CHALMERS

Efficacy of Pipelining to Reduce Energy of Floating-Point Adders and Multipliers

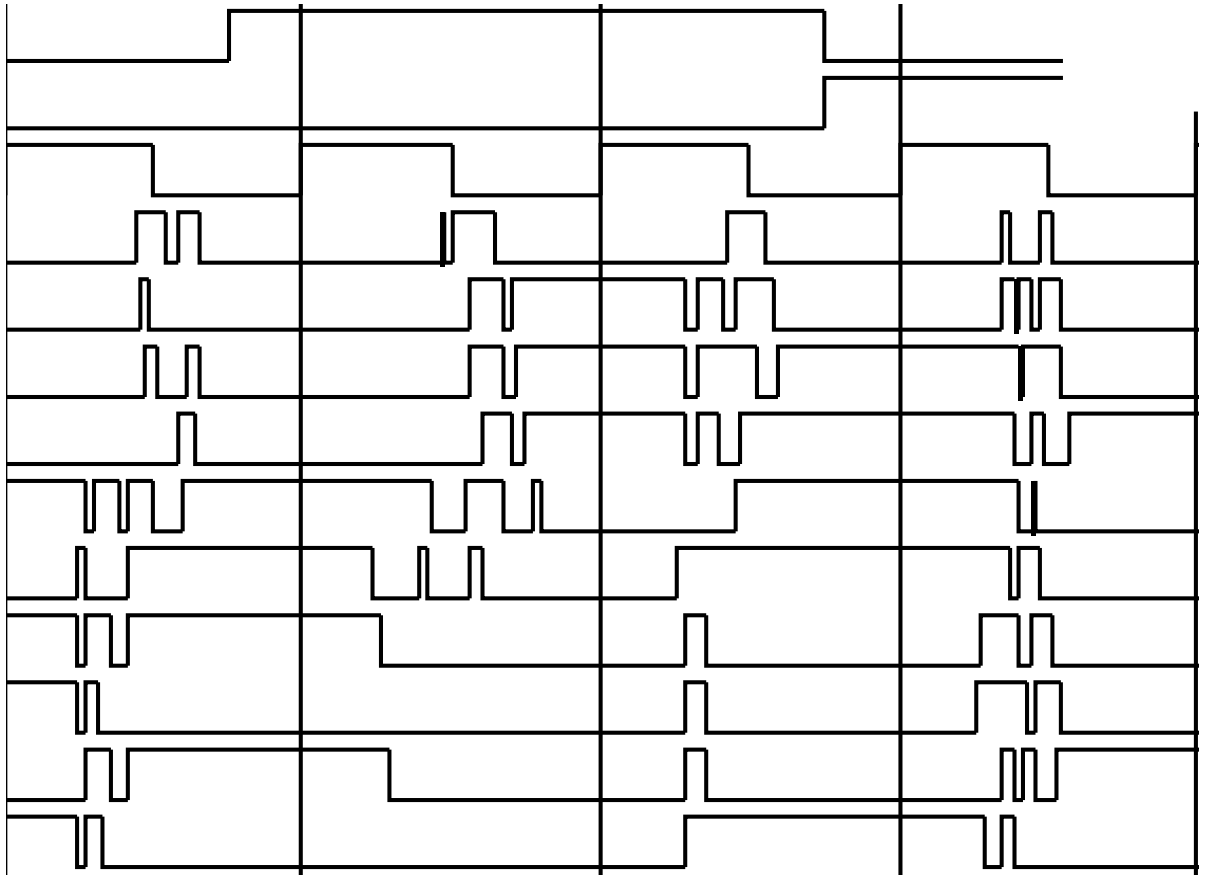
Per Larsson-Edefors and Erik Börjeson

Chalmers University of Technology
Gothenburg, Sweden

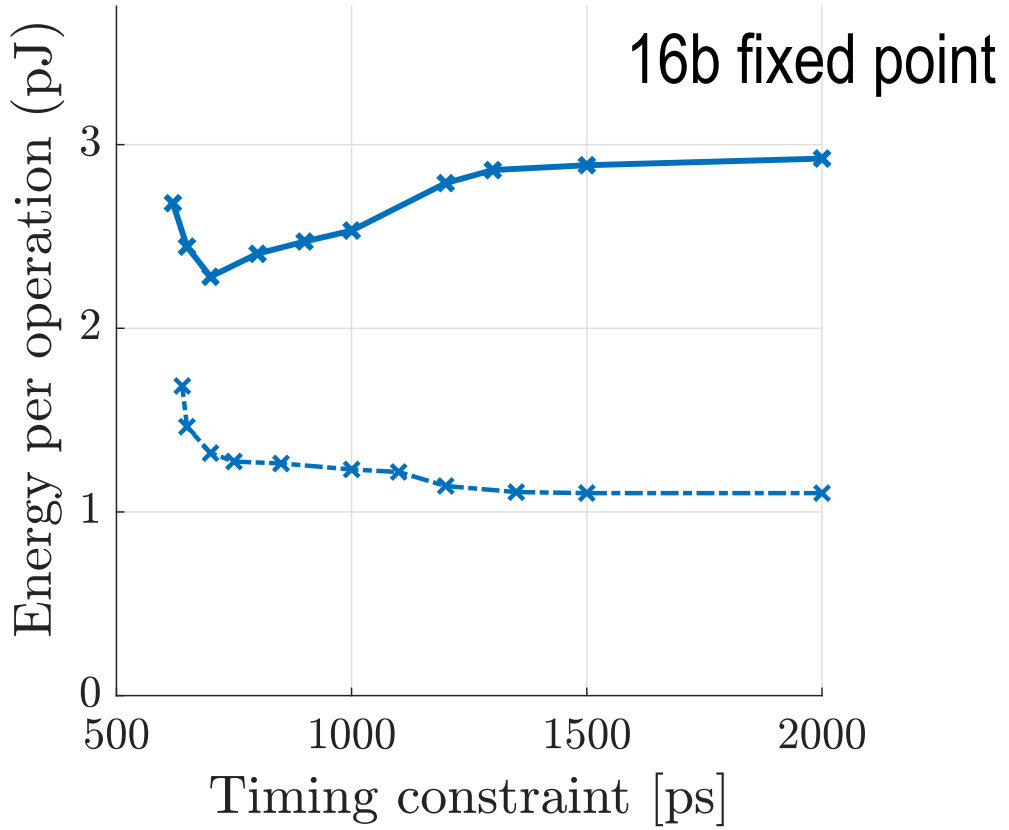
perla@chalmers.se

Stiftelsen för Strategisk Forskning

Problem: Glitch Energy Dissipation

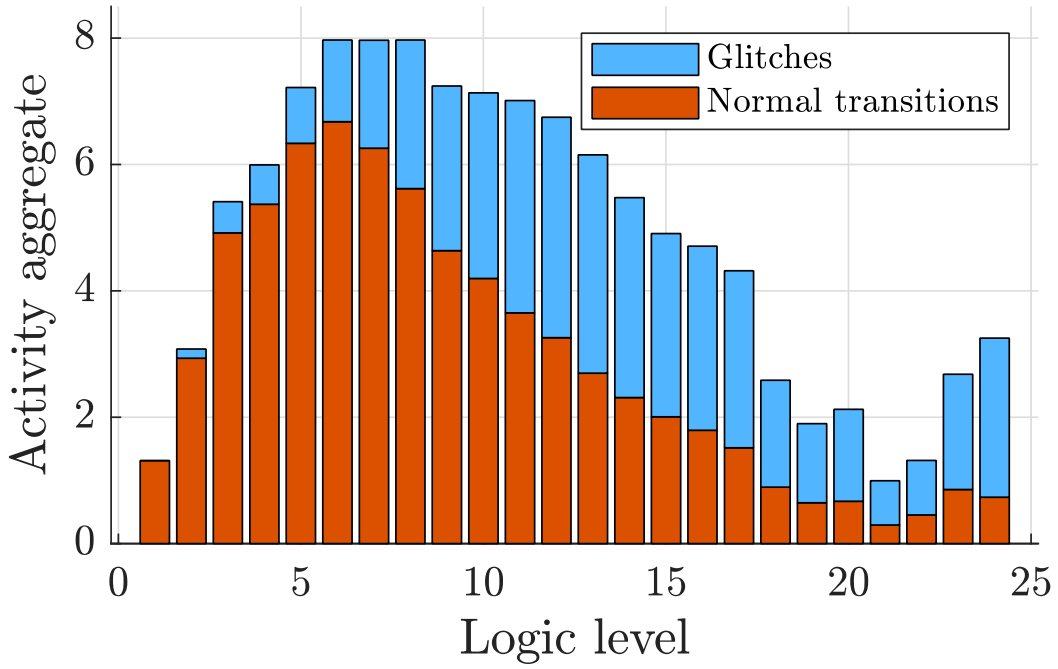


ARITH'25: *Glitches dominate energy dissipation in complex-valued integer multipliers.*

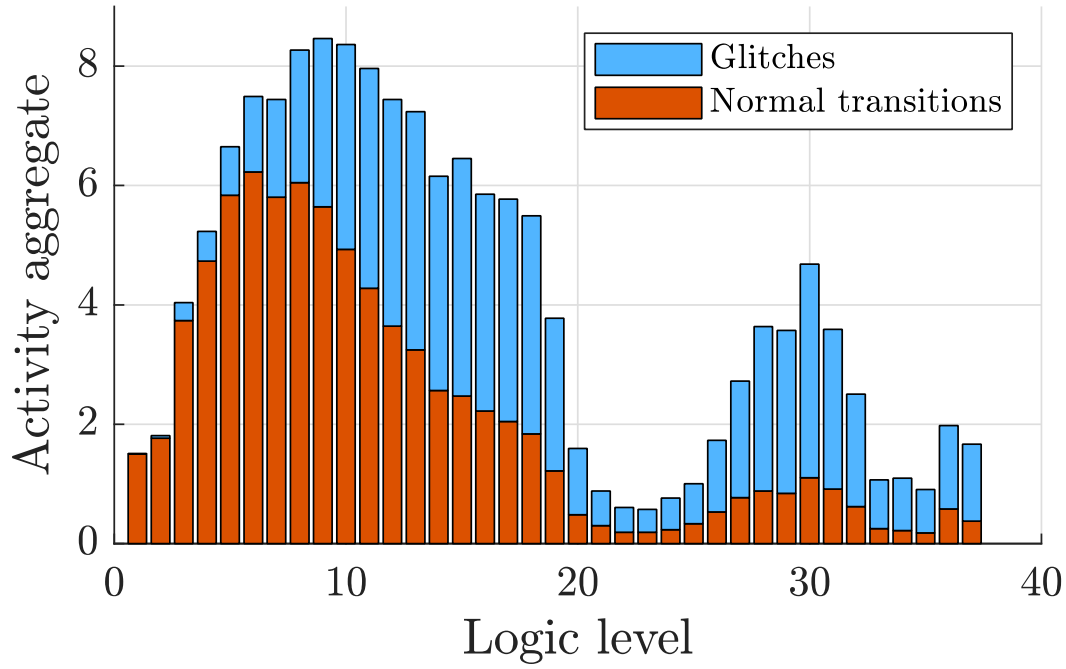


Activity Profile of Integer vs Floating-Point Multiplier

24b fixed-point multiplier



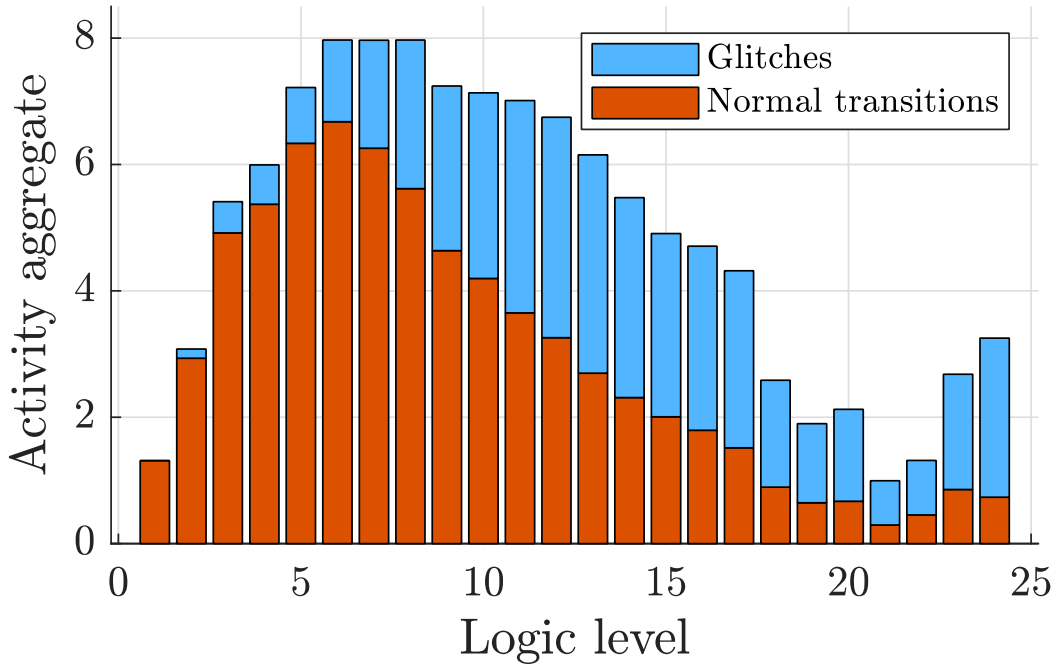
32b floating-point multiplier



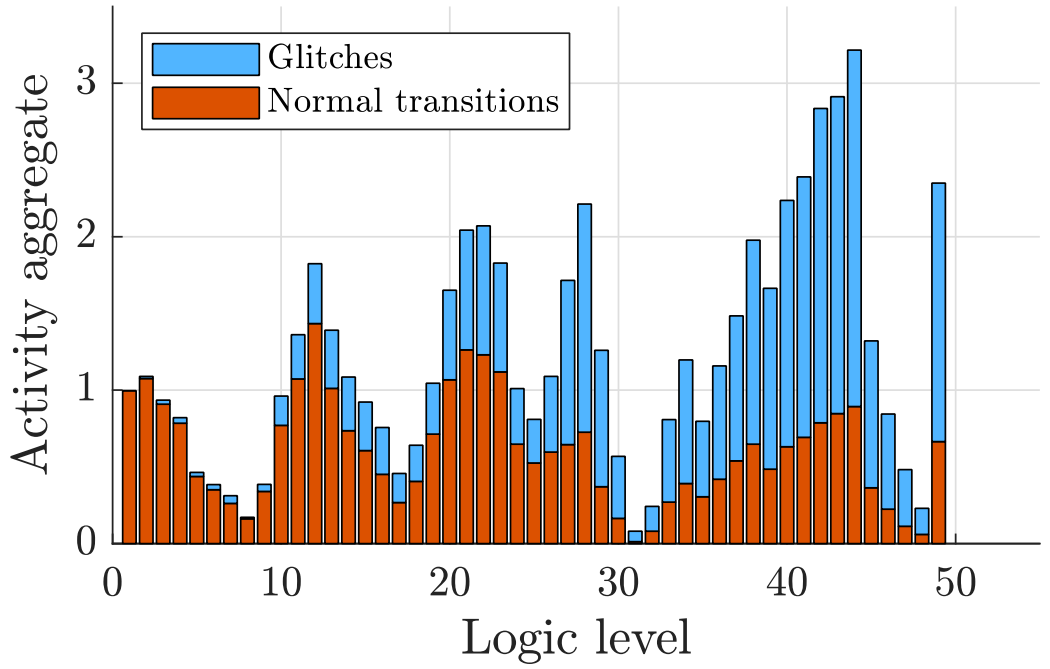
Glitches appear to be a problem
in floating-point multipliers too.

Activity Profile of Integer Multiplier vs Floating-Point Adder

24b fixed-point multiplier



32b floating-point adder



In floating-point arithmetic,
glitches appear to be a problem in adders too.

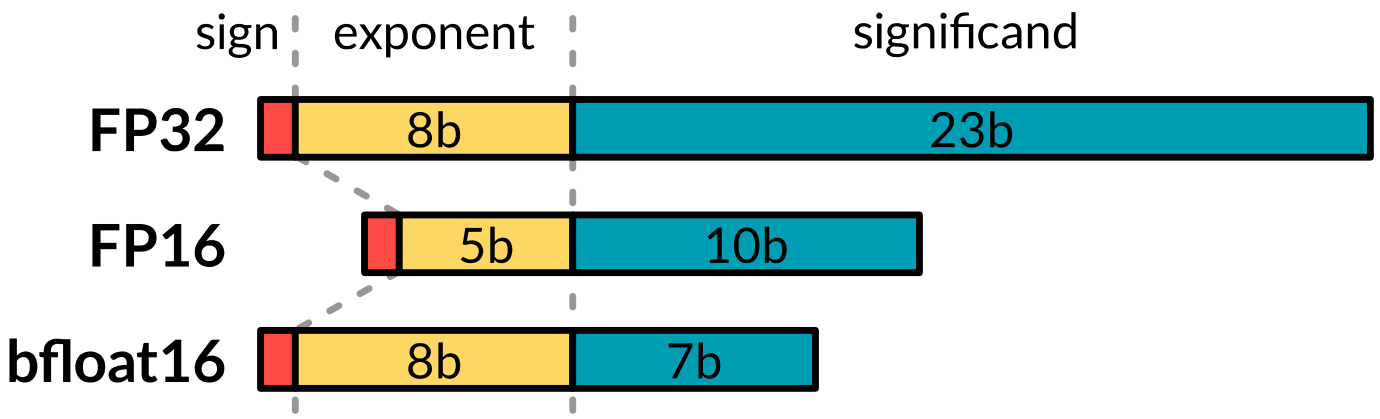
This Study

- There exists one practical method to suppress glitches: insert registers.
- But adding registers means we add capacitance which increases energy dissipation.
- ***When does it make sense to pipeline floating-point arithmetic to save energy?***

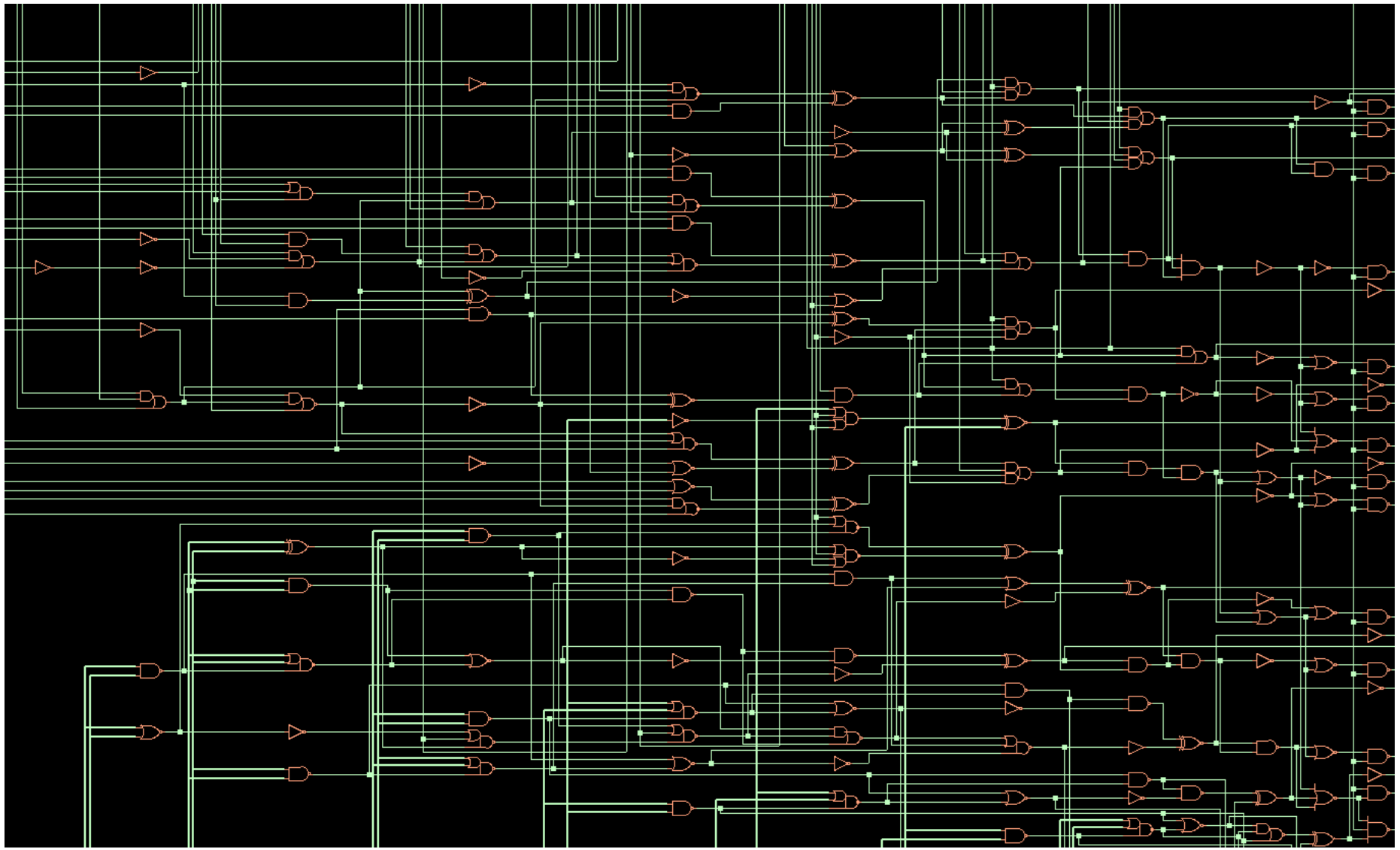
Method

- There exists one practical method to suppress glitches: insert registers.
- But adding registers means we add capacitance which increases energy dissipation.
- ***When does it make sense to pipeline floating-point arithmetic to save energy?***

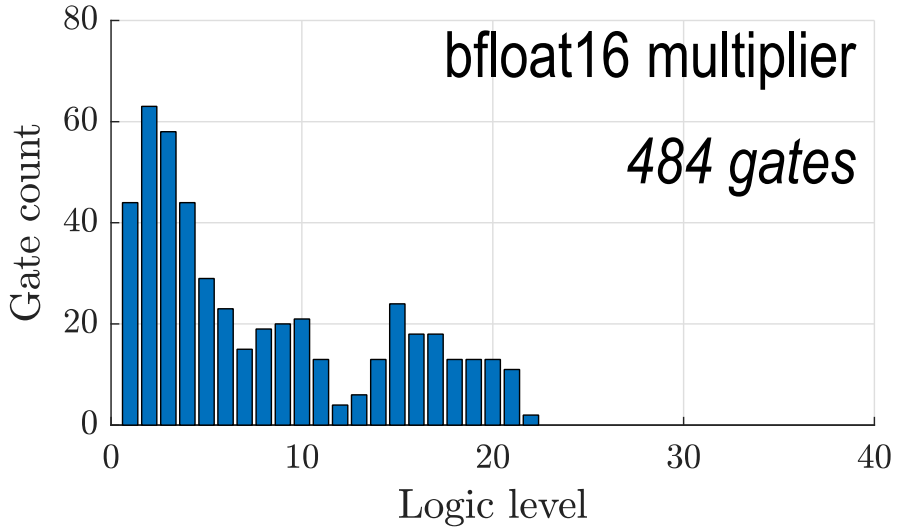
1) FloPoCo Nfloat HDL.
2) Formats selected to capture impact of different field widths.
3) ASIC design flow.
4) ASAP7 predictive 7nm FinFET tech.
5) Power simulation of synthesized netlists.

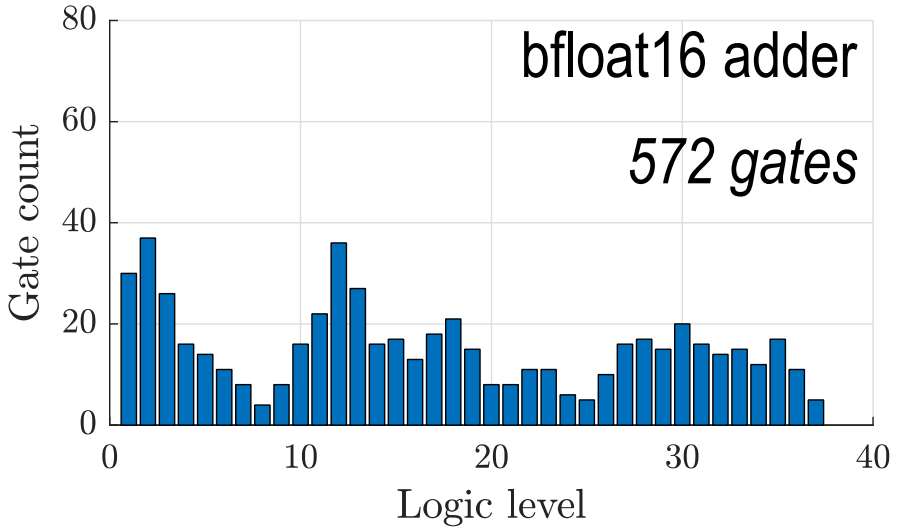


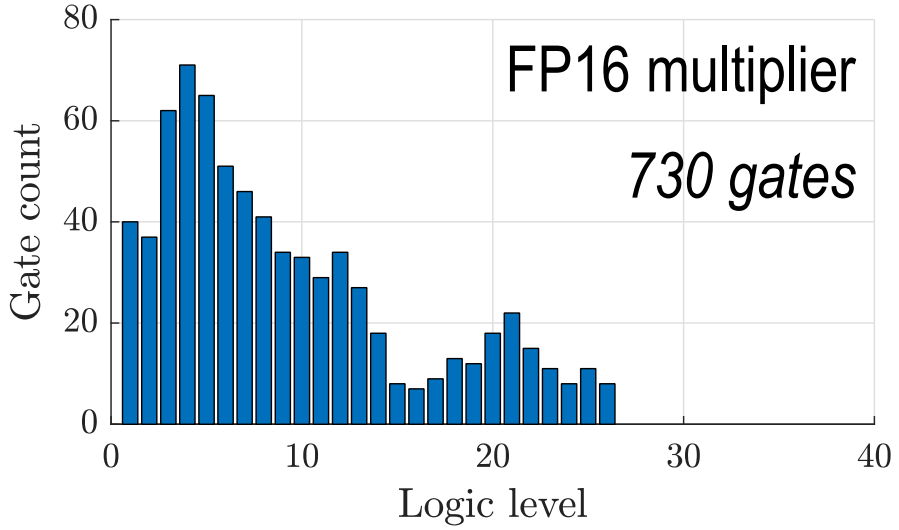
How Can We Visualize Key Properties of Gate Netlists?

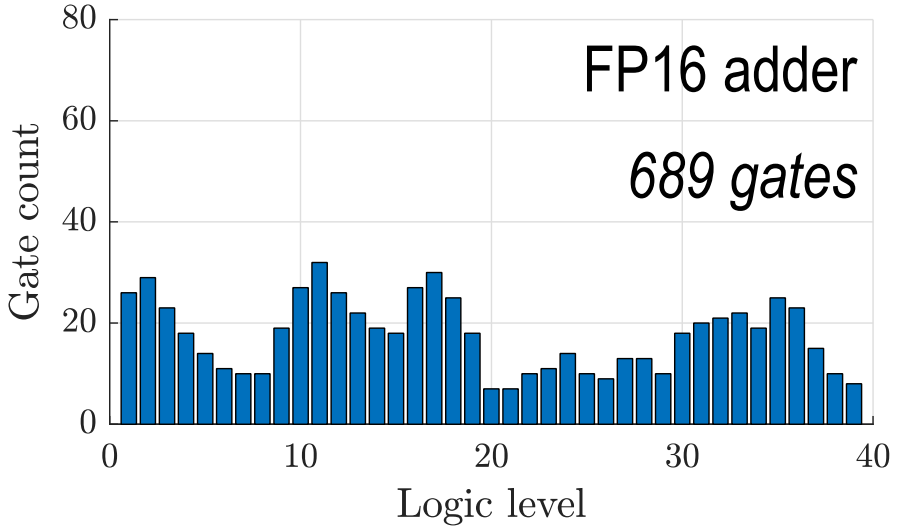


Logic Levels of Different Arithmetic Circuits



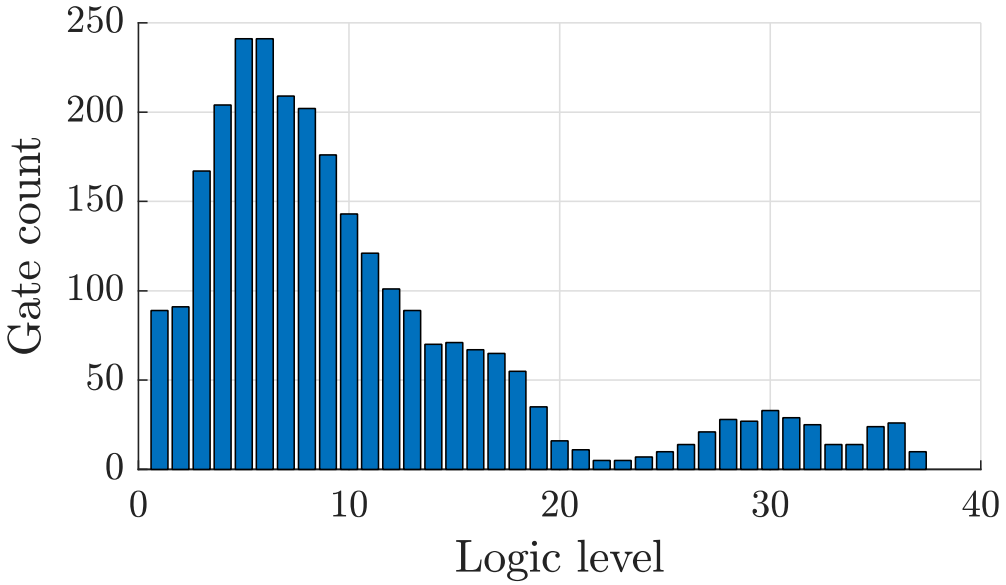


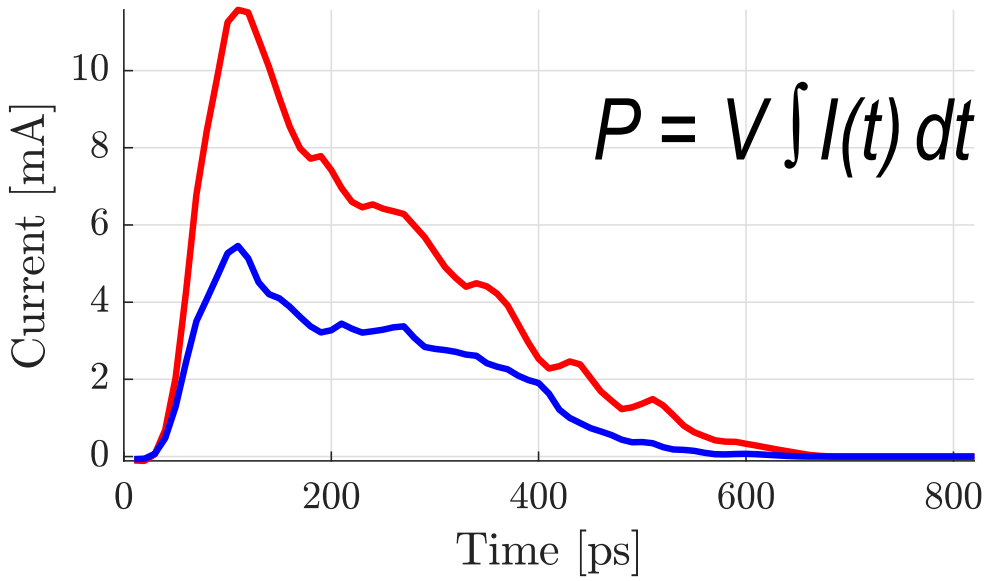




Multiplier circuits are “short and fat”, adders “long and thin”.

Current per Cycle Depends on Logic Gate Distribution …





… as well as gate types, gate sizes and choice of data for input operands.

Input Vector Generation for Power Analysis

$$P_{sw} = f\, V_{DD}^2\, \Sigma(C_i\, \alpha_i)$$

$$P_{clk} = f\, V_{DD}^2\, C_{clk},\ where\ \alpha = 1$$

- For pipelined designs, we must establish a realistic proportion: P_{sw} vs P_{clk}.
- Thus, data for operands must have realistic switching statistics ($\alpha = 0.1$) .

1. Generate data vectors for input operands in all three formats.
2. Run power analysis using logic simulation of synthesized netlists.

Example of data vectors *bfloat16*:

Sign	Exponent	Significand
0	11000110	0110001
0	11000110	1110101
1	11110111	1111101
1	01110110	1111101
1	01010111	0011101
0	01100111	0011100
1	01100101	0011110
1	11100101	0011111
1	11011010	0111011
1	11011111	0011100
0	01011111	0111111

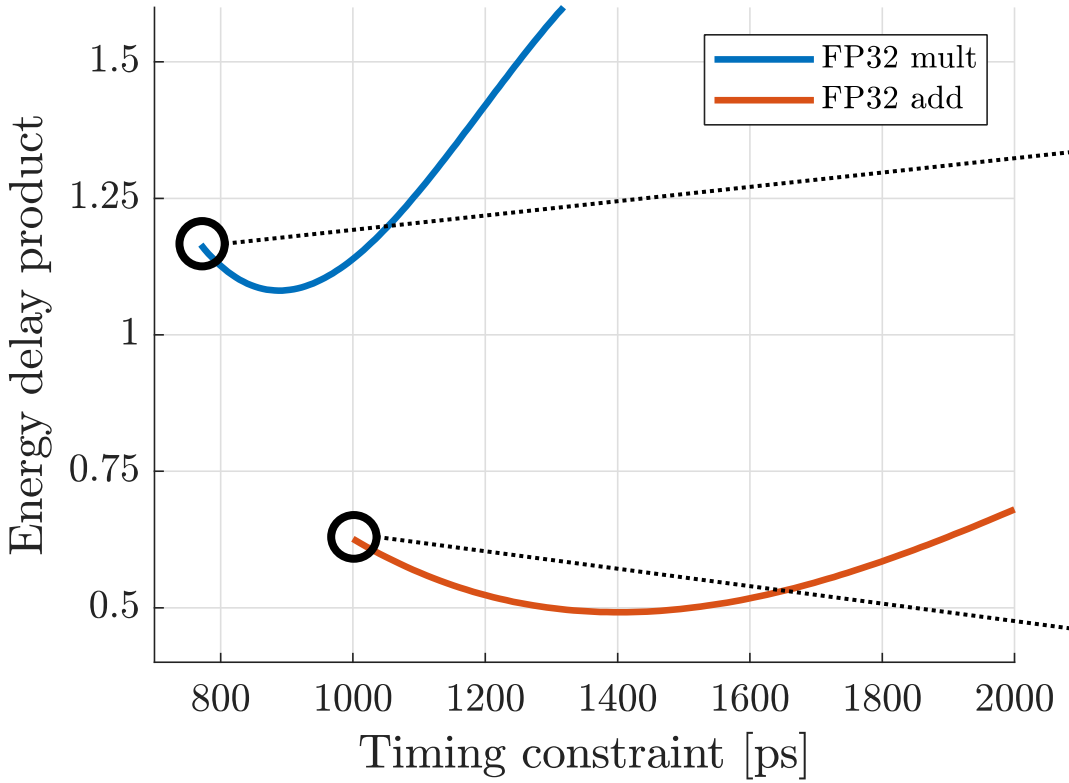
Input Vector Generation + Output Reference for Simulation

- Output reference data generated for netlist logic verification.

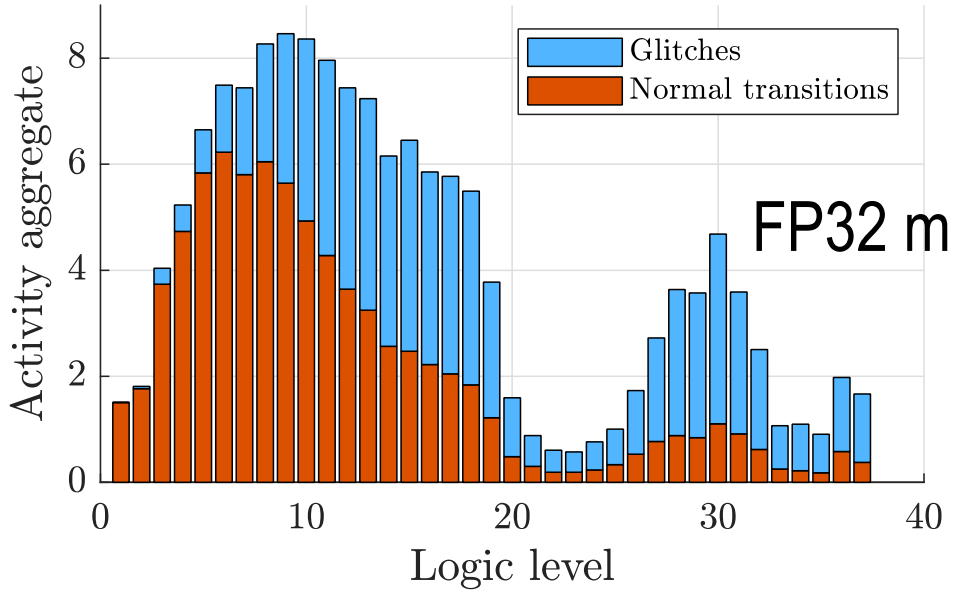
Example of data vectors *bfloat16*:

Sign	Exponent	Significand	Output reference
0	11000110	0110001	0110001100110011
0	11000110	1110101	0110001101110101
1	11110111	1111101	1111101111111101
1	01110110	1111101	1011101101111101
1	01010111	0011101	1010101110011101
0	01100111	0011100	0011001110011100
1	01100101	0011110	1011001010011110
1	11100101	0011111	1111001010011111
1	11011010	0111011	1110110100111011
1	11011111	0011100	1110111110011100
0	01011111	0111111	1011111101100110

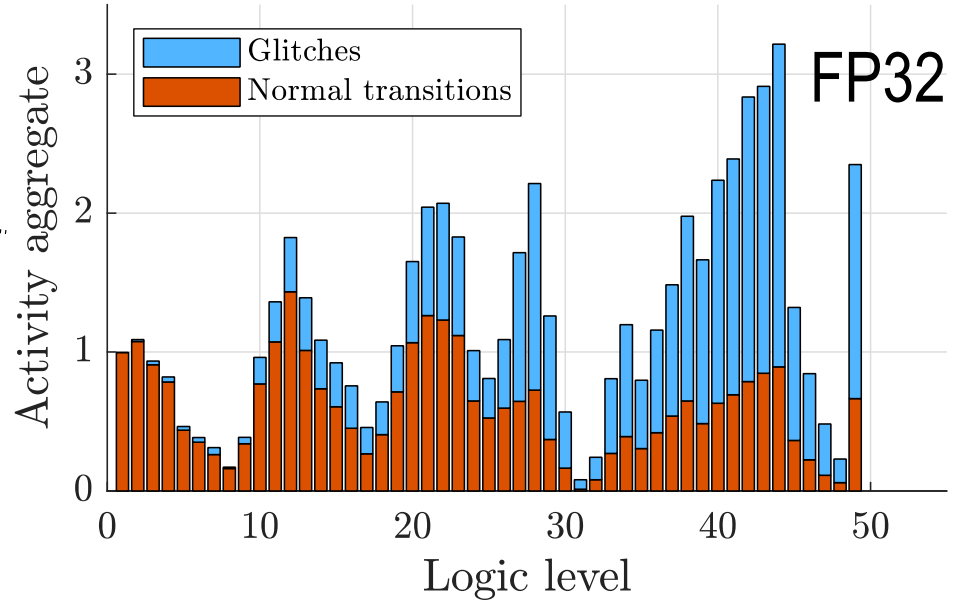
Energy-Delay Product and Activity Profile



$E = P / f$ $EDP = E \cdot TC$

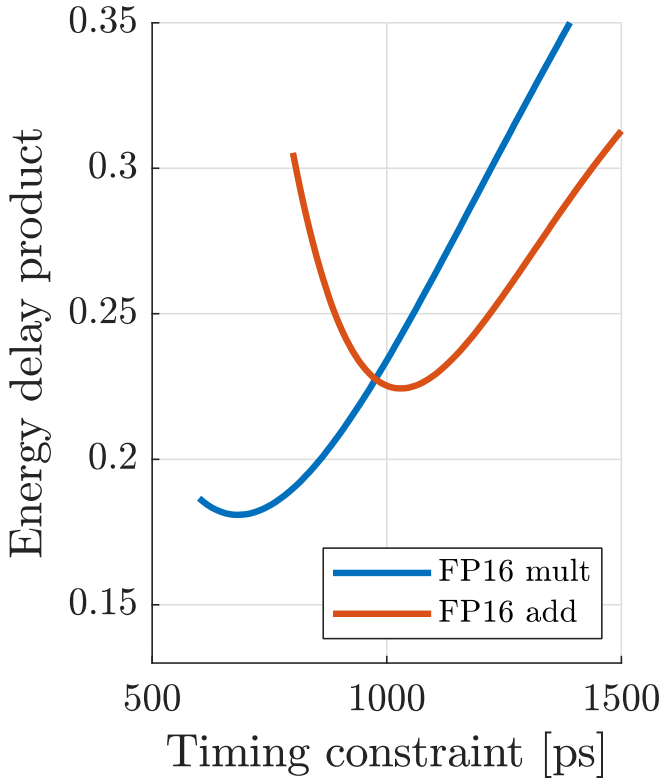


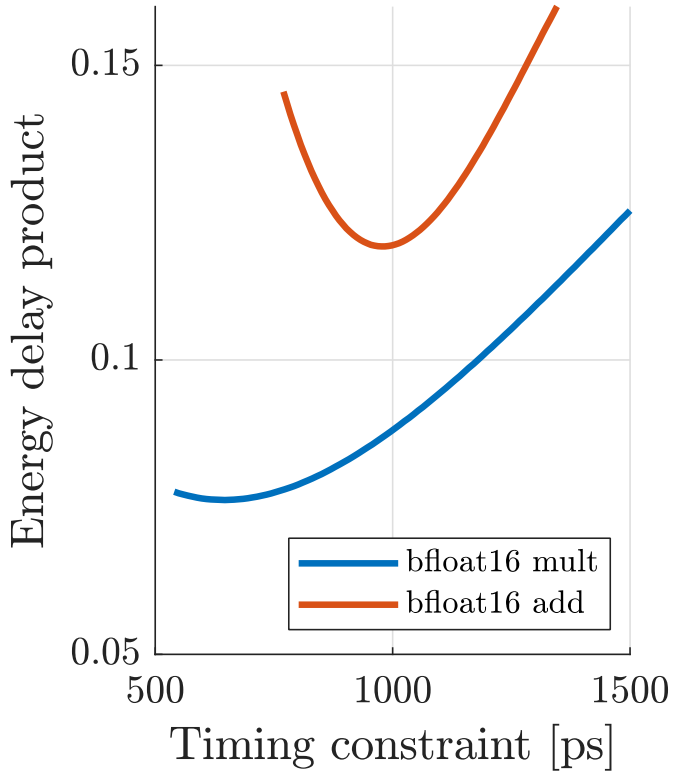
FP32 multiplier



FP32 adder

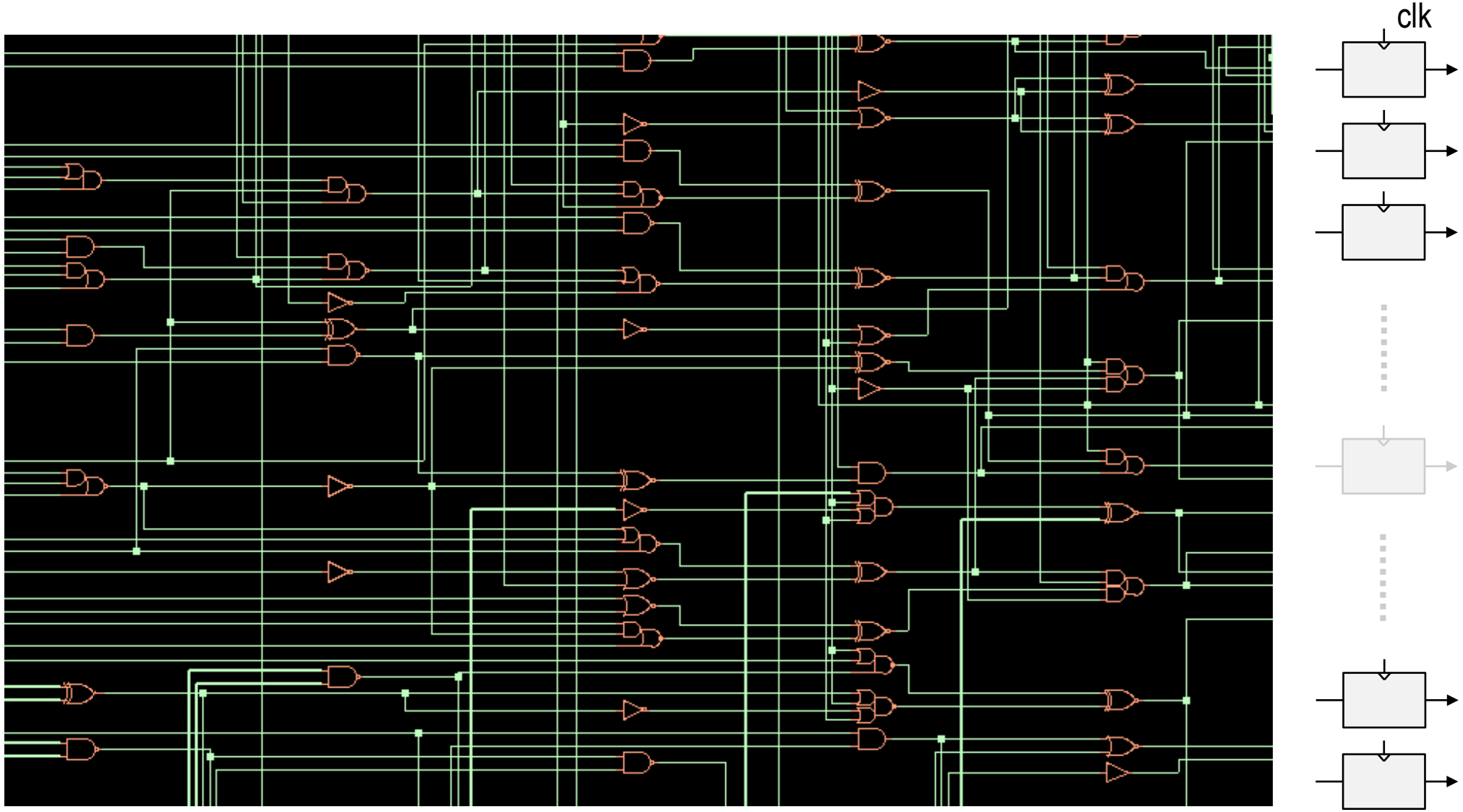
Energy-Delay Product of Arithmetic for Short Formats



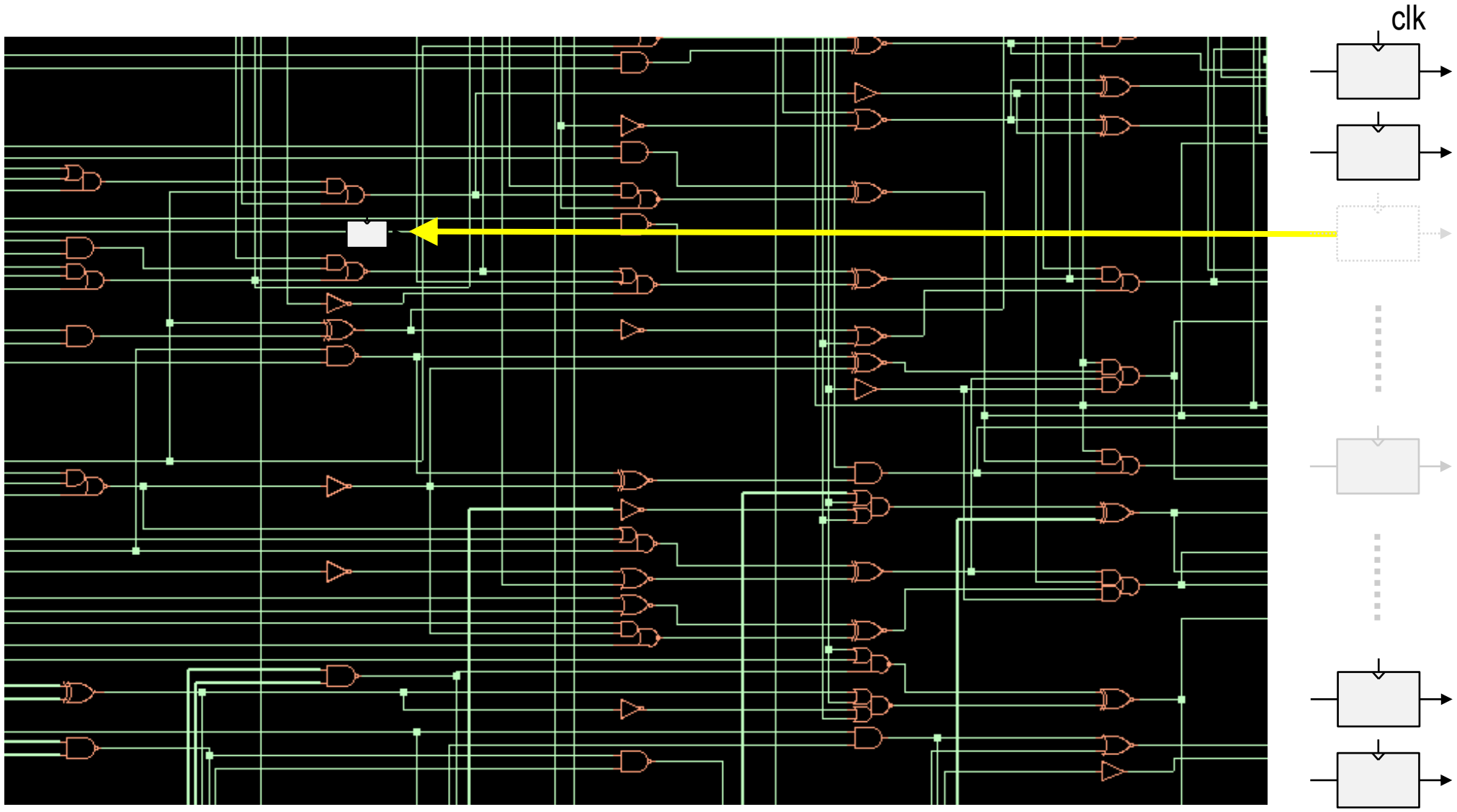


As formats get shorter, multiplier energy decreases faster than adder energy.

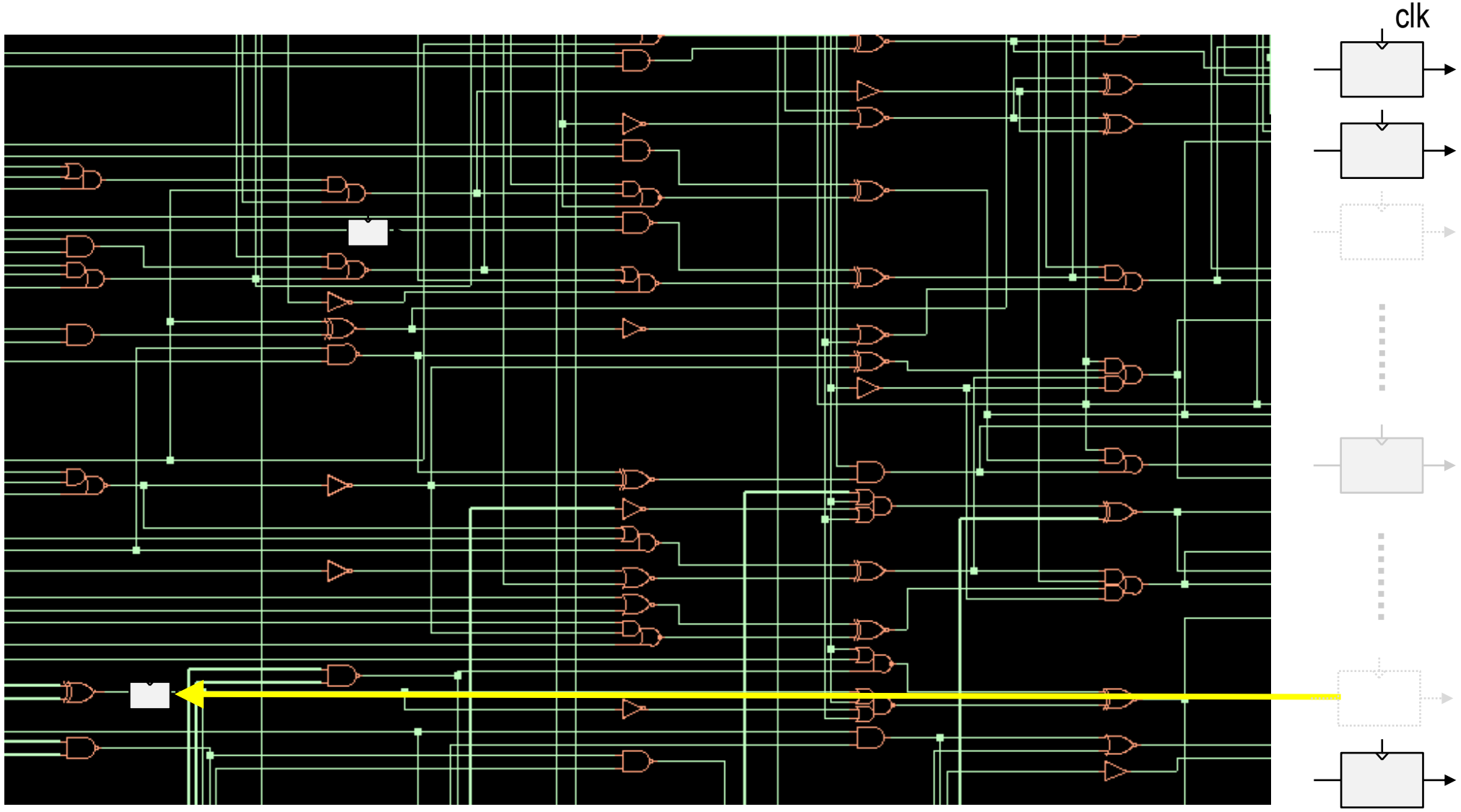
Pipelining/Retiming: Add Registers



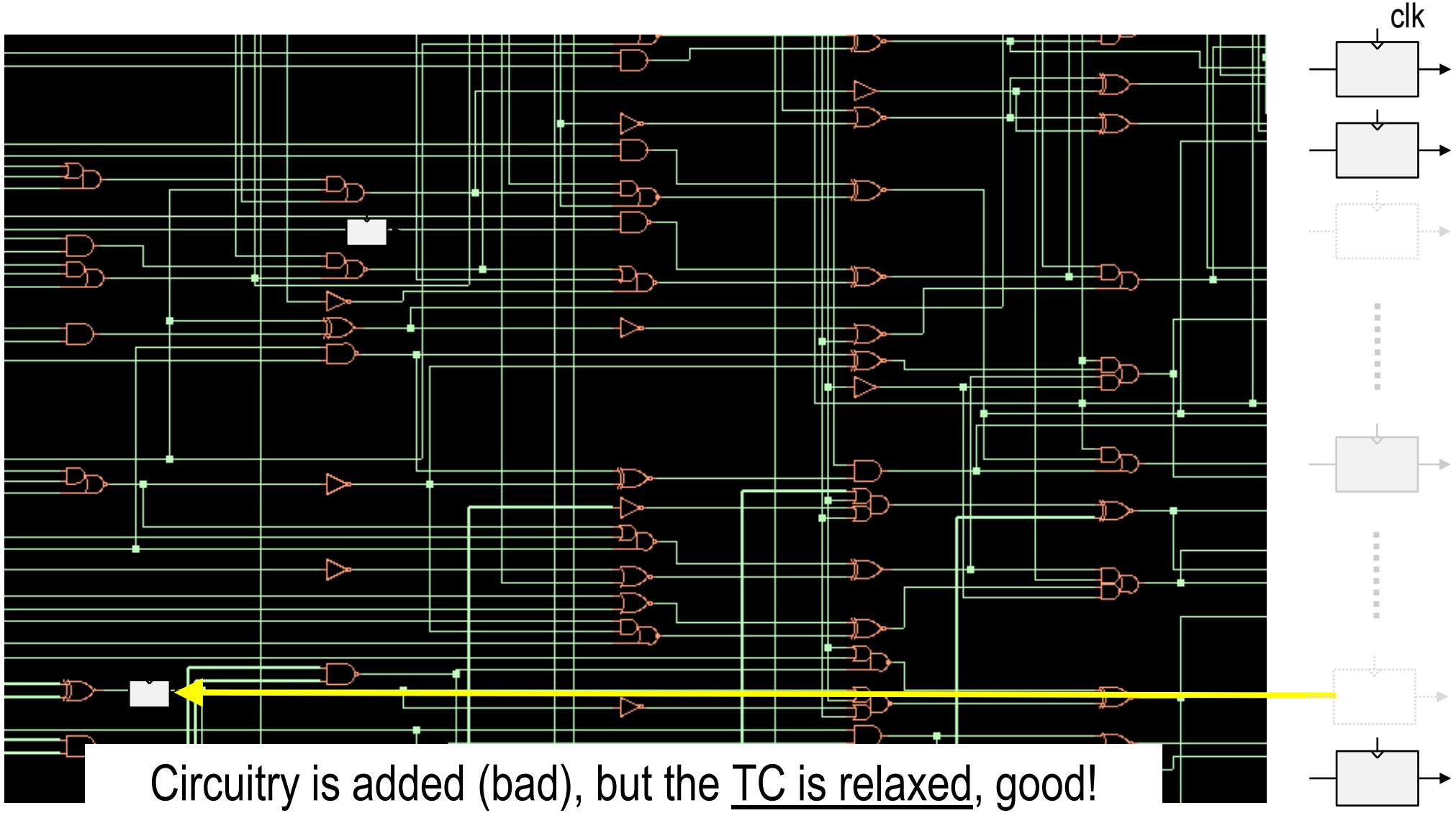
Pipelining/Retiming: Move Registers 1(2)



Pipelining/Retiming: Move Registers 2(2)



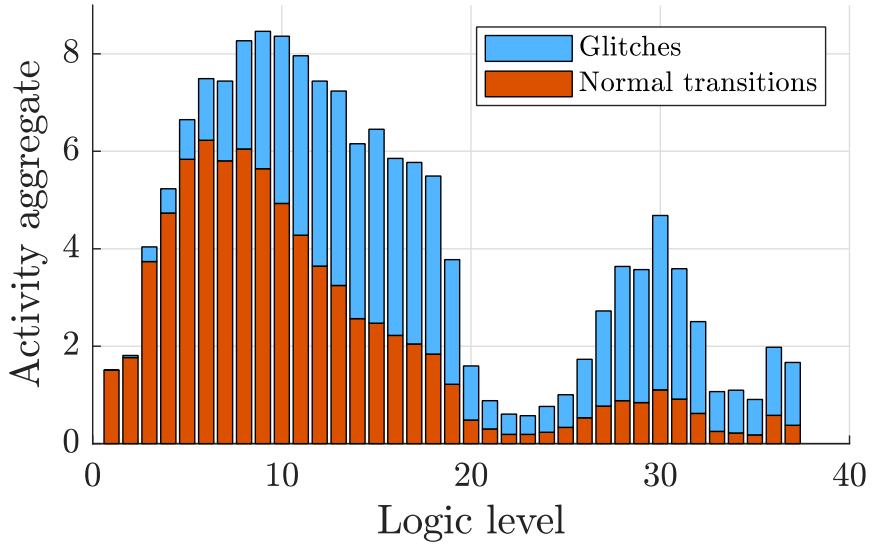
Pipelining/Retiming: Move Registers 2(2)



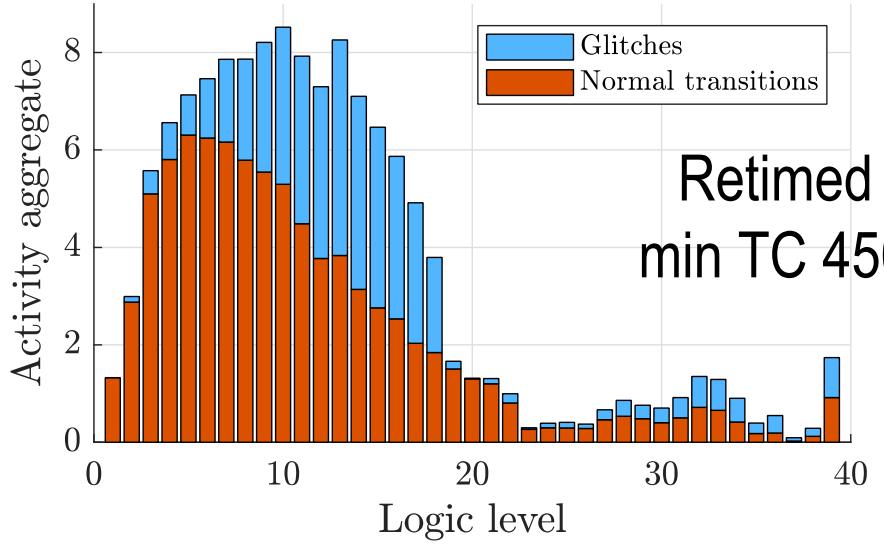
Circuitry is added (bad), but the TC is relaxed, good!

Effect of Pipelining and Timing Constraint (TC): FP32 Mult

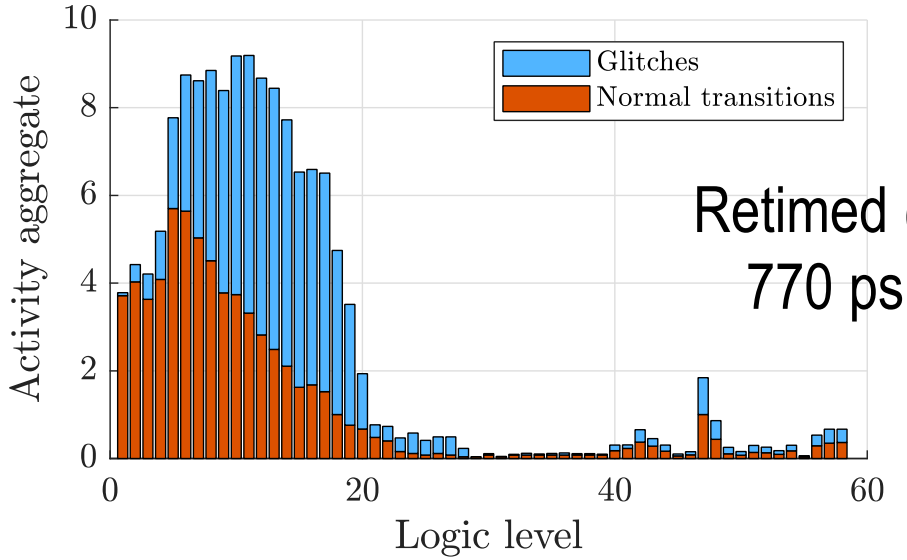
Combinational @ min TC 770 ps



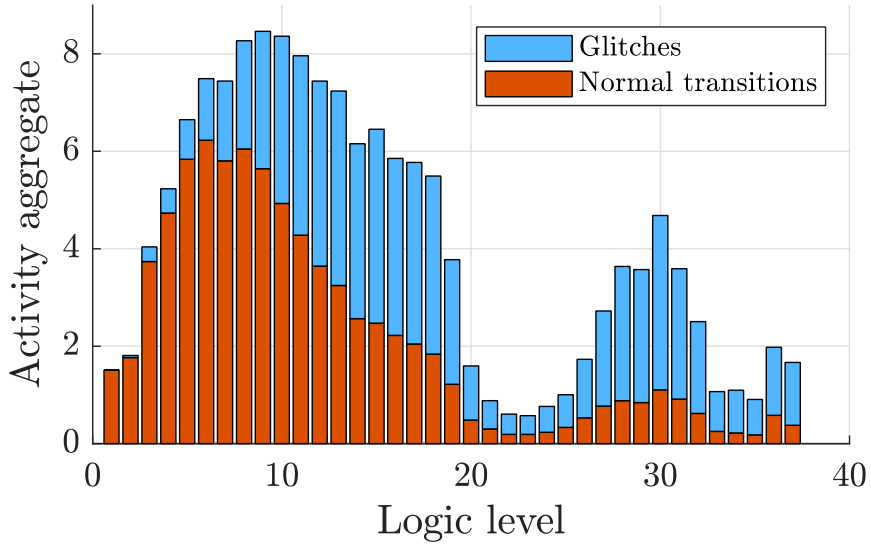
Retimed @ min TC 450 ps

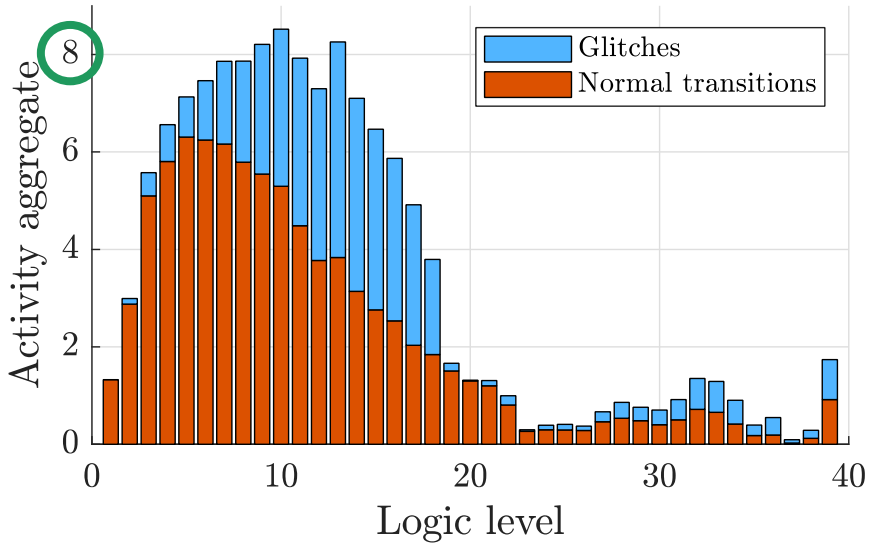


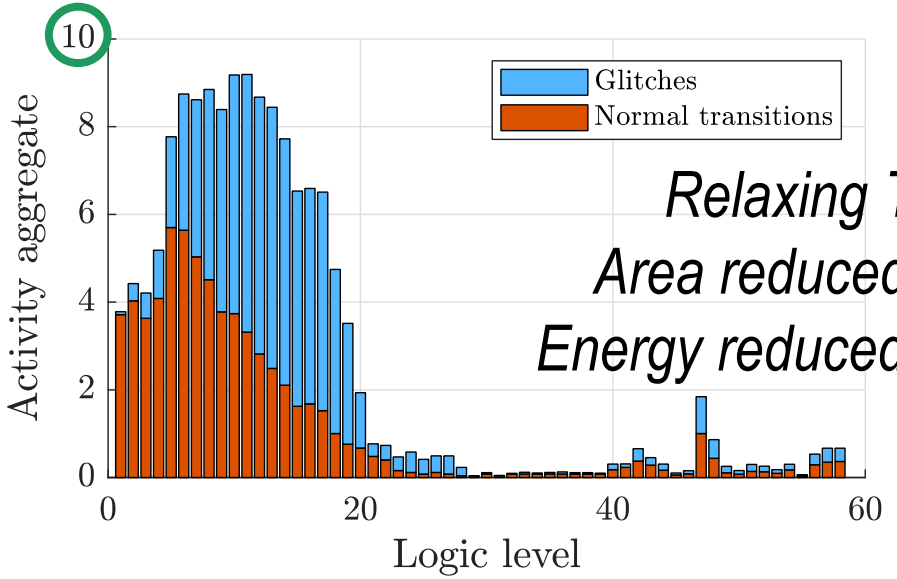
Retimed @ 770 ps



Effect of Pipelining and Timing Constraints: FP32 Mult



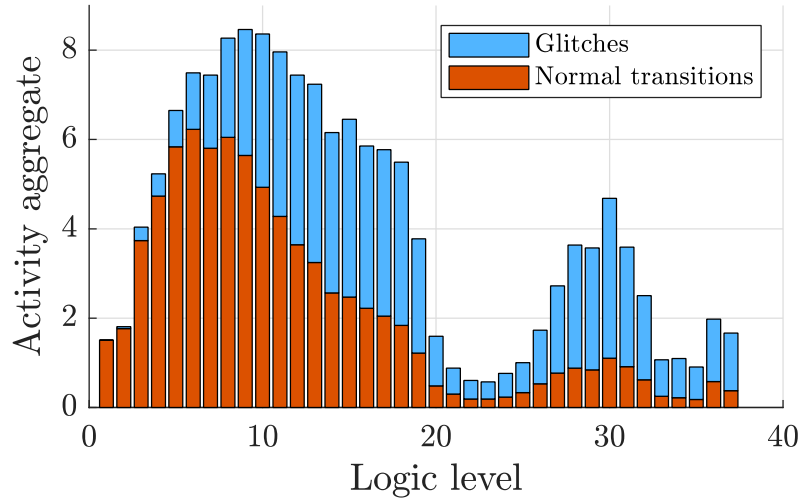




Relaxing TC:
Area reduced by ¼
Energy reduced by 10%

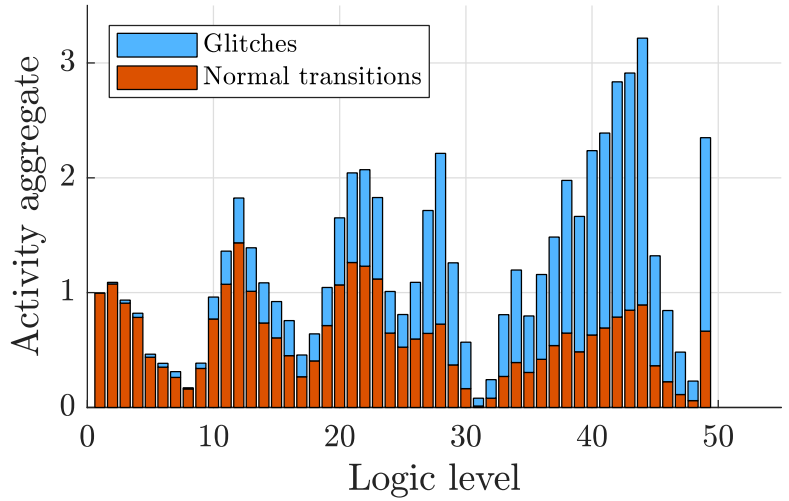
Pipelining Appears to Be Effective for Adders

Combinational FP32 multiplier



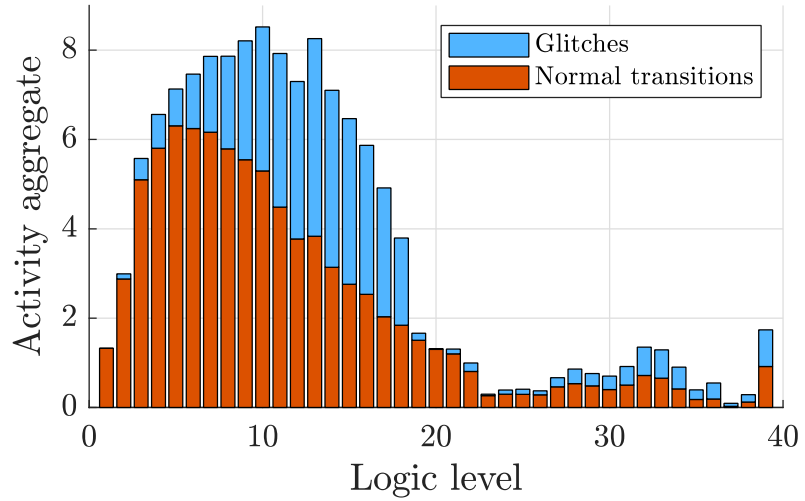
Min TC 770 ps

Combinational FP32 adder



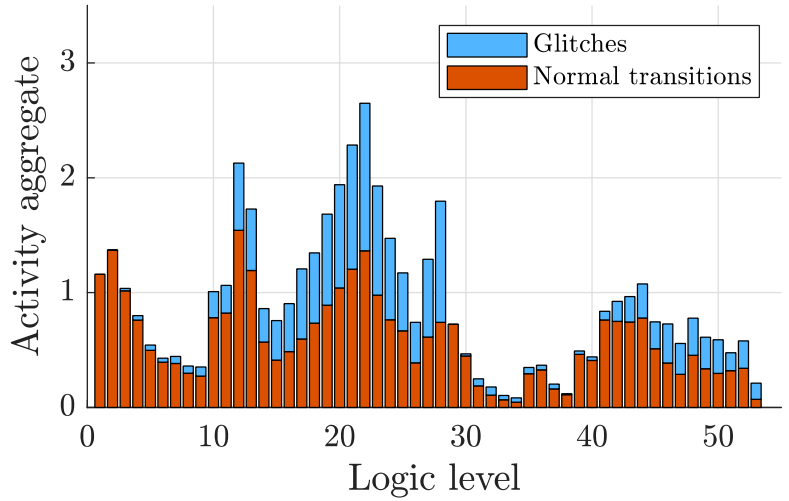
Min TC 1000 ps

Pipelined FP32 multiplier



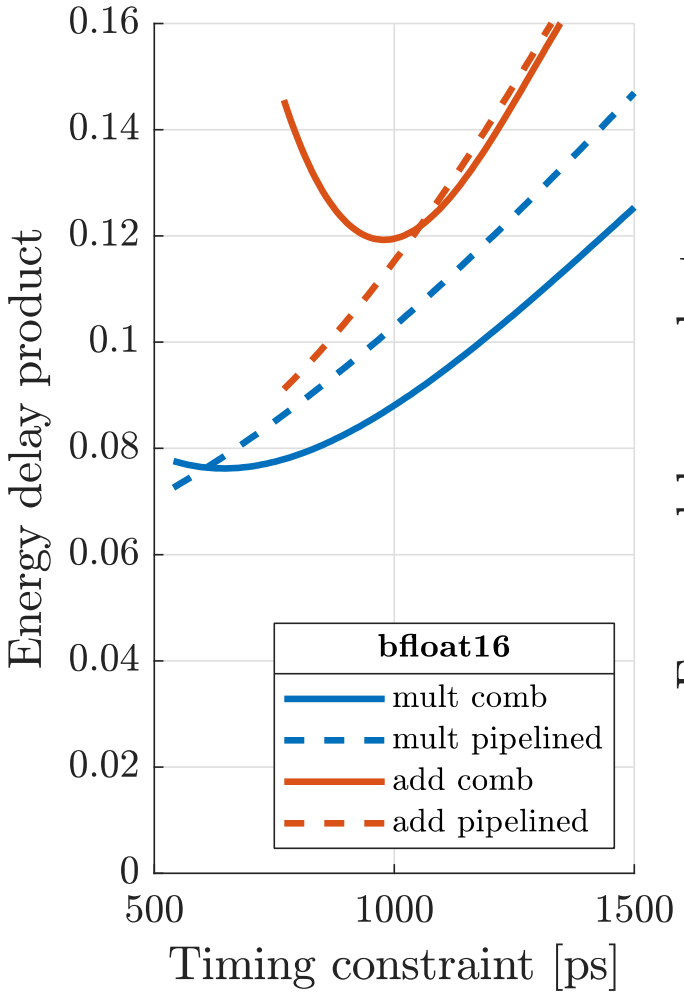
Min TC 450 ps

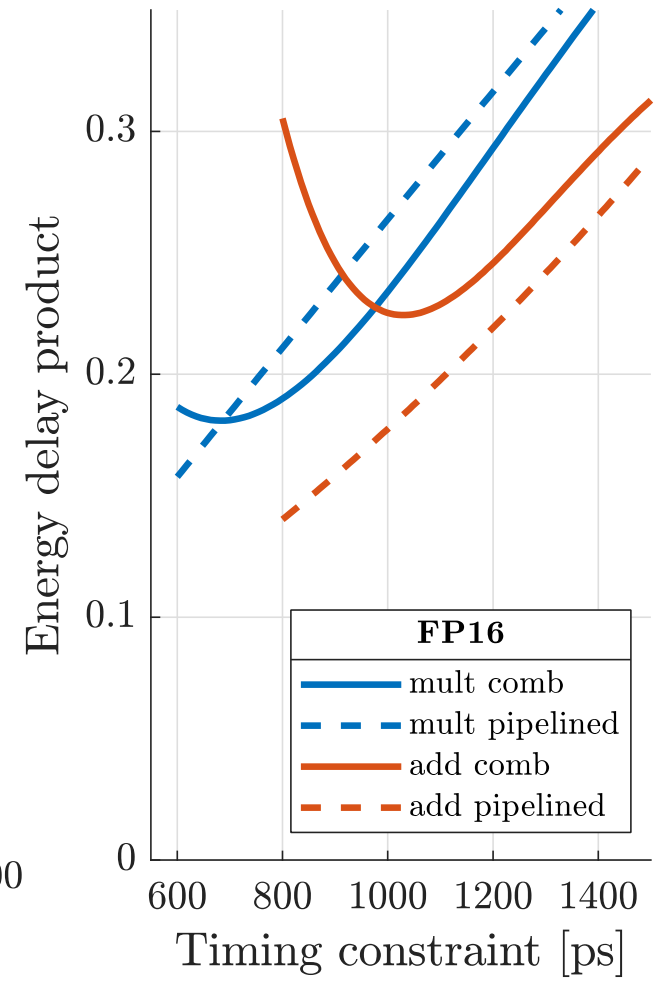
Pipelined FP32 adder

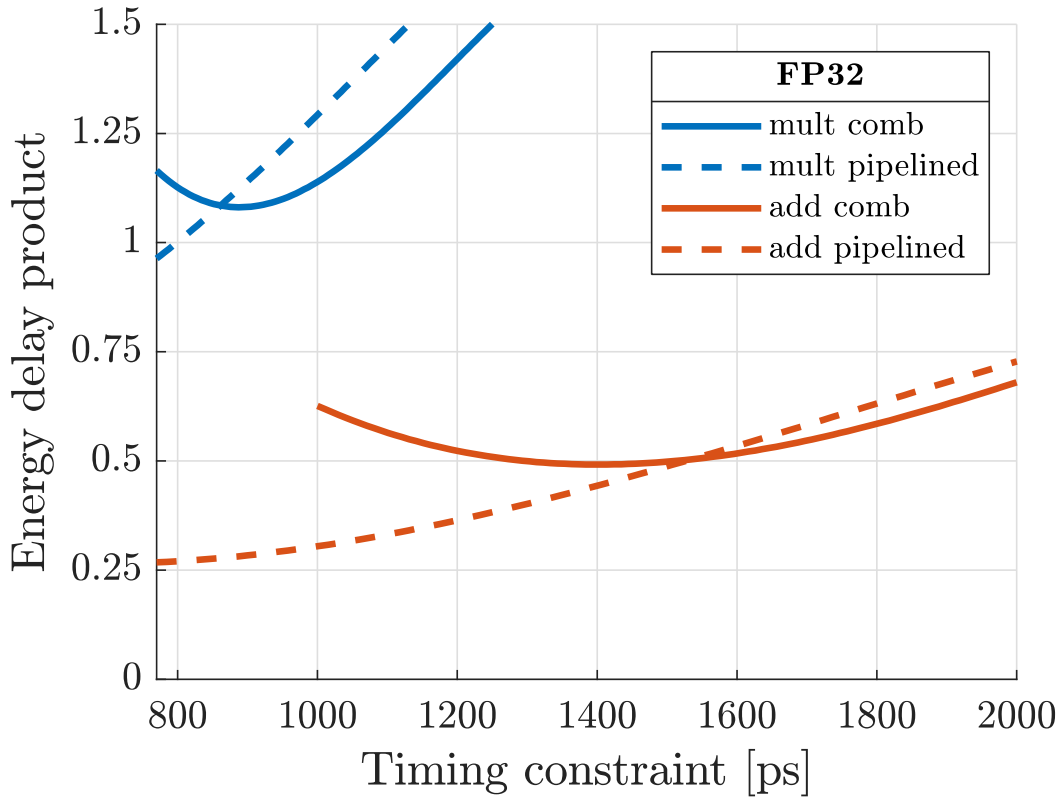


Min TC 570 ps

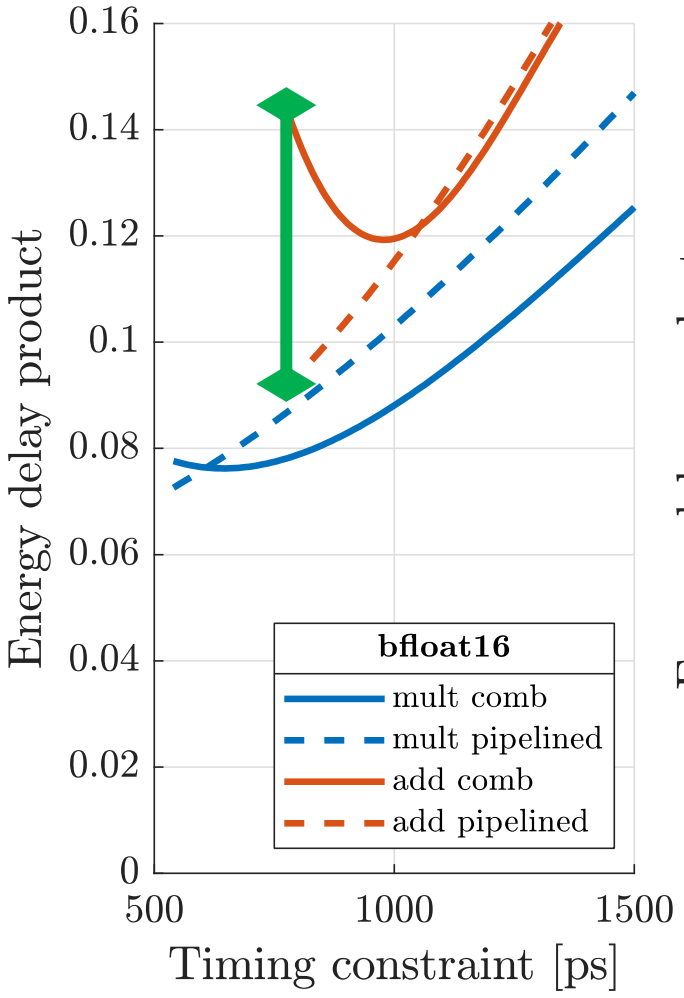
When Does It Make Sense to Pipeline (to Save Energy)?

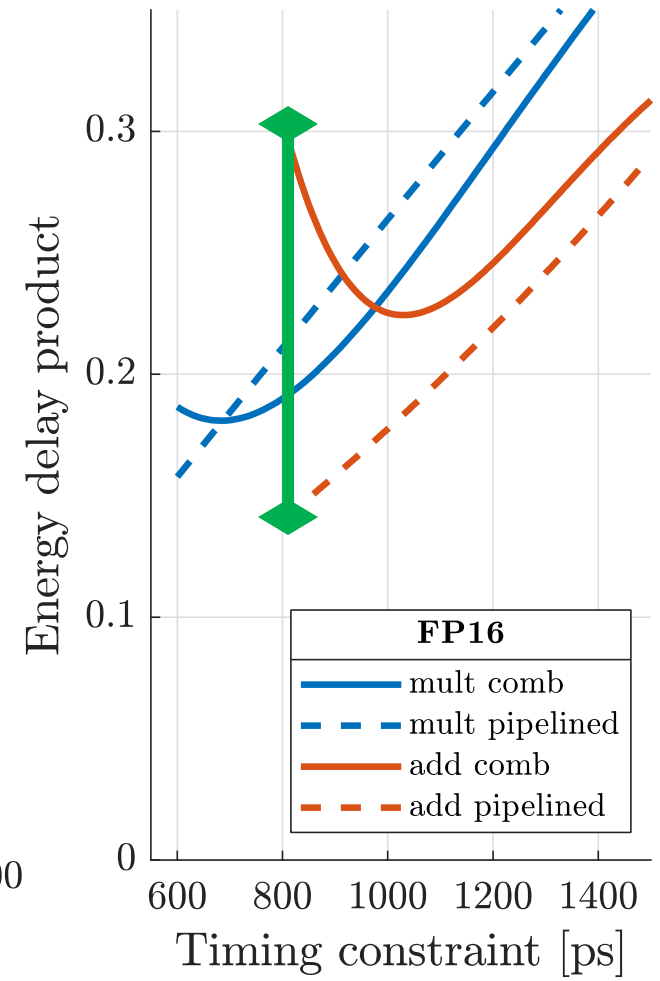


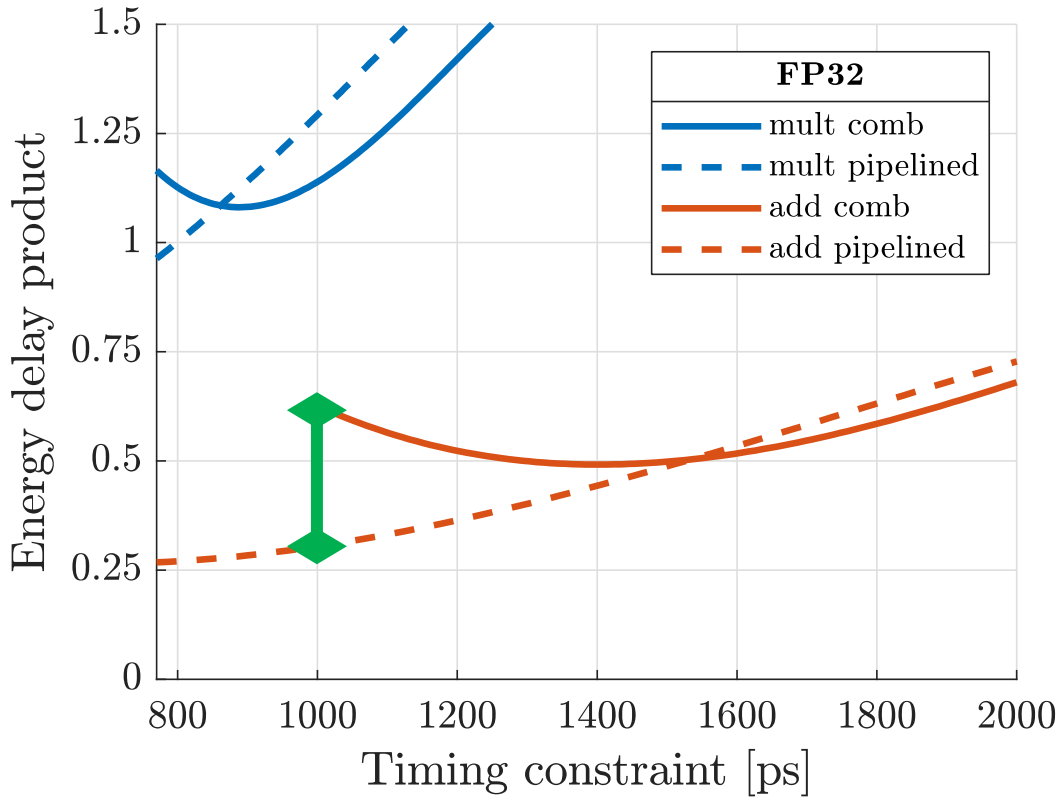




When Does It Make Sense to Pipeline (to Save Energy)?



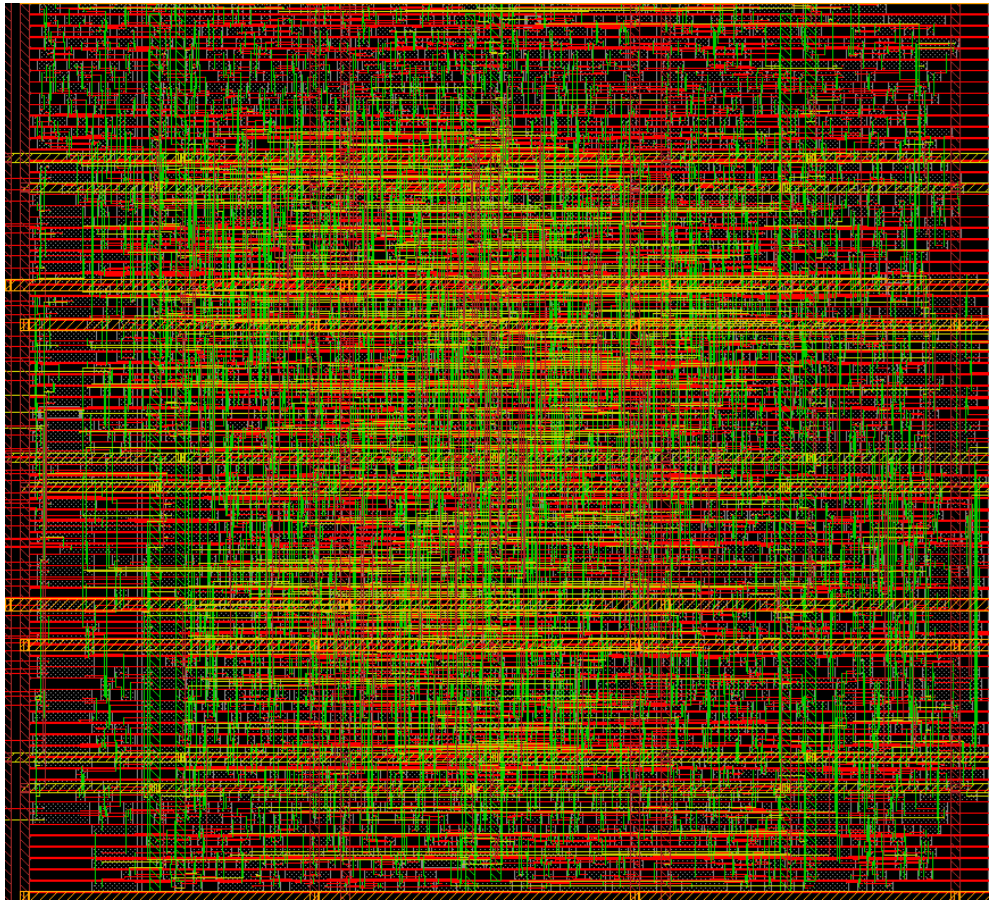


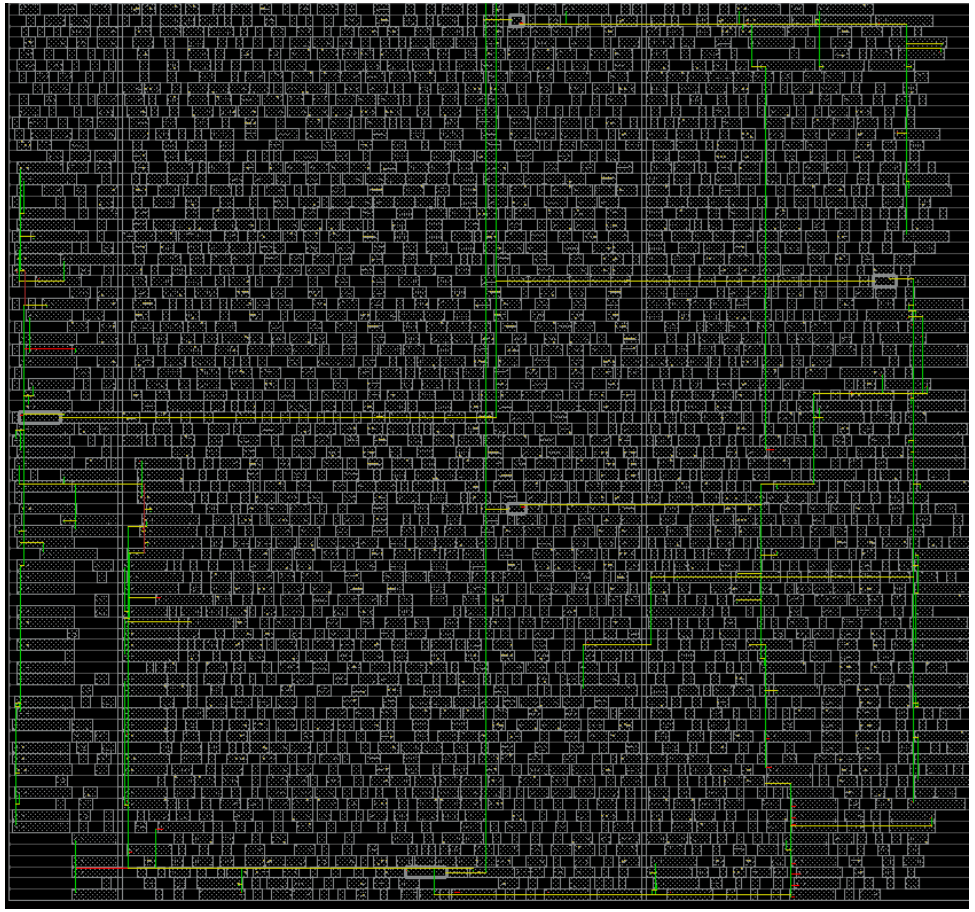


Adder energy reduction 38-54%.

Significant in terms of absolute energy.

PnR Clock-Tree Synthesis Adds <4% Energy





This Study

- Pipelining/register insertion can reduce energy by inhibiting propagation of glitches, which dissipate a significant portion of total energy.
- However, added registers, clock tree and clock buffers increase energy dissipation.
- ***When does it make sense to pipeline floating-point arithmetic to save energy?***

Conclusion

- Pipelining saves energy for both adders and multipliers at iso-delay,
  but the effect is the greatest for adders: 38-54% for Nfloat implementations.
- Main takeaways:
  1. Adders are known to have large logic depths, so they are intrinsically slow.
     Pipelining balances timing and reduces energy simultaneously.
  2. Shorter exponent and longer significand fields lead to
     greater adder energy savings (relevant in accumulators).
  3. As formats get shorter, adder energy begins to dominate.
     If the architecture permits, pipeline the adder even if it is fast enough.