Regular Expression

Operators

- ‘*’ repeat (unary)
- ‘|’ or (binary)
- ‘’ concat (binary)

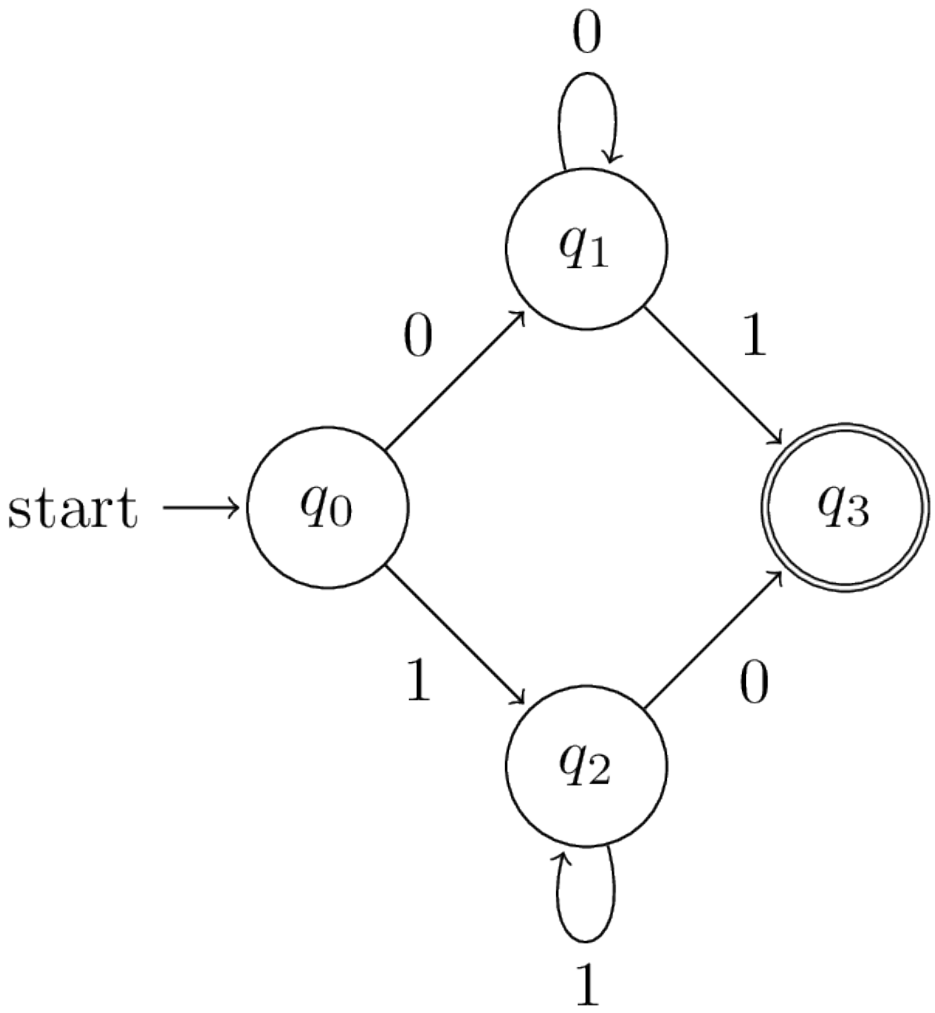
Example: ((a|b)*)(cd)

Regular Expression Matching

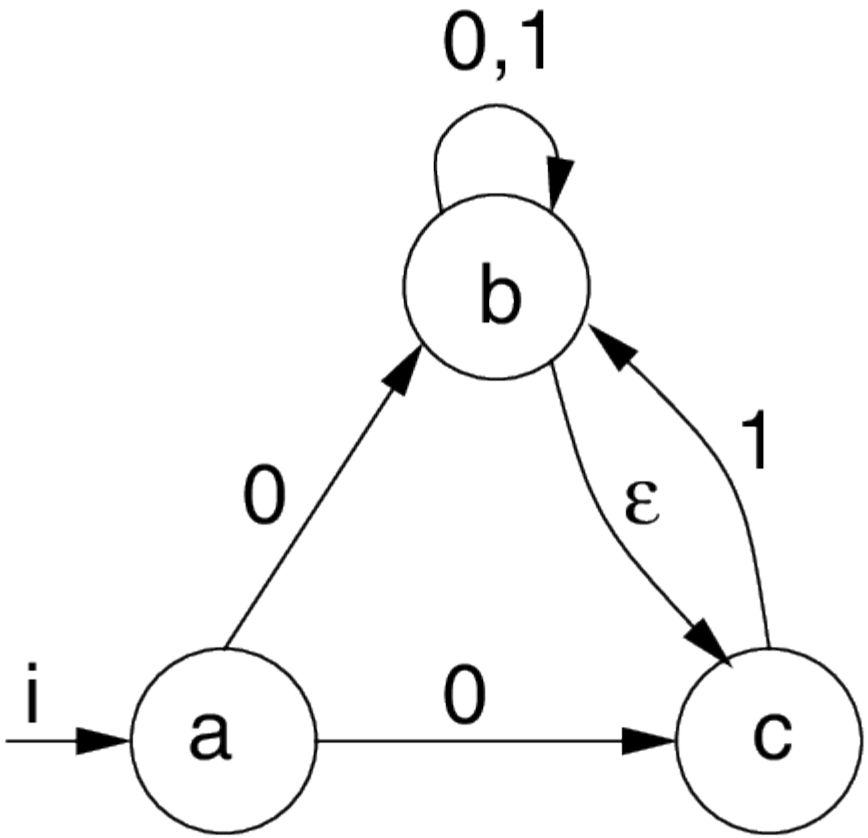
Regular Expression

- pattern for searching in data
- pattern ⇔ regular language
- can be detected by an finite automata
- can be deterministic or nondeterministic

DFA



NFA



Simulation on CPU

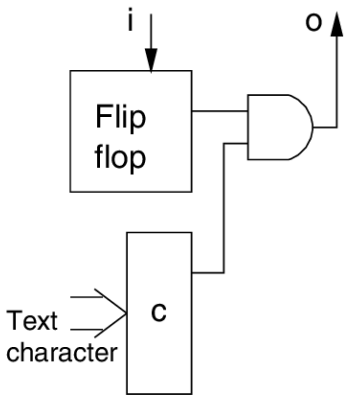
Create NFA from grammar	O(n)
Translate NFA into DFA	O(2^n)
Simulate DFA	O(1) per char

Simulation on CPU

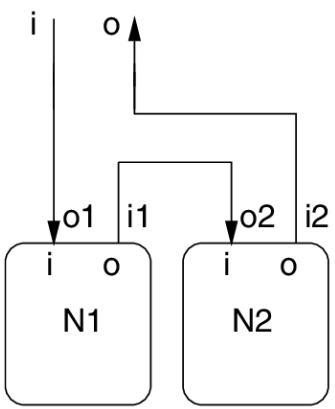
Create NFA from grammar	O(n)
Simulate NFA	O(n) per char

Simulation on FPGA

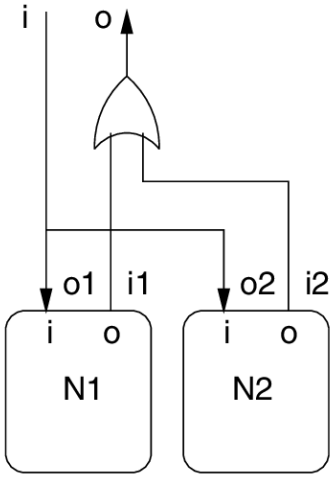
Create NFA from grammar	O(n)
Translate NFA into wiring	O(n)
Simulate NFA	O(1) per char



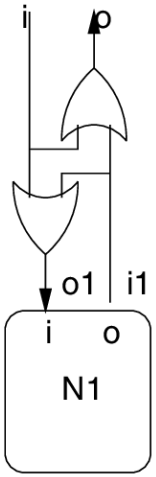
single character



$r_1 r_2$

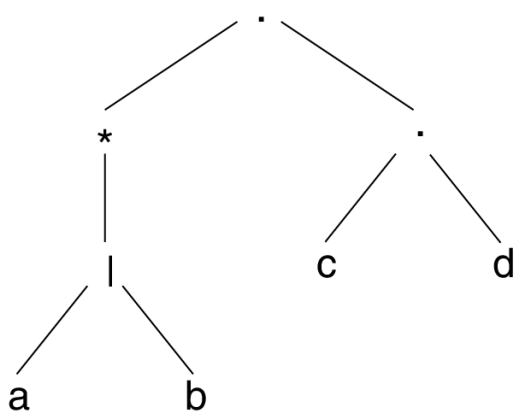


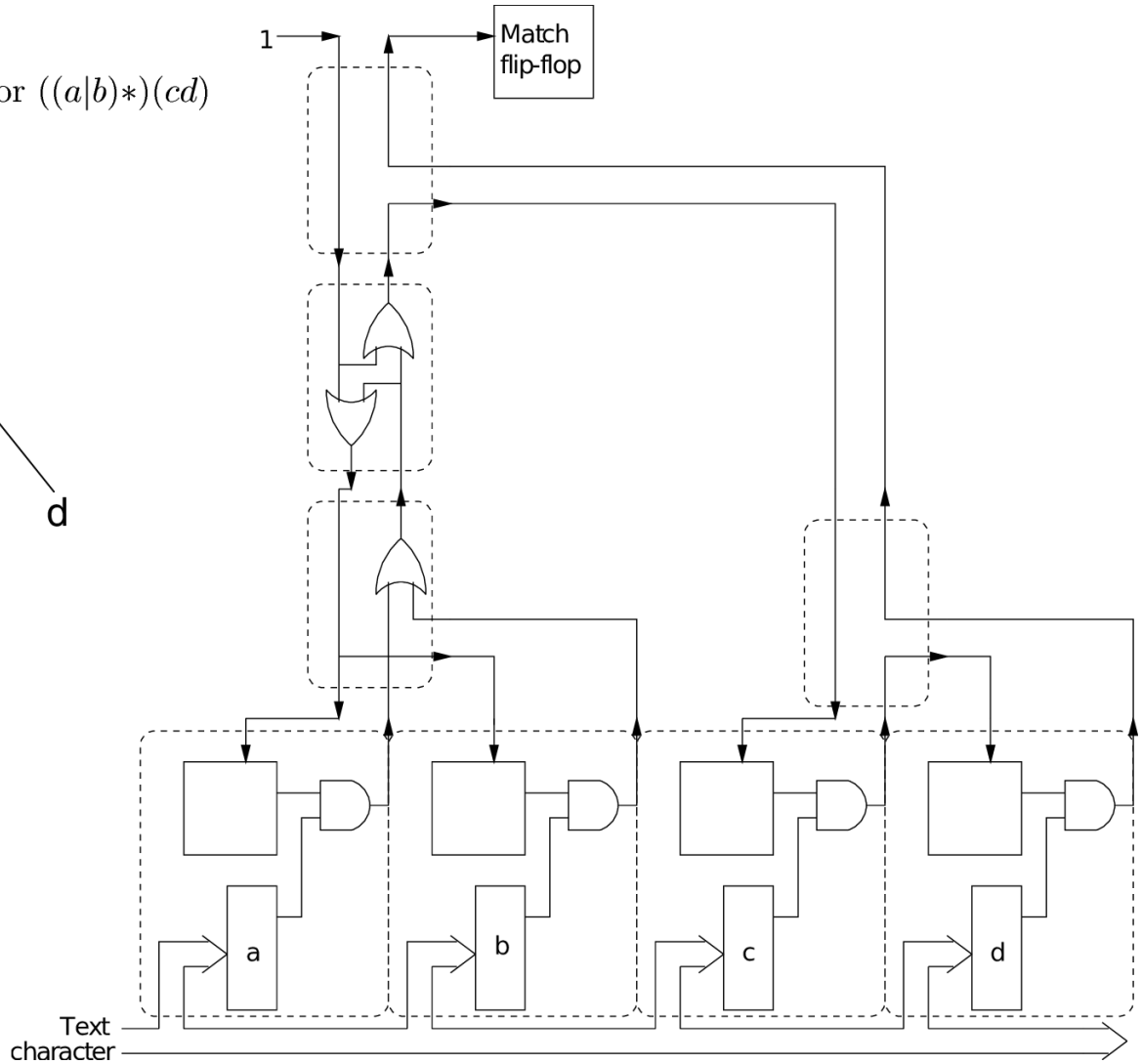
$r_1 | r_2$



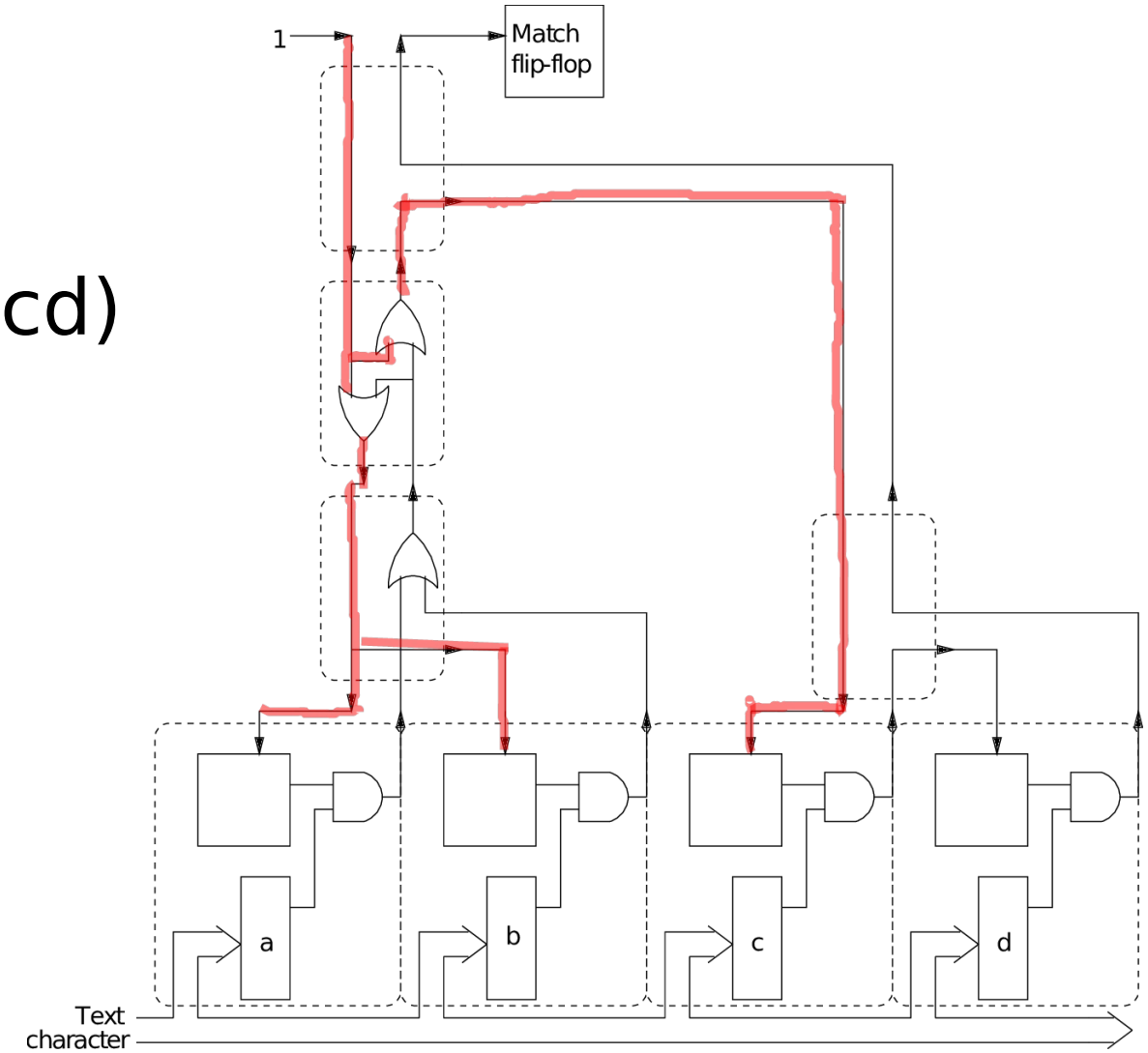
$r_1 *$

Syntax tree for $((a|b)*)(cd)$



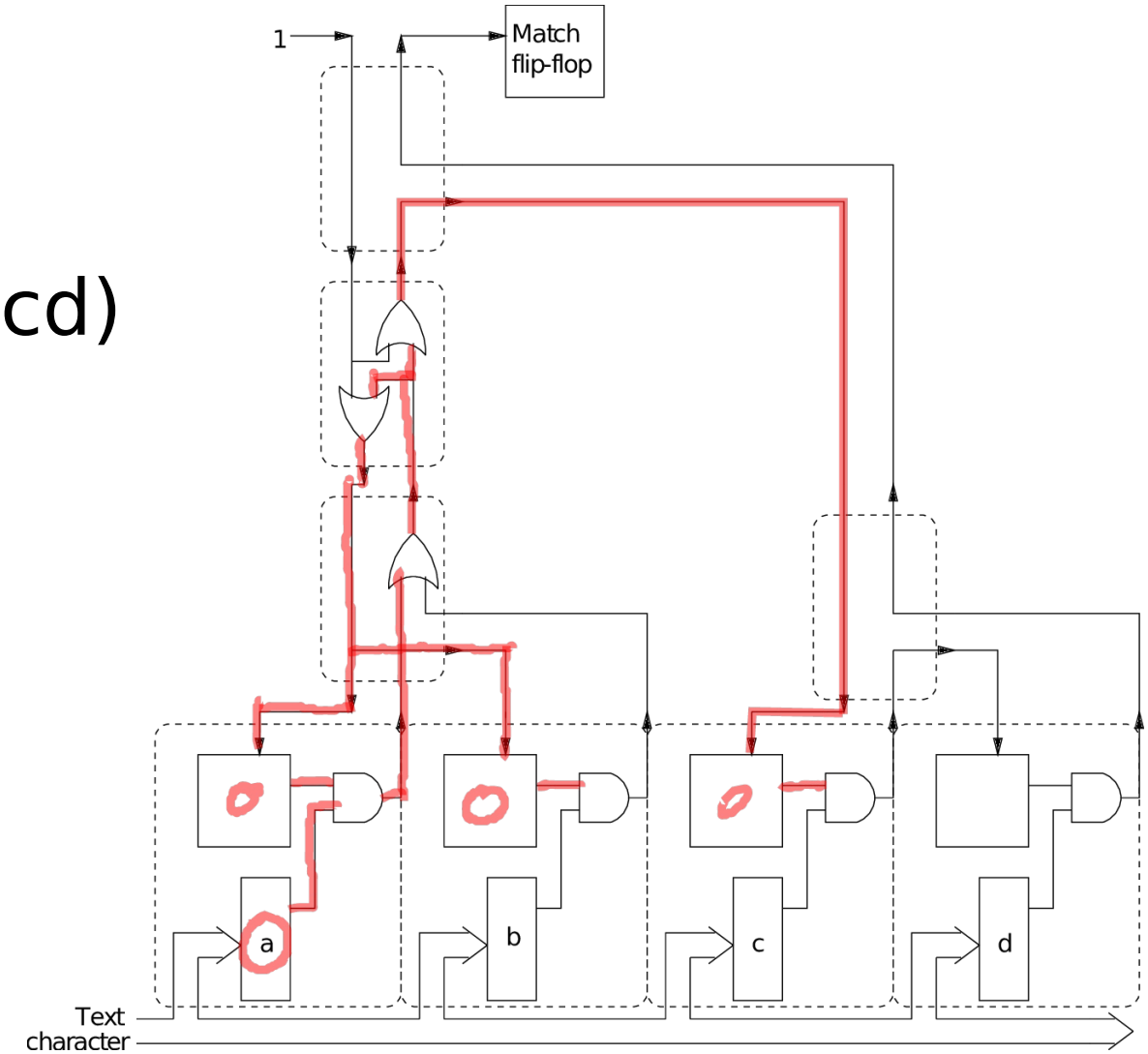


((a|b)*)(cd)
abcd



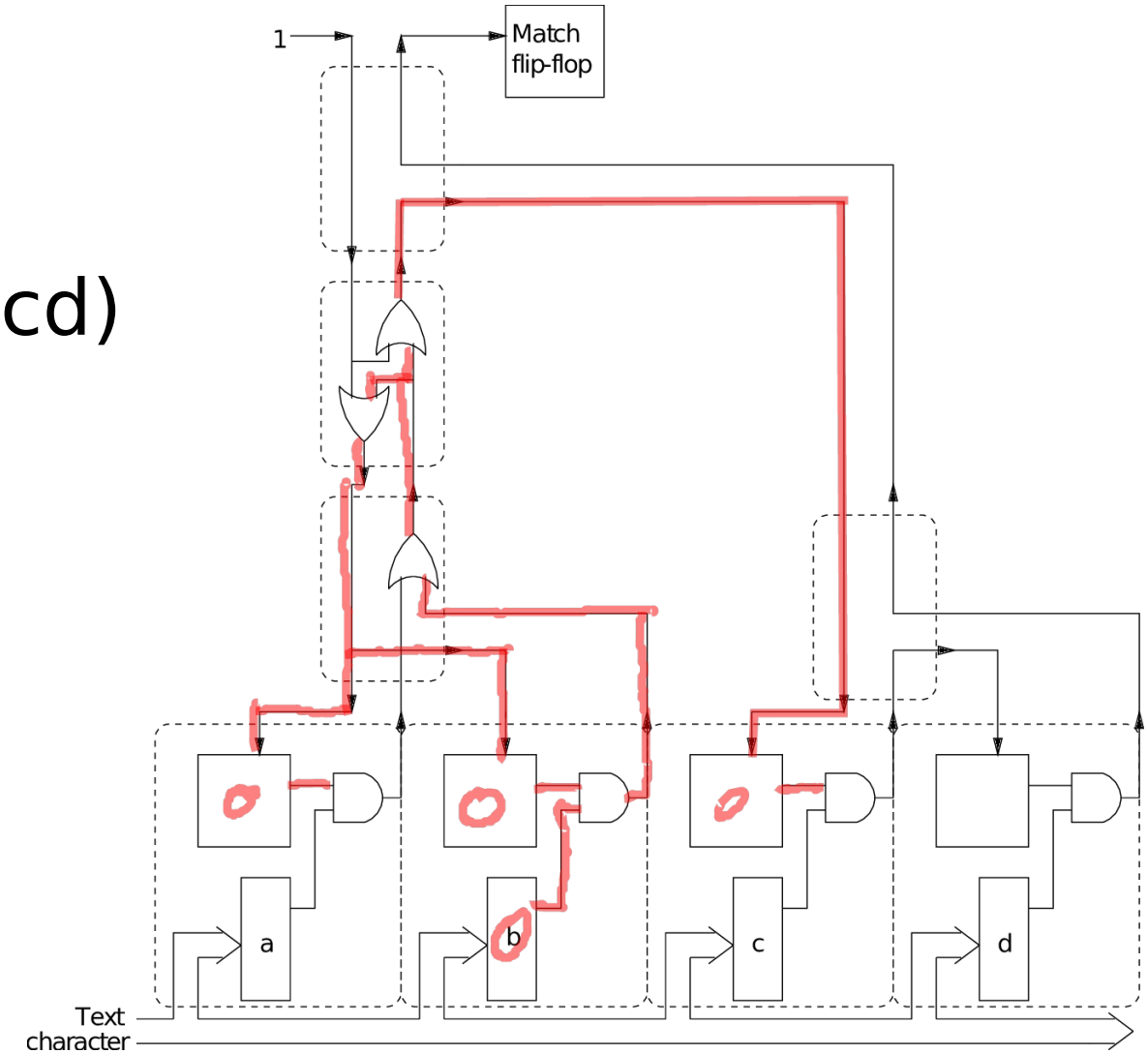
((a|b)*)(cd)

abcd



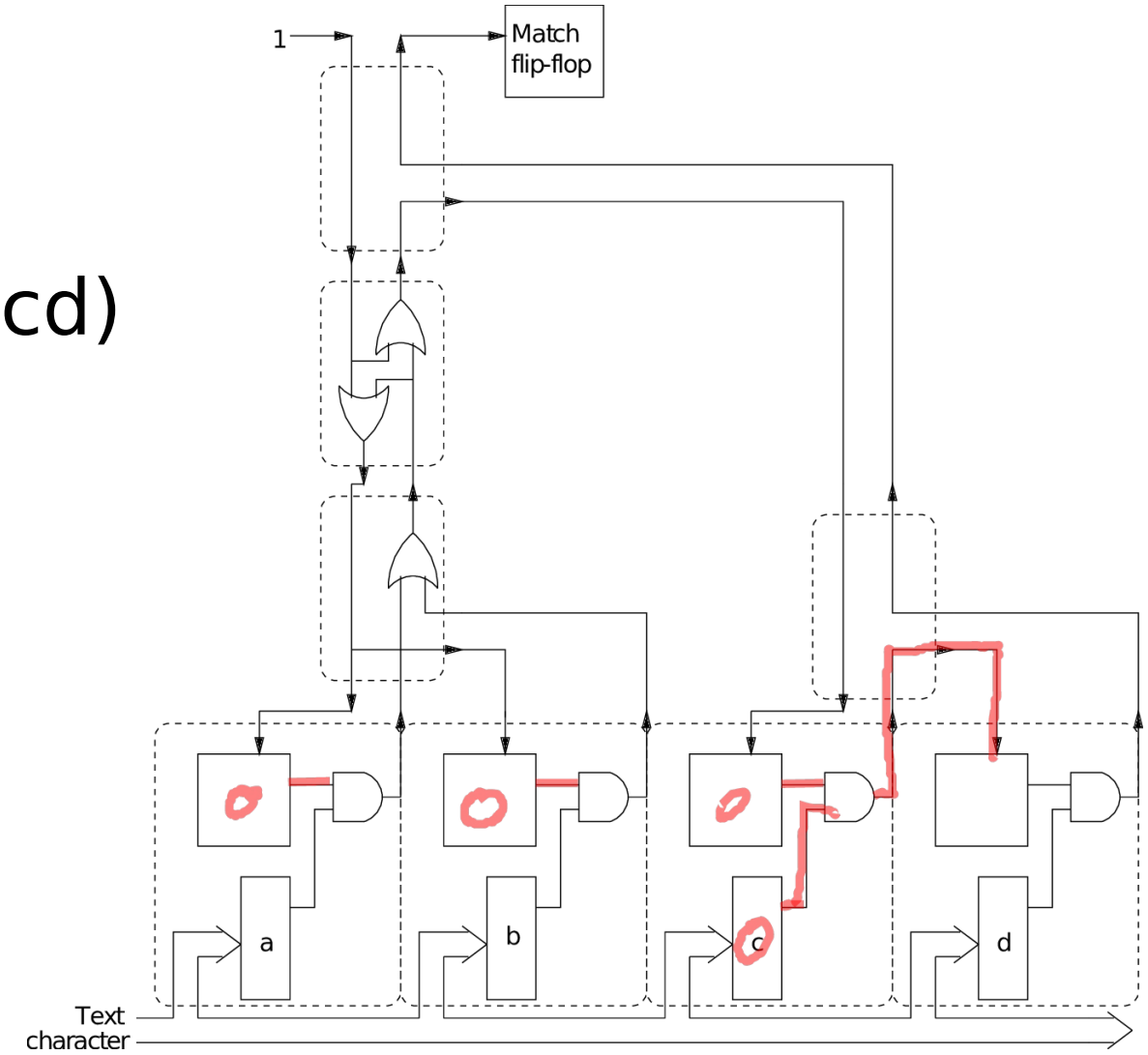
((a|b)*)(cd)

abcd



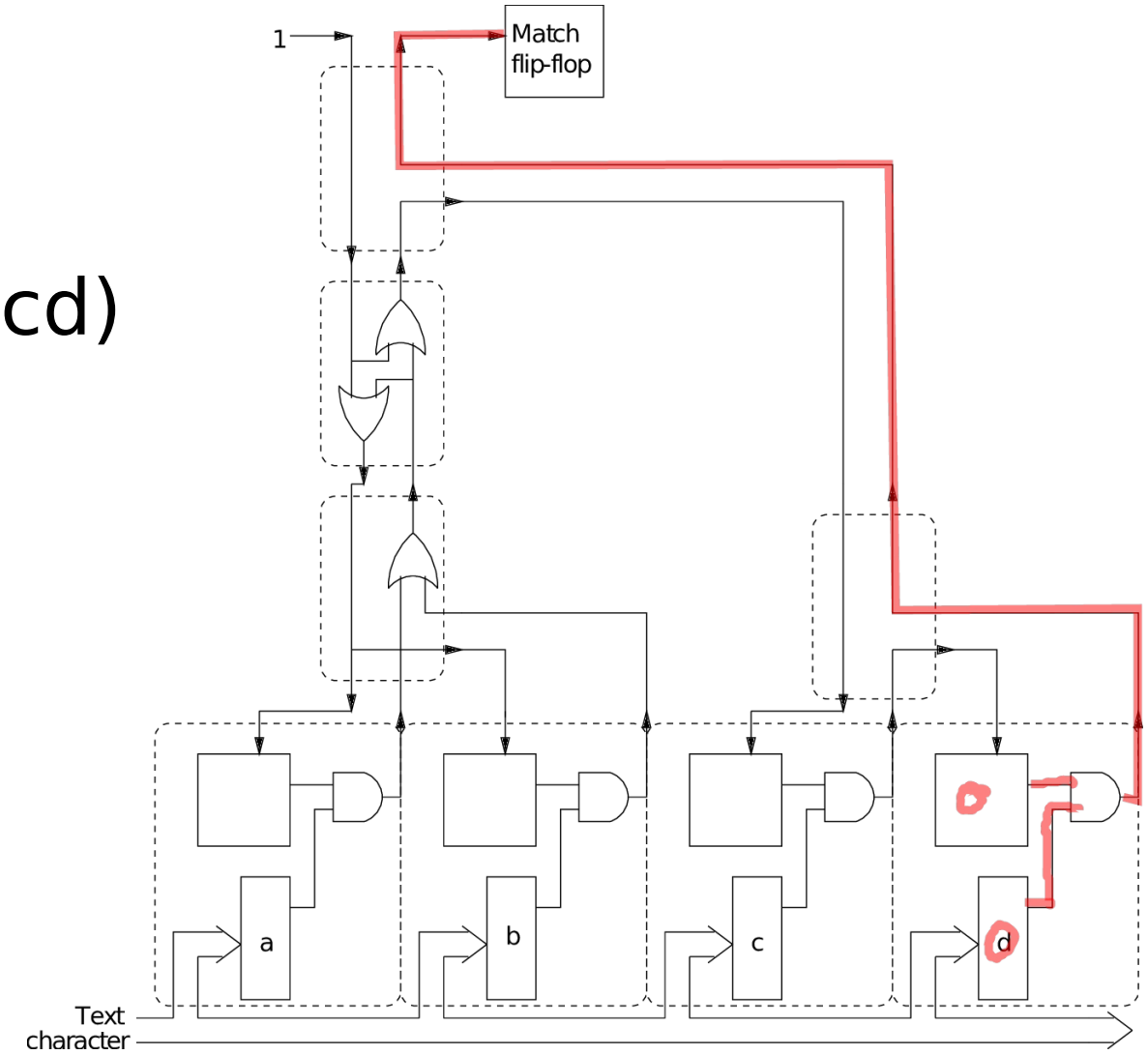
((a|b)*)(cd)

abcd

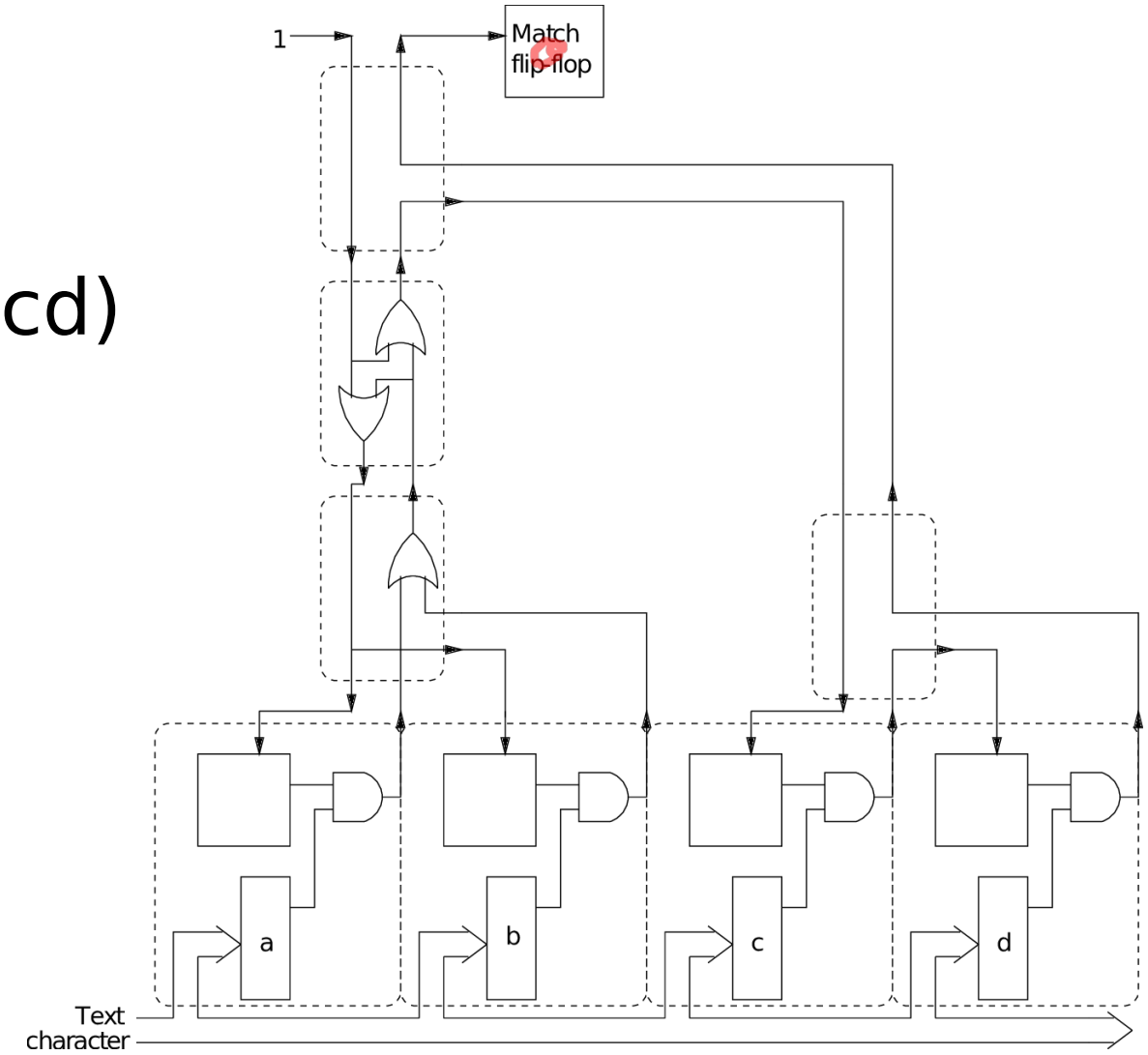


((a|b)*)(cd)

abcd



((a|b)*)(cd)
abcd



Drop of Complexity

- completely parallelized
- linear in energy
- still minimal

(a|b)*a(a|b)(a|b)...(a|b)
k times (a|b)

k	NFA area	Construction time	Time per text character
8	10 × 7 CLBs	21 ms	10.70 ns
9	11 × 8 CLBs	39 ms	11.68 ns
10	12 × 8 CLBs	32 ms	11.99 ns
11	13 × 9 CLBs	34 ms	12.17 ns
12	14 × 9 CLBs	31 ms	12.69 ns
13	15 × 10 CLBs	29 ms	12.32 ns
14	16 × 10 CLBs	33 ms	12.70 ns
15	17 × 11 CLBs	34 ms	11.89 ns
16	18 × 11 CLBs	34 ms	12.55 ns
17	19 × 12 CLBs	37 ms	13.06 ns
18	20 × 12 CLBs	37 ms	13.24 ns
19	21 × 13 CLBs	31 ms	14.98 ns
28	30 × 16 CLBs	39 ms	17.42 ns

Table 5: NFA area, NFA construction time and time per text character for the FPGA implementation.

k	Text file size (bytes)	CPU time	Maximum memory	Time per character
8	2560	0.00 s	580 KB	–
9	5632	0.00 s	580 KB	–
10	12288	0.00 s	580 KB	–
11	26624	0.00 s	580 KB	–
12	57344	0.00 s	580 KB	–
13	122880	0.005 s	580 KB	–
14	262144	0.01 s	580 KB	–
15	557056	0.03 s	580 KB	53.86 ns
16	1179648	0.04 s	580 KB	33.91 ns
17	2490368	0.08 s	580 KB	32.12 ns
18	5242880	0.17 s	580 KB	32.42 ns
19	11010048	0.34 s	580 KB	30.88

Table 2: Results for text search and minimal DFA construction (best case).

Dense Linear Algebra

- fix point/floating point
- math operations in a pipeline
- reduce pipeline
- performance

fix point/floating point

fix point

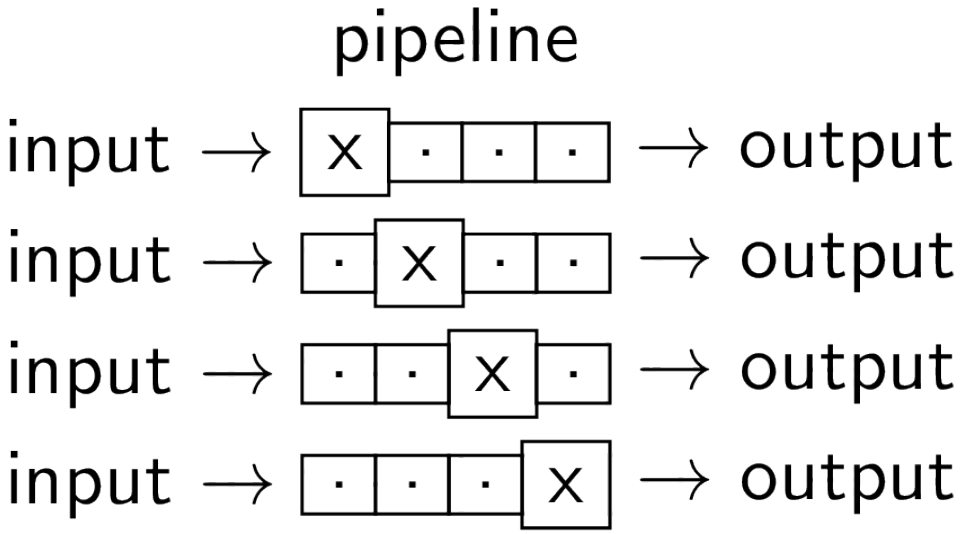
- almost like int
- operations are simple

floating point

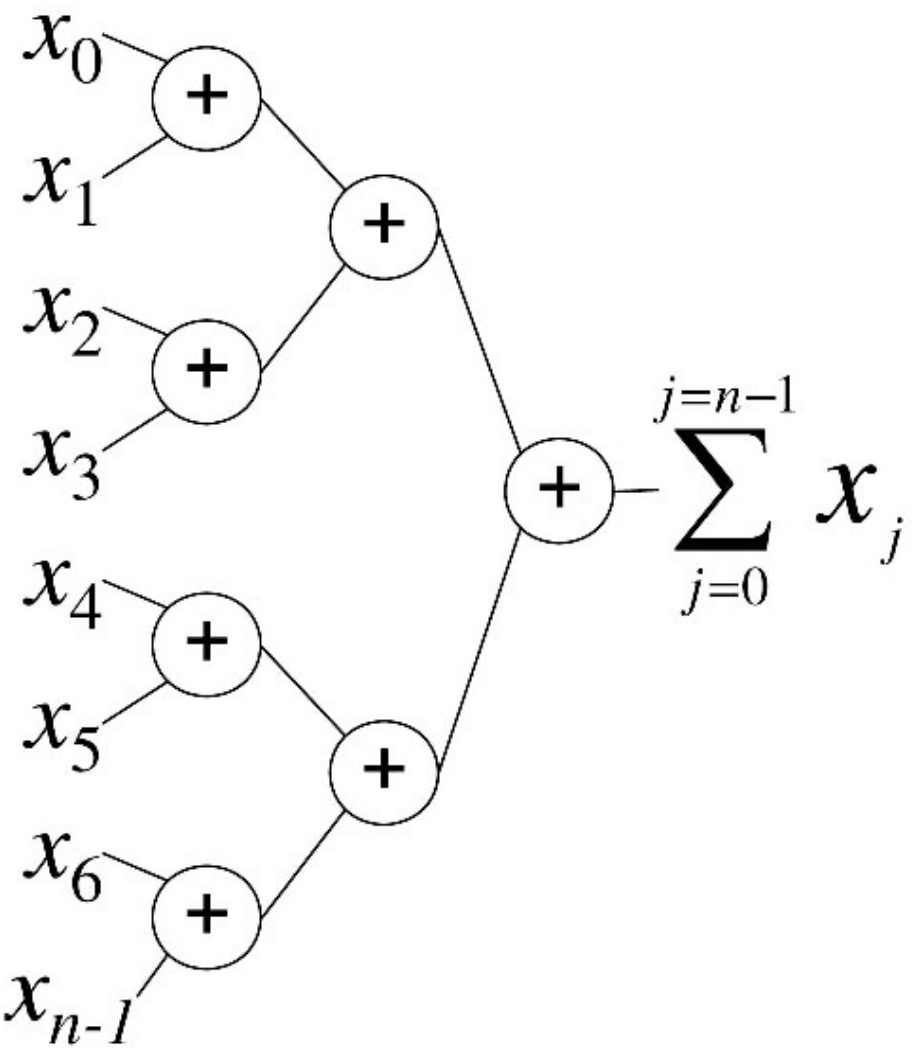
- complex behavior
- exponent
- denormalized numbers
- NaN, inf

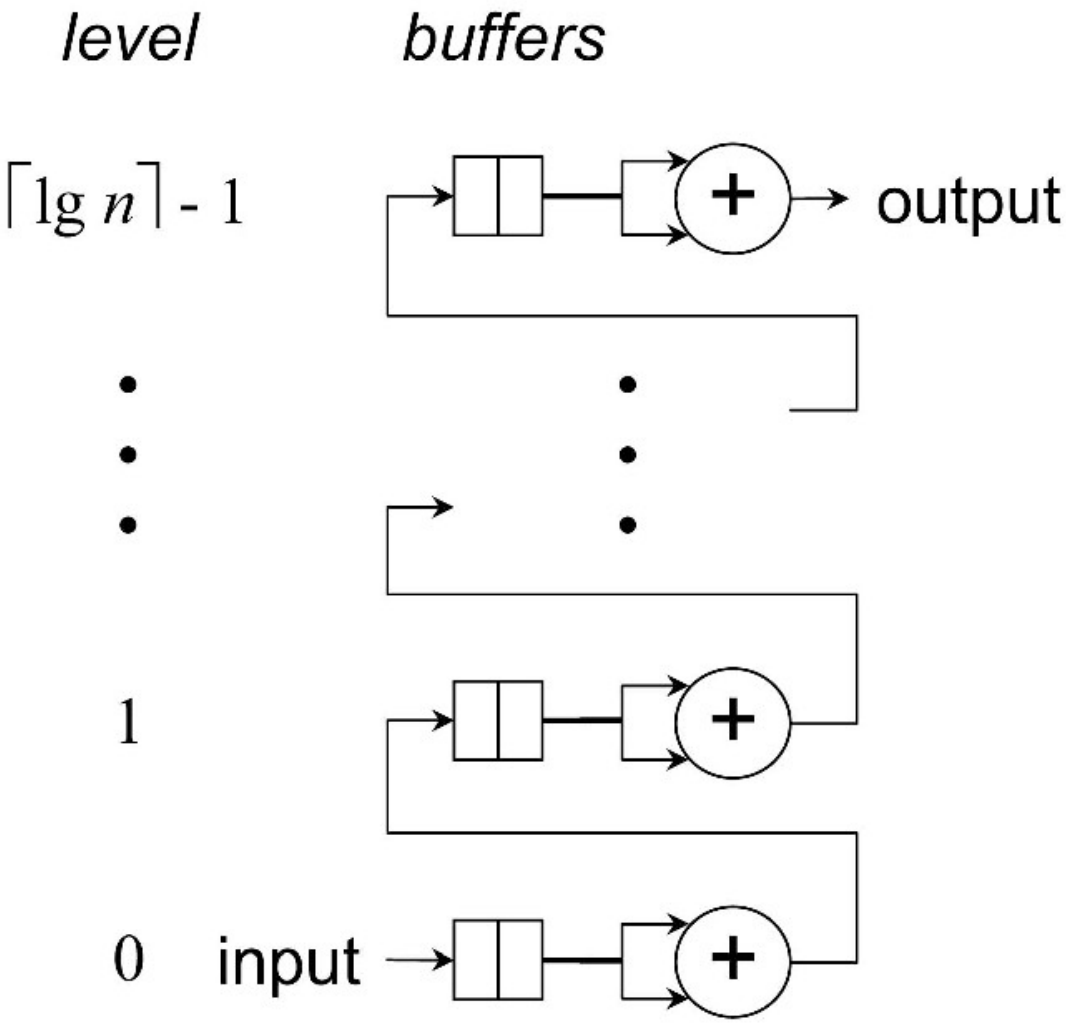
math operations in a pipeline

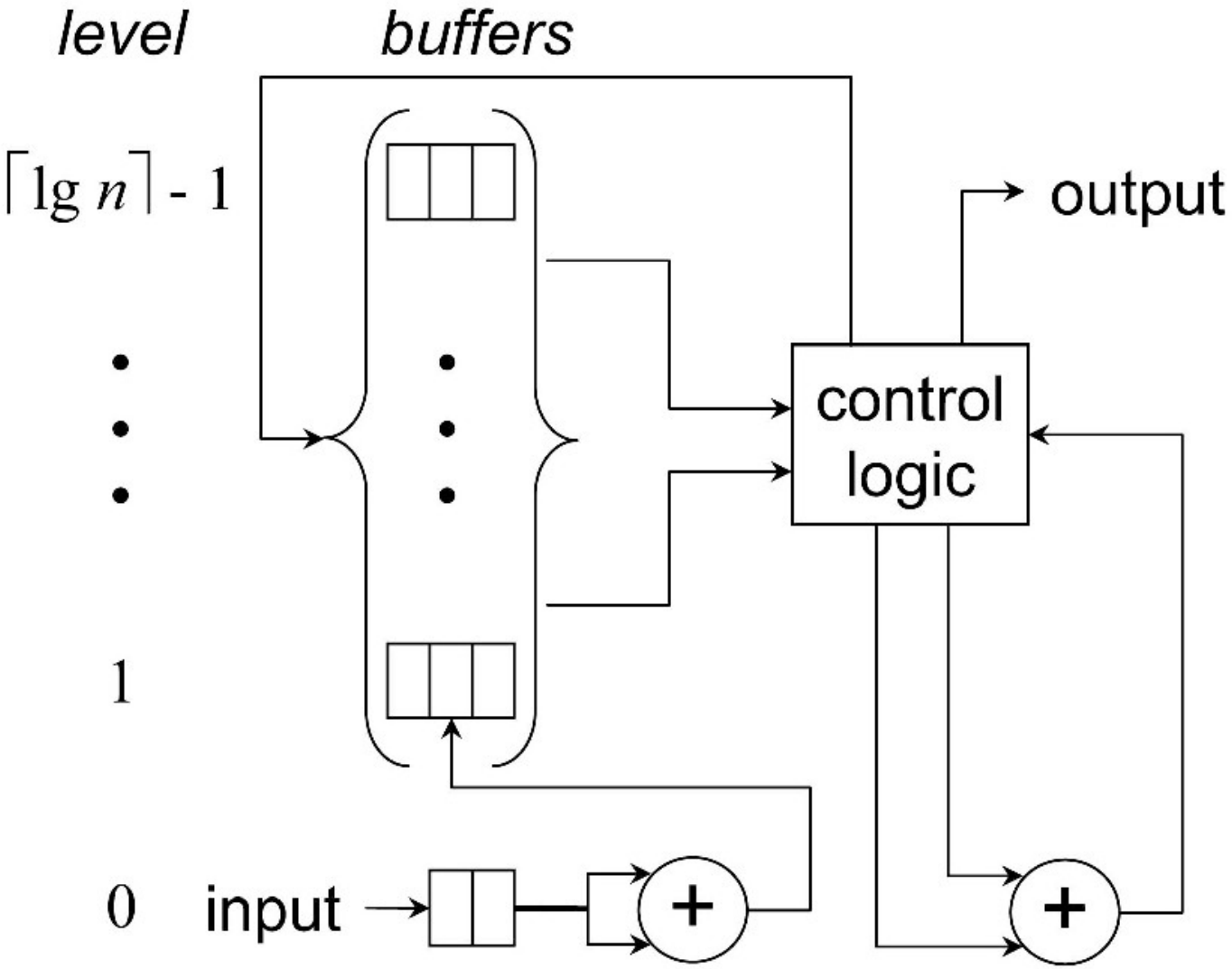
- take α cycles for a result
- accept input each cycle

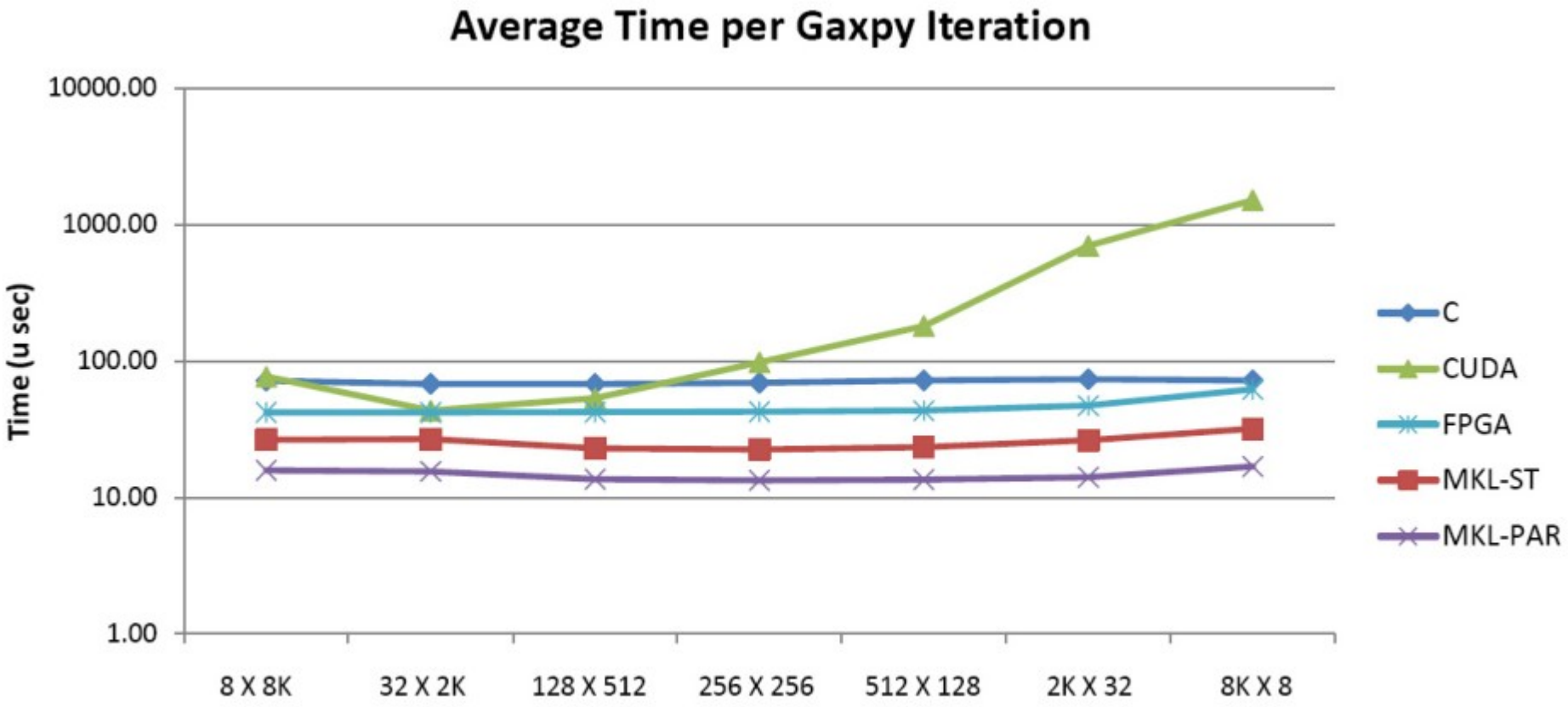


reduce tree

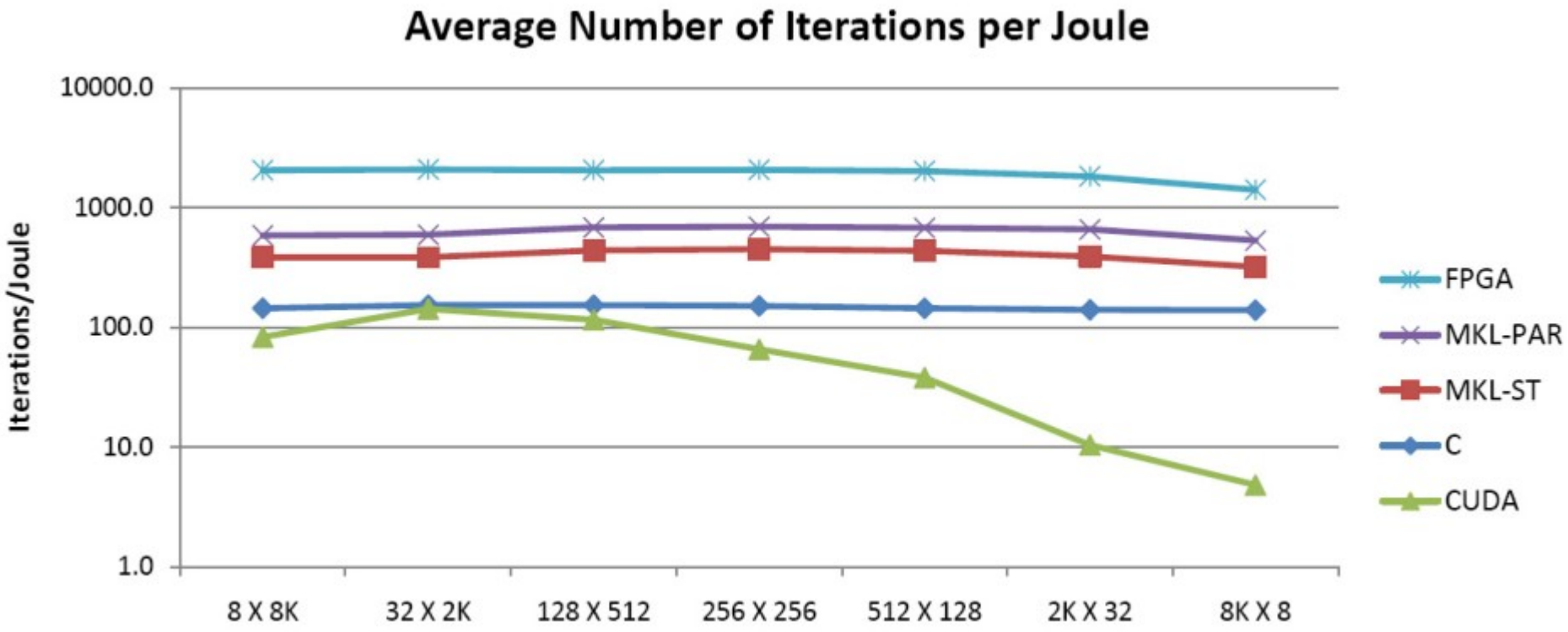



level
buffers
⌈lg n⌉ - 1
output
1
0
input

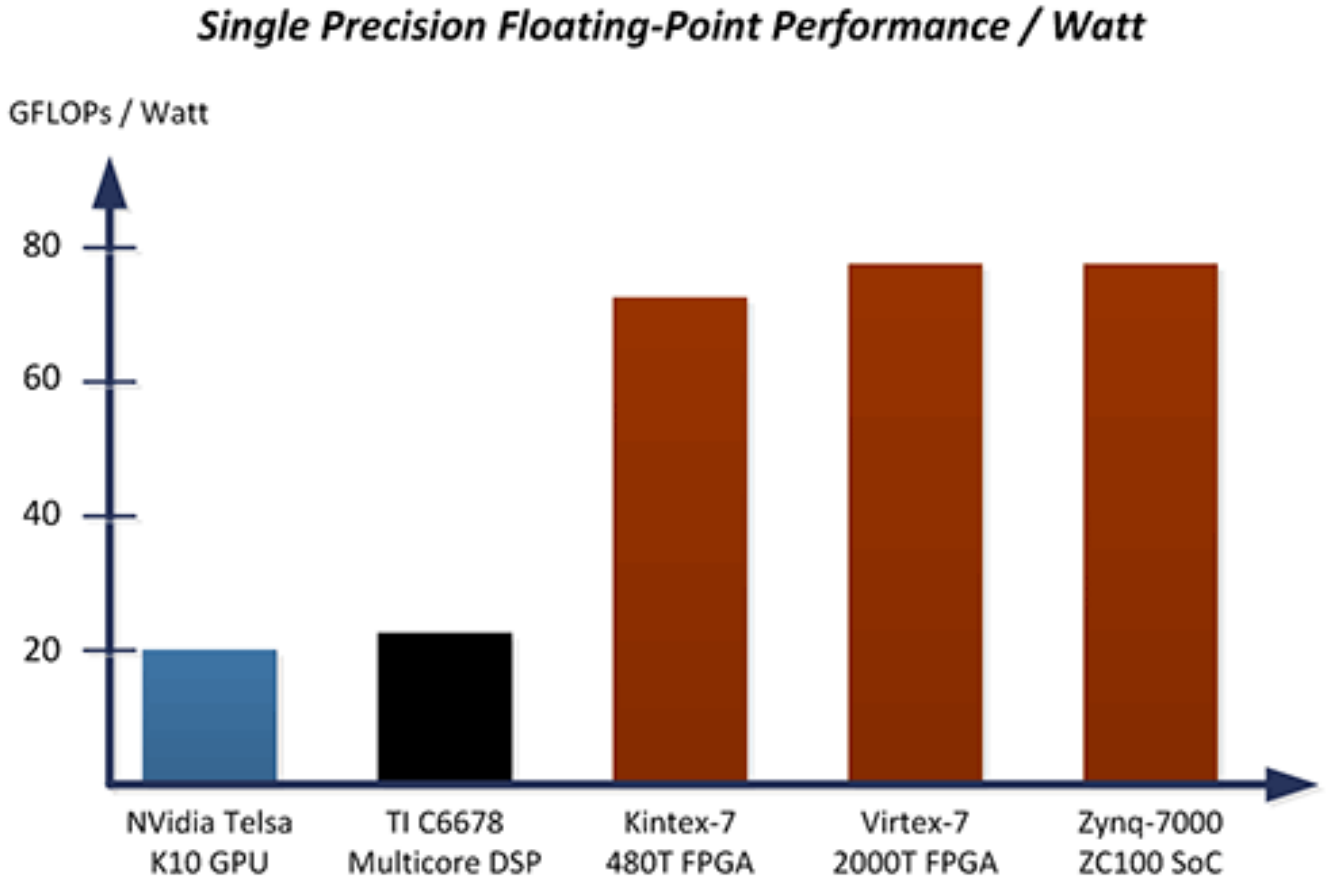

level
buffers
⌈lg n⌉ - 1
output
control logic
1
0
input
+
+


Average Time per Gaxpy Iteration
Time (u sec)
10000.00
1000.00
100.00
10.00
1.00
8 X 8K
32 X 2K
128 X 512
256 X 256
512 X 128
2K X 32
8K X 8
C
CUDA
FPGA
MKL-ST
MKL-PAR

$y = Ax + y$


Average Number of Iterations per Joule
Iterations/Joule
10000.0
1000.0
100.0
10.0
1.0
8 X 8K
32 X 2K
128 X 512
256 X 256
512 X 128
2K X 32
8K X 8
FPGA
MKL-PAR
MKL-ST
C
CUDA

y = Ax + y



source: Xilinx

Summary

very scalable
many useful applications
might become a default component of personal computers

Image Sources

- High-Performance Reduction Circuits Using Deeply Pipelined Operators on FPGAs
- BLAS Comparison on FPGA, CPU and GPU
- Fast Regular Expression Matching using FPGAs