DEPARTMENT OF
INFORMATION SYSTEMS
AND COMPUTER SCIENCE

Pipelining Circuits

Or How to Make Your Circuit Look Like a Factory

Lecture Time!

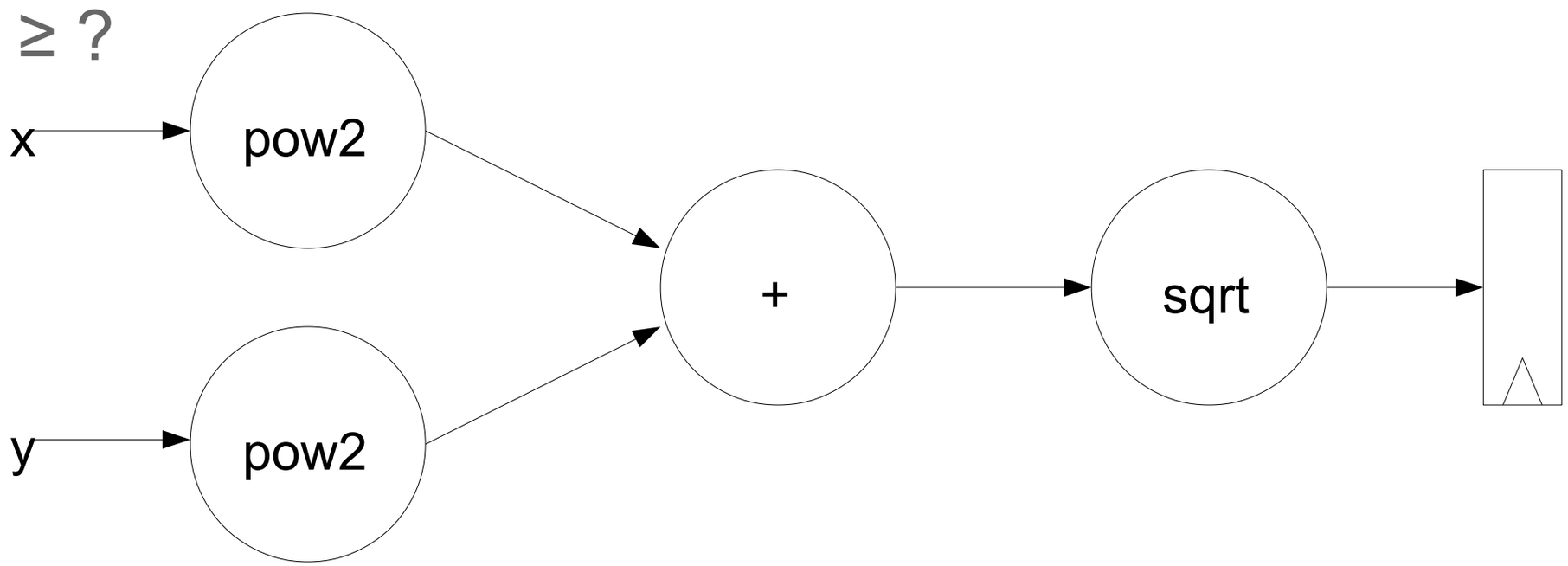
- Registers: Ideal and Non-Ideal
- Pipelining: Basics
- Performance: Throughput and Latency

Registers

- Since registers (edge-triggered flip-flops) are required for pipelining, it is important to remember that they are physical devices and have limitations.
  - These limitations are expressed in their timing attributes, namely t_{pdCQ}, t_{cdCQ}, t_s, and t_h.
- Ideal registers (which don't exist in real life) are registers with $t_{pdCQ} = t_{cdCQ} = t_s = t_h = 0$ (zero).
- Non-ideal registers are registers that are not ideal ~~(duh)~~ and exist in real life.
  - At least one of their stats is non-zero.

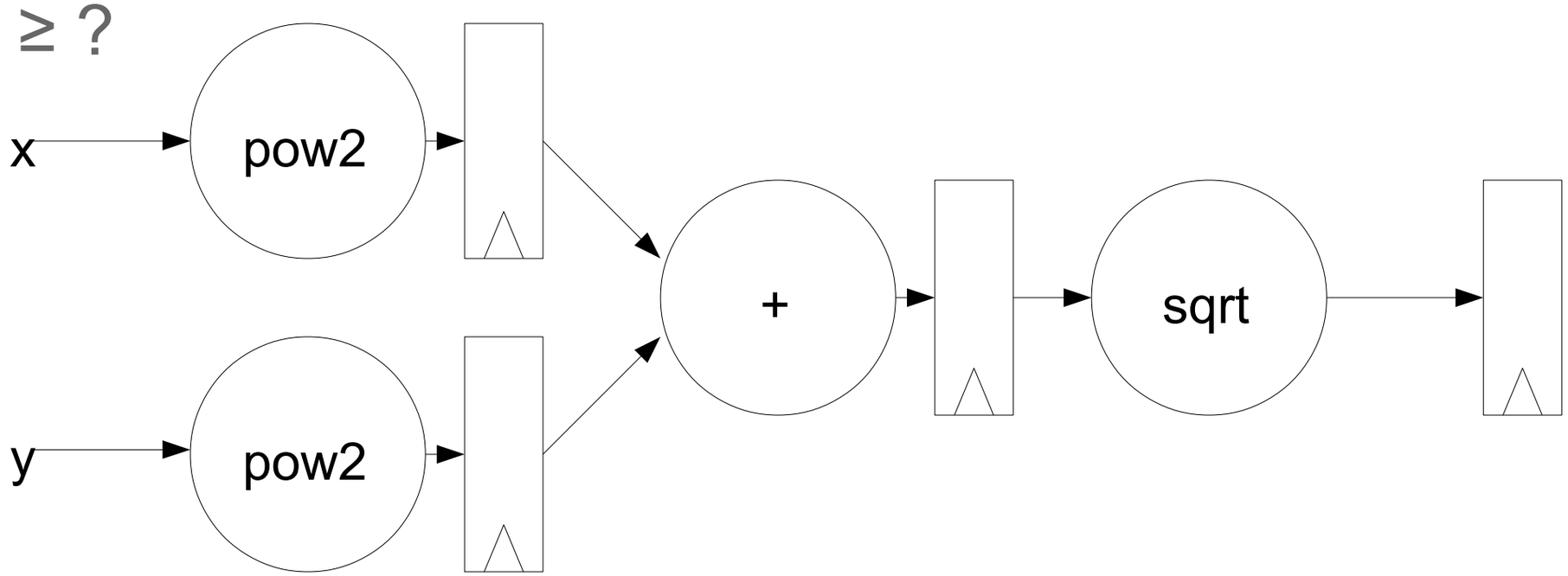
Example: Pythagorean Circuit

- Assume ideal registers.
- Assume that inputs are also coming from registers.
- pow2: t_{pd} = 5 ns, t_{cd} = 3 ns
- adder: t_{pd} = 2 ns, t_{cd} = 1 ns
- sqrt: t_{pd} = 6 ns, t_{cd} = 1 ns
- $t_{clk} \geq$?



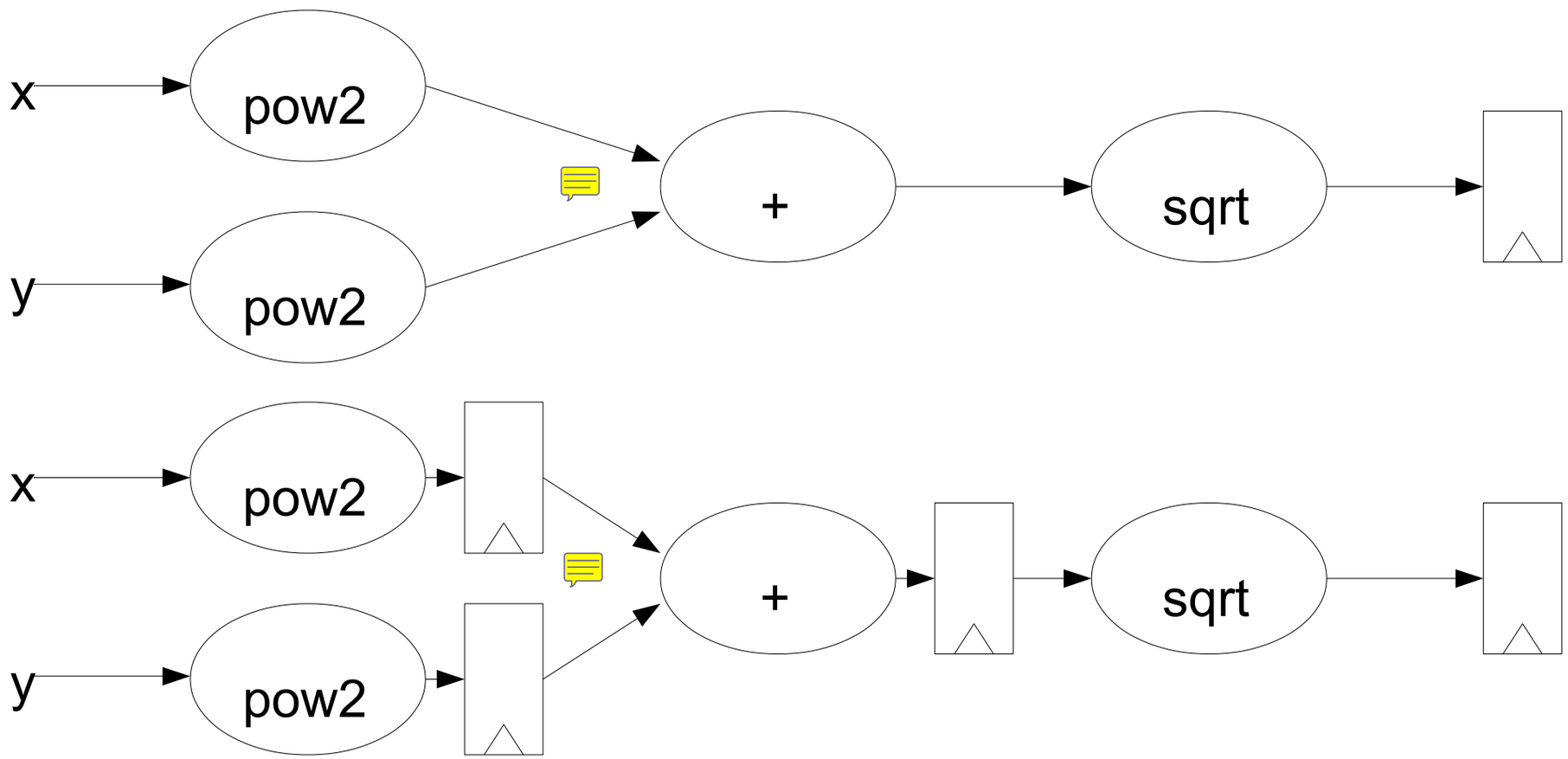
Pipelining

- Pipelining means dividing the circuit into shorter stages, resulting in a lower minimum clock period!
- pow2: t_{pd} = 5 ns, t_{cd} = 3 ns
- adder: t_{pd} = 2 ns, t_{cd} = 1 ns
- sqrt: t_{pd} = 6 ns, t_{cd} = 1 ns
- $t_{clk} \geq$?



Measuring Performance

- Compare the two circuits.
  - When both of them are started up, how long until the first output emerges?
  - After any given output, how long until the next output?

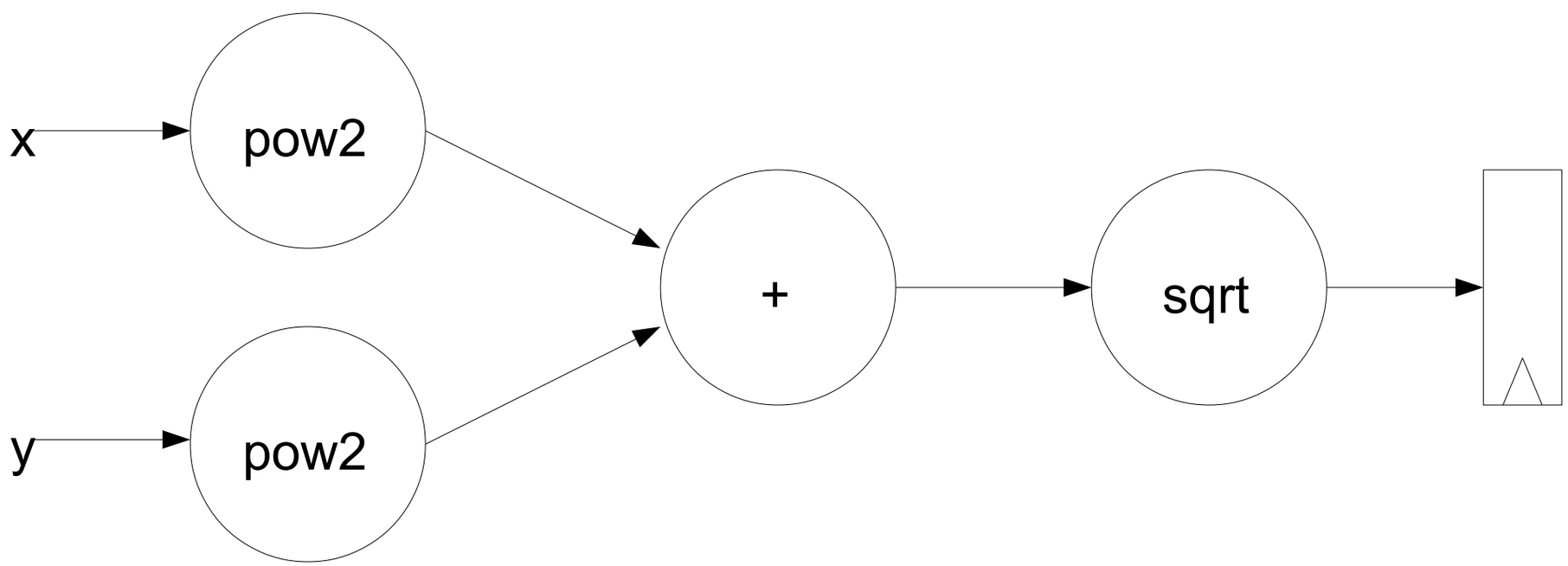


Throughput and Latency

- Throughput = # of outputs / t_{clk_min}
  - # of outputs is usually 1 (one).
  - This represents the rate at which outputs emerge once all stages in the system have data.
    - In other words, once the pipeline is full.
- Latency = # of stages * t_{clk_min}
  - # of stages is 1 (one) in an unpipelined system.
  - This represents the time for the system to generate an output for a given set of inputs.

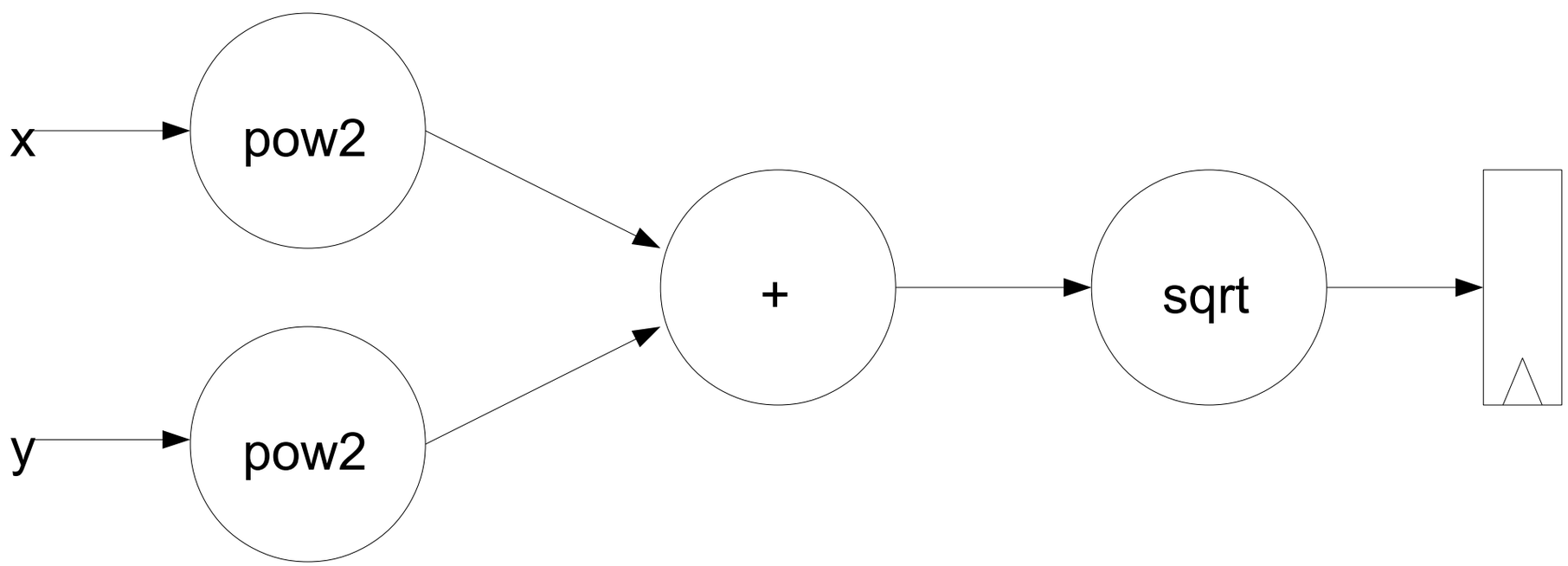
Unpipelined

- $t_{clk} \geq 13$ ns
- Throughput = ?
- Latency = ?



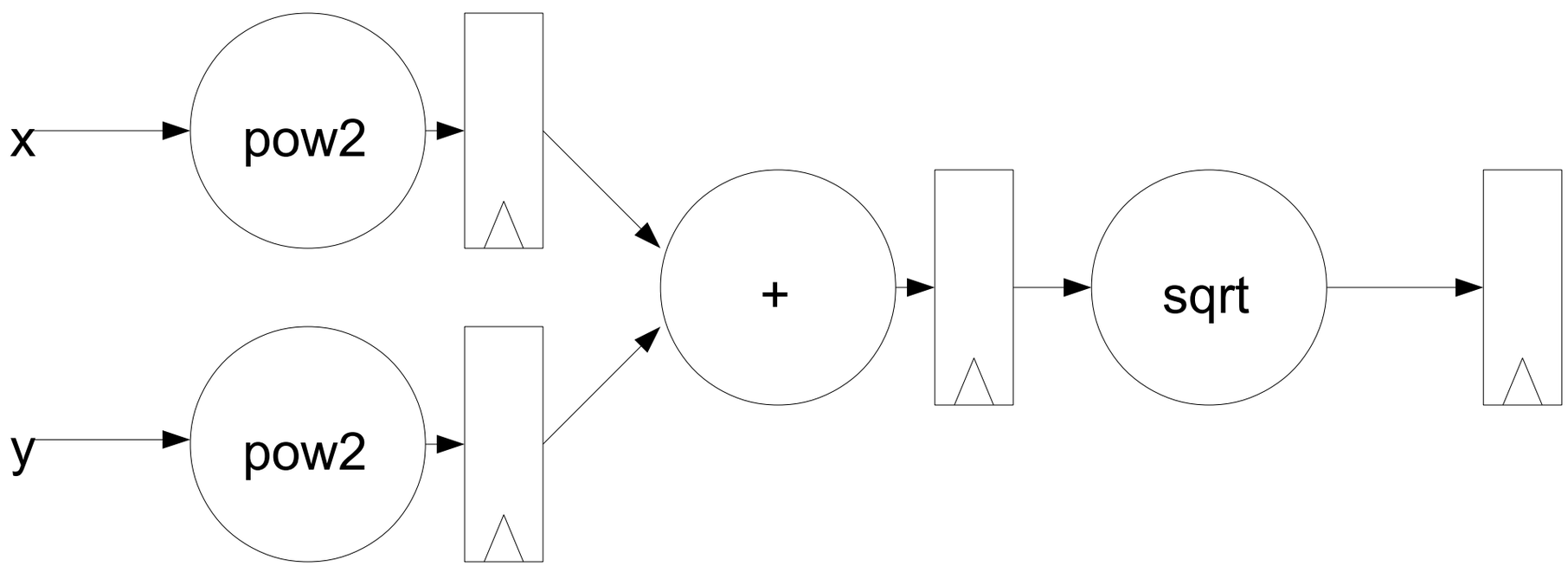
Unpipelined

- $t_{clk} \geq 13$ ns
- Throughput = 1 / t_{clk} = 1 / 13 ns = 77 MHz
- Latency = 1 * t_{clk} = 1 * 13 ns = 13 ns



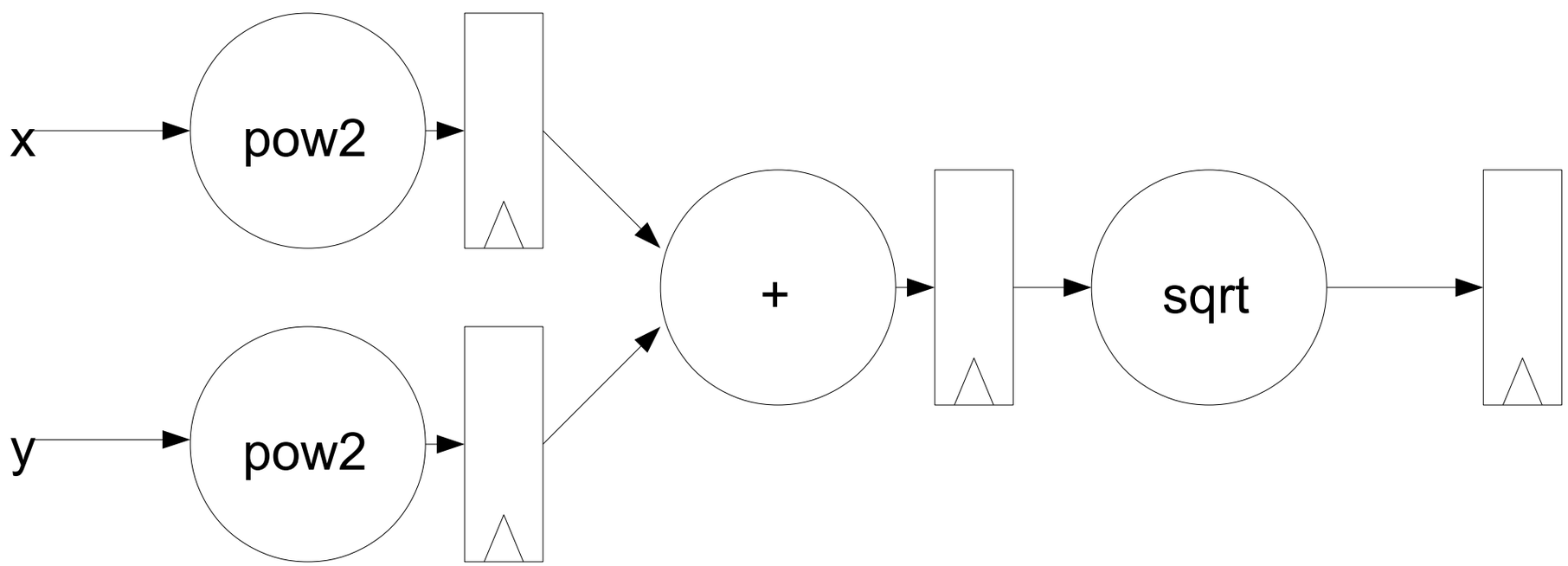
Pipelined

- $t_{clk} \geq 6$ ns
- Throughput = ?
- Latency = ?



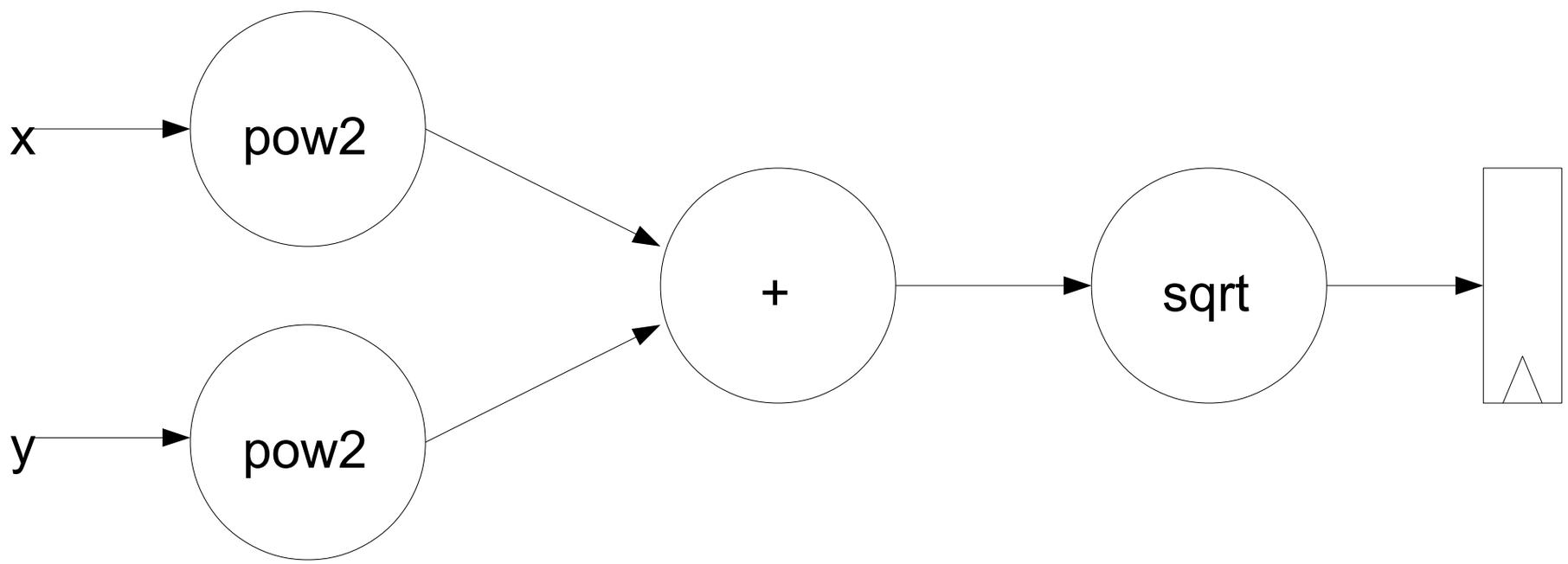
Pipelined

- $t_{clk} \geq 6$ ns
- Throughput = 1 / t_{clk} = 1 / 6 ns = 166 MHz
- Latency = 3 * t_{clk} = 3 * 6 ns = 18 ns



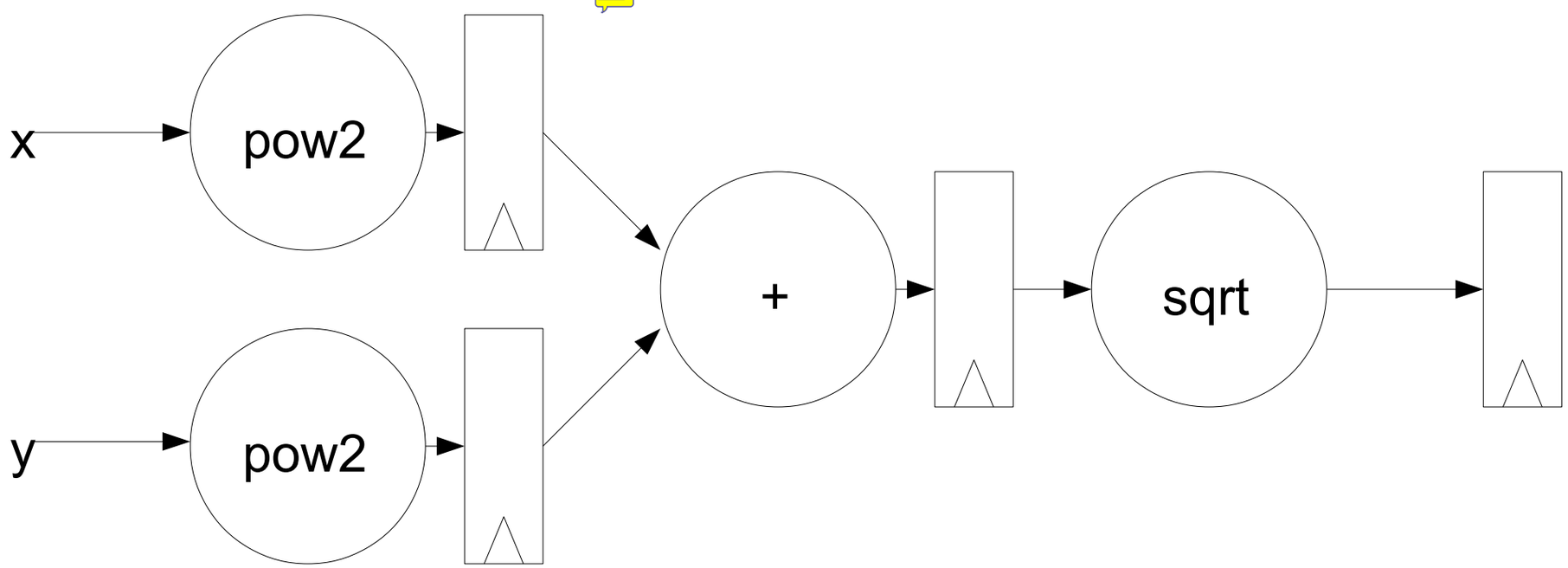
What About Non-Ideal Registers?

- t_{pdCQ} = 3 ns, t_{cdCQ} = 1 ns, t_s = 2 ns, t_h = 1 ns
- pow2: t_{pd} = 5 ns, t_{cd} = 3 ns
- adder: t_{pd} = 2 ns, t_{cd} = 1 ns
- sqrt: t_{pd} = 6 ns, t_{cd} = 1 ns
- $t_{clk} \geq$?, Throughput = ?, Latency = ?



What About Non-Ideal Registers?

- t_{pdCQ} = 3 ns, t_{cdCQ} = 1 ns, t_s = 2 ns, t_h = 1 ns
- pow2: t_{pd} = 5 ns, t_{cd} = 3 ns
- adder: t_{pd} = 2 ns, t_{cd} = 1 ns
- sqrt: t_{pd} = 6 ns, t_{cd} = 1 ns
- $t_{clk} \geq$?, Throughput = ?, Latency = ?

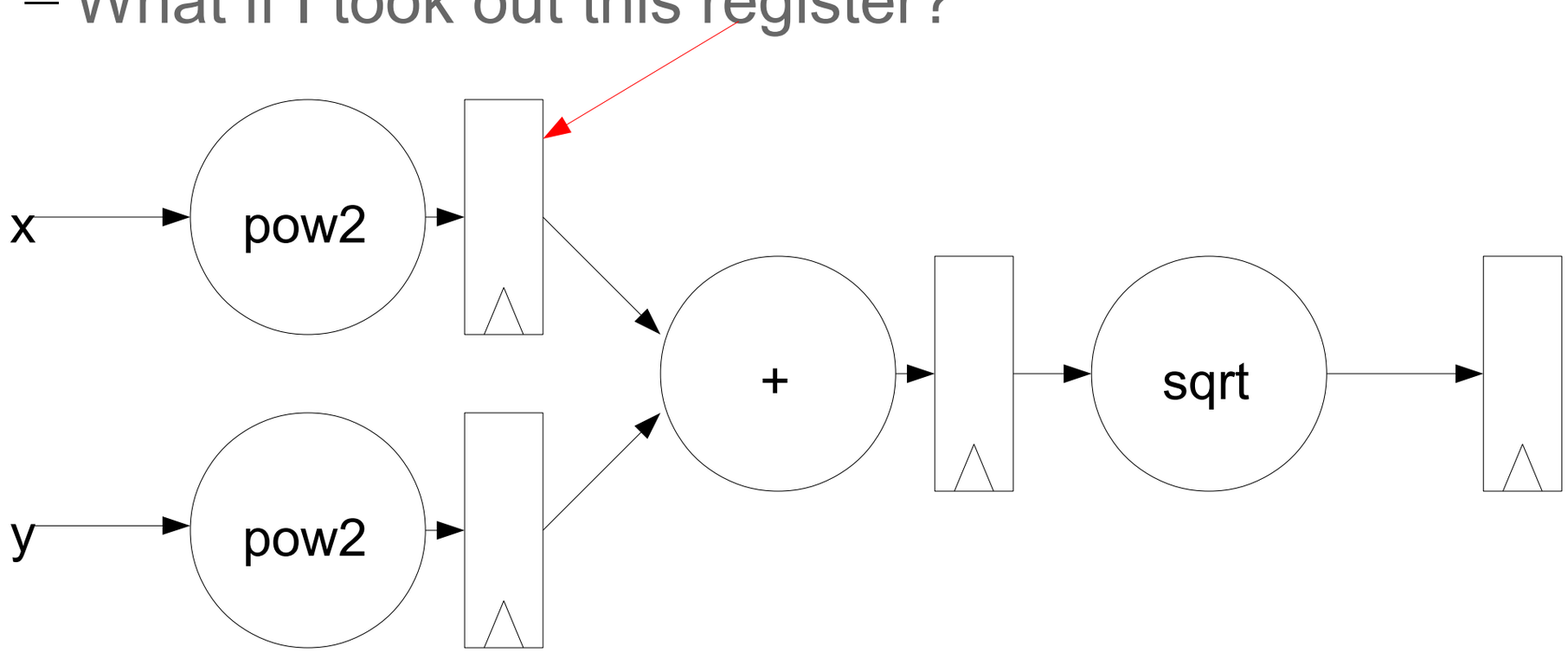


Unpipelined VS Pipelined

- In general, pipelining will:
  - Increase throughput…
  - … at expense of slightly longer latency per object.
- Is this a good tradeoff?
  - Will the circuit most likely encounter several inputs in a row?
  - How many lines of code does an average program have?

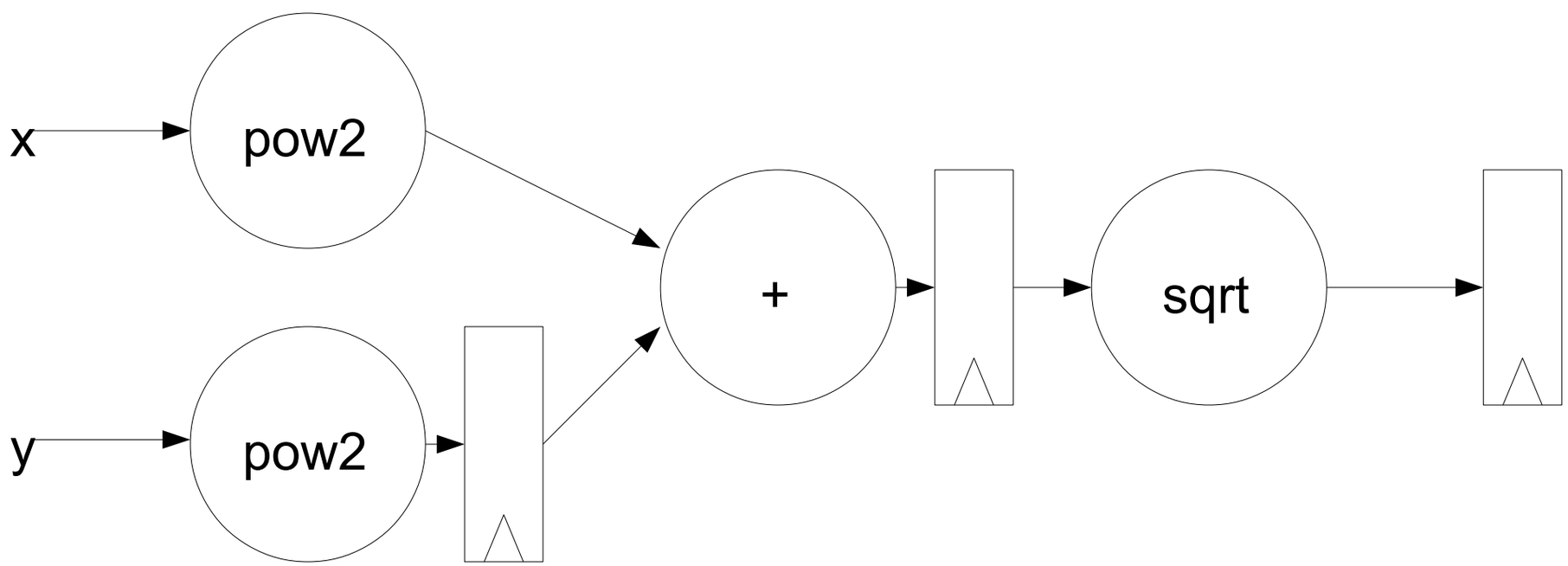
Pipelining Guideline #1

- The circuit should contain the same number of registers along any path from any input to any output.
  - This insures that every computational unit in the circuit sees inputs in phase!
  - What if I took out this register?



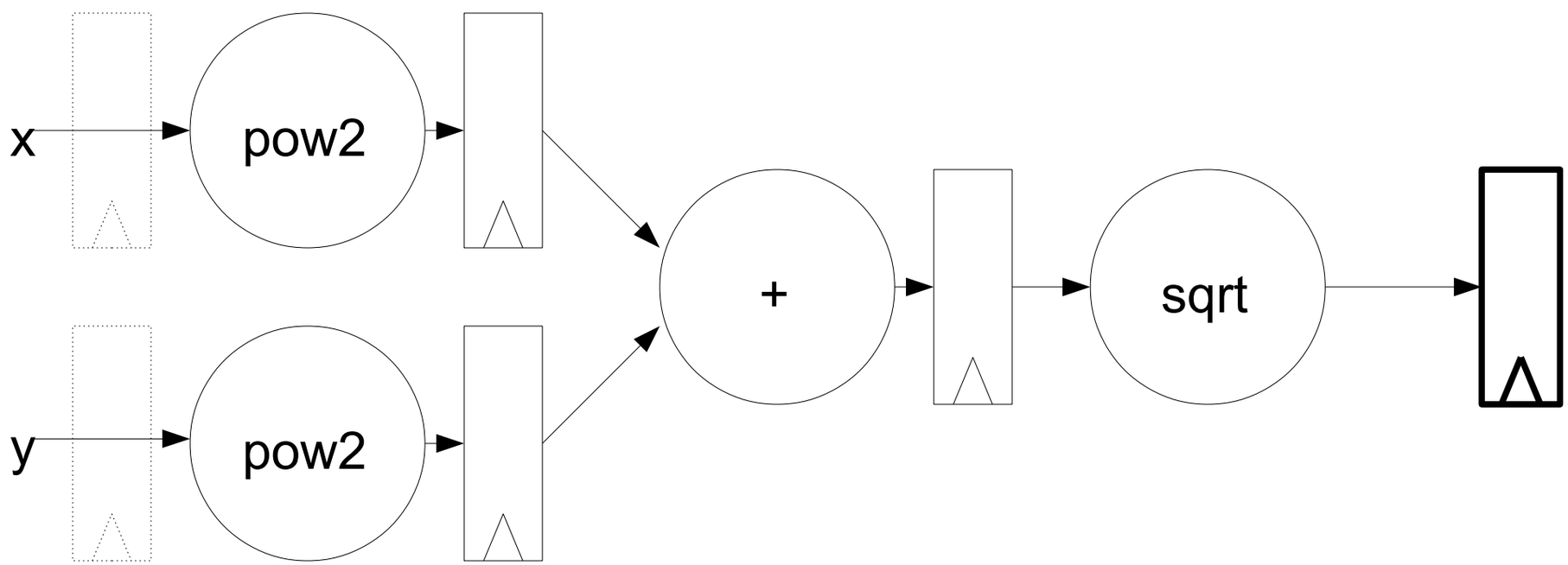
Ill-formed Pipelines

- Adder will get x and y from different pairs, resulting in an incorrect calculation!
  - x and y pairs become unsynchronized due to the unequal number of registers in the paths leading to the adder.



Pipelining Guideline #2

- There must ALWAYS be a register on each output.
- Inputs are assumed to be coming from registers (perhaps from another pipelined circuit) and therefore must not be drawn.



Pipelining Guideline #3

- t_{pdCQ} and t_s are constant.
  - This is the <u>overhead</u> caused by FFs that CANNOT be eliminated (except by removing the FFs) and will therefore limit throughput!
  - Even if we increase the number of stages, we cannot bring t_{clk_min} lower than $t_{pdCQ} + t_s$.
  - We can however, still increase throughput. How?
    - Anyone here know what SMP means?
    - For now, we will assume that this solution is unavailable to us due to cost issues.

Pipelining Guideline #4

- When pipelining a circuit, these are your priorities:
  - First, *maximize throughput!* (minimize clock period)
  - Then, *minimize number of stages for that max throughput!* (minimize latency)
  - Finally, *minimize number of registers for a given throughput and latency!* (minimize cost)

Pipelining Guideline #4

- Given a circuit, throughput is usually limited by a device that has the longest t_{pd}.
  - So find that first and isolate it using registers!
  - Don't forget that there are already registers present in the outermost areas of the circuit!
    - Where input/s come from and where output/s emerge!
- Once you've done that, you can divide the rest of circuit into stages, using the isolated device/s' t_{pd} as a guide.
  - Example / exercise time!