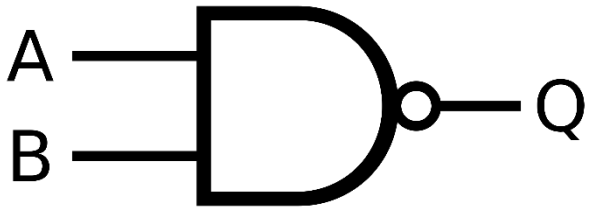
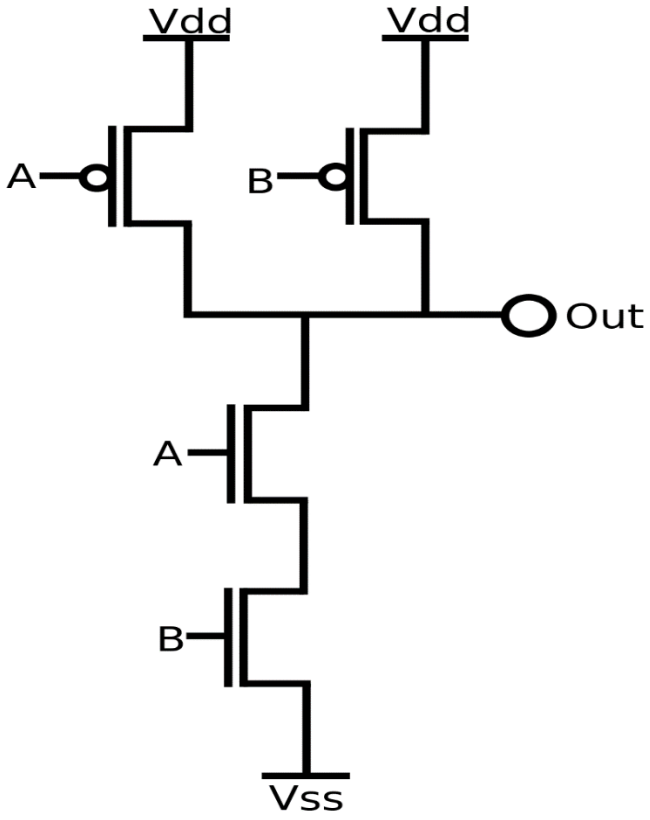
CPU – principles of operation

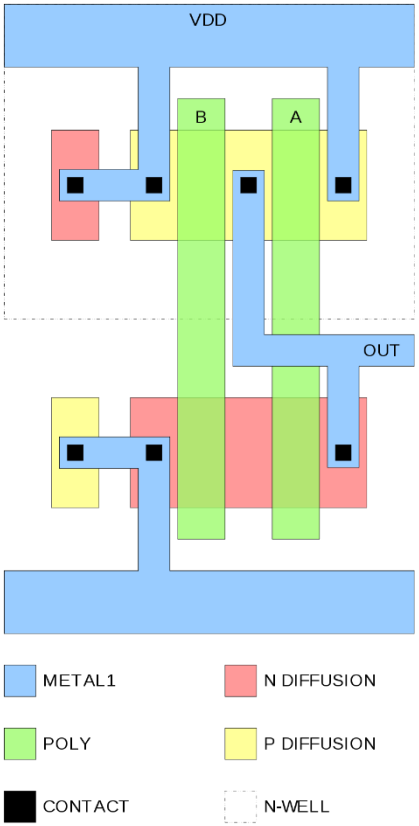
Transistor level

- CPU and similar chips mostly use CMOS technology (1963)
  - C=Complementary – both polarities of transistors available
    - A mix of n-type and p-type transistors on the same chip
    - The mix allows control of both incoming and outgoing current at a gate
  - MOS=Metal-Oxide-Semiconductor – a particular transistor technology (1959)
    - A kind of FET=Field-Effect-Transistor – a transistor controlled by voltage on "Gate"
      - Behaviorally similar to a triode vacuum tube (1908) but much smaller voltage, size, price
    - Also called Unipolar – only one kind of current-carriers used (in one transistor)
      - n-type transistors use electrons, p-type use holes
      - The first transistors (1947) were Bipolar (both electrons and holes used to transport current) and controlled by current – no longer used in computing after ~1980
  - DRAM (1968) and Flash (1980) chips use slightly different technologies
    - the memory effect is provided by a specially constructed transistor
    - usually not mixable with CPU in the same chip
      - only low-density devices like embedded CPUs can mix CMOS logic and Flash

Logic (gate) level

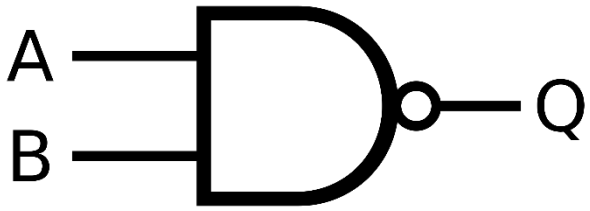


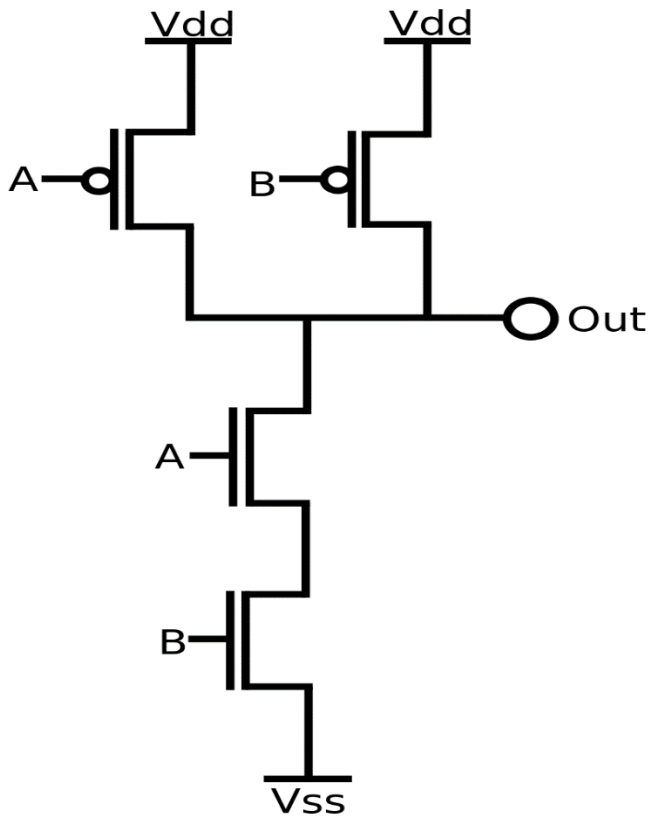


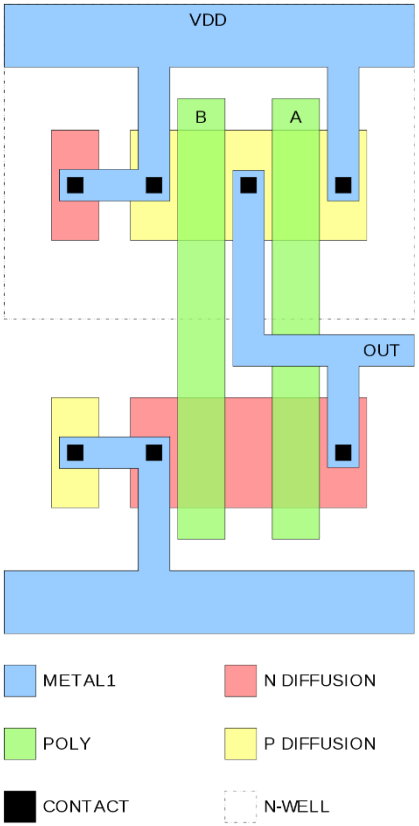


- Four transistors constitute a **NAND gate**
  - Two inputs: A, B
  - Output = !(A&&B)
  - High voltage on both inputs closes the p-type transistors (current sources, top) and opens the n-type transistors (current sink, bottom)

Logic level

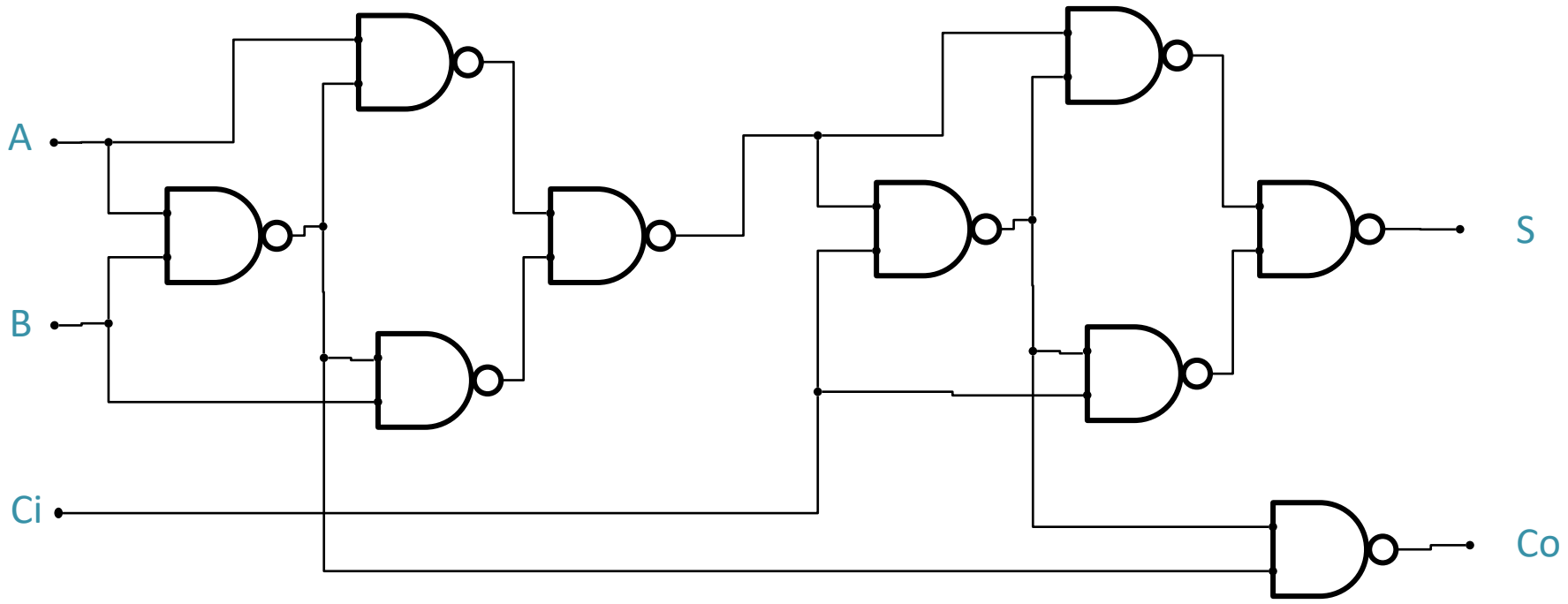






- Power consumption
  - Gates and wires connected to Out form a capacitor which must be charged and discharged in each cycle – proportional to frequency, reduced by Dennard scaling
  - Short-circuit current (all transistors are open during state changes)
  - Leakage through closed transistors – increased by geometry scaling down

Full 1-bit adder

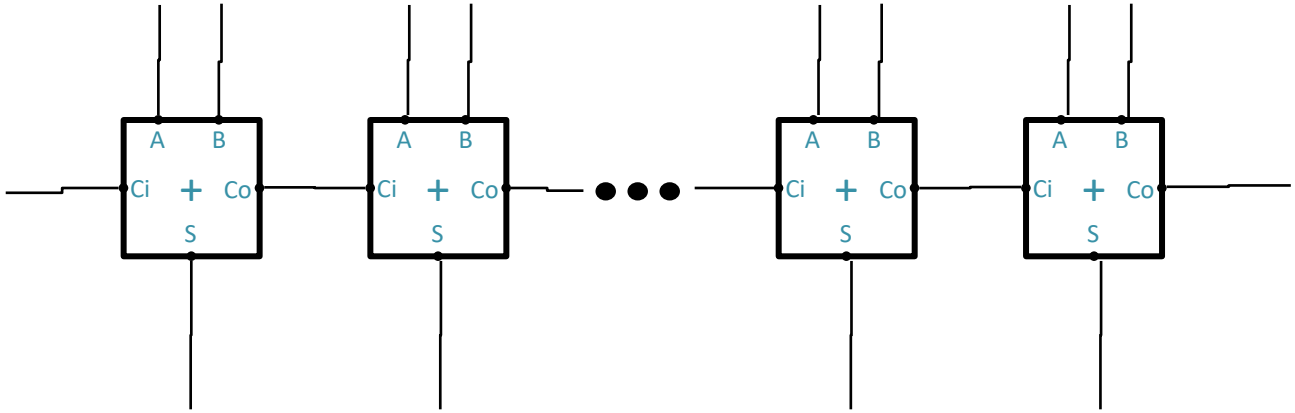


- Full adder = 1-bit addition with carry in and carry out
  - The sum of three 0/1 inputs may be 0/1/2/3 – representable by 2-bit output

```
(Co<<1)|S = A+B+Ci
```

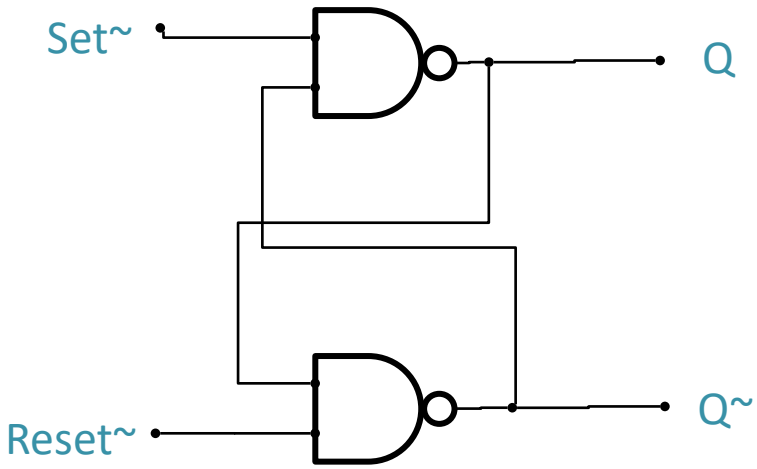
  - The circuit is made smaller with three-input NAND and NOR gates

N-bit adder



- N-bit addition with carry in and carry out
  - O(N) delay - too slow for large N
  - Recursive “dynamic-programming” construction used for large N
    - O(log N) delay, but significantly more gates needed

Combinational vs. sequential circuits

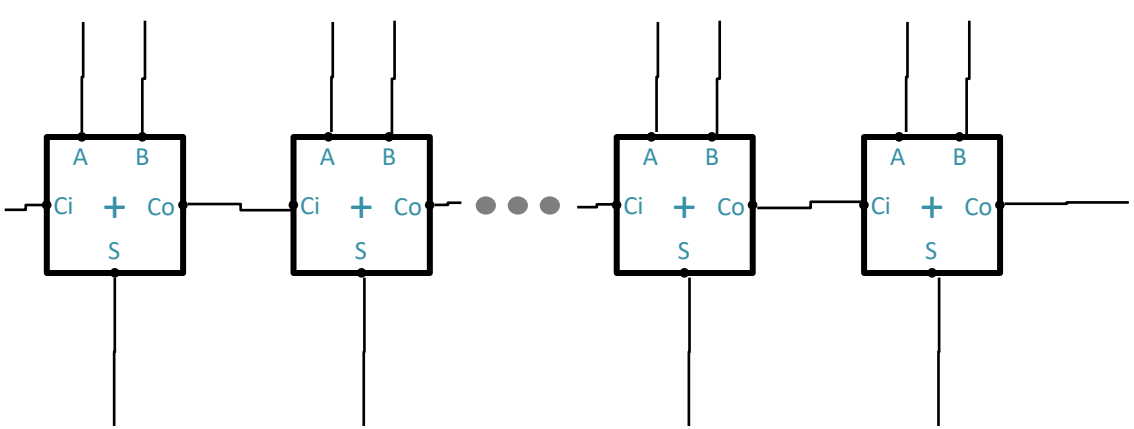


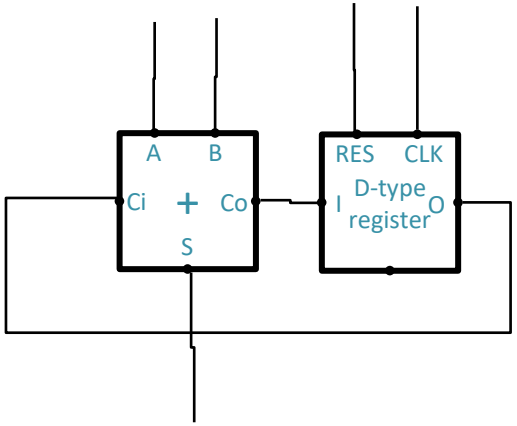
- **Combinational** circuit = directed acyclic graph = state-less circuit
  - The signal is only delayed, proportionally to the longest path length
- **Sequential** circuit = cyclic graph
  - Cycles cause state-full behavior (usually through positive feed-back)
  - Various kinds of registers
    - *"Register"* is originally a digital-design term – any state-full circuit used to store information
    - The simplest case of register is the 1-bit Set/Reset Latch depicted above
    - A *CPU register* (as referenced in machine code) is, in digital-design terminology, a part of small *Static RAM*
    - *Static RAM* is an *addressable* array of n-bit registers – one of them is activated by a *binary decoder* (a combinational circuit)
  - More sophisticated cases (e.g. clock dividers)

Digital-circuit design

- ***Process*** (*lithography*) – the technology of the chip-producing ***foundry*** *(fab)*
  - Often simply denoted by geometric resolution in nanometers
  - Decides the geometrical, electrical, and timing properties of transistors
- ***Physical design*** – produces *masks* for manufacturing
- ***Transistor Level*** – network of transistors (and other elements)
  - Often skipped as each gate type has its optimized physical design
- ***Logic (Gate) Level*** – network of gates, latches, etc.
- **RTL** (Register-Transfer Level) design
  - Circuits limited to Combinational circuits + Registers
    - Registers controlled by a common clock
  - Originally a principle of systematic (but not optimal) gate-level design
  - Later an intermediate level of design, specified as program-like text (not schematically)
  - Now usually generated from system-level design
- ***System Level*** design
  - Done in **System-C**, **Verilog** (1986), or **VHDL** (1981)
    - Effectively parallel-programming languages
      - Verilog and VHDL also allow direct RTL/Logic/Transistor-level design, analog and simulation models
      - There is probably more work done than in gcc/clang/msvc/javac together
    - Restrictions (w.r.t. general programming) allow transformation to RTL designs
      - The transformation shares some algorithms with parallelizing compilers
- Electronic Design Automation (EDA) software is a 10-billion USD/year industry
  - The largest player is Synopsys, Inc. (market cap USD 35B, cf. Microsoft 1700B)

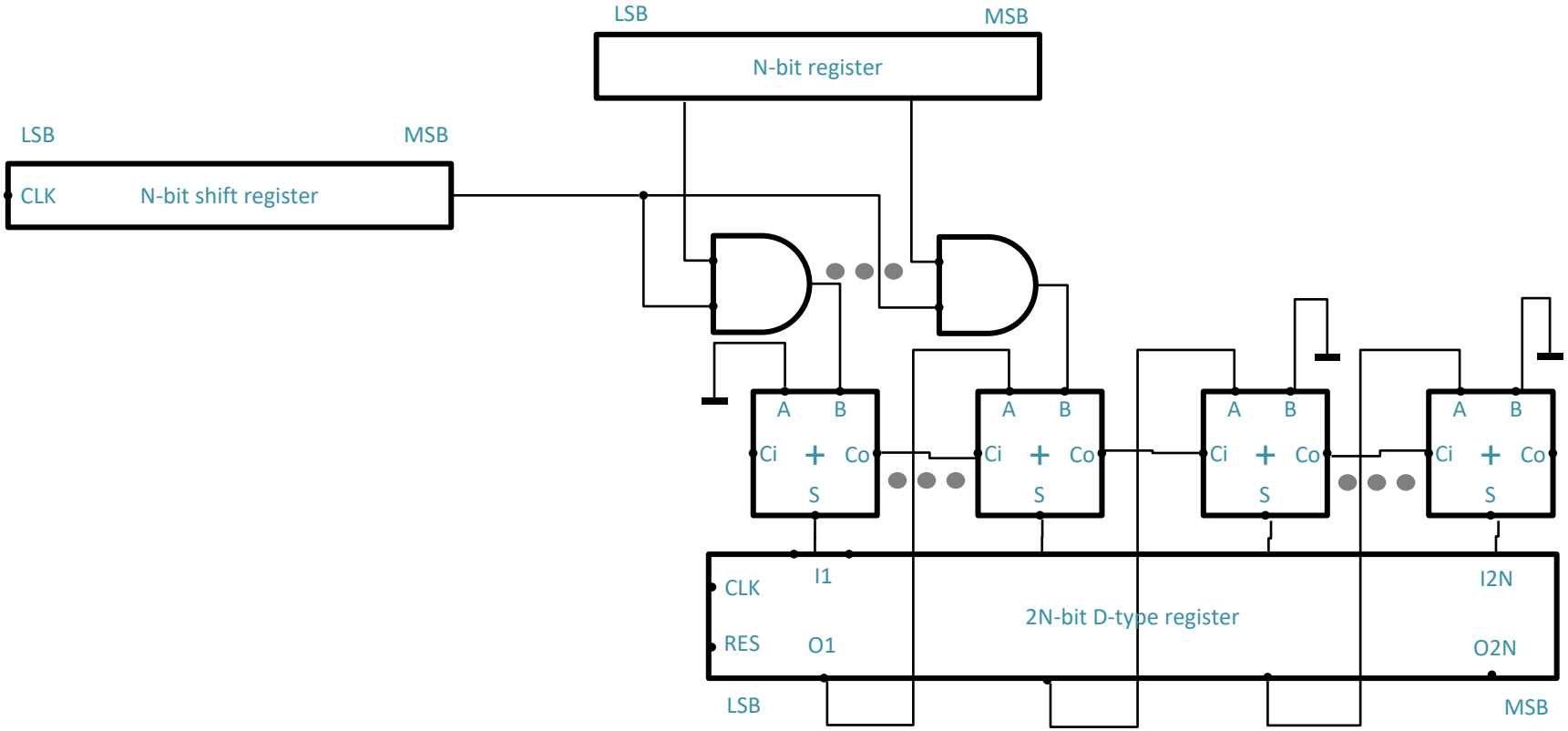
Combinational vs. sequential N-bit adder





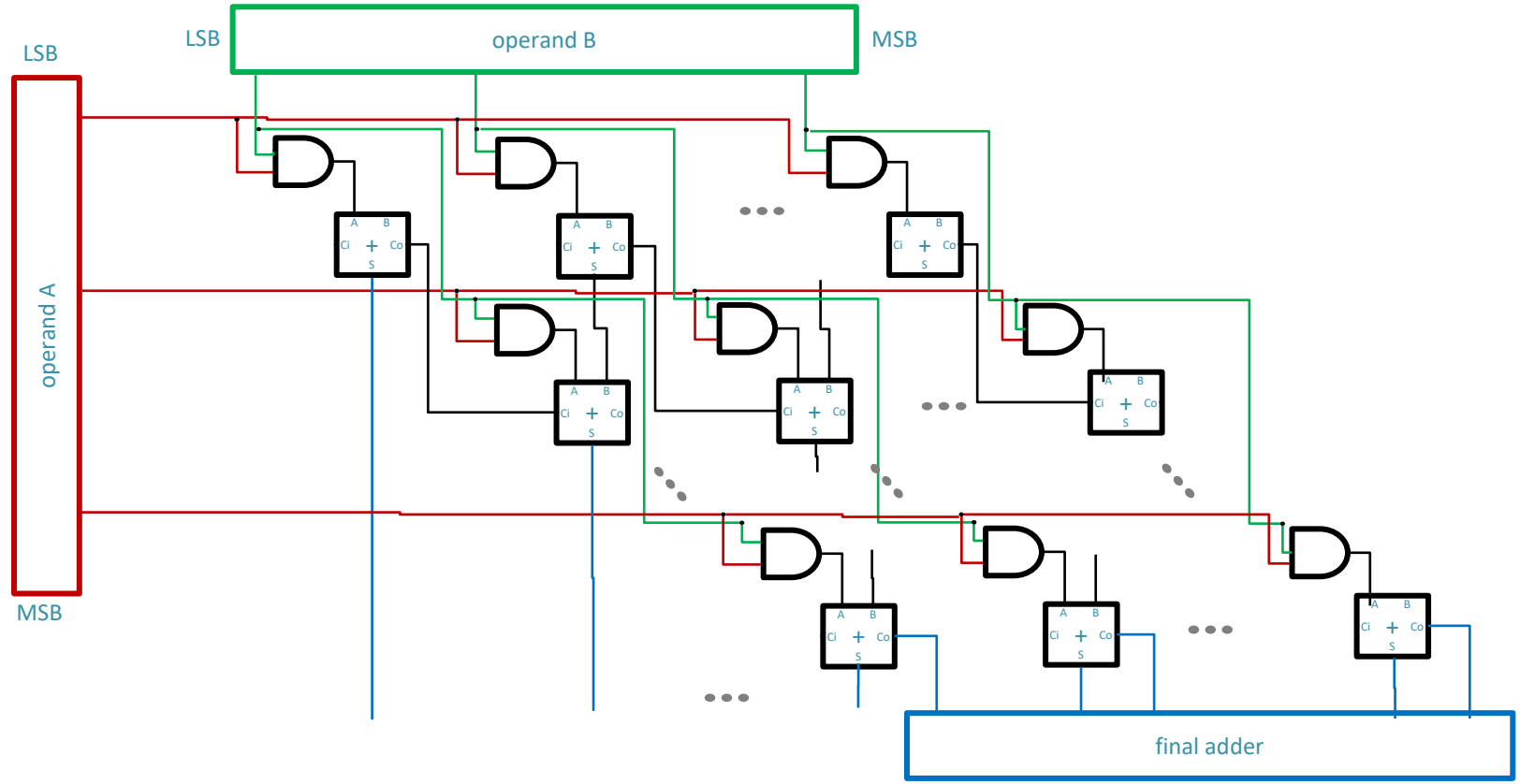
- Sequential N-bit adder
  - Input bits serialized on 2 wires, synchronized, in LSB-to-MSB order
  - Requires clock to sample inputs and to control the (D-type) register
    - Due to delay, outputs (Co, S) must be sampled later, at the end of clock cycle
  - Slower even than naïve combinational but O(1) gates

Sequential N*N-to-2N-bit multiplier



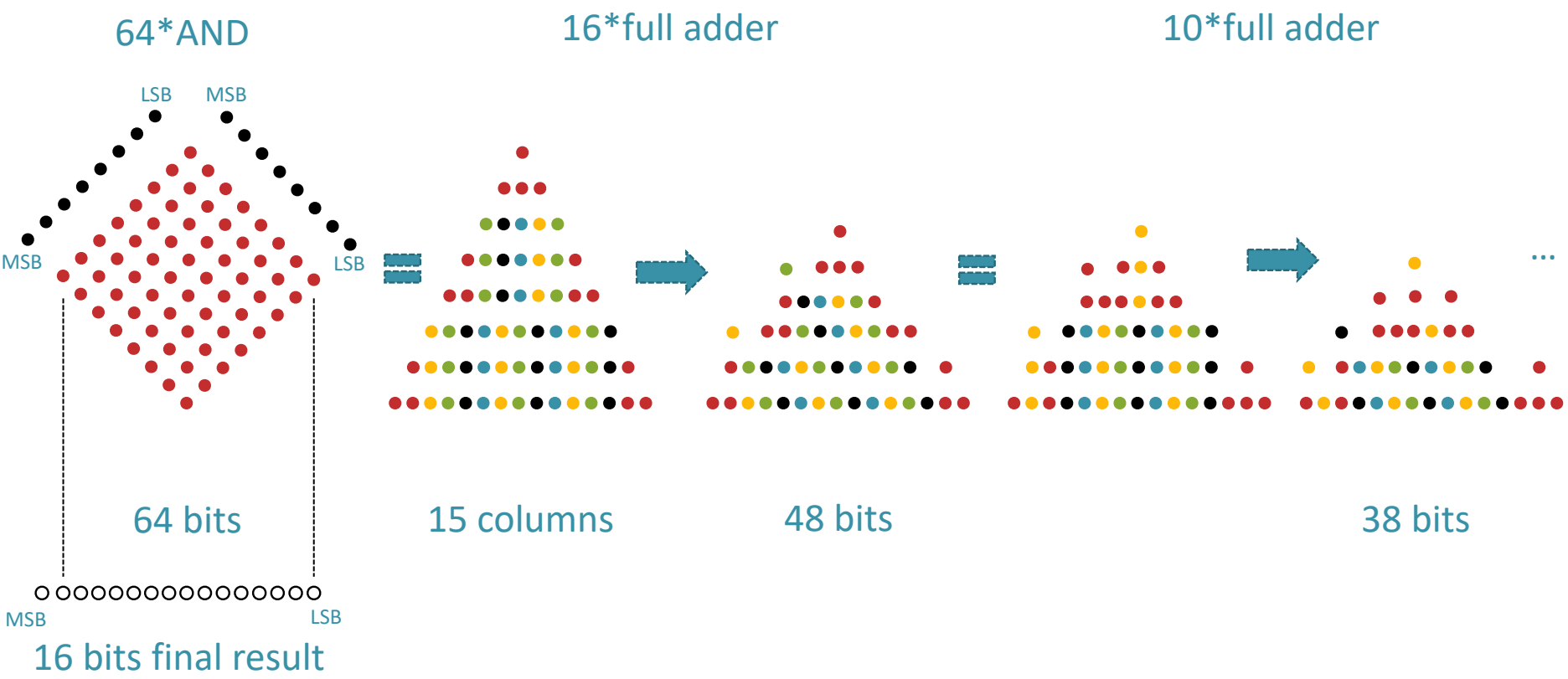
- The school multiplication algorithm in binary digits
  - Shown mirrored wrt. school convention, i.e. with LSB on the left and MSB on the right
  - One operand fixed, one being shifted by a shift register
  - 1*N-bit multiplication implemented by N AND gates
  - Addition of N N-bit results implemented by 2N full adders with shifting loopback
  - Total delay O(N*N); may be improved to O(N*log N) with optimized 2N-bit adder

Combinational N*N-to-2N-bit multiplier



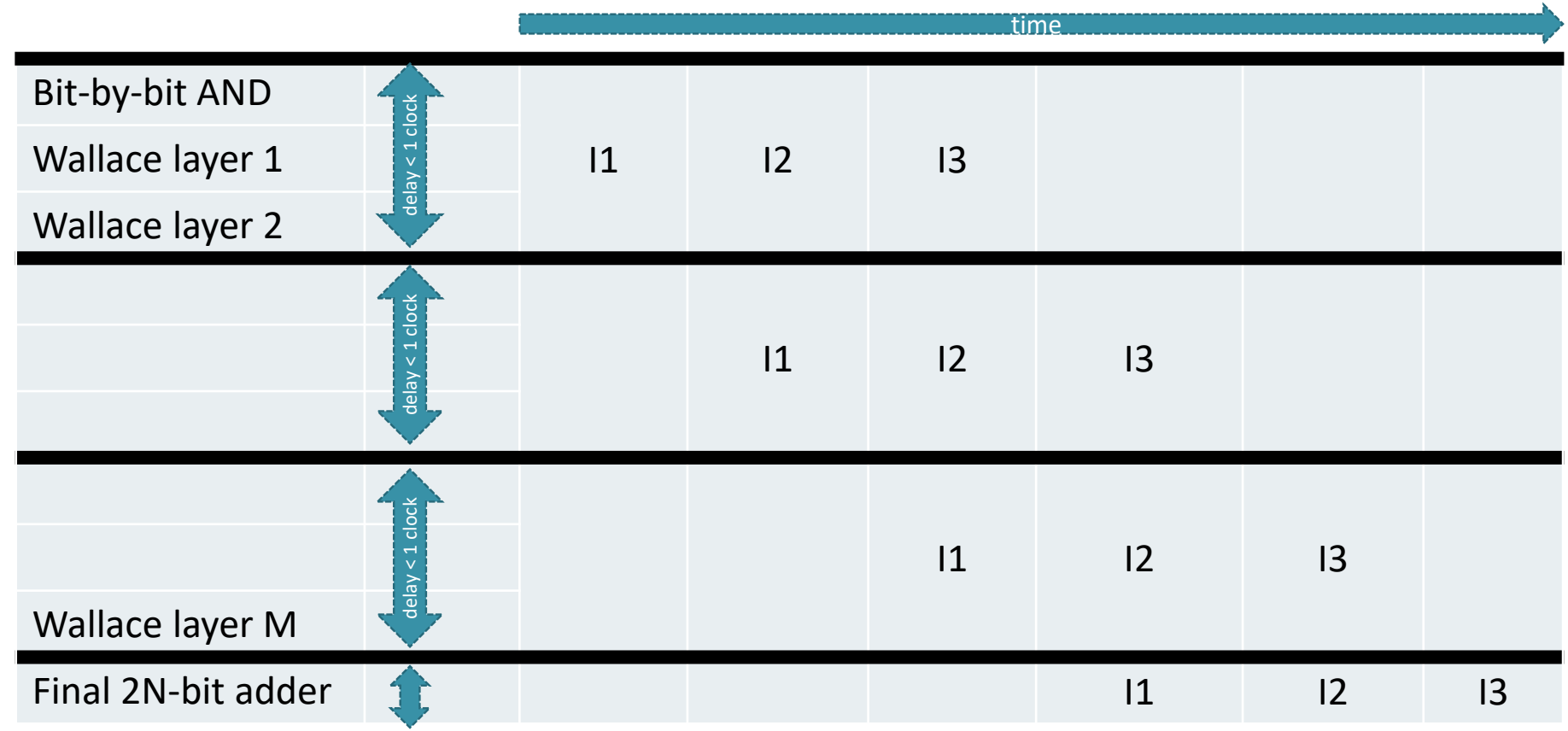
- The school multiplication algorithm in binary digits
  - Shown mirrored wrt. school convention, i.e. with LSB on the left and MSB on the right
  - 1*N-bit multiplication implemented by N*N AND gates
  - Addition of N N-bit results implemented by N*N full adders, then by a final N+N-bit adder
  - Total delay O(N)
    - In real designs (1992), reduced to half by radix-4 Booth encoding of operands

Wallace-tree multiplier (invented 1964, used after 2000)



- 1-bit multiplier = AND gate
- N-bit multiplier must add N*N 1-bit results, properly shifted
  - Wallace-tree: Instead of adding bits in the columns, recursively reduce number of operands by full adders
    - A full adder converts three bits (in the same column) into two bits (in two adjacent columns)
    - Each layer reduces the number of bits in a column from M to (M/3)+(M%3) but sends (M/3) bits to the left
    - After $O(\log_{3/2} N*N) = O(\log N)$ layers, the remaining small columns are summed by optimized 2N-bit adder
    - This algorithm is done in the circuit generator – the generated circuit is combinational with O(log N) delay

Wallace-tree multiplier staged



- The Wallace-tree consists of tens of layers ($\log_{1.22} 64 = 21$)
  - Each layer is a set of independent full adders
- The total delay may be longer than the desired system clock period
- Registers inserted after each K layers, delay between registers less than a clock
- A new multiplication may be started in each clock period = **pipelining**

Arithmetic units in general

- Sequential implementation
  - Hardware equivalent of loop in a program
  - The same hardware used repeatedly during the same operation
  - **Minimum number of transistors**, large latency
    - The registers incur additional delay
- Purely combinational implementation
  - Hardware equivalent of mathematical formula
  - Each piece of hardware used only once during the same operation
    - Idle for most of the time
  - Large number of transistors, **minimum latency**
  - Advanced (non-iterative) algorithms possible (e.g. Wallace multiplier)
- Staged combinational implementation
  - Hardware equivalent of unrolled loop (but iterations may differ)
  - Each piece of hardware used once during a clock period
    - May be used for another operation in the next period
  - Even larger number of transistors, latency similar to sequential implementation
    - Latency may be lower if a stage corresponds to more iterations of the algorithm
  - **Large throughput** due to pipelining

Arithmetic units in general

- When increasing transistor speed is no longer possible (2005)...
- ... there are two ways to improve throughput:
  - Replace sequential by staged combinational implementation
    - Allows **pipelining**
  - Implement more than one arithmetic unit
    - Multiple units in the same pipeline (SIMD instructions)
      - A SIMD instruction may also use two pipelines in parallel
    - Multiple pipelines (processing instructions of the same thread)
      - Hyperthreading: More than one thread shares the same set of pipelines
    - Multiple cores/sockets (processing instructions of different threads)
- The price:
  - Increased number of transistors
  - Increased complexity of synchronization (additional energy consumption)
- Throughput (number of operations per time) is improved
- Latency (time required to finish an operation) is NOT improved
  - Latency may be improved by advanced circuit designs

Arithmetic units in general

- The reality in Intel/AMD CPUs (2010 to 2020):
  - Integer addition/subtraction is single-clock (unstaged combinational)
    - FP addition/subtraction usually 3 clocks due to shifting/normalization, pipelined
  - Multiplication is staged combinational
    - Latency usually 3 to 5 clocks
    - Pipelined: Throughput 1 instruction per clock per pipeline
  - Division is always iterative (sequential)
    - Latency 20 to 80 clocks (FP often faster than integer)
      - Latency may depend on actual values
      - The pipeline is blocked – very small throughput
  - Bit operations (AND/OR/XOR) are single-clock
    - Shifts/rotations may cost more than one clock
      - In terms of chip layout, ALUs are rather wide (across bits) than deep (across stages)
      - Any information travelling across (many) bits is problematic
  - SIMD operations: usually the same latency as their scalar counterpart
    - Only few iterative operations (FDIV) available in the SIMD instruction set
    - Some wide-vector instructions implemented as two half-width operations pipelined
      - This fact heavily influenced the SIMD instruction set