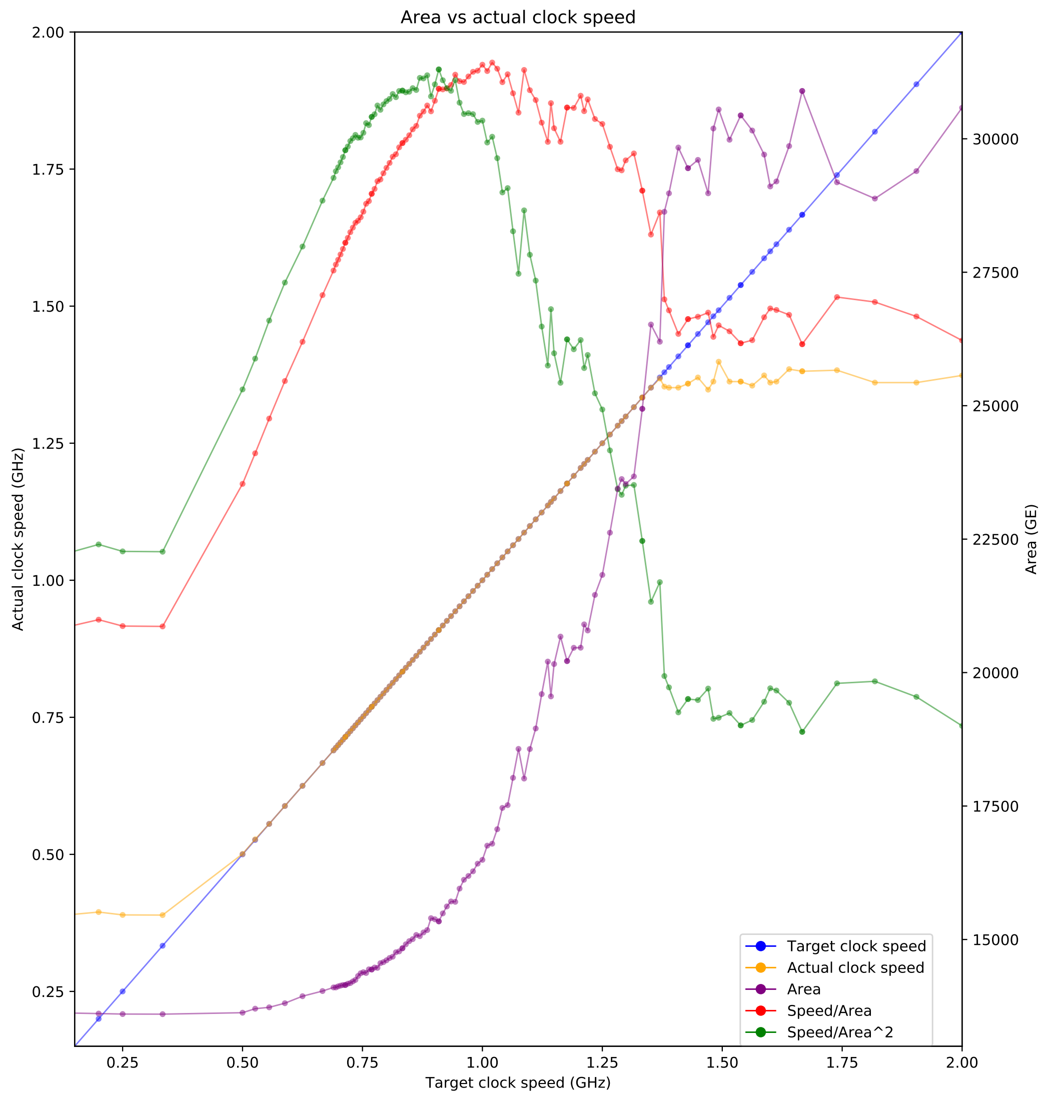

Figure 7.1: Actual clock speed vs target clock speed

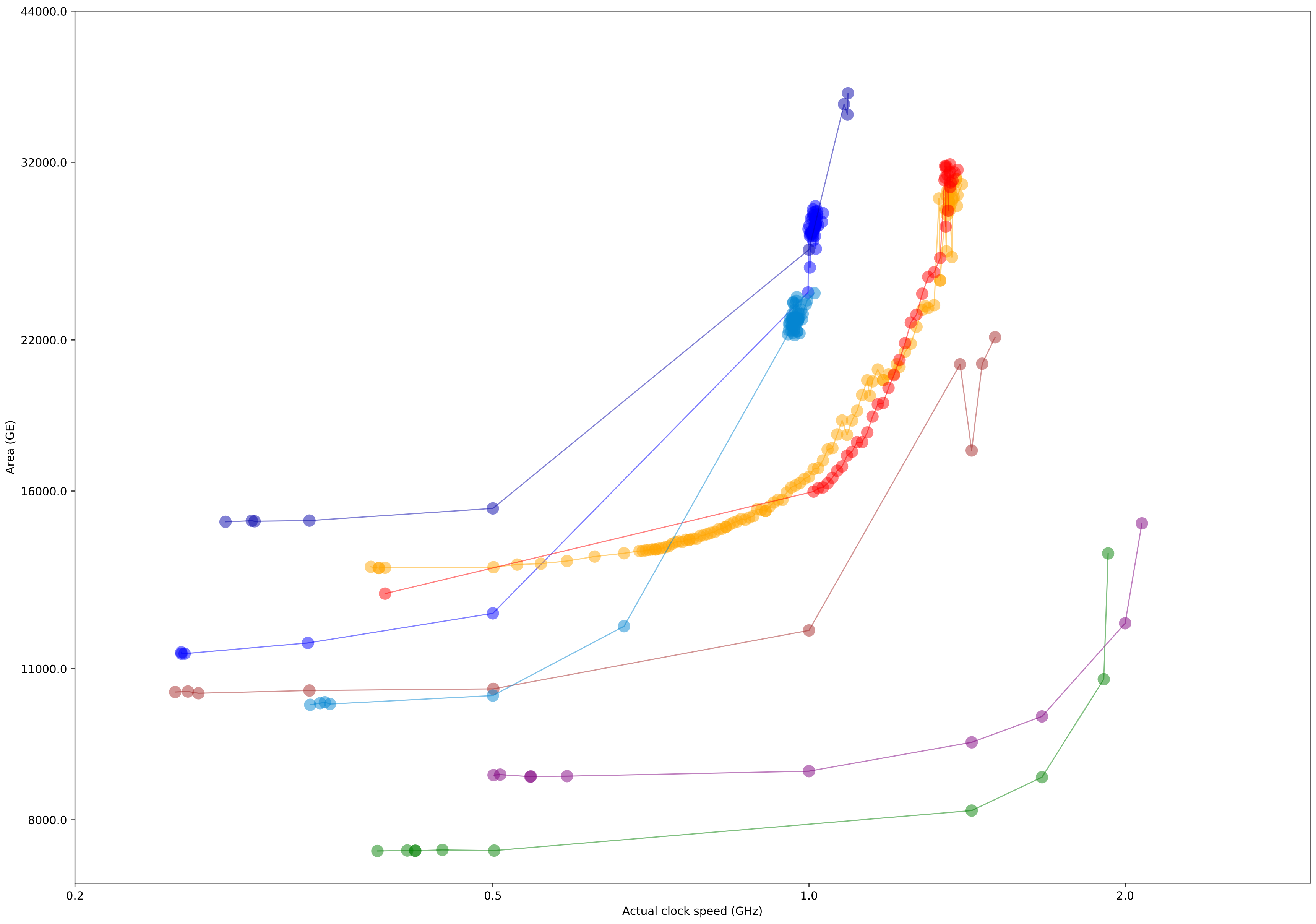

Figure 7.2: Area vs actual clock speed

References

[1] K. Gaj, Jan. 2021. Personal communication.

[2] K. Gaj, J. Kaps, V. Amirineni, M. Rogawski, E. Homsirikamol, and B. Y. Brewster. ATHENa — Automated Tool for Hardware EvaluatioN: Toward fair and comprehensive benchmarking of cryptographic hardware using FPGAs. In *2010 International Conference on Field Programmable Logic and Applications*, pages 414–421, 2010.

[3] E. Homsirikimal and W. Diehl. Cryptotvgen.

[4] J.-P. Kaps, W. Diehl, M. Tempelmeier, E. Homsirikamol, and K. Gaj. Hardware API for lightweight cryptography. Technical report, George Mason University, Oct. 2019.

[5] K. Mohajerani, R. Haeussler, R. Nagpal, F. Farahmand, A. Abdulgadir, J.-P. Kaps, and K. Gaj. FPGA benchmarking of round 2 candidates in the NIST lightweight cryptography standardization process: Methodology, metrics, tools, and results. Technical Report 2020-1207, IACR, Feb. 2021.

A Area Details

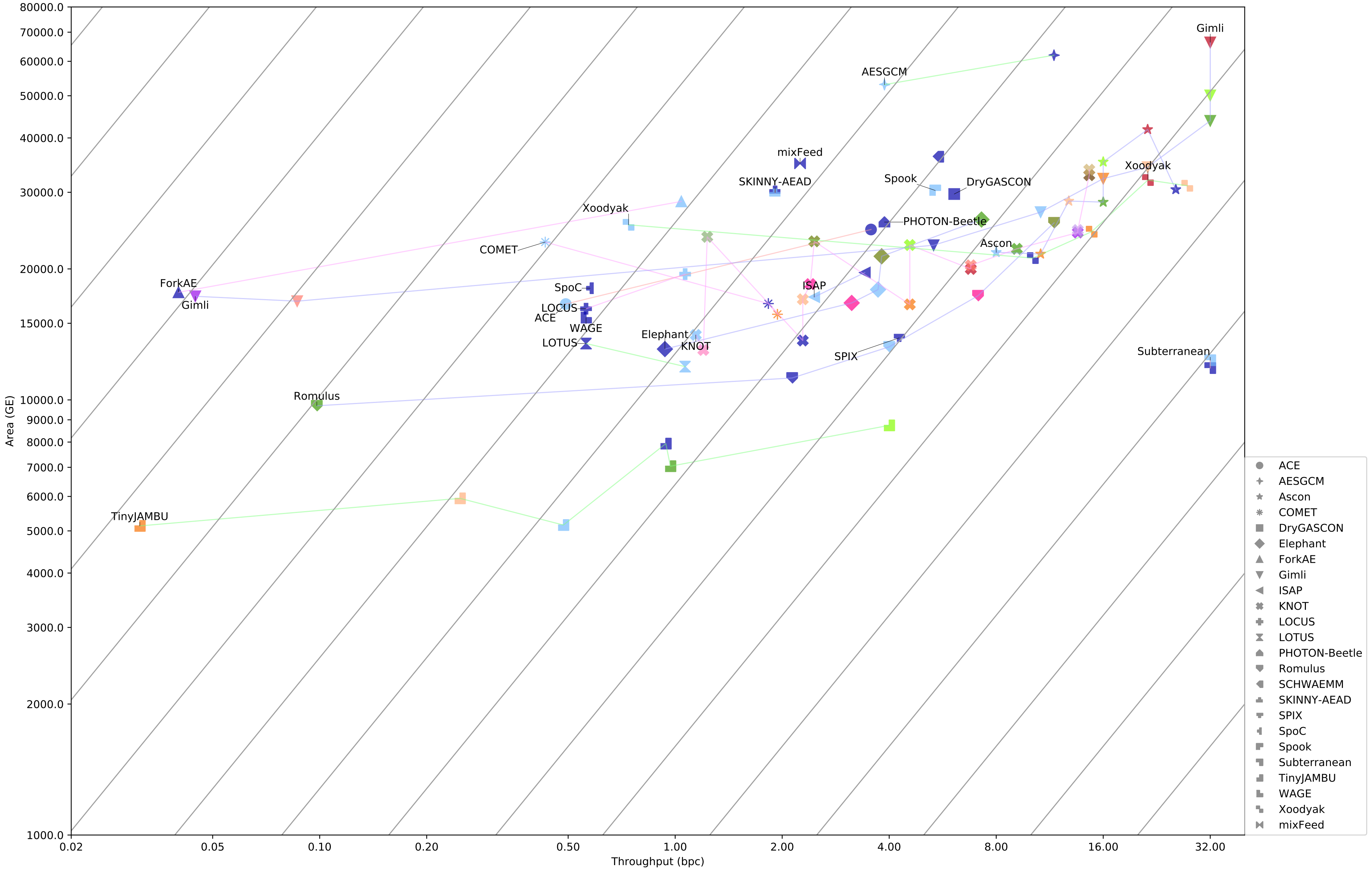

Figure A.1: Area vs throughput for configuration A1

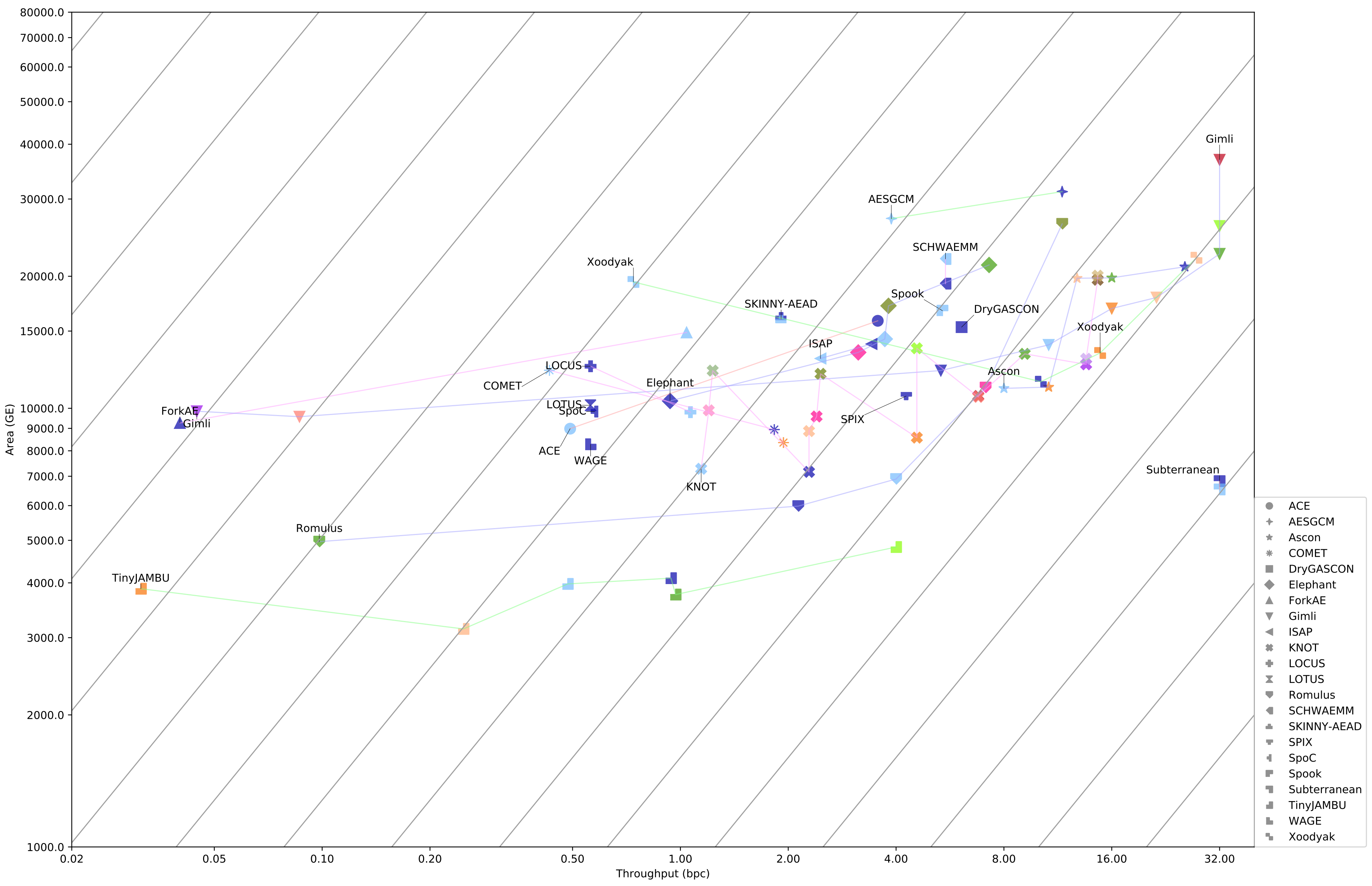

Figure A.2: Area vs throughput for configuration A2

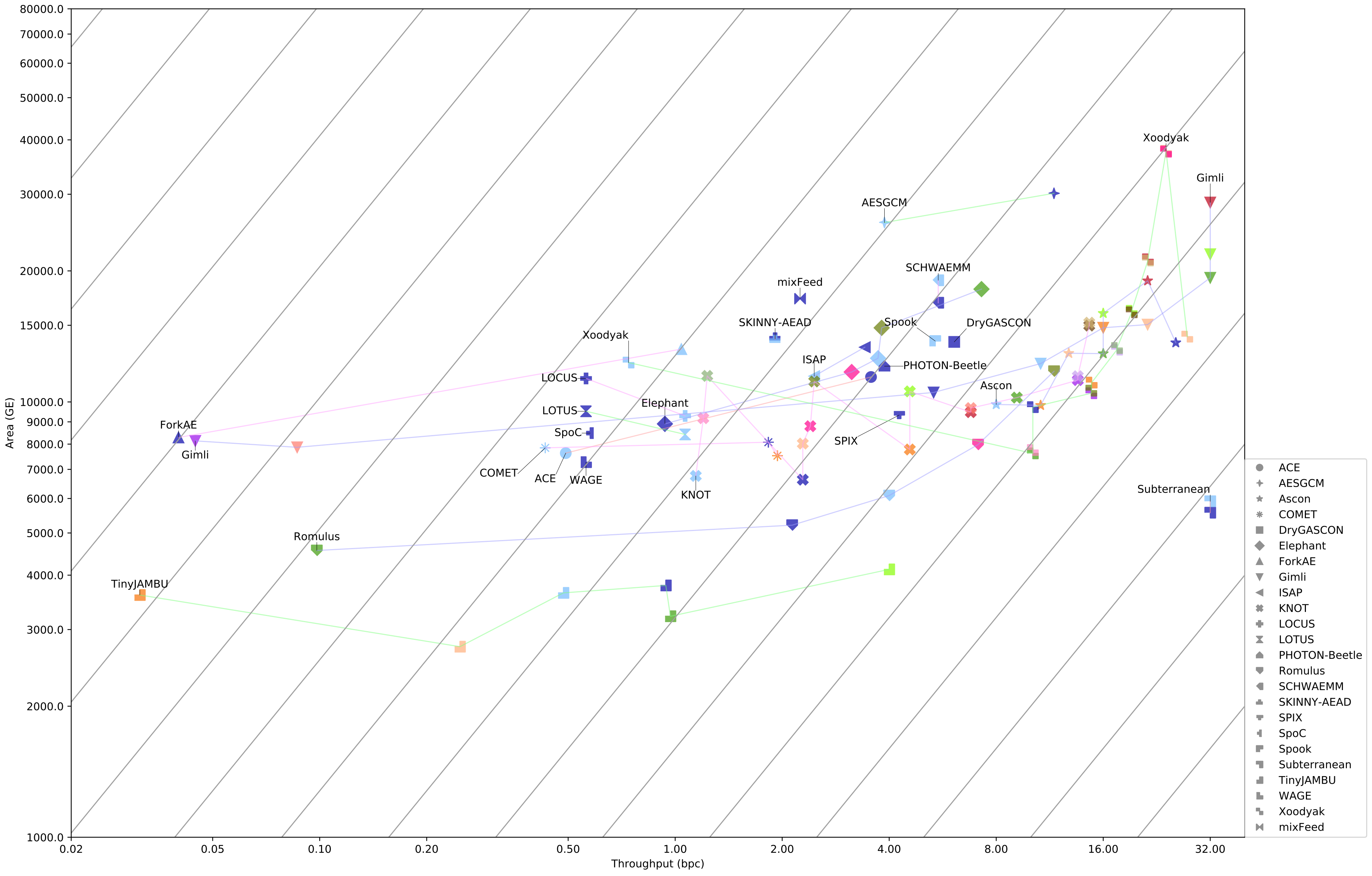

Figure A.3: Area vs throughput for configuration B1

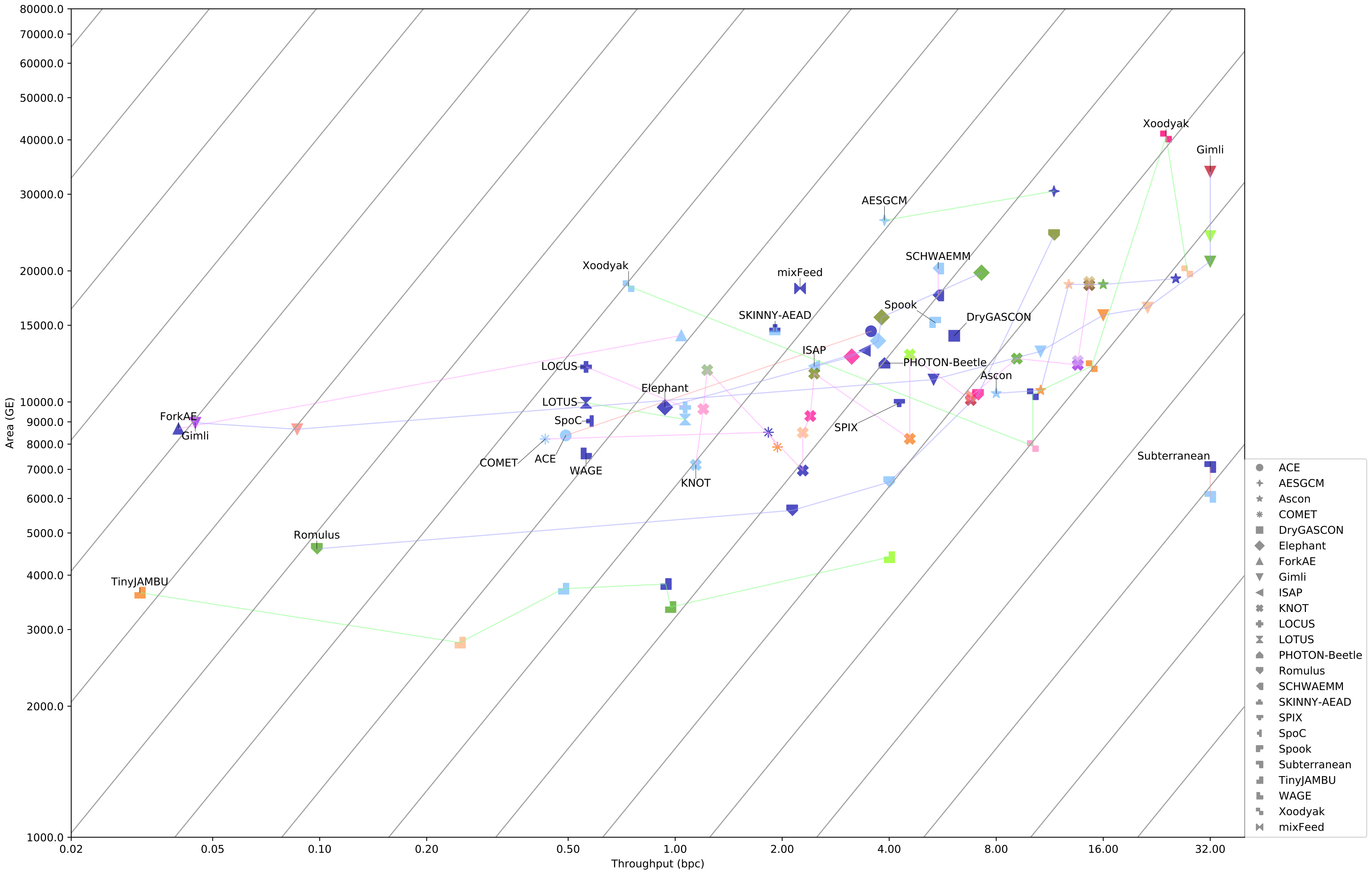

Figure A.4: Area vs throughput for configuration B2

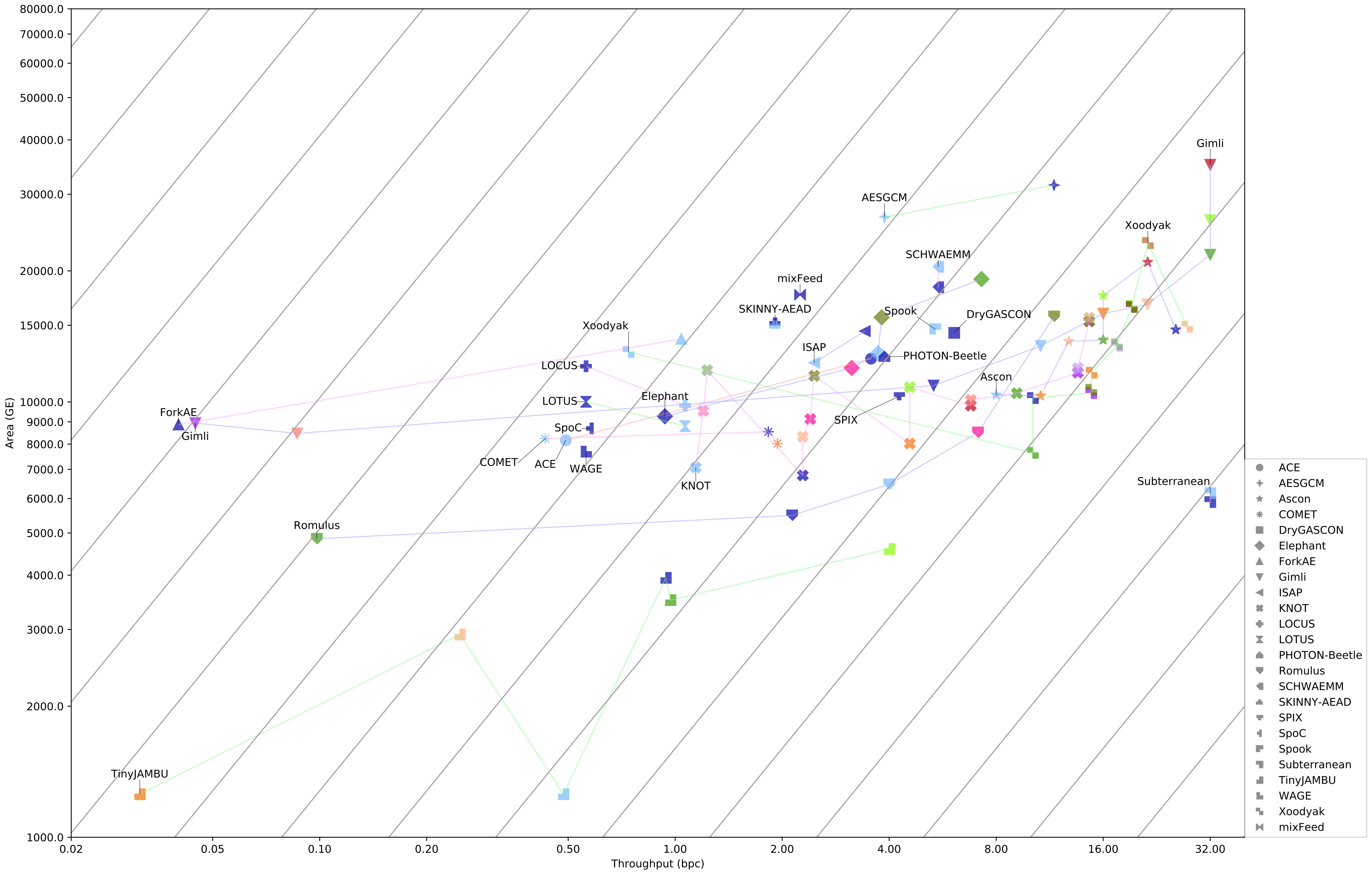

Figure A.5: Area vs throughput for configuration C1

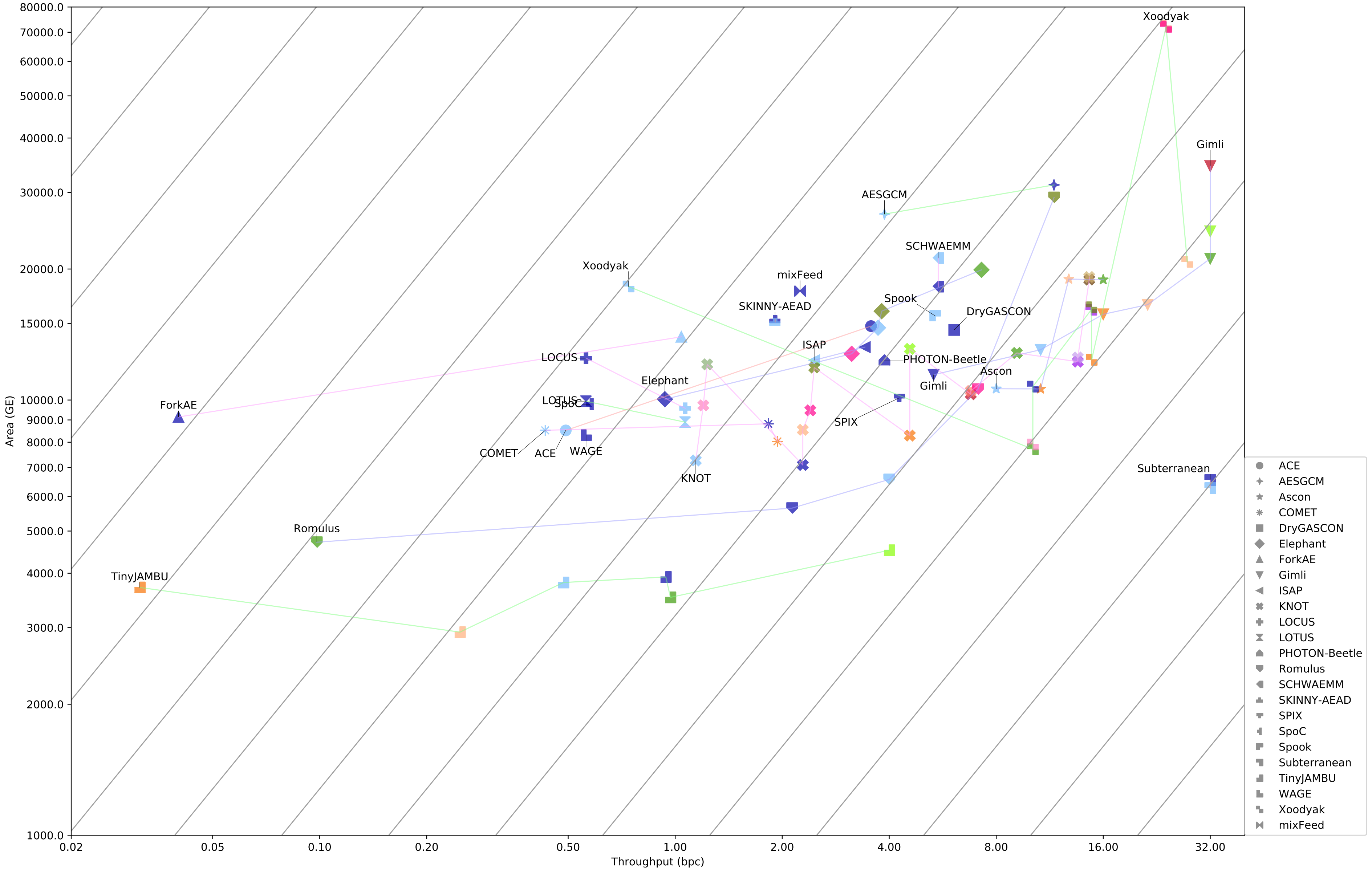

Figure A.6: Area vs throughput for configuration C2

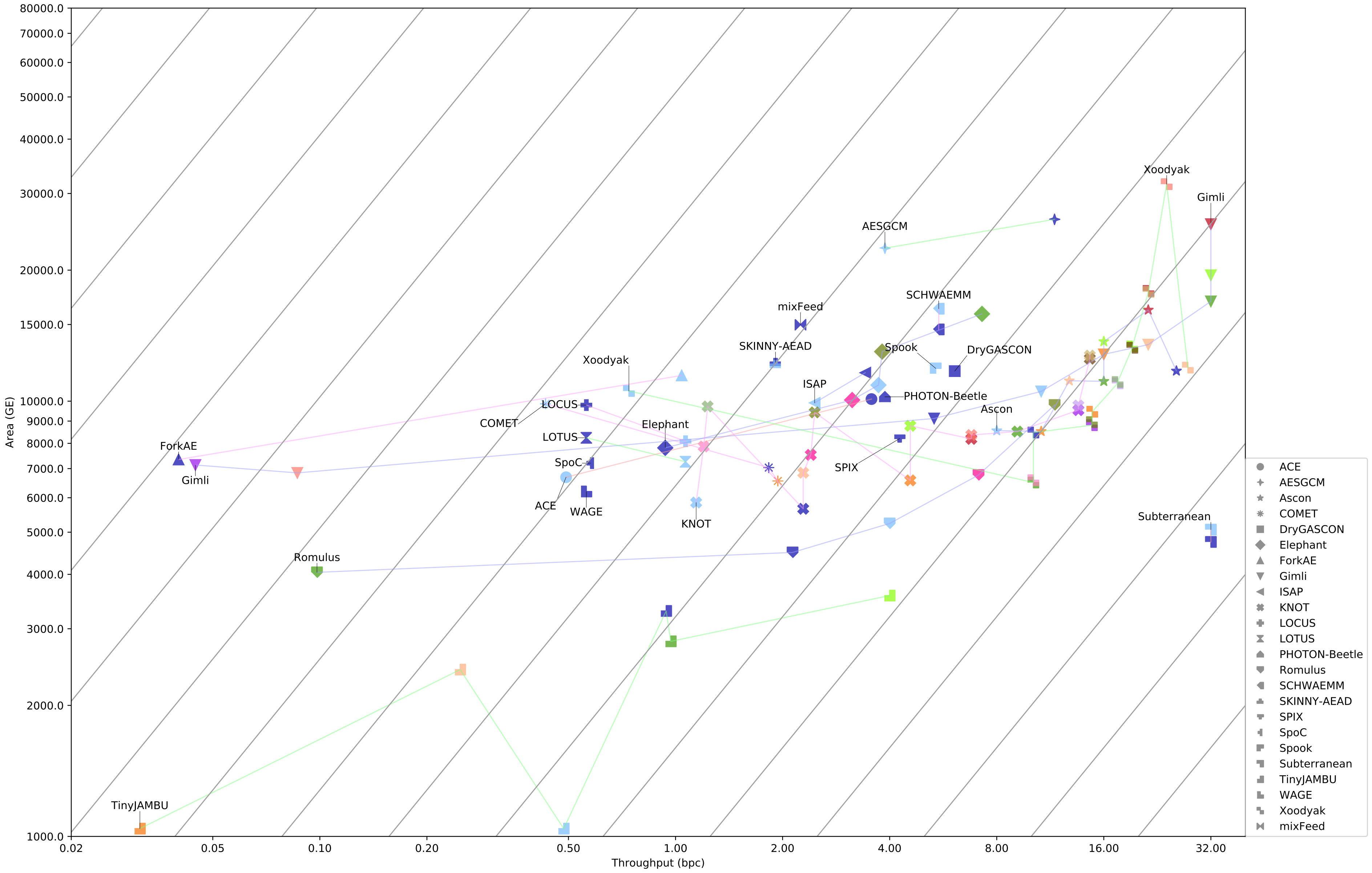

Figure A.7: Area vs throughput for configuration D1

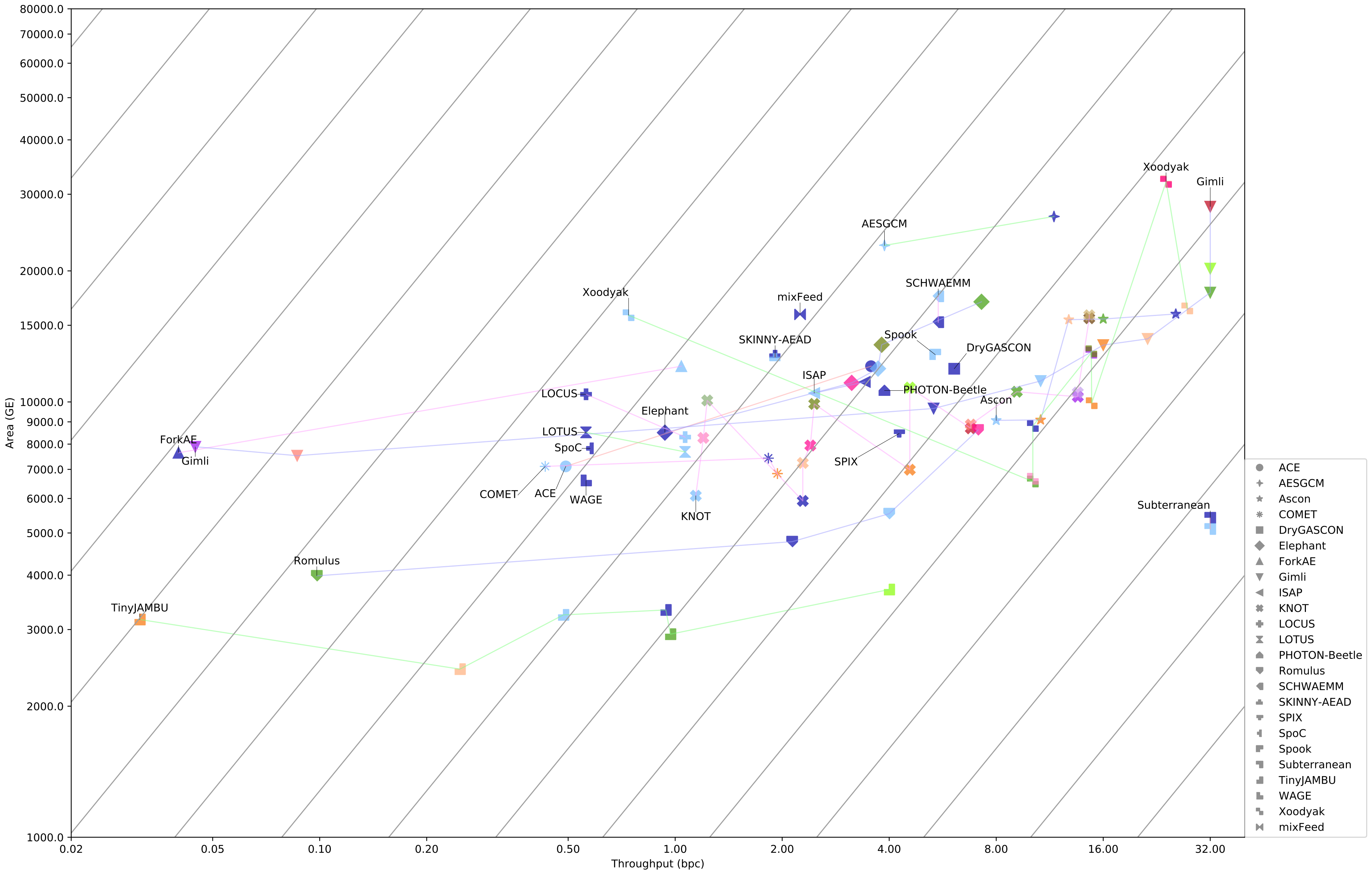

Figure A.8: Area vs throughput for configuration D2

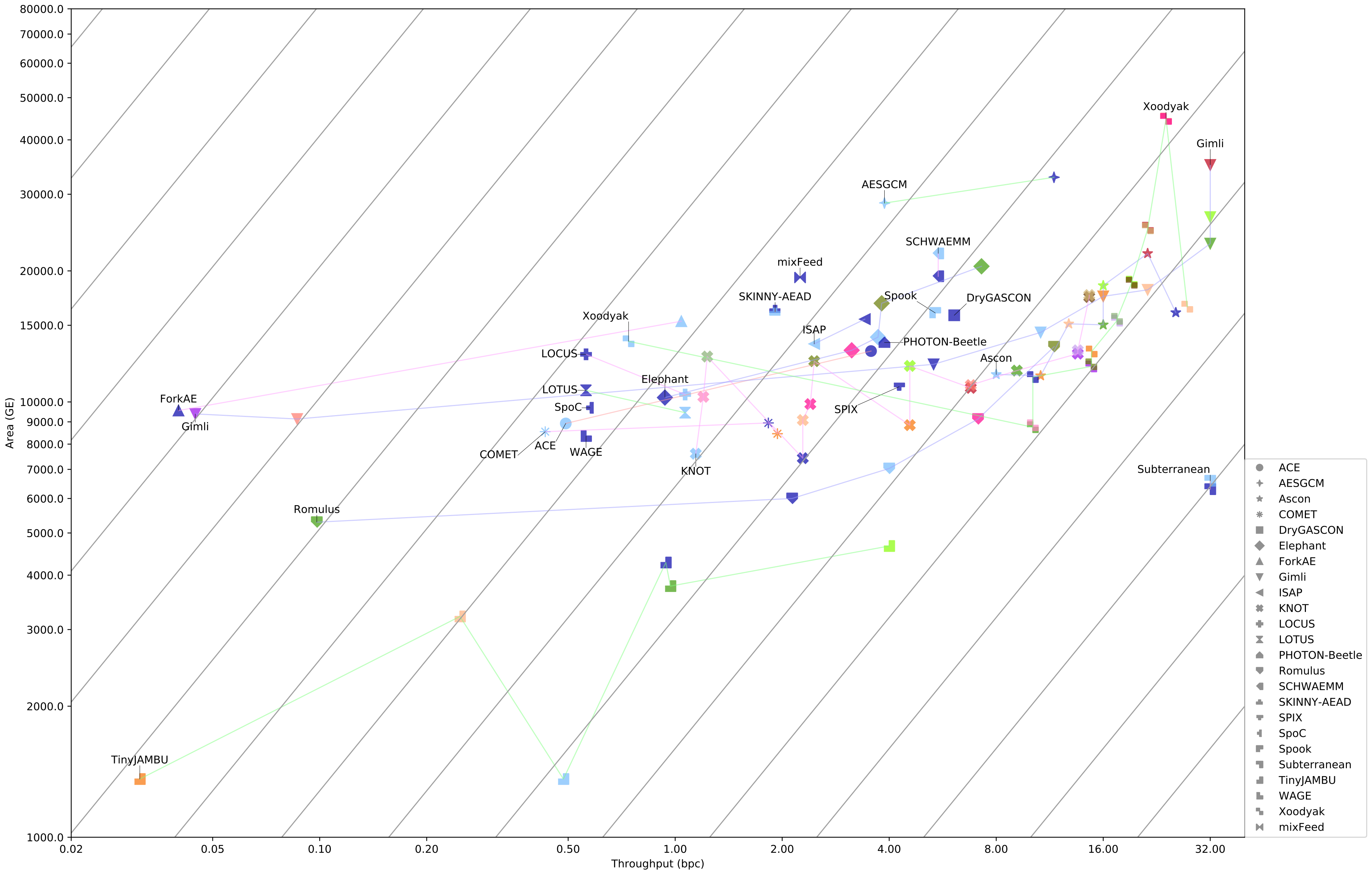

Figure A.9: Area vs throughput for configuration E1

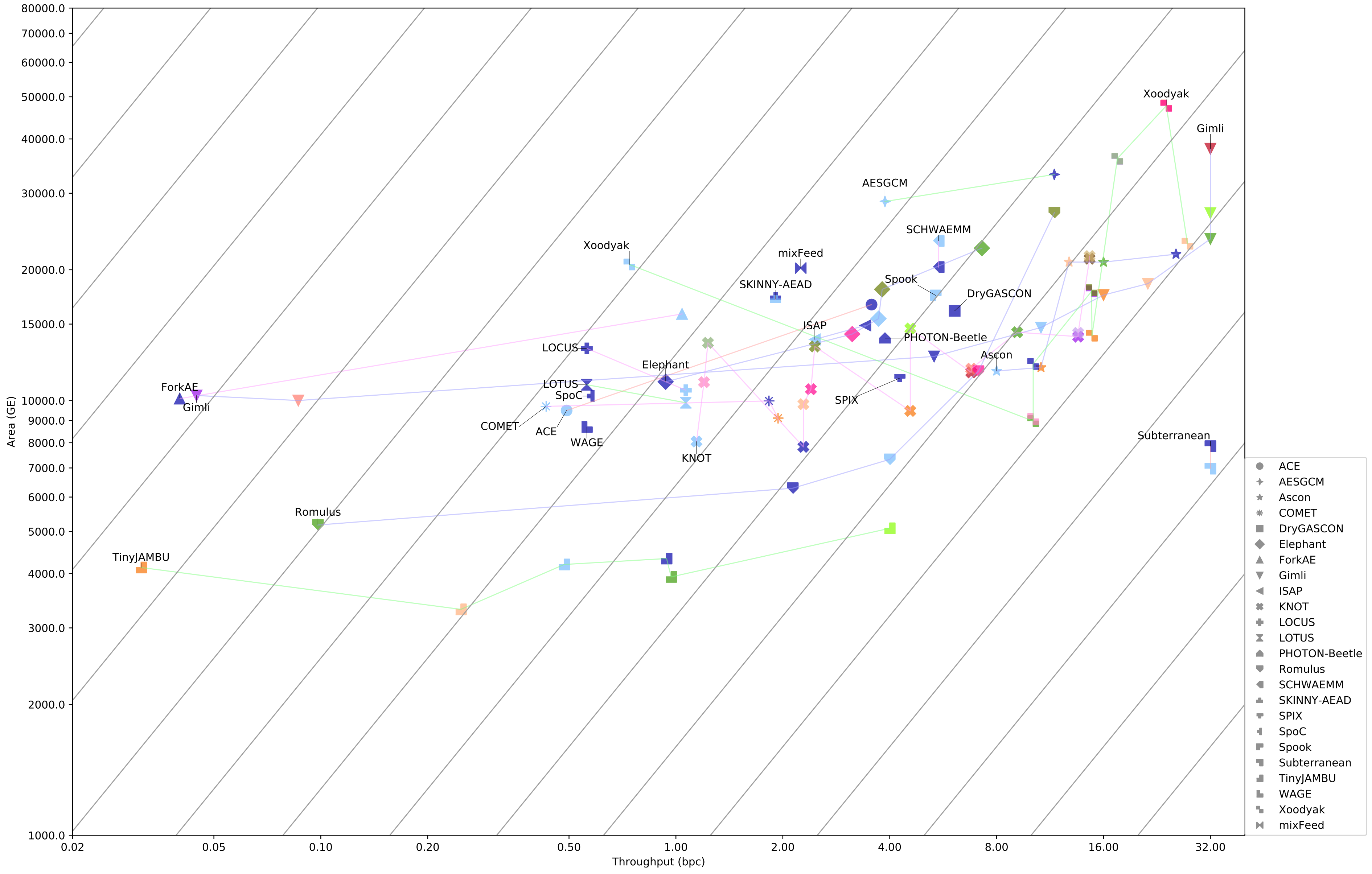

Figure A.10: Area vs throughput for configuration E2

B Energy Details

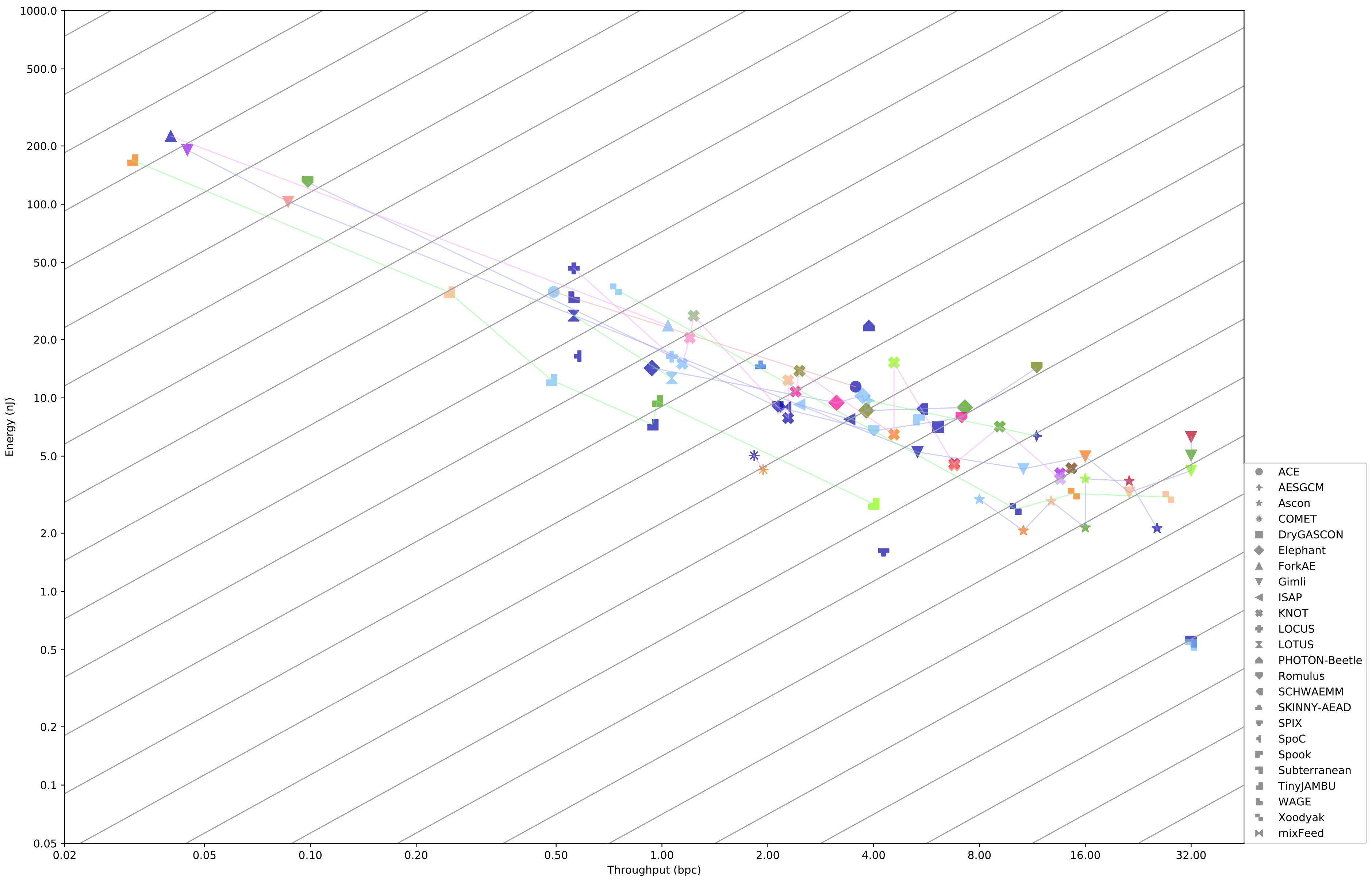

Figure B.1: Energy vs throughput for configuration A1 at 50 MHz

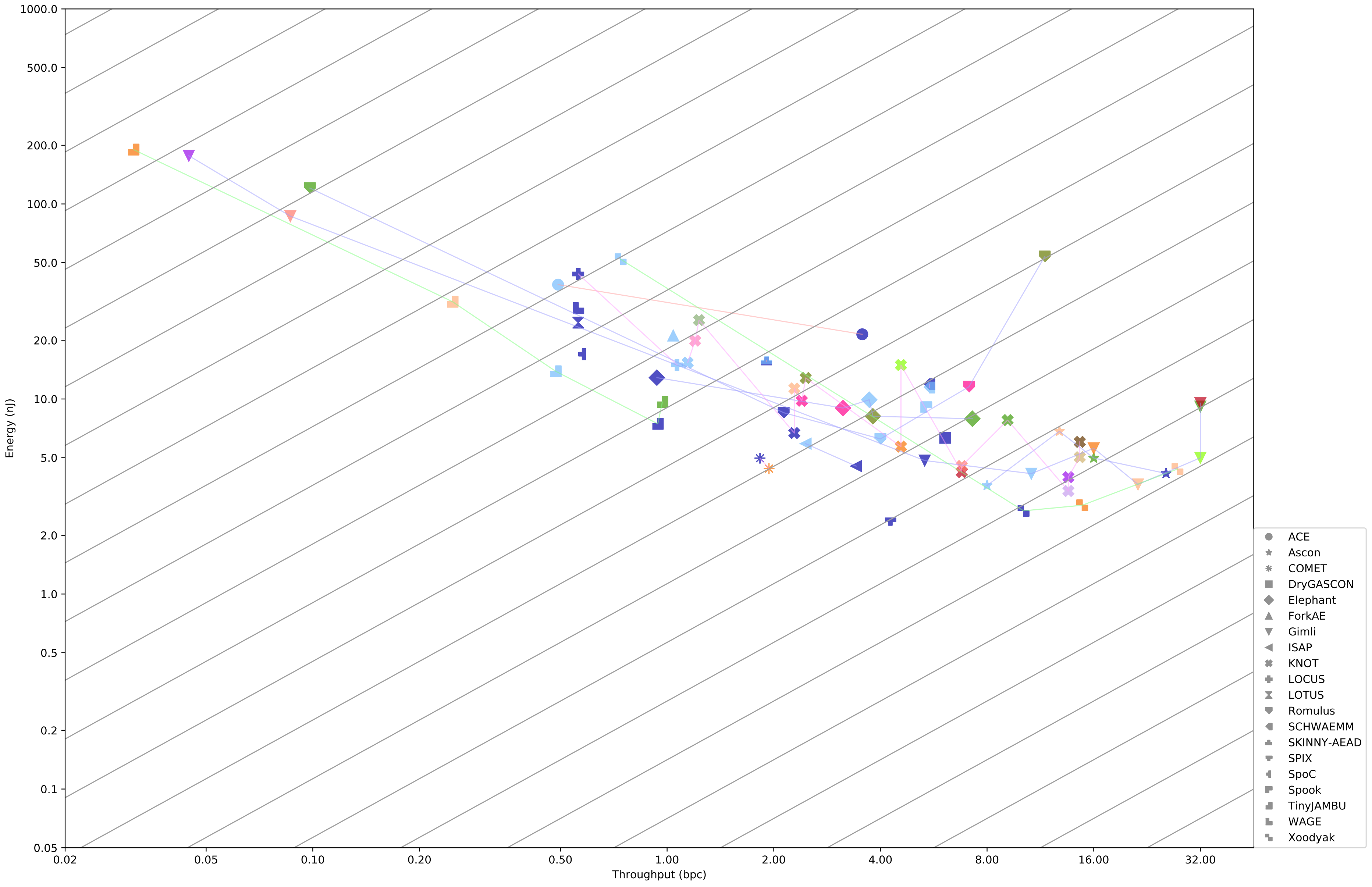

Figure B.2: Energy vs throughput for configuration A2 at 50 MHz

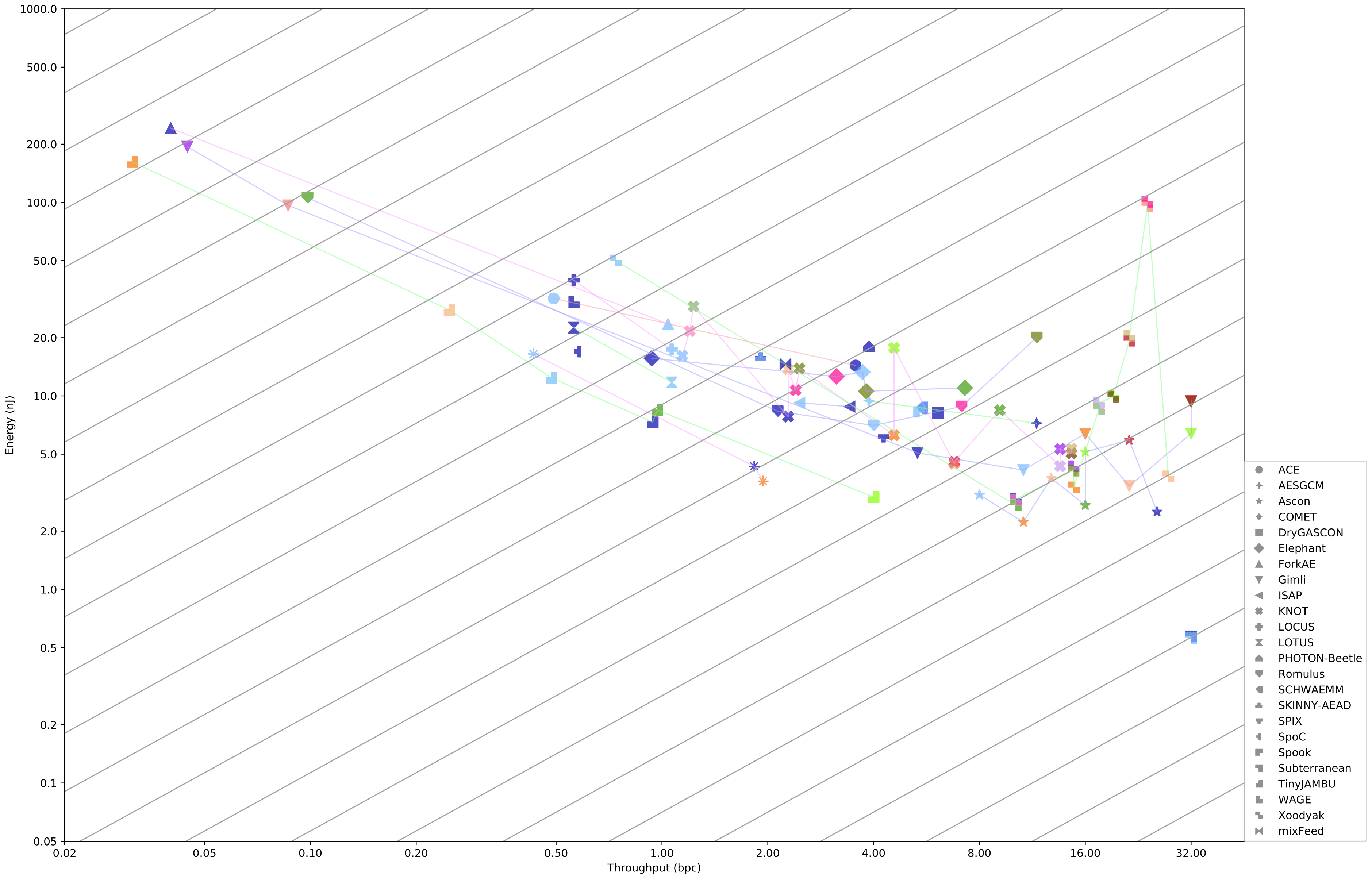

Figure B.3: Energy vs throughput for configuration B1 at 50 MHz

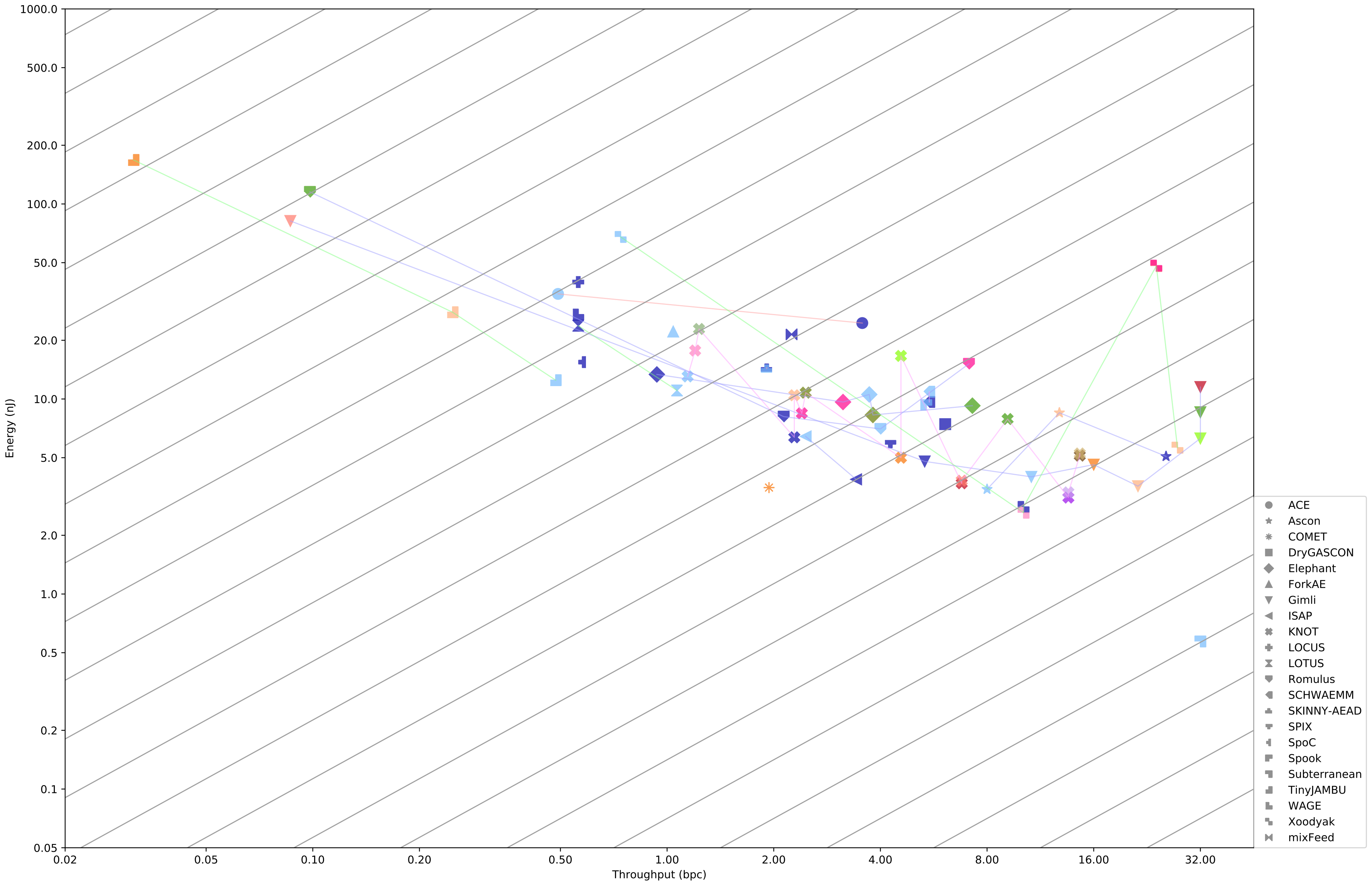

Figure B.4: Energy vs throughput for configuration B2 at 50 MHz

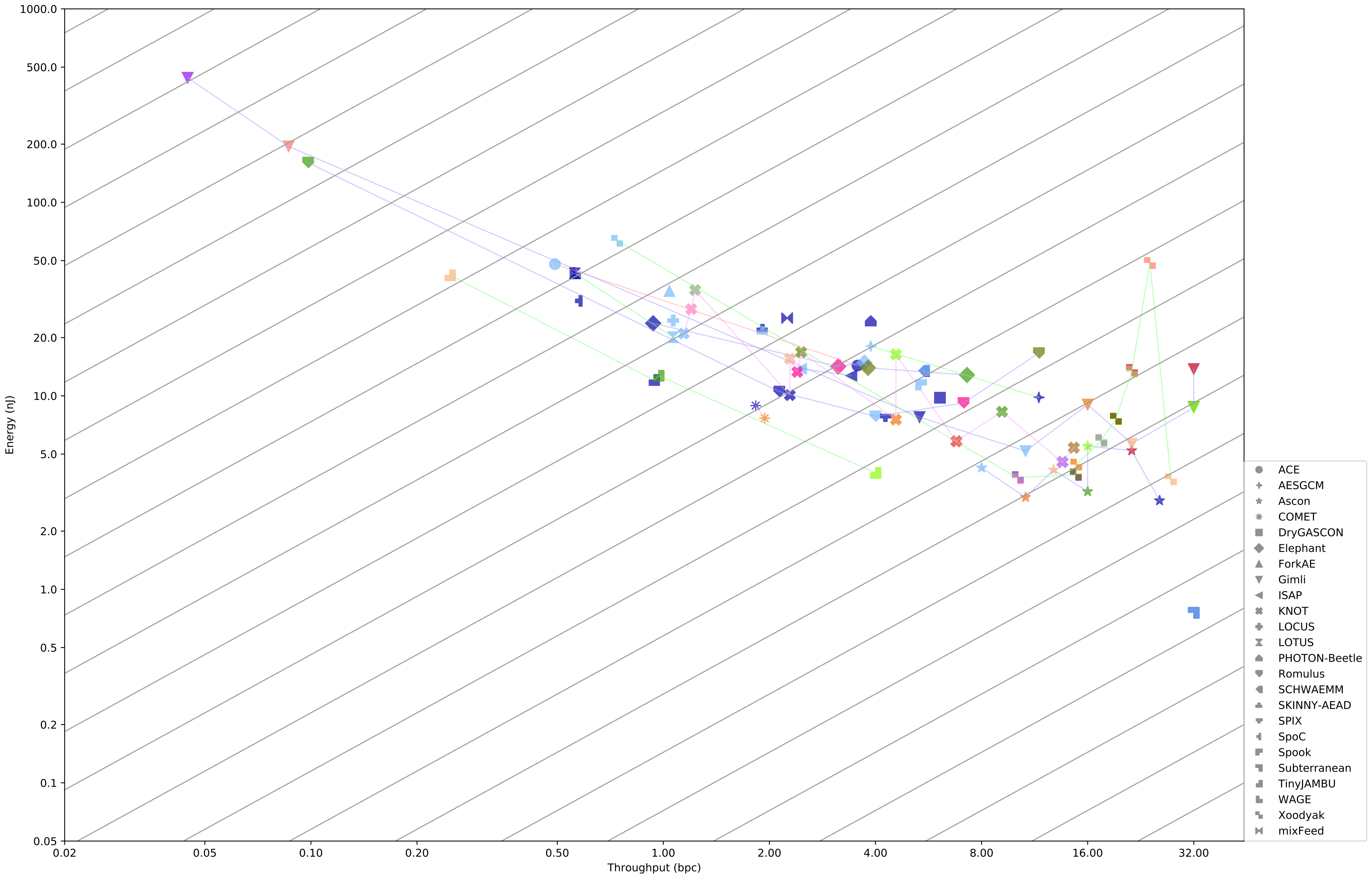

Figure B.5: Energy vs throughput for configuration D1 at 50 MHz

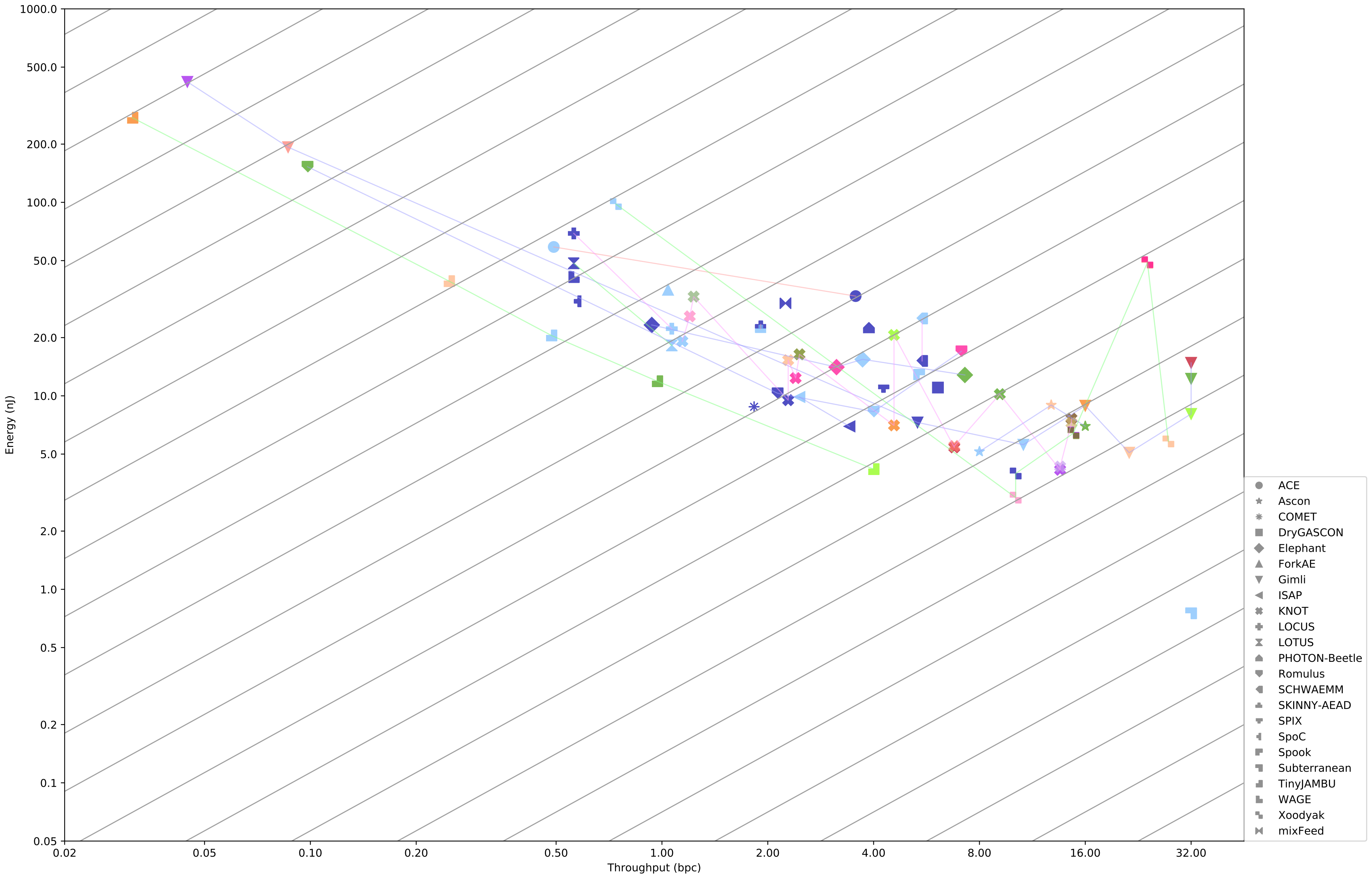

Figure B.6: Energy vs throughput for configuration D2 at 50 MHz

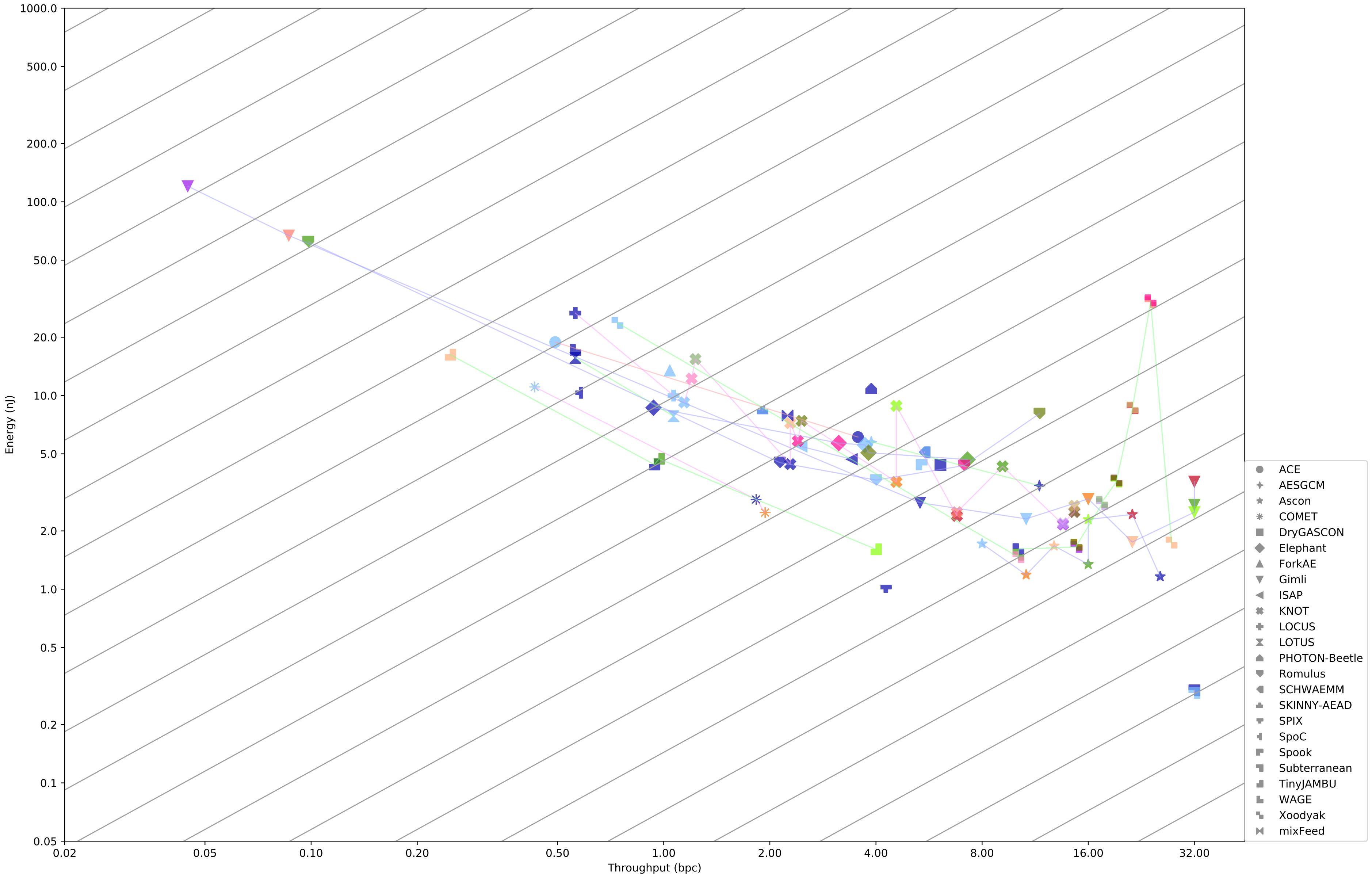

Figure B.7: Energy vs throughput for configuration E1 at 50 MHz

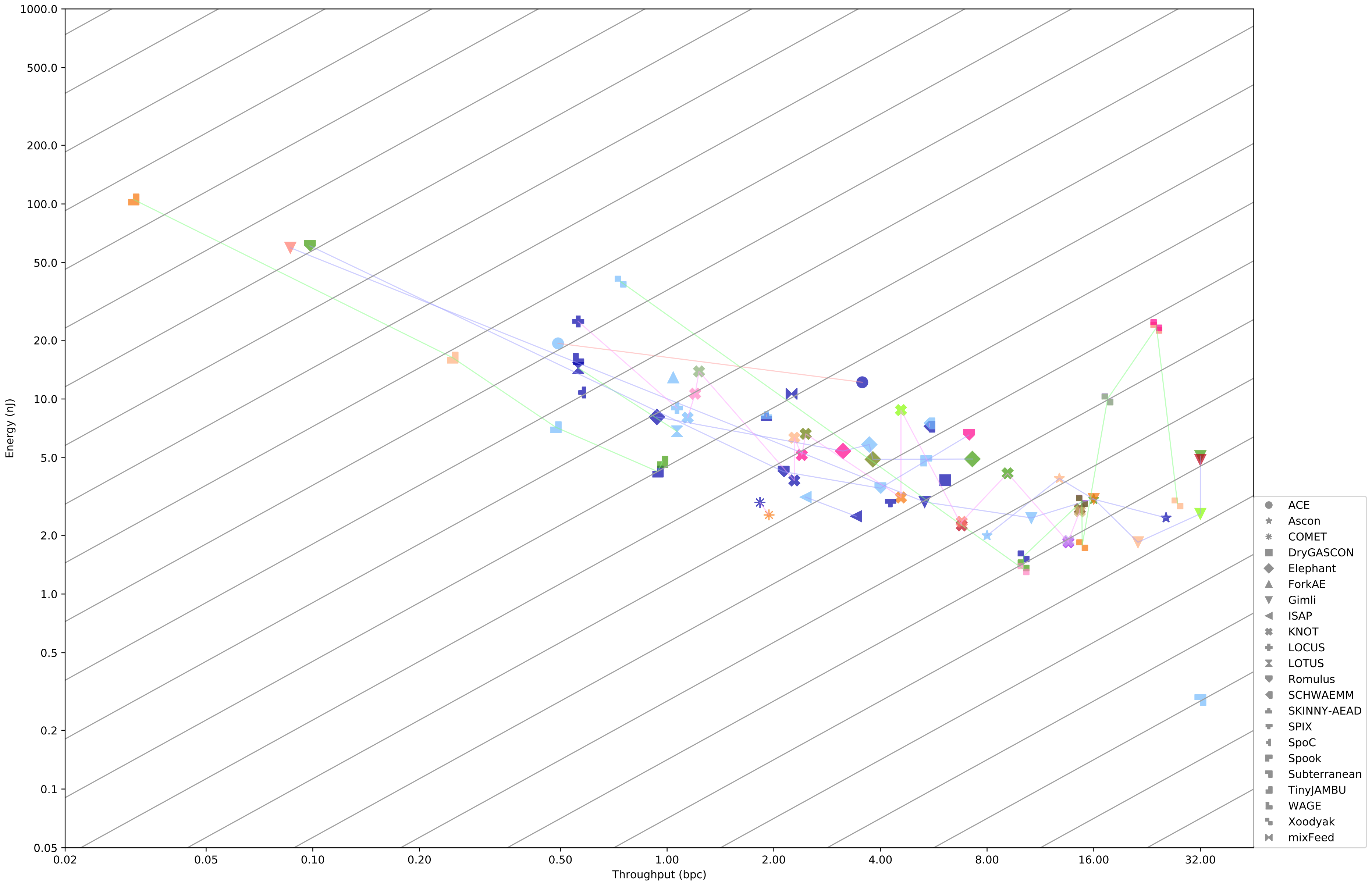

Figure B.8: Energy vs throughput for configuration E2 at 50 MHz

C Area × Energy Details

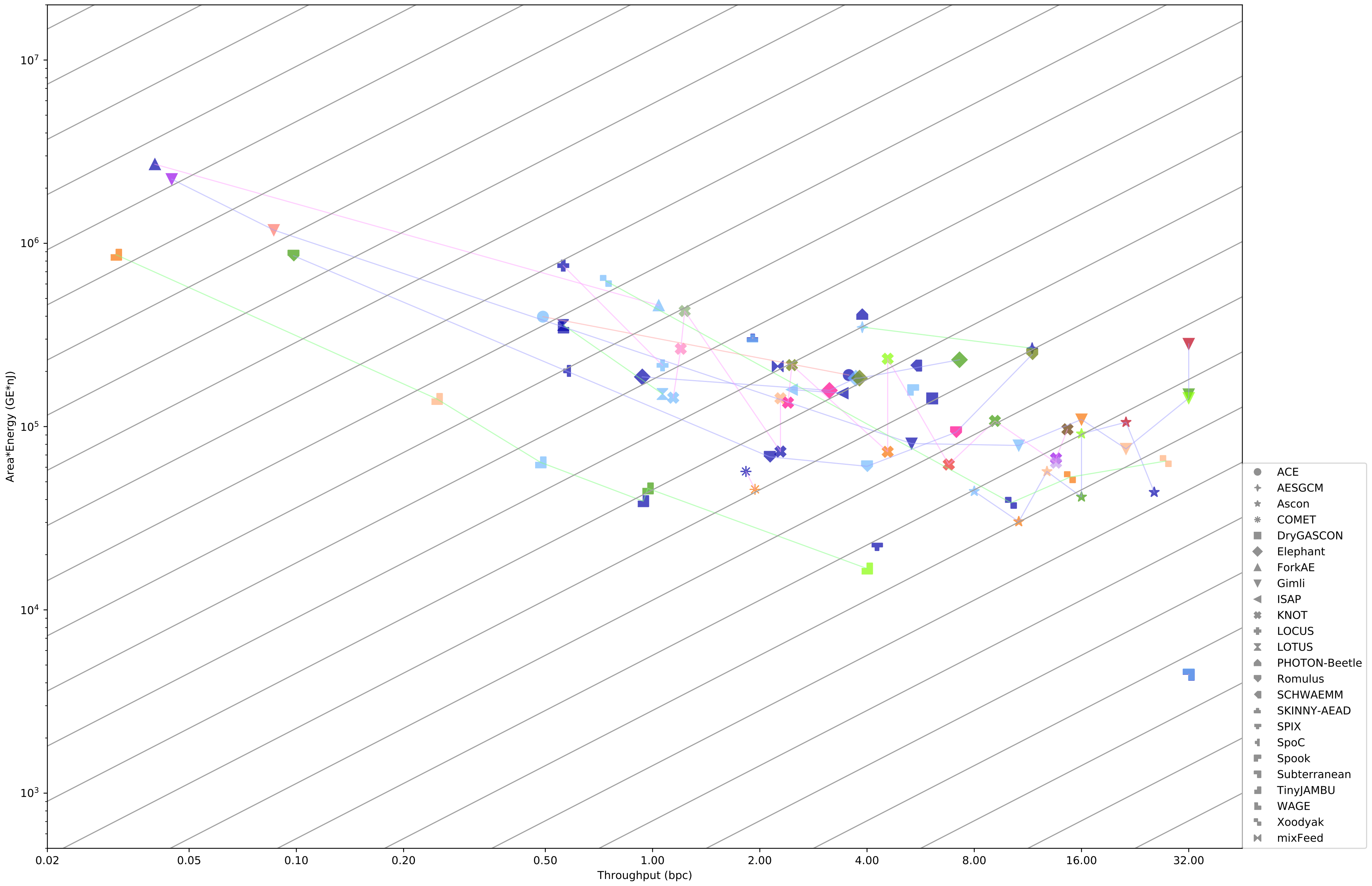

Figure C.1: Area×energy vs throughput for configuration A1 at 50 MHz

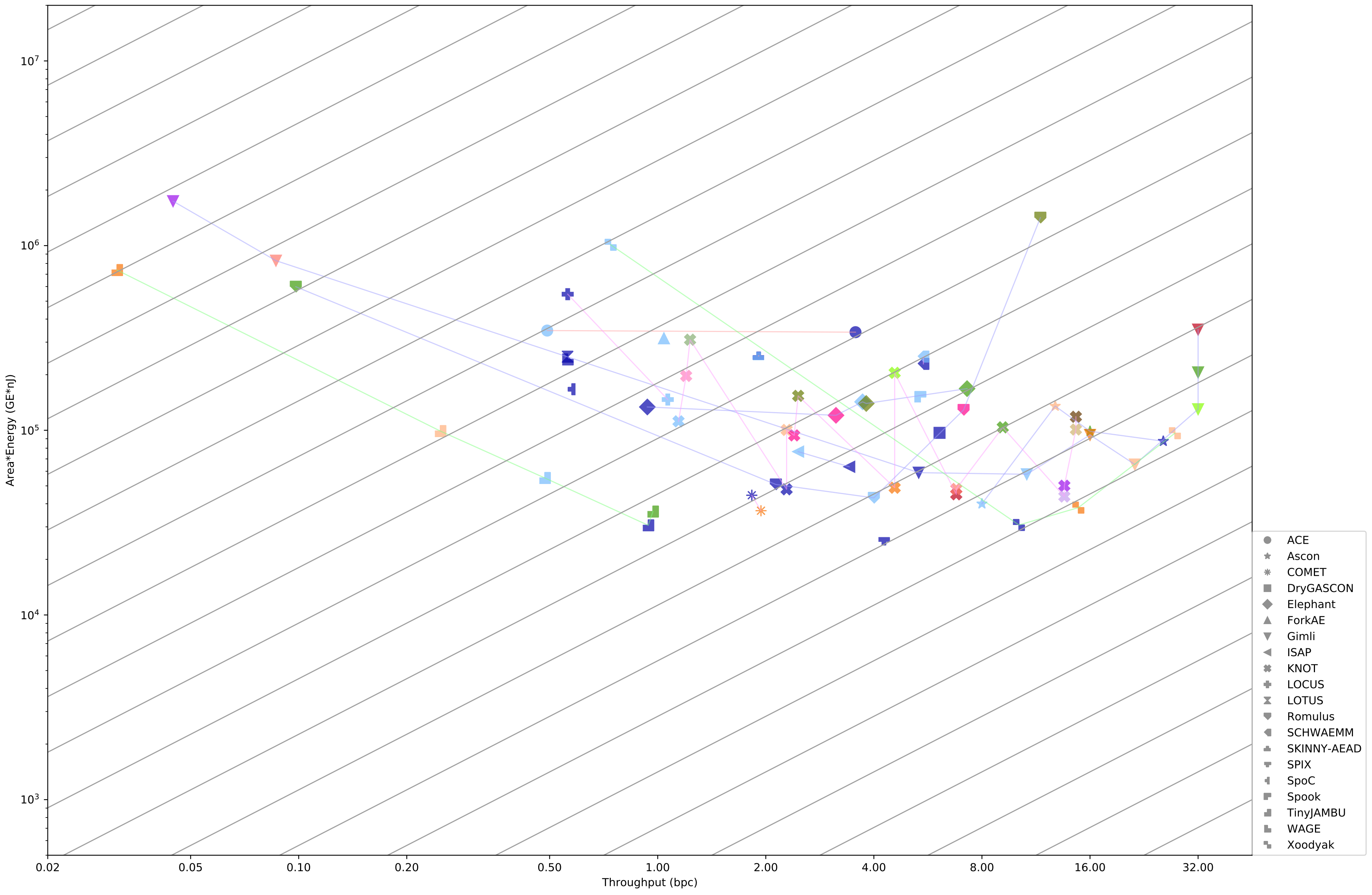

Figure C.2: Area×energy vs throughput for configuration A2 at 50 MHz

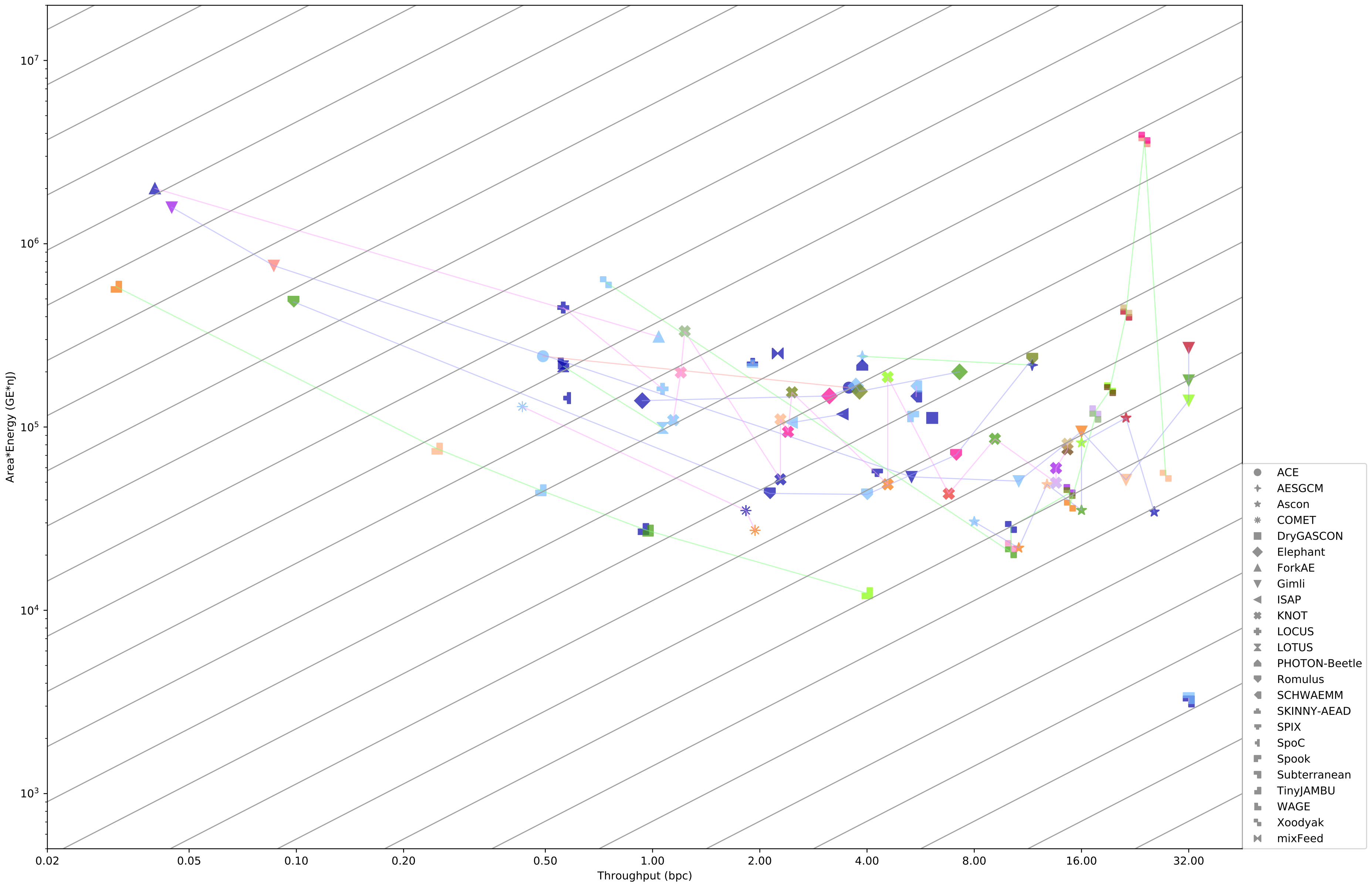

Figure C.3: Area×energy vs throughput for configuration B1 at 50 MHz

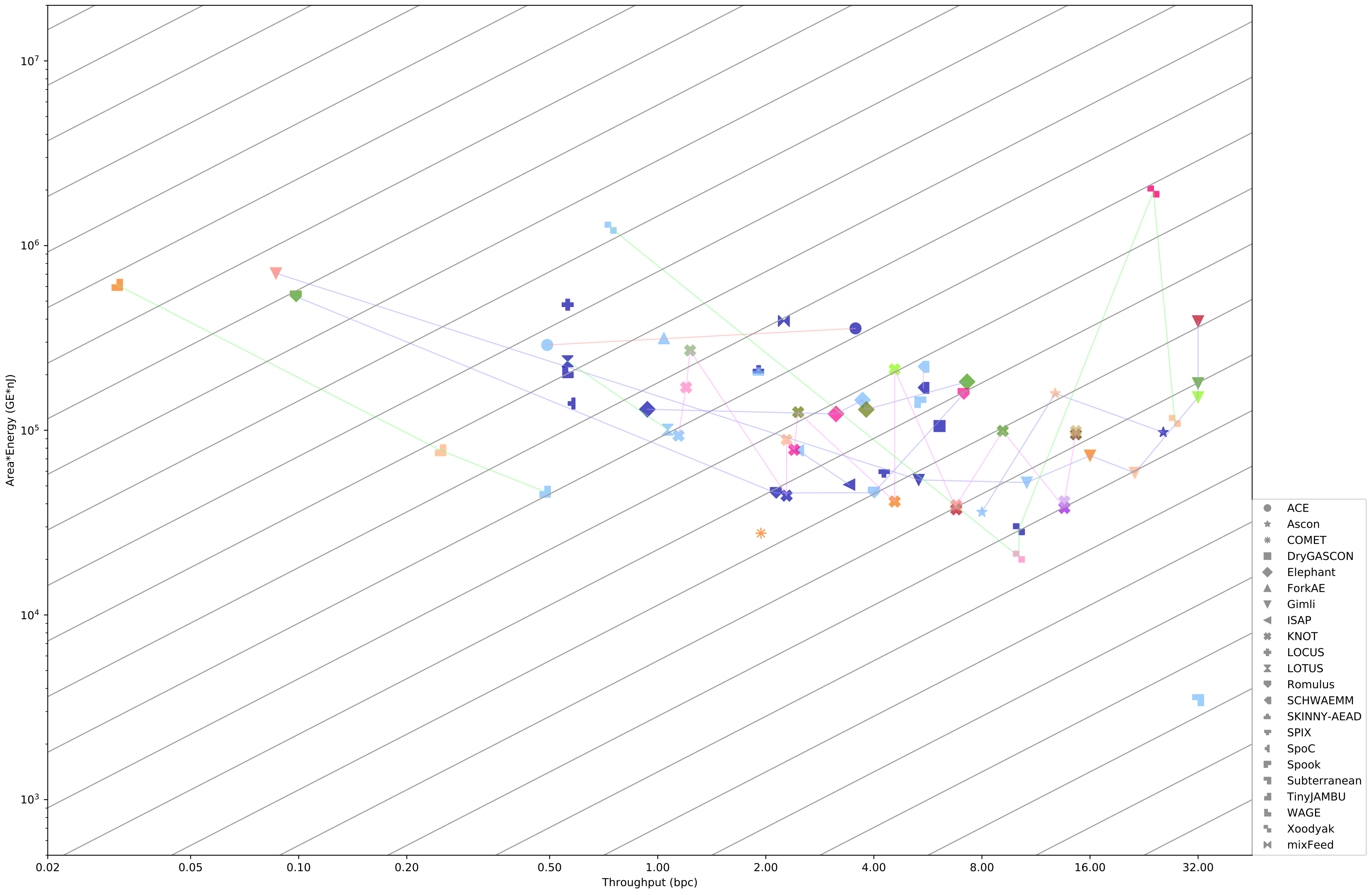

Figure C.4: Area×energy vs throughput for configuration B2 at 50 MHz

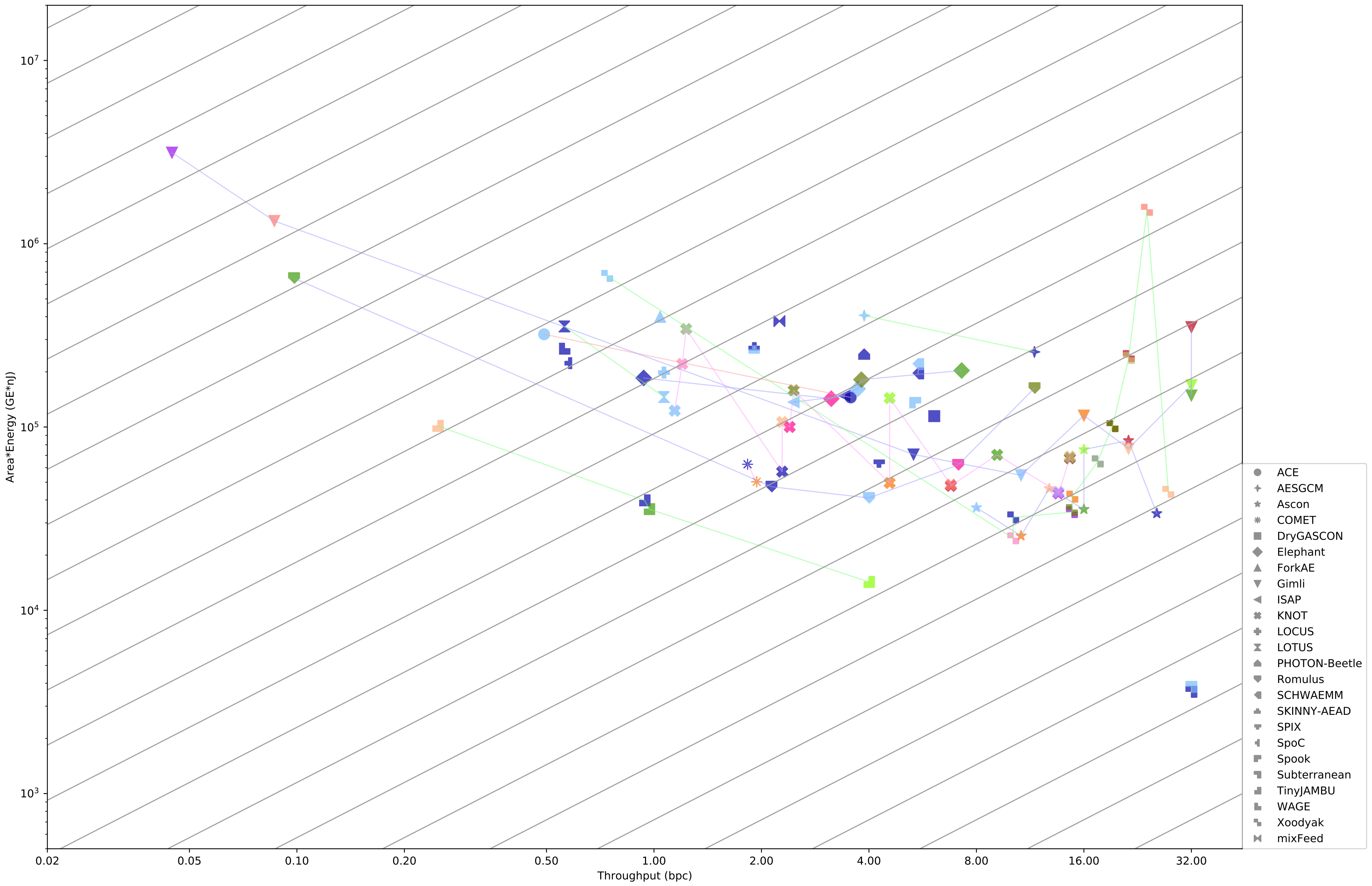

Figure C.5: Area×energy vs throughput for configuration D1 at 50 MHz

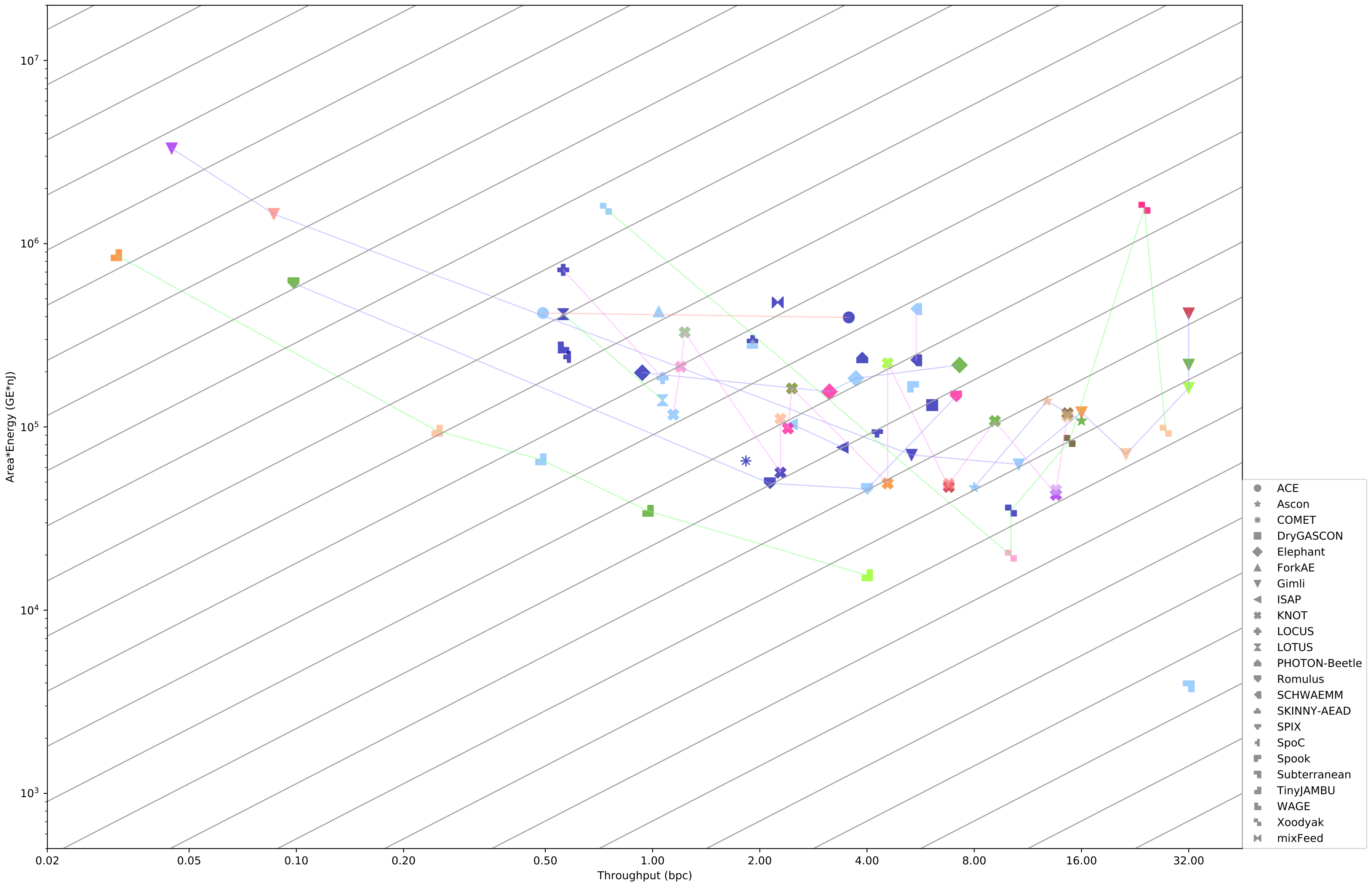

Figure C.6: Area×energy vs throughput for configuration D2 at 50 MHz

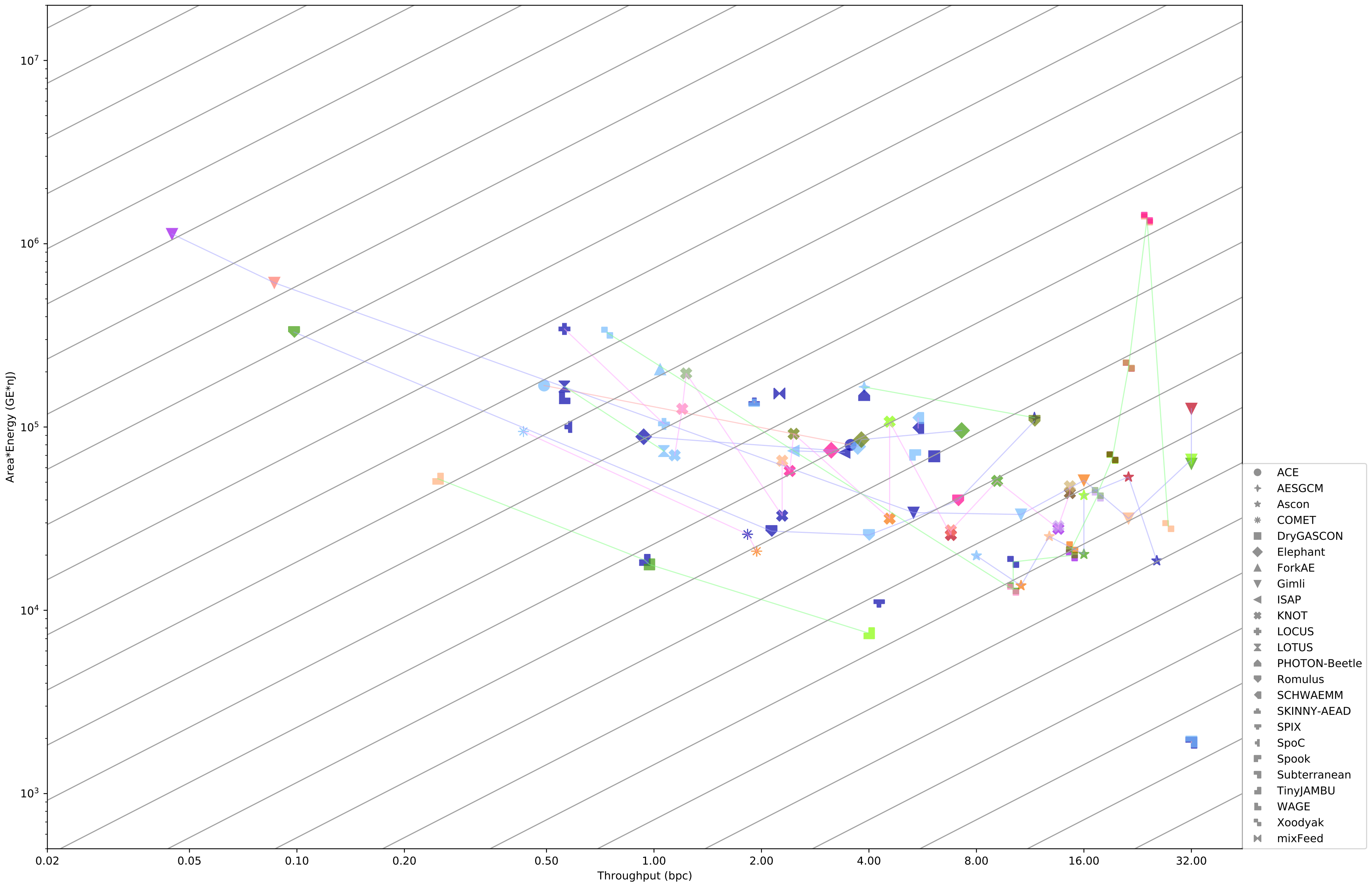

Figure C.7: Area×energy vs throughput for configuration E1 at 50 MHz

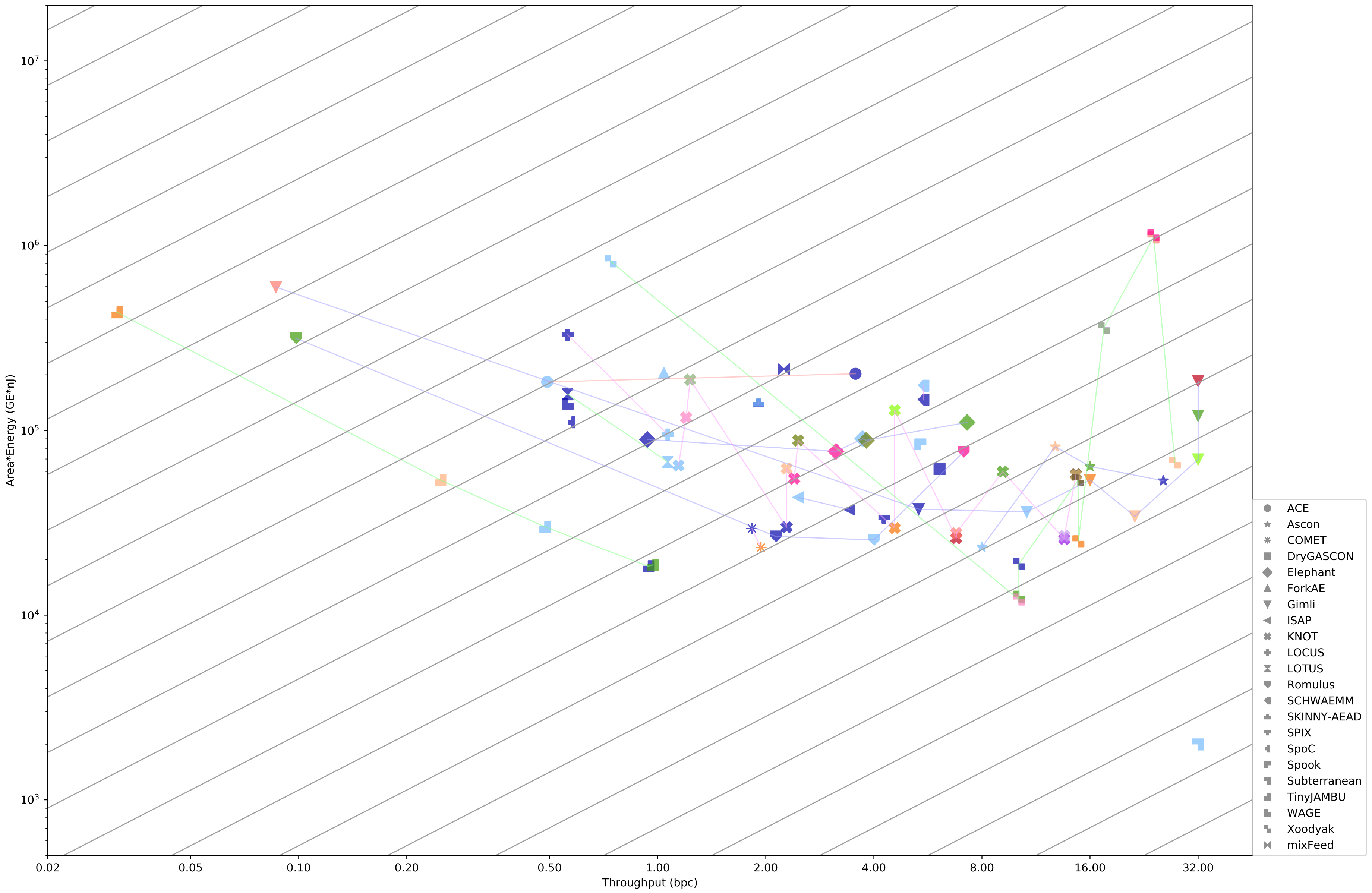

Figure C.8: Area×energy vs throughput for configuration E2 at 50 MHz

D Table of Average Scaled Results

This table summarizes the area, energy, and area×energy results. The first triple of columns show the averaged-scaled results from Figures 4.1, 5.1 and 6.1. The second triple of columns shows the ratio of throughput to each of the metrics. This essentially shows how efficient the cipher instance is with respect to the given metric. The third triple of columns gives the index of the cipher instance with respect to all other cipher instances for that metric.

The final column shows on average how the cipher instance ranks across all three metrics. This average ranking should be interpreted carefully and narrowly. First, cipher instances with higher throughputs benefit when measuring energy and area×energy. This means that most of the overall high ranking instances have high throughputs. These high-throughput instances might not be the be best choice for low-bandwidth applications. Second, the ranking is merely an ordering of instances, the ranking does not indicate the relative distance between an instance and other instances.

Table 3: Summary of average scaled data

		Throughput	Average scaled values			Ratio of throughput to			Index			Average
	Cipher	(bpc)	area	energy	area×energy	area	energy	area×energy	area	energy	area×energy	index
0	Subterranean_ST-v2	32.00	0.55	0.05	0.03	58.60	597.24	1051.39	0	0	0	0.00
1	Subterranean_GMU-v1	32.00	0.56	0.06	0.03	57.51	498.71	960.12	1	1	1	1.00
2	Ascon_GMU-v1	25.60	1.49	0.29	0.46	17.18	89.60	56.13	3	2	2	2.33
3	Xoodyak_GMU2-v2	27.57	1.56	0.40	0.60	17.63	69.52	46.31	2	3	6	3.67
4	Gimli_GT-v4	21.33	1.47	0.37	0.54	14.54	57.77	39.28	7	4	8	6.33
5	Xoodyak_XT-v7	10.14	0.70	0.26	0.19	14.43	39.11	53.80	8	14	3	8.33
6	Xoodyak_GMU2-v1	14.85	1.07	0.35	0.37	13.93	43.00	40.12	10	9	7	8.67
7	Gimli_GT-v6	32.00	2.16	0.56	1.20	14.82	56.72	26.64	5	5	16	8.67
8	Xoodyak_XT-v1	10.14	0.70	0.27	0.19	14.54	37.16	52.10	6	17	4	9.00
9	Ascon_Graz-v2	10.67	0.93	0.23	0.23	11.41	46.41	46.91	19	7	5	10.33
10	Xoodyak_XT-v2	14.81	1.13	0.36	0.42	13.14	40.72	34.85	12	11	11	11.33
11	Xoodyak_XT-v8	14.81	1.14	0.36	0.43	13.00	41.07	34.80	13	10	12	11.67
12	Gimli_GT-v5	32.00	1.87	0.80	1.47	17.10	39.84	21.74	4	13	21	12.67
13	KNOT-v2x2h	13.57	1.09	0.34	0.39	12.50	40.10	35.12	17	12	10	13.00
14	Ascon_Graz-v4	16.00	1.44	0.34	0.52	11.09	47.58	30.74	21	6	15	14.00
15	KNOT-v2x2	13.57	1.06	0.35	0.39	12.78	38.42	34.52	14	15	13	14.00
16	Ascon_Graz-v6	21.33	1.85	0.47	0.87	11.52	45.05	24.54	18	8	18	14.67
17	Xoodyak_GMU-v1	10.14	0.93	0.27	0.27	10.91	37.20	37.98	22	16	9	15.67
18	Xoodyak_XT-v4	19.24	1.51	0.57	0.86	12.77	33.68	22.41	15	19	20	18.00
19	Xoodyak_XT-v10	19.24	1.52	0.66	0.98	12.68	29.14	19.63	16	22	24	20.67
20	Gimli_GT-v3	16.00	1.41	0.52	0.76	11.37	30.87	21.06	20	21	22	21.00
21	Ascon_Graz-v5	16.00	1.56	0.44	0.68	10.25	36.60	23.43	26	18	19	21.00
22	Xoodyak_XT-v9	17.50	1.26	0.81	1.44	13.92	21.62	12.17	11	28	30	23.00
23	Xoodyak_XT-v3	17.50	1.25	0.83	1.46	14.03	21.15	12.02	9	29	31	23.00
24	Gimli_GT-v7	32.00	2.96	0.96	2.77	10.80	33.25	11.54	23	20	32	25.00
25	Ascon_Graz-v1	8.00	0.93	0.32	0.32	8.56	24.62	24.99	33	26	17	25.33
26	TinyJAMBU_TJT-v3	4.00	0.39	0.29	0.12	10.16	13.74	33.74	27	35	14	25.33

Table 3: Summary of average scaled data (cont'd)

		Cipher	Average scaled values			Ratio of throughput to			Index			Average
	Throughput	(bpc)	area	energy	area×energy	area	energy	area×energy	area	energy	area×energy	index
27	KNOT-v2x4	14.62	1.53	0.51	0.78	9.53	28.41	18.74	28	24	25	25.67
28	KNOT-v2x4h	14.62	1.56	0.51	0.81	9.36	28.53	18.13	29	23	26	26.00
29	Gimli_GT-v2	10.67	1.17	0.45	0.51	9.15	23.92	20.77	30	27	23	26.67
30	Ascon_Graz-v3	12.80	1.44	0.46	0.73	8.86	27.58	17.61	31	25	28	28.00
31	Xoodyak_XT-v11	21.37	2.02	1.09	2.19	10.58	19.54	9.74	24	30	38	30.67
32	KNOT-v2x1	6.79	0.89	0.41	0.38	7.60	16.60	17.70	35	32	27	31.33
33	Xoodyak_XT-v5	21.37	2.03	1.11	2.22	10.51	19.25	9.63	25	31	39	31.67
34	KNOT-v2x1h	6.79	0.91	0.42	0.40	7.46	16.32	17.10	36	34	29	33.00
35	KNOT-v1x4	9.14	1.04	0.78	0.82	8.78	11.68	11.14	32	36	33	33.67
36	Romulus-v3	7.11	0.84	0.92	0.82	8.47	7.72	8.68	34	41	40	38.33
37	Romulus-v2	4.00	0.58	0.61	0.37	6.94	6.54	10.78	38	44	34	38.67
38	Gimli_GT-v1	5.33	1.00	0.51	0.53	5.34	10.43	9.98	42	37	37	38.67
39	SPIX-v1	4.27	0.88	0.45	0.42	4.86	9.39	10.07	43	38	36	39.00
40	KNOT-v1x2	4.57	0.73	0.60	0.45	6.30	7.57	10.22	41	42	35	39.33
41	DryGASCON-v1	6.10	1.28	0.71	0.95	4.77	8.63	6.42	44	39	41	41.33
42	Romulus-v4	11.64	1.61	1.81	2.00	7.23	6.41	5.81	37	45	42	41.33
43	AESGCM-v1	11.63	2.75	0.71	2.05	4.22	16.32	5.68	48	33	43	41.33
44	Elephant-v5	7.27	1.72	0.88	1.58	4.23	8.30	4.59	47	40	48	45.00
45	Spook-v2	5.39	1.34	0.82	1.15	4.03	6.57	4.68	49	43	47	46.33
46	Romulus-v1	2.13	0.49	0.78	0.41	4.31	2.73	5.26	45	58	45	49.33
47	SCHWAEMM-v1	5.49	1.60	0.97	1.62	3.44	5.66	3.39	52	46	51	49.67
48	Xoodyak_XT-v12	24.03	3.58	4.34	15.04	6.72	5.53	1.60	39	47	64	50.00
49	KNOT-v1x1	2.29	0.61	0.71	0.46	3.72	3.20	4.97	50	54	46	50.00
50	Xoodyak_XT-v6	24.03	3.58	4.64	16.21	6.71	5.18	1.48	40	48	65	51.00
51	COMET_CI-v3	1.94	0.71	0.49	0.36	2.74	3.95	5.33	61	51	44	52.00
52	Elephant-v2	3.72	1.20	1.05	1.31	3.11	3.55	2.85	54	52	54	53.33
53	SCHWAEMM-v2	5.49	1.80	1.07	2.01	3.05	5.16	2.73	55	49	56	53.33
54	KNOT-v1x4h	4.57	1.07	1.63	1.75	4.29	2.81	2.62	46	57	58	53.67
55	Elephant-v4	3.81	1.39	0.93	1.37	2.74	4.10	2.79	62	50	55	55.67
56	COMET_CI-v1	1.83	0.76	0.60	0.48	2.40	3.07	3.85	64	56	49	56.33
57	KNOT-v3	2.40	0.82	0.96	0.83	2.93	2.50	2.89	58	59	53	56.67
58	Elephant-v3	3.14	1.10	0.99	1.15	2.85	3.15	2.72	60	55	57	57.33
59	TinyJAMBU_TJT-v2	0.97	0.31	0.88	0.29	3.15	1.10	3.39	53	70	50	57.67
60	PHOTON-Beetle-v1	3.88	1.11	1.99	2.33	3.49	1.95	1.66	51	62	62	58.33
61	KNOT-v1x2h	2.29	0.75	1.25	0.95	3.04	1.83	2.42	56	64	59	59.67
62	ISAP-v1	3.42	1.19	1.39	1.85	2.87	2.46	1.85	59	60	60	59.67
63	ACE_GMU-v1	3.56	1.20	1.78	2.17	2.97	2.00	1.64	57	61	63	60.33
64	TinyJAMBU_GMU-v1	0.94	0.35	0.80	0.29	2.70	1.18	3.21	63	69	52	61.33
65	AESGCM-v2	3.88	2.37	1.20	2.95	1.64	3.24	1.32	69	53	66	62.67
66	KNOT-v4	2.46	1.03	1.33	1.38	2.39	1.85	1.79	65	63	61	63.00
67	ISAP-v2	2.46	1.08	1.69	2.00	2.29	1.46	1.23	66	65	67	66.00

Table 3: Summary of average scaled data (cont'd)

		Cipher	Average scaled values			Ratio of throughput to			Index			Average
	Throughput	(bpc)	area	energy	area×energy	area	energy	area×energy	area	energy	area×energy	index
68	SKINNY-AEAD-v2	1.91	1.32	1.56	2.17	1.44	1.23	0.88	70	67	71	69.33
69	KNOT-v1x1h	1.14	0.63	1.48	0.98	1.81	0.77	1.17	68	72	68	69.33
70	SKINNY-AEAD-v1	1.91	1.34	1.57	2.21	1.42	1.22	0.86	71	68	72	70.33
71	mixFeed-v1	2.25	1.61	1.73	2.90	1.40	1.30	0.77	73	66	73	70.67
72	TinyJAMBU_GMU-v2	0.48	0.26	1.26	0.45	1.83	0.38	1.07	67	78	69	71.33
73	LOTUS-v2	1.07	0.78	1.36	1.12	1.37	0.78	0.96	74	71	70	71.67
74	KNOT-v3h	1.20	0.84	1.97	1.76	1.42	0.61	0.68	72	74	75	73.67
75	LOCUS-v2	1.07	0.86	1.74	1.56	1.24	0.61	0.68	75	73	74	74.00
76	Elephant-v1	0.94	0.85	1.60	1.44	1.10	0.59	0.65	77	75	76	76.00
77	KNOT-v4h	1.23	1.05	2.72	2.88	1.17	0.45	0.43	76	76	77	76.33
78	ForkAE-v2	1.04	1.25	2.42	3.18	0.83	0.43	0.33	79	77	79	78.33
79	SpoC-v1	0.58	0.81	2.00	1.70	0.71	0.29	0.34	81	79	78	79.33
80	WAGE-v1	0.56	0.69	3.01	2.17	0.81	0.19	0.26	80	82	82	81.33
81	TinyJAMBU_TJT-v1	0.25	0.26	2.95	0.80	0.96	0.08	0.31	78	86	80	81.33
82	LOTUS-v1	0.56	0.87	2.81	2.59	0.65	0.20	0.22	83	81	83	82.33
83	COMET_CI-v2	0.43	0.82	2.01	1.50	0.53	0.21	0.29	86	80	81	82.33
84	ACE_UW-v1	0.49	0.74	3.58	2.77	0.67	0.14	0.18	82	84	84	83.33
85	Xoodyak_GMU-v2	0.74	1.39	5.30	7.92	0.53	0.14	0.09	84	83	86	84.33
86	LOCUS-v1	0.56	1.05	4.31	4.85	0.53	0.13	0.12	85	85	85	85.00
87	Romulus-v5	0.10	0.42	11.40	5.05	0.23	0.01	0.02	87	87	87	87.00
88	Gimli_TUM-v2	0.09	0.77	12.16	9.79	0.11	0.01	0.01	89	88	88	88.33
89	TinyJAMBU_GMU-v3	0.03	0.26	17.48	6.14	0.12	0.00	0.01	88	89	89	88.67
90	Gimli_TUM-v3	0.04	0.80	25.62	21.36	0.06	0.00	0.00	90	90	90	90.00
91	ForkAE-v1	0.04	0.79	24.72	20.39	0.05	0.00	0.00	91	91	91	91.00