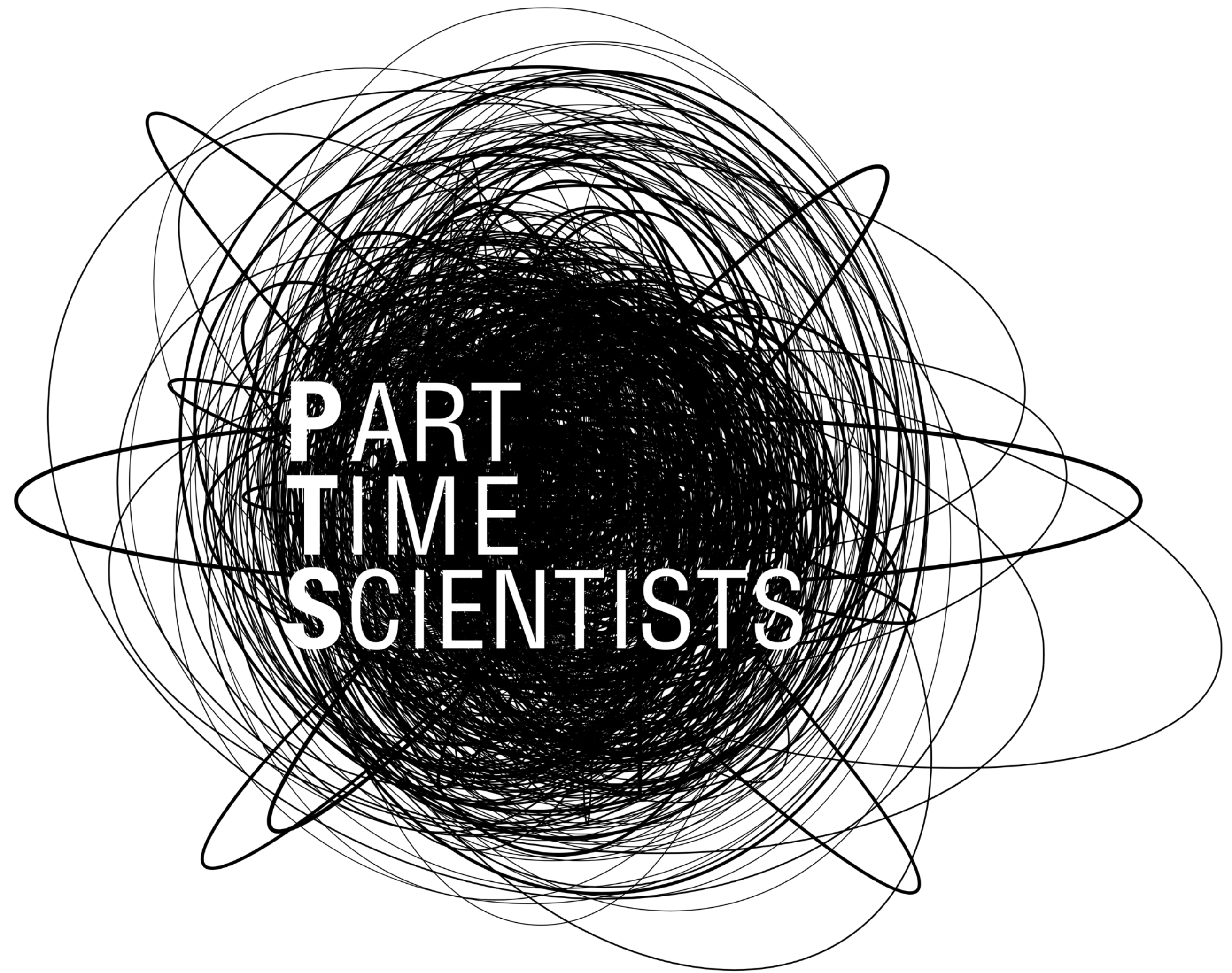


FPGA 101

Field programmable gate arrays in action

About me



- Karsten Becker
- Head of electronics @Part-Time Scientists
- PhD candidate @TUHH
 - FPGA Architecture

What is an FPGA

- Programmable Logic

Programmable logic (LUT)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Programmable logic (LUT)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

X	Y	C	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Programmable logic (LUT)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

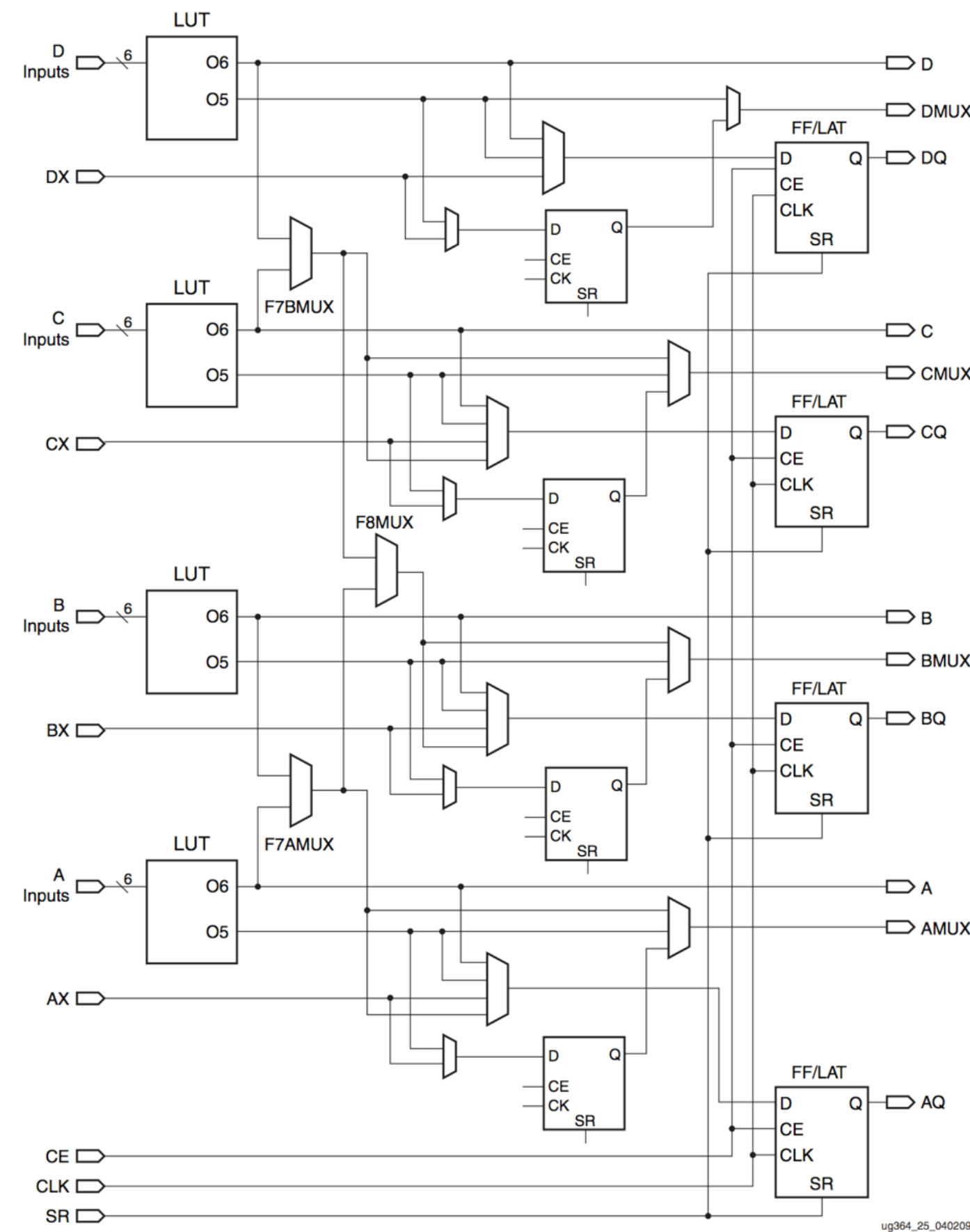
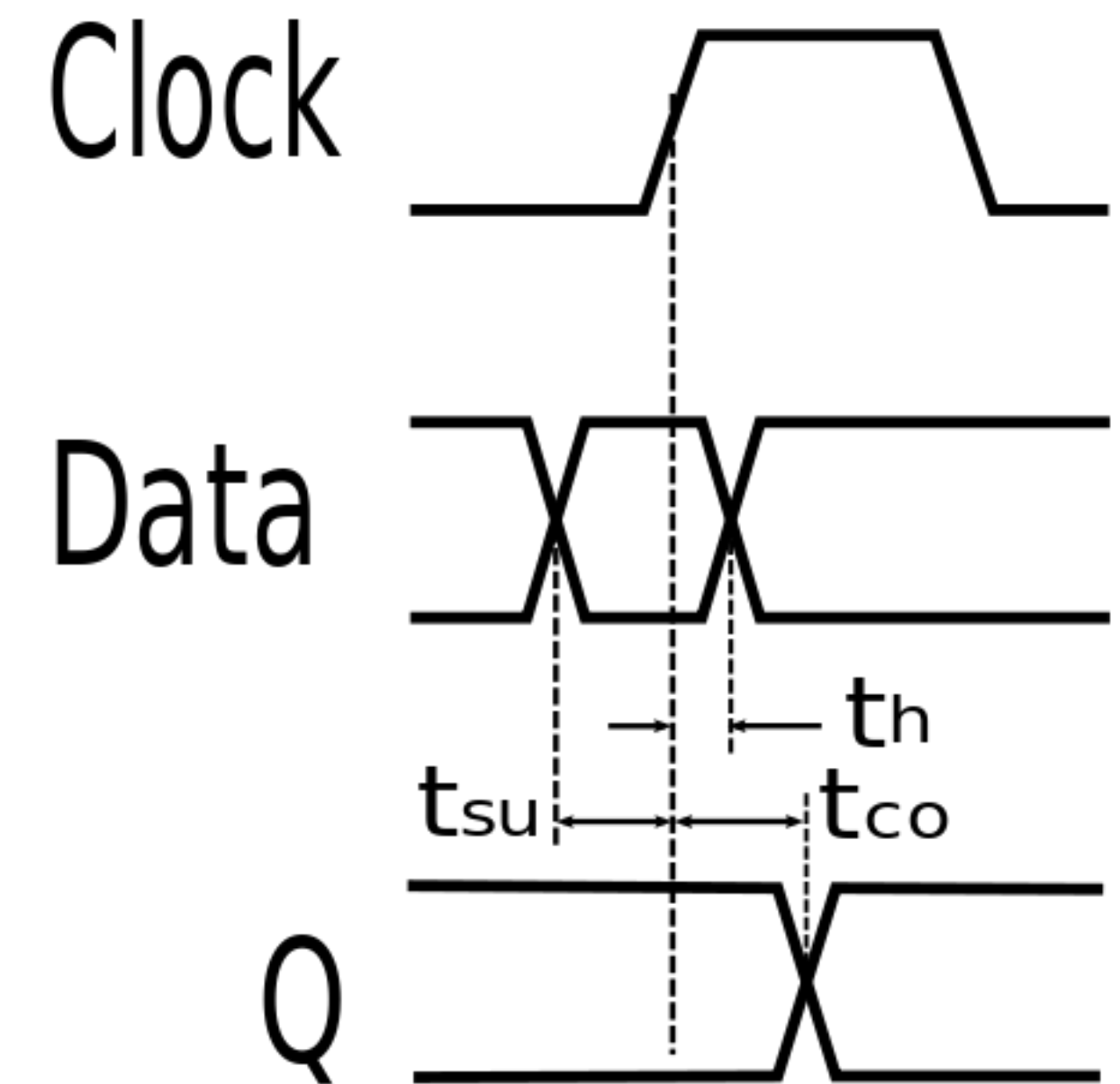
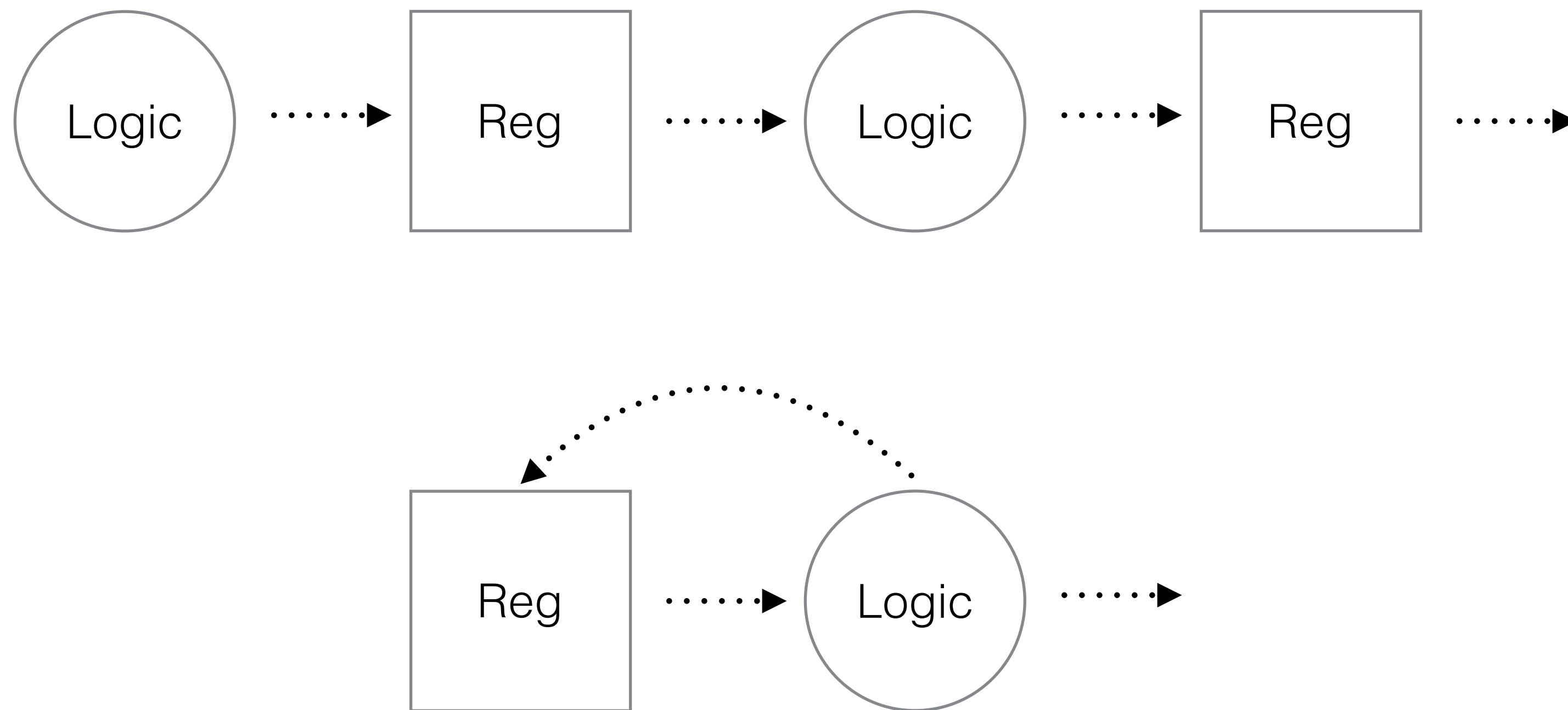


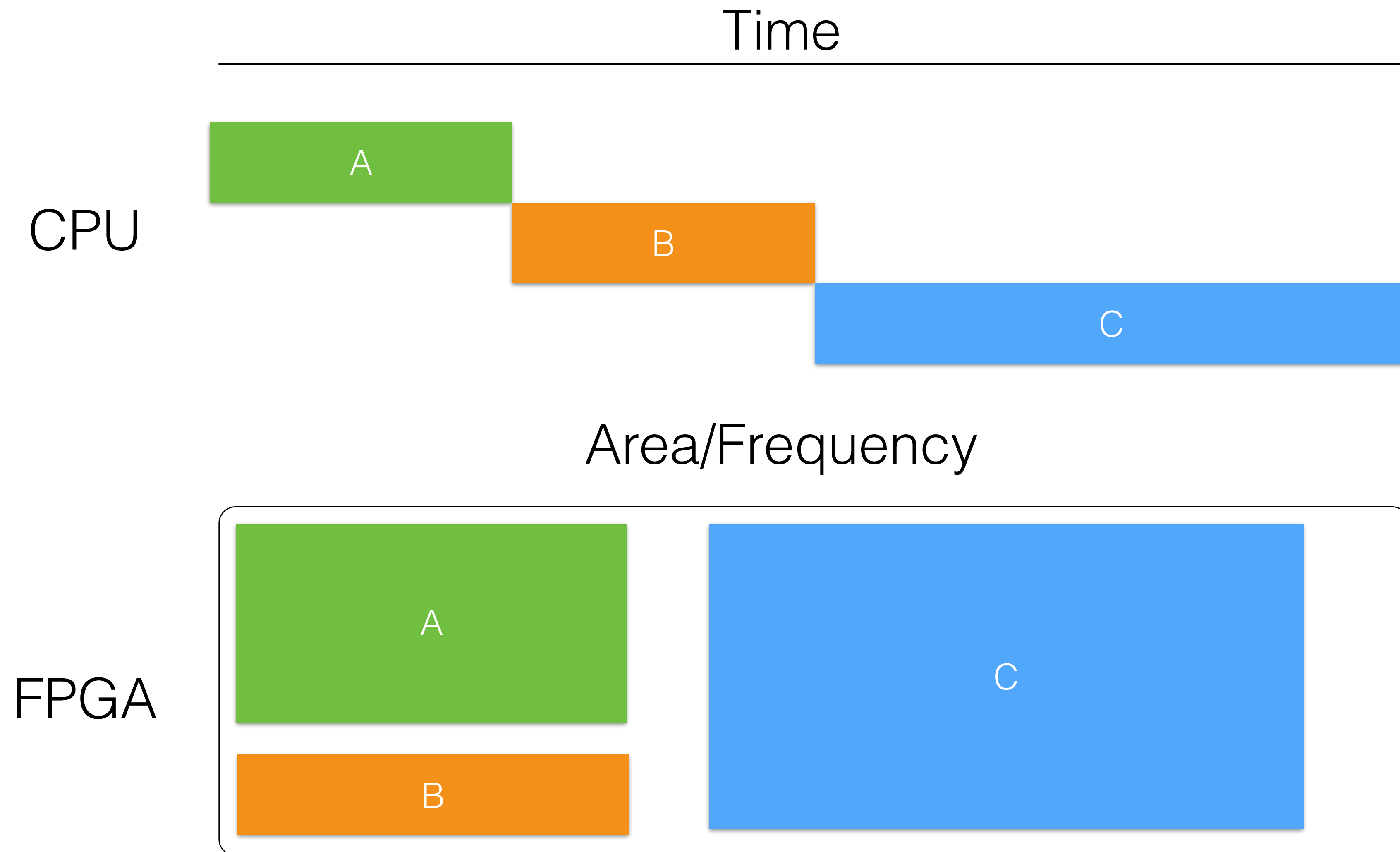
Figure 25: Simplified Virtex-6 FPGA Slice

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Registers



FPGA vs CPU



Fabric

- Composed of
 - Lookup tables
 - Flip-flops
 - Multiplexer
 - Routing resources
 - Clock Management
 - Carry Logic
 - Shift-register
- Hard blocks
 - DSP Units
 - Multiplier
 - Block RAM
 - High-speed IO
 - CPUs
 - Memory controller
 - ADC/DACs

Cool FPGA Projects

- Borgventilator
 - Pixel transformation to polar coordinates
 - Transmission as 8b10 coding
 - Gamma correction
 - Precise timing
 - Go to: Das Labor (Hackerspace) somewhere here



Cool FPGA Projects

- HDMI Overlay
 - Detect HDMI Sync signal
 - Encrypt Overlay
 - Select overlay stream vs content stream
 - Does not decrypt content stream!
 - See bunny @28C3

Programming

High Level tools

Matlab Simulink, C to HDL generators,
OpenCL to HDL generators

Code generation

Hardware Description Language

VHDL, Verilog, MyHDL, PSHDL

Synthesis/Mapping

Netlist

Place&Route

Configuration file

Code Example (counter)

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity MyFirstModule is
    port (
        clk : in std_logic;
        rst : in std_logic
    );
end;
architecture pshdlGenerated of MyFirstModule is
    signal counter : unsigned(11 downto 0);
begin
    process(clk)
    begin
        if RISING_EDGE(clk) then
            if rst = '1' then
                counter <= (others => '0');
            else
                counter <= (counter + 1);
            end if;
        end if;
    end process;
end;
```

PSHDL

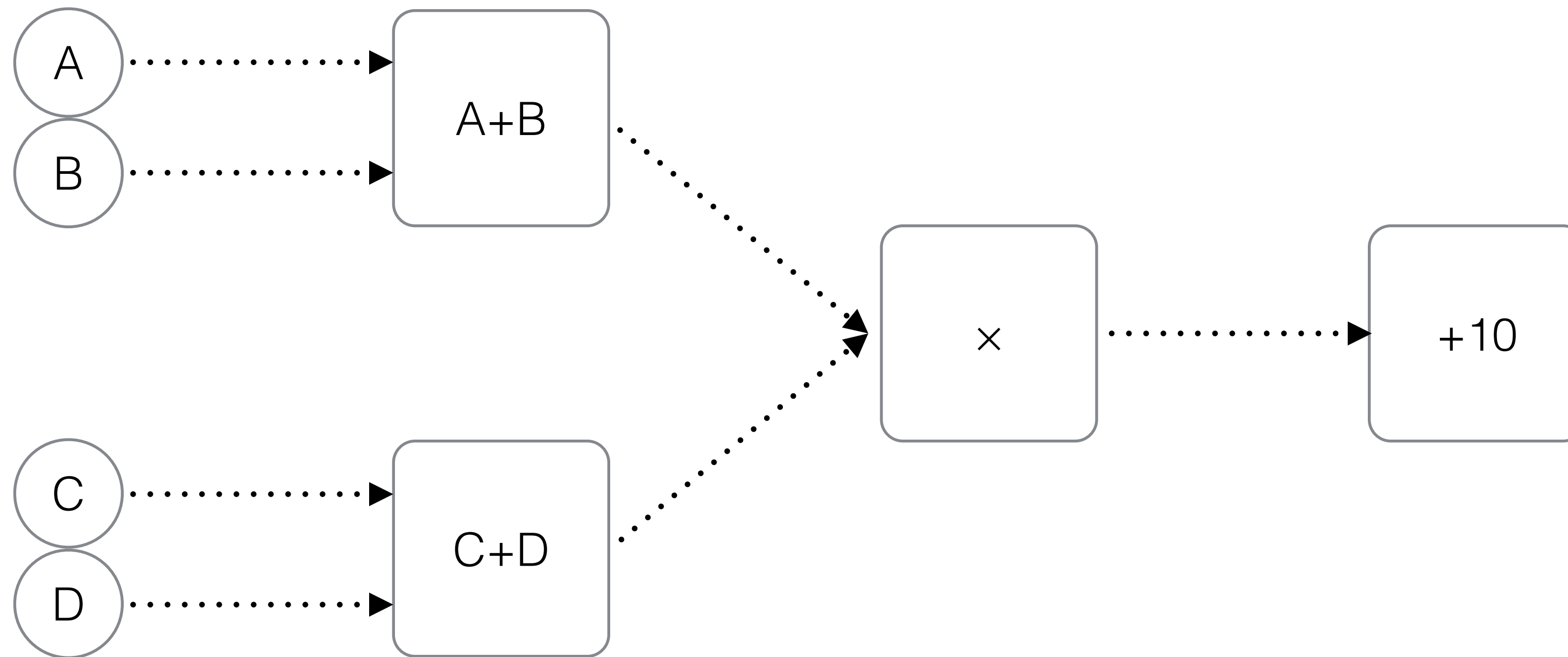
```
module MyFirstModule{
    register uint<12> counter=counter+1;
}
```

Verilog

```
module counter (
    out,
    clk,
    reset
);
    output [7:0] out;
    input clk, reset;
    reg [7:0] out;
    always @(posedge clk)
    if (reset) begin
        out <= 8'b0 ;
    end else begin
        out <= out - 1;
    end
endmodule
```

Pipelining

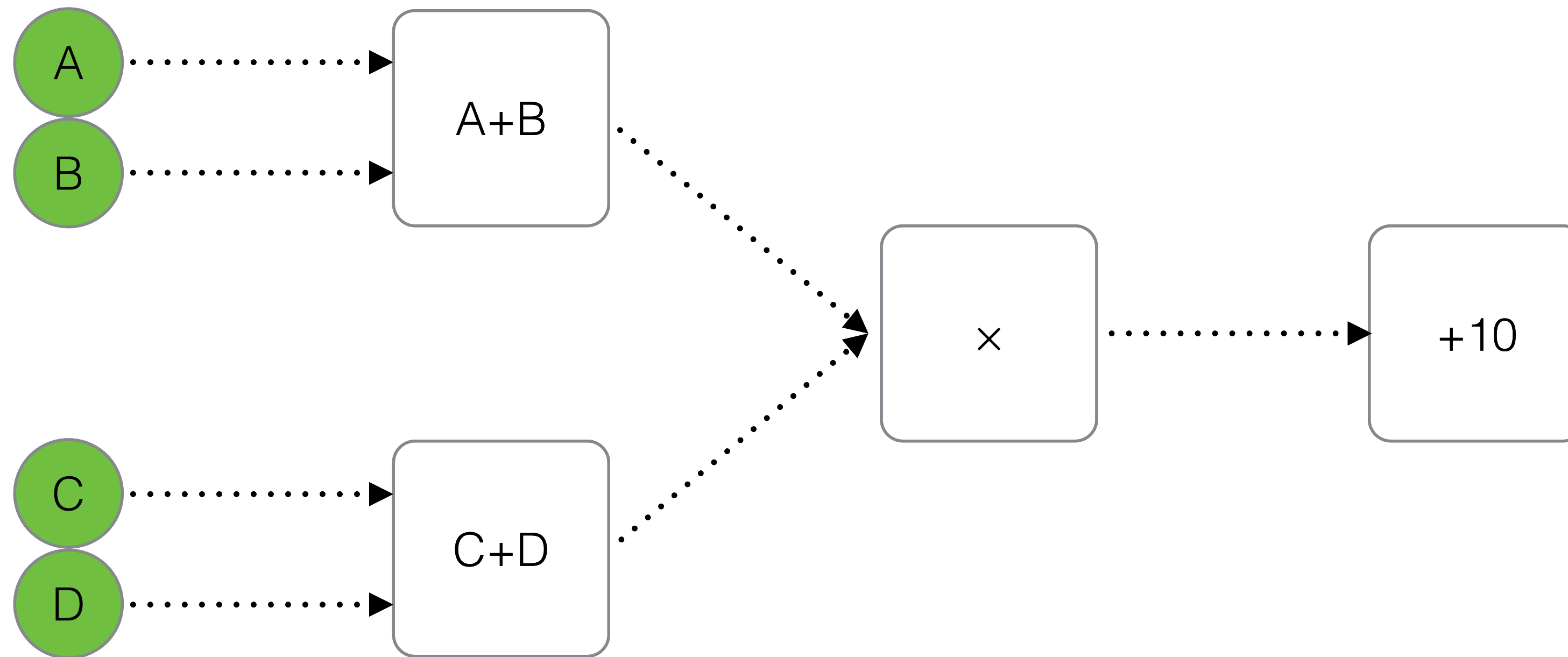
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

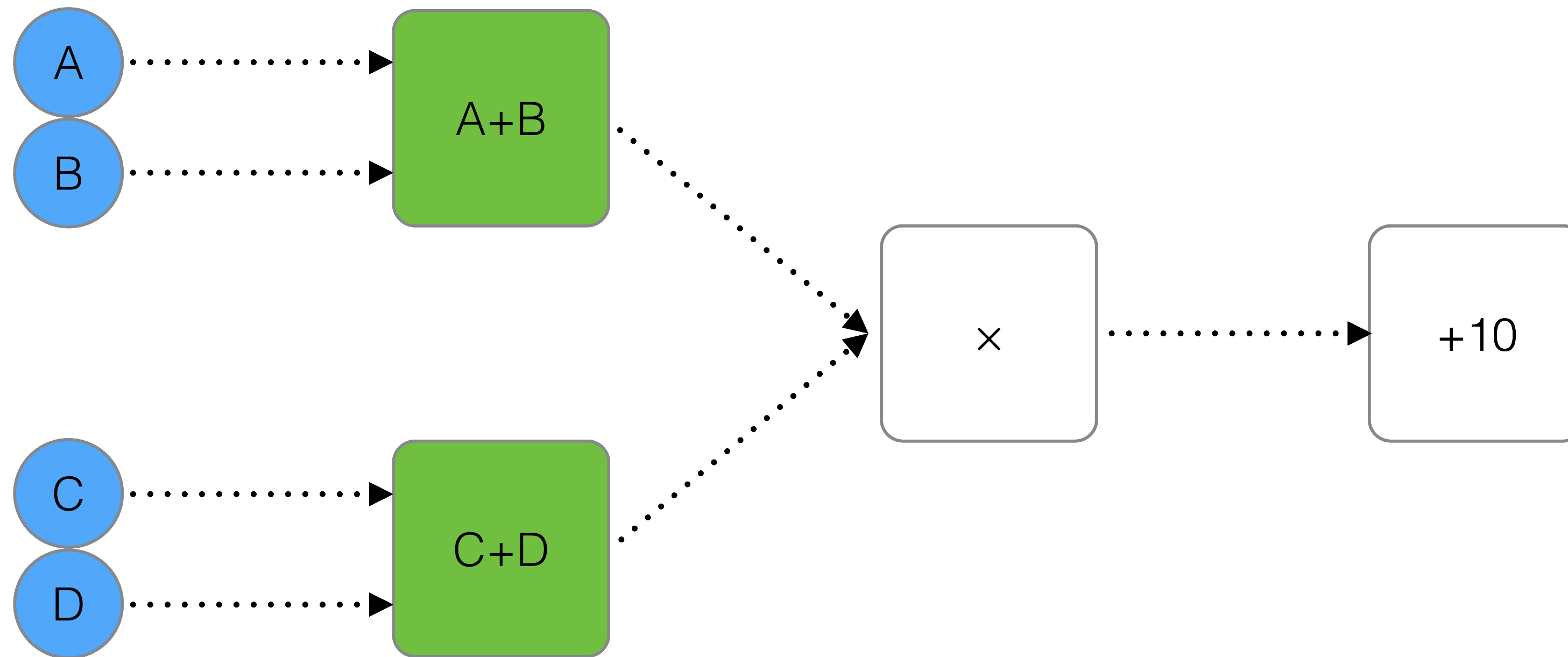
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

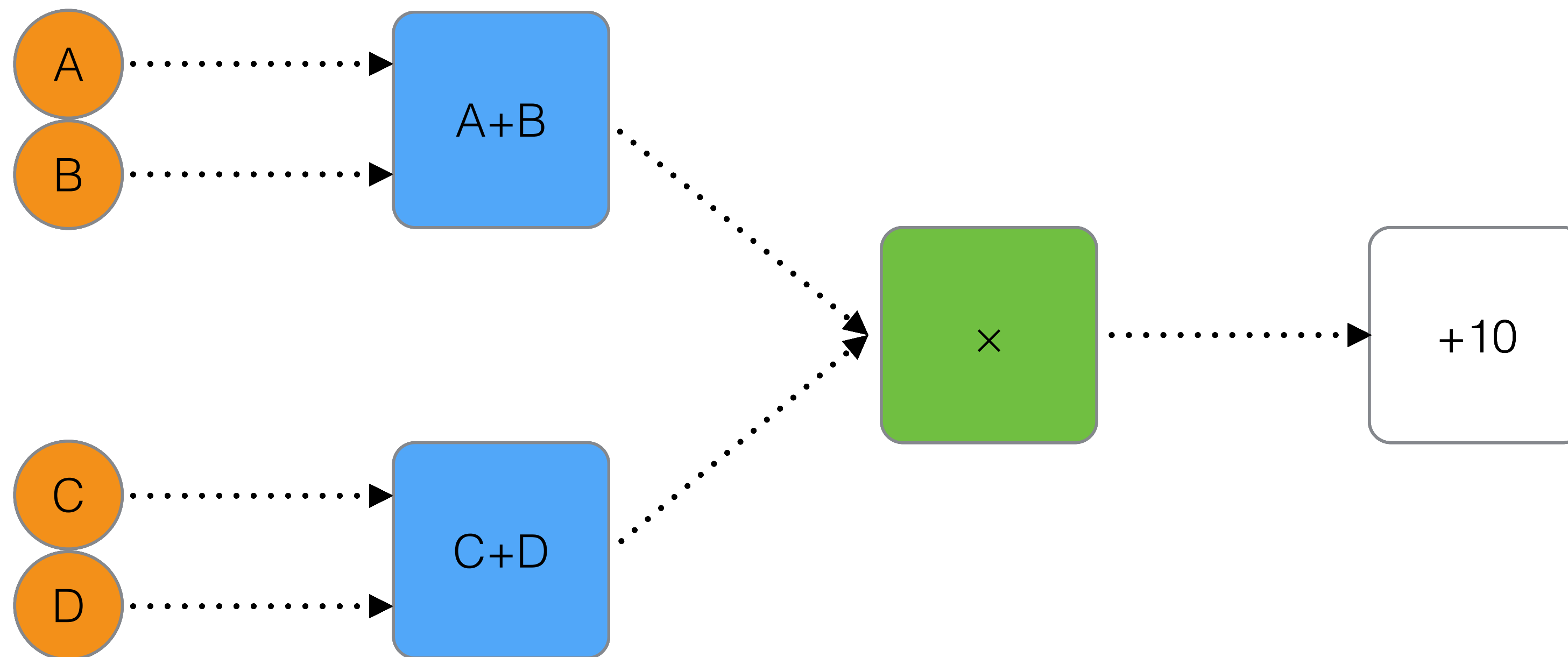
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

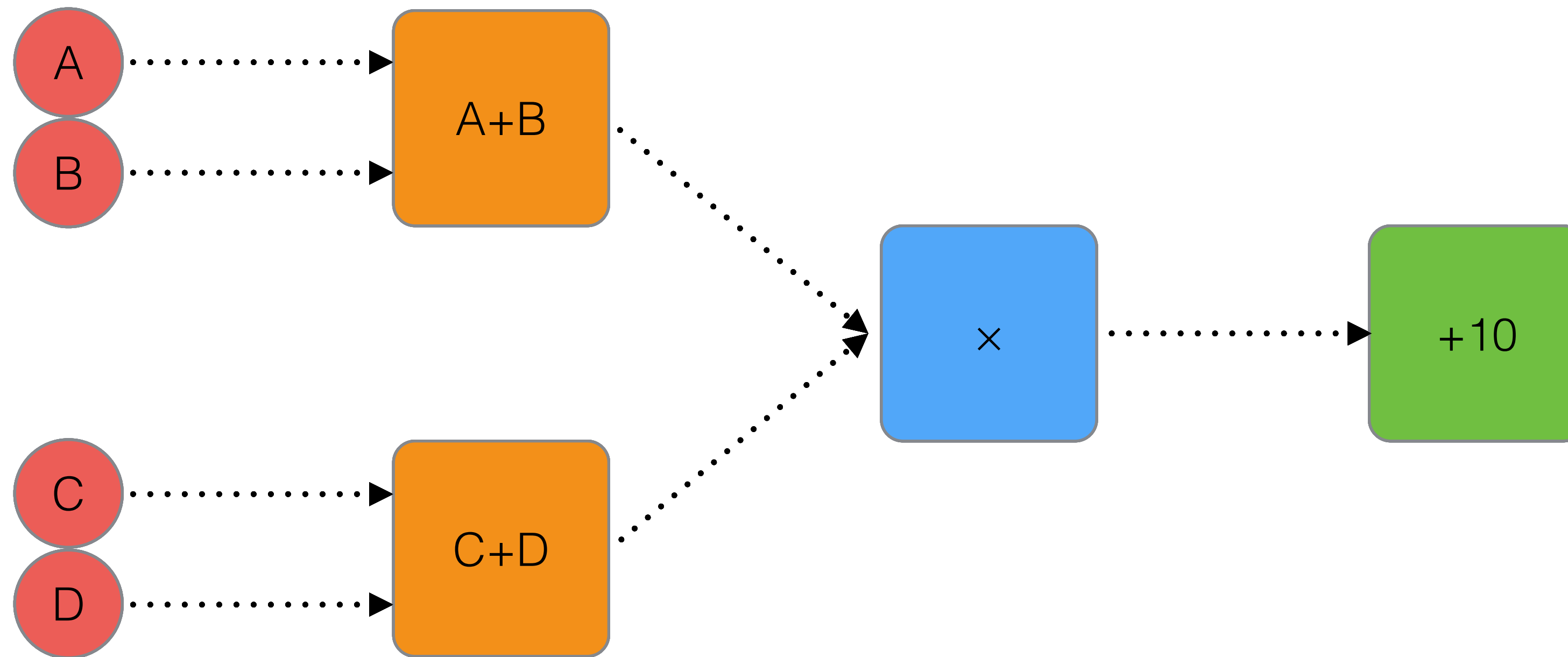
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

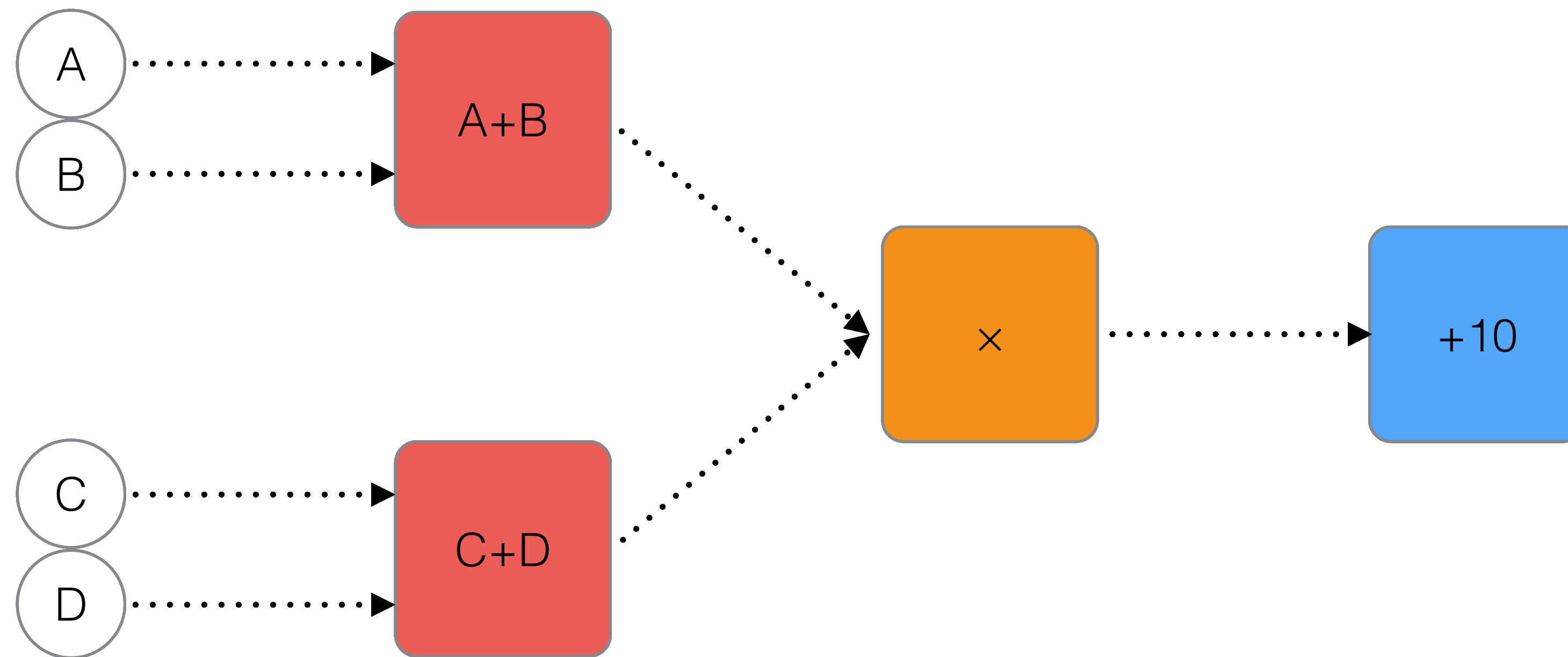
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

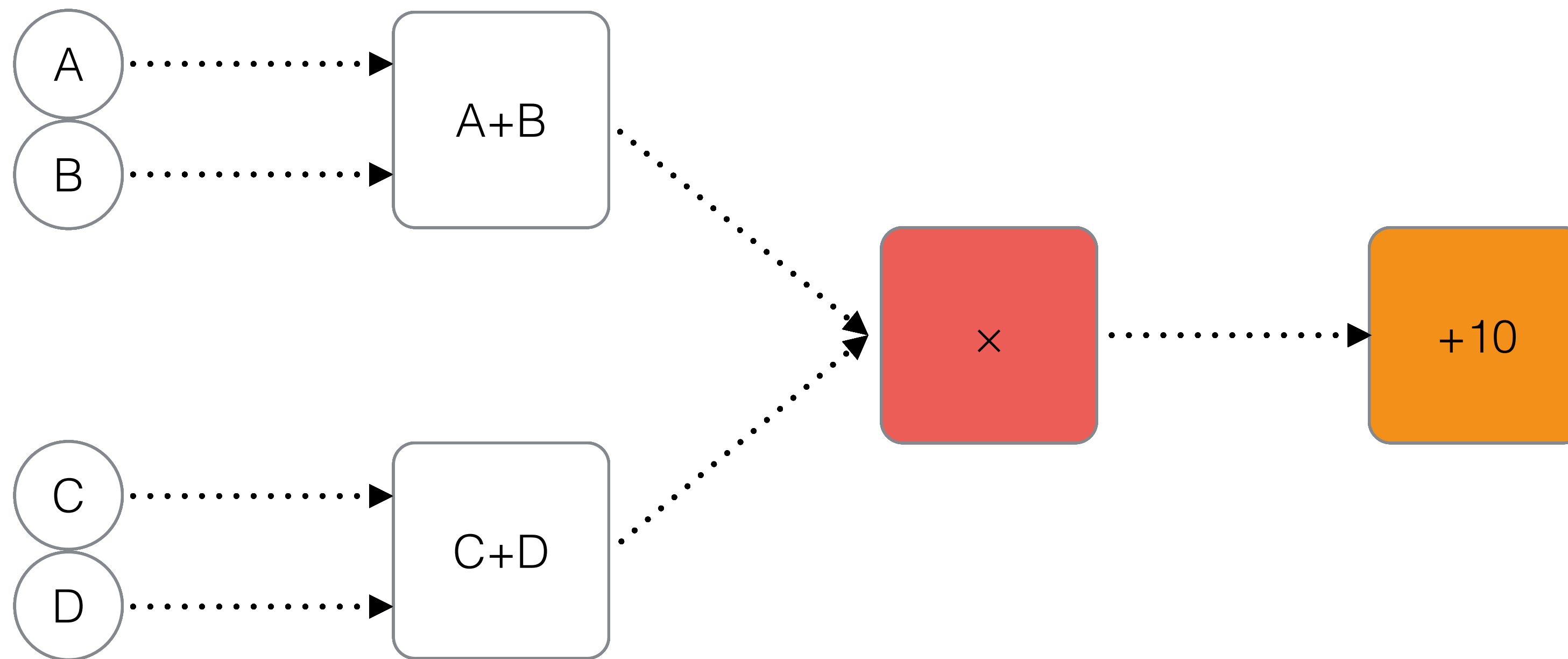
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

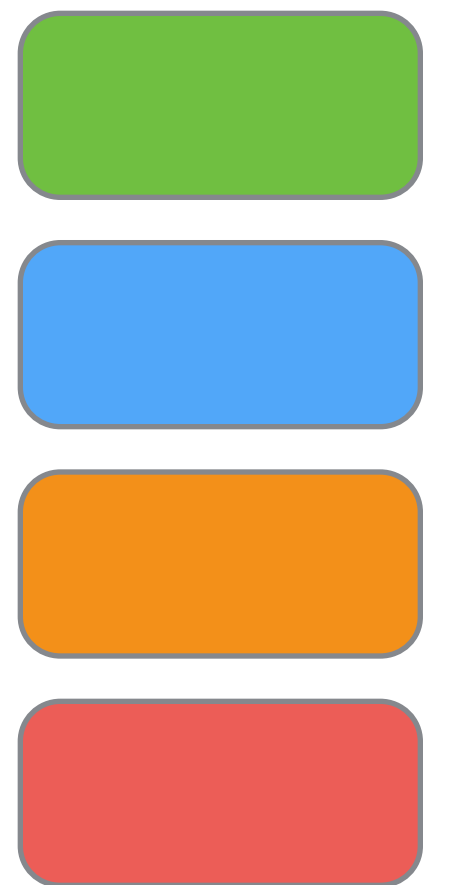
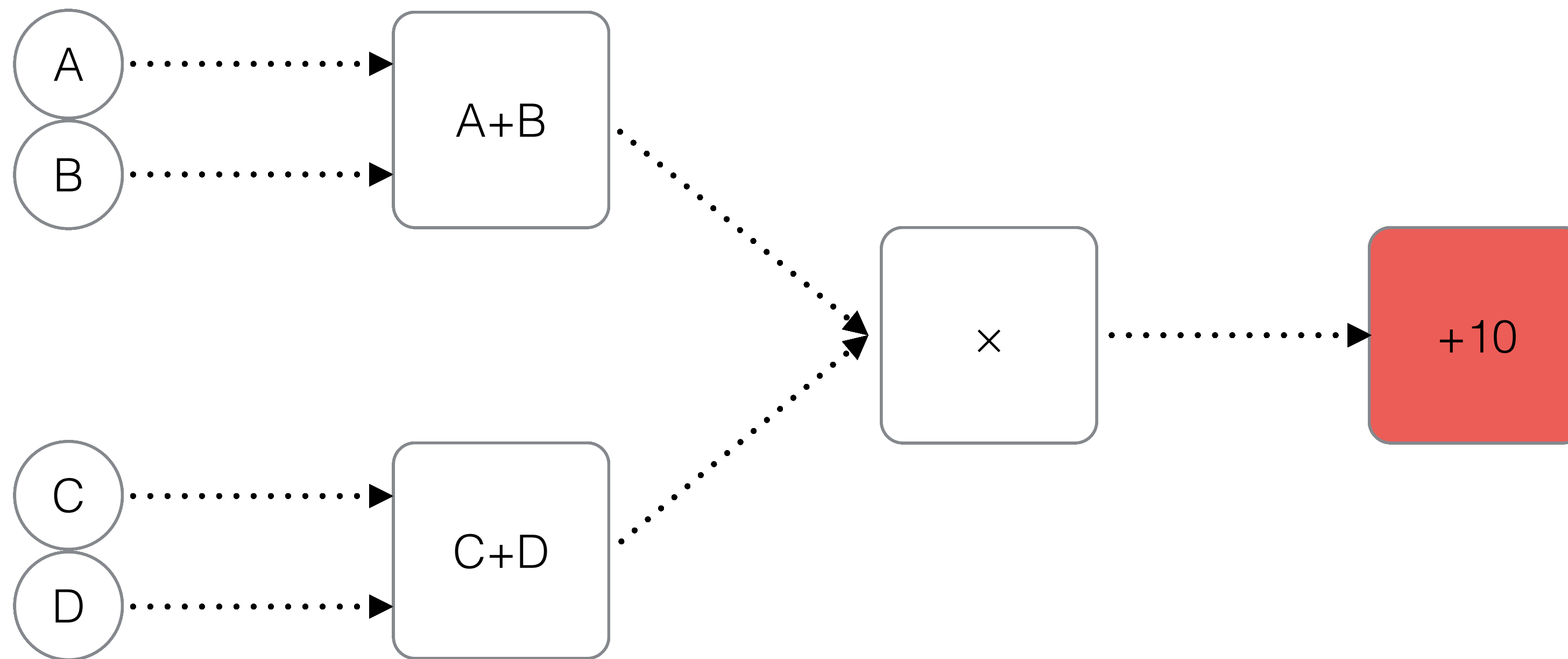
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

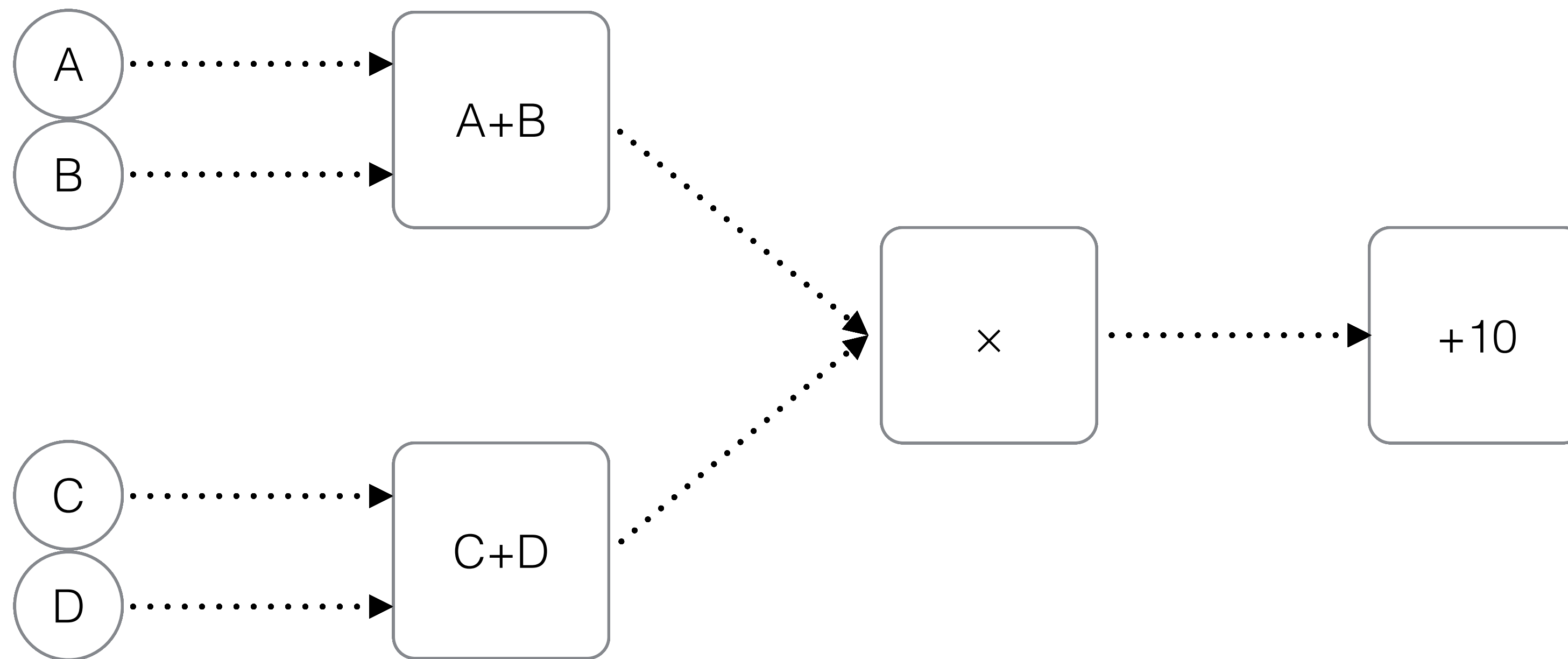
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

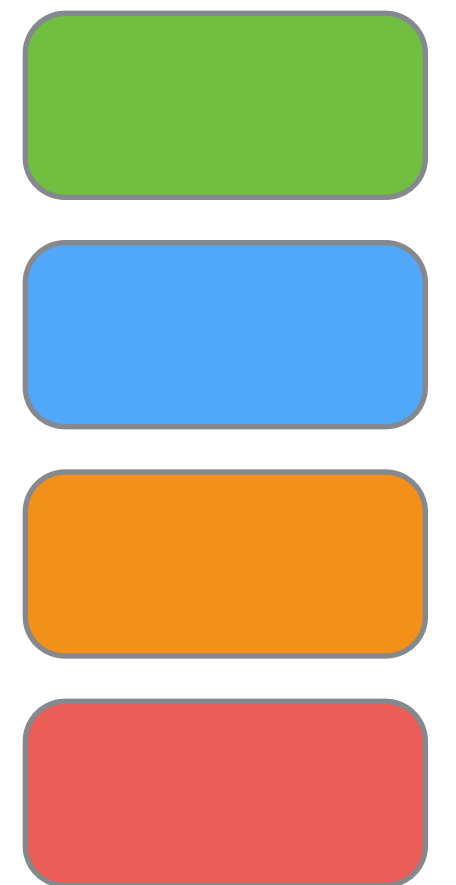
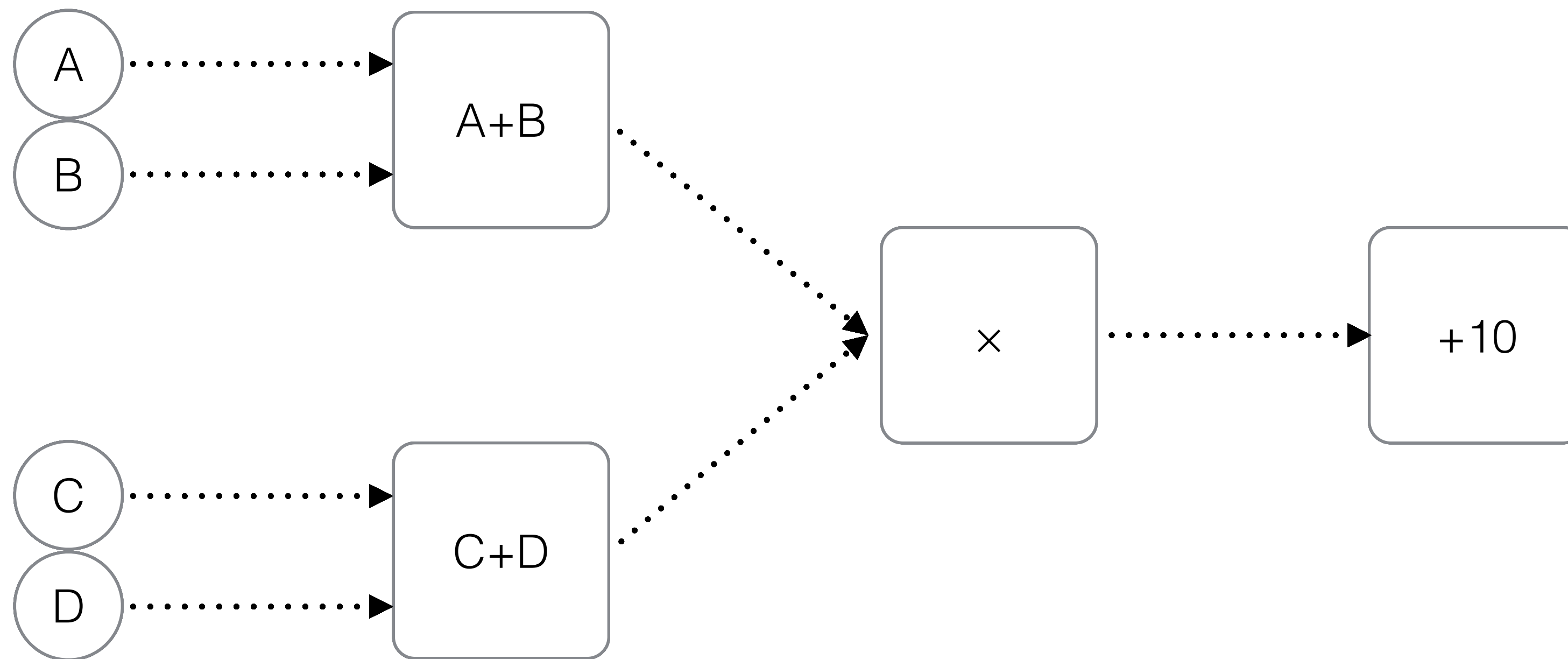
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

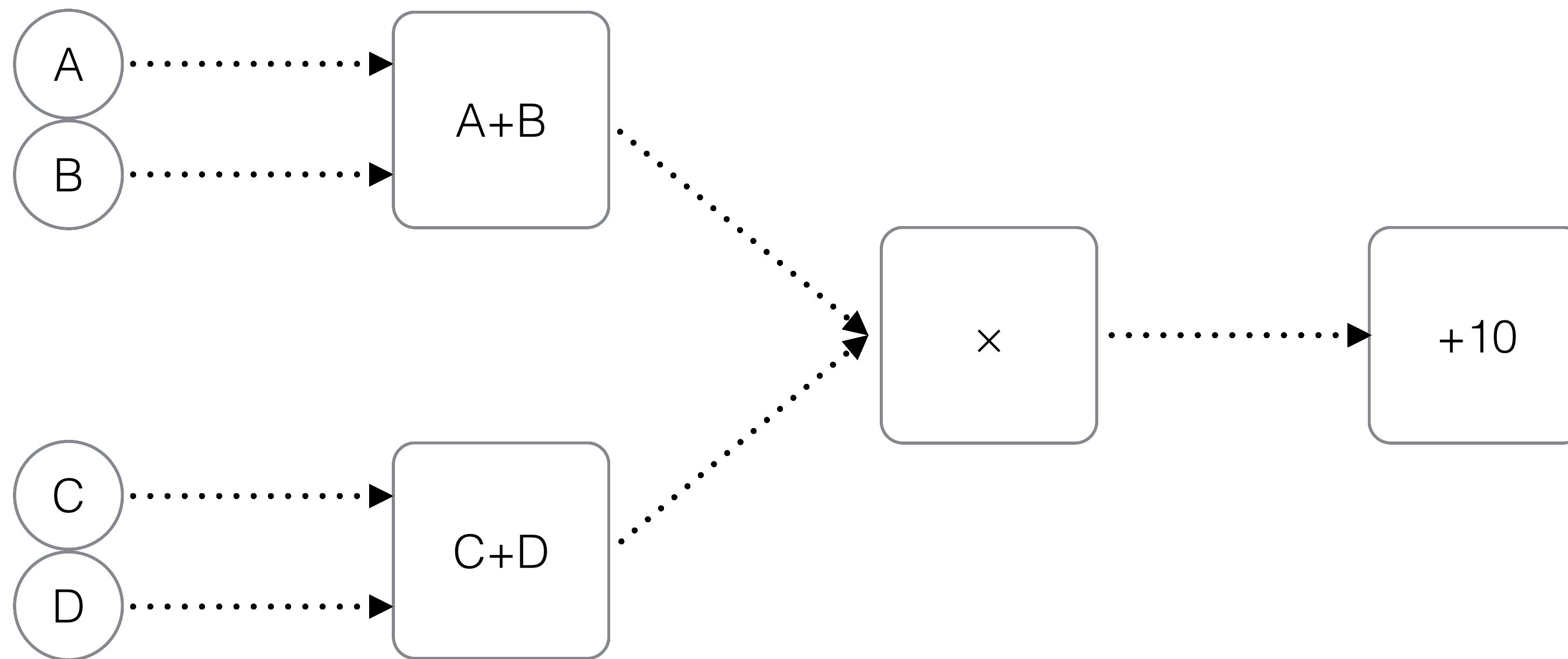
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

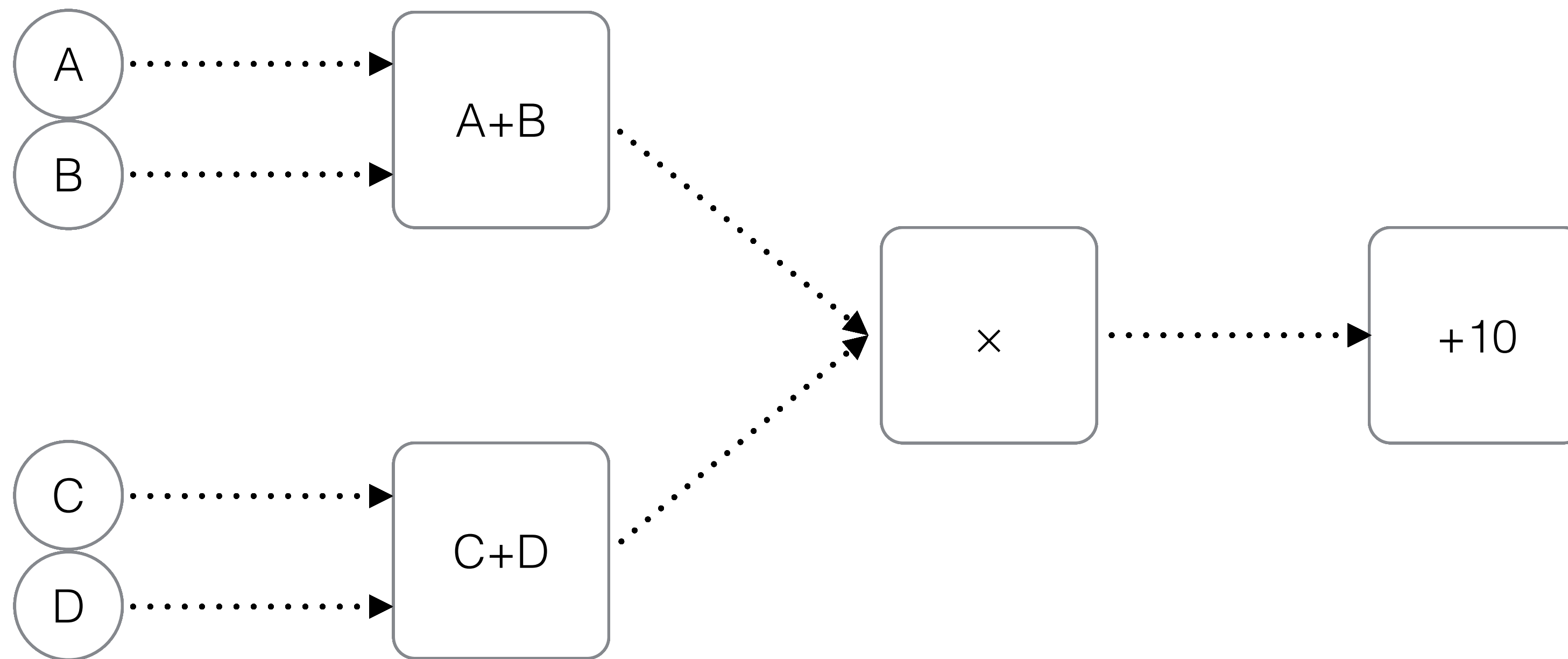
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

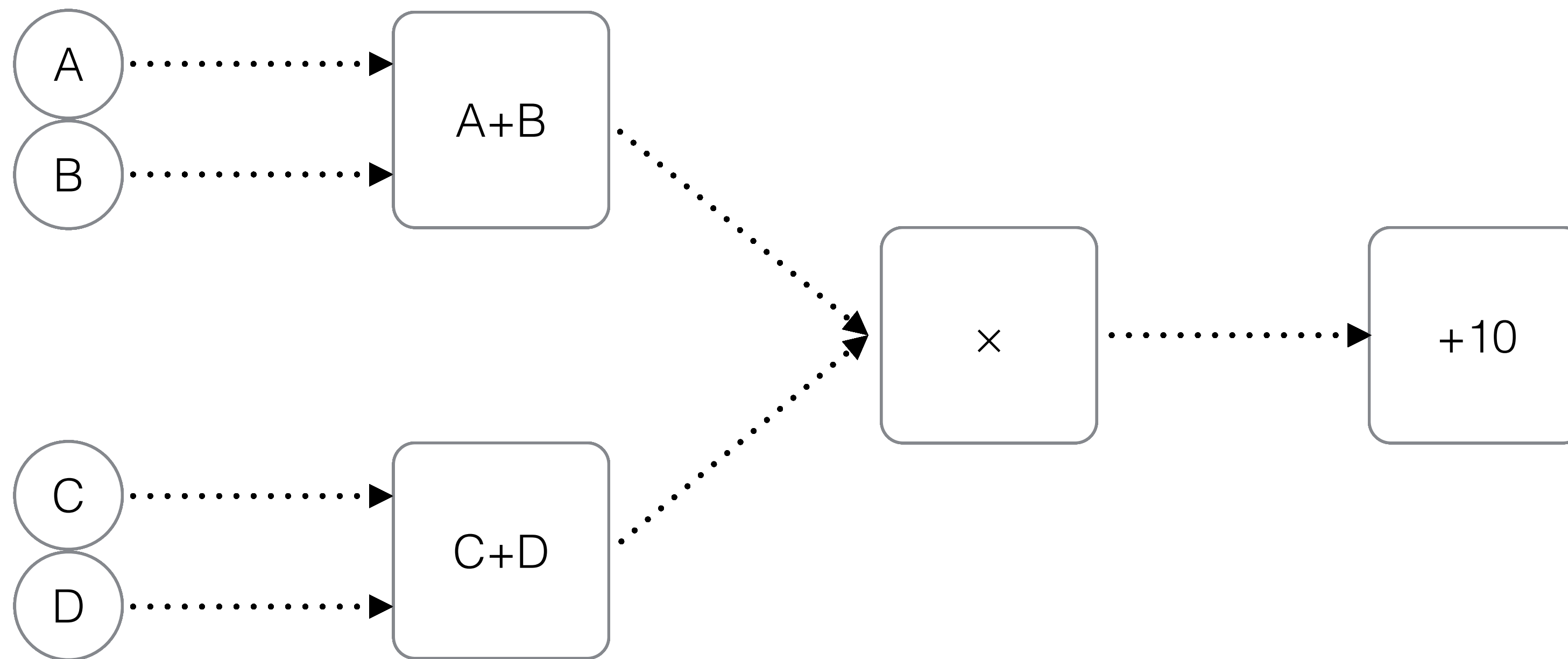
Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

Pipelining

Calculate $(A+B)*(C+D)+10$



Every Clock the data proceeds one step

(High-speed) IO

- Tons of GPIO!
 - Toggle frequency as high as fabric frequency
 - Speeds of up to 400->800 MBit/s with shift registers
- Some dedicated very high speed transceiver
 - 1->25 GB/s
- Very detailed control of characteristics

CPUs

- CPUs are good at
 - things that don't repeat very often
 - Large complex pieces of SW like operating system, network stack or dynamic memory allocation
 - quick turnaround times during development

Combining CPU and FPGA

- Connect via bus/memory interface on same PCB
- Soft CPU realized in fabric
- Hard CPU realized in silicon on FPGA die

Accelerating parts of application

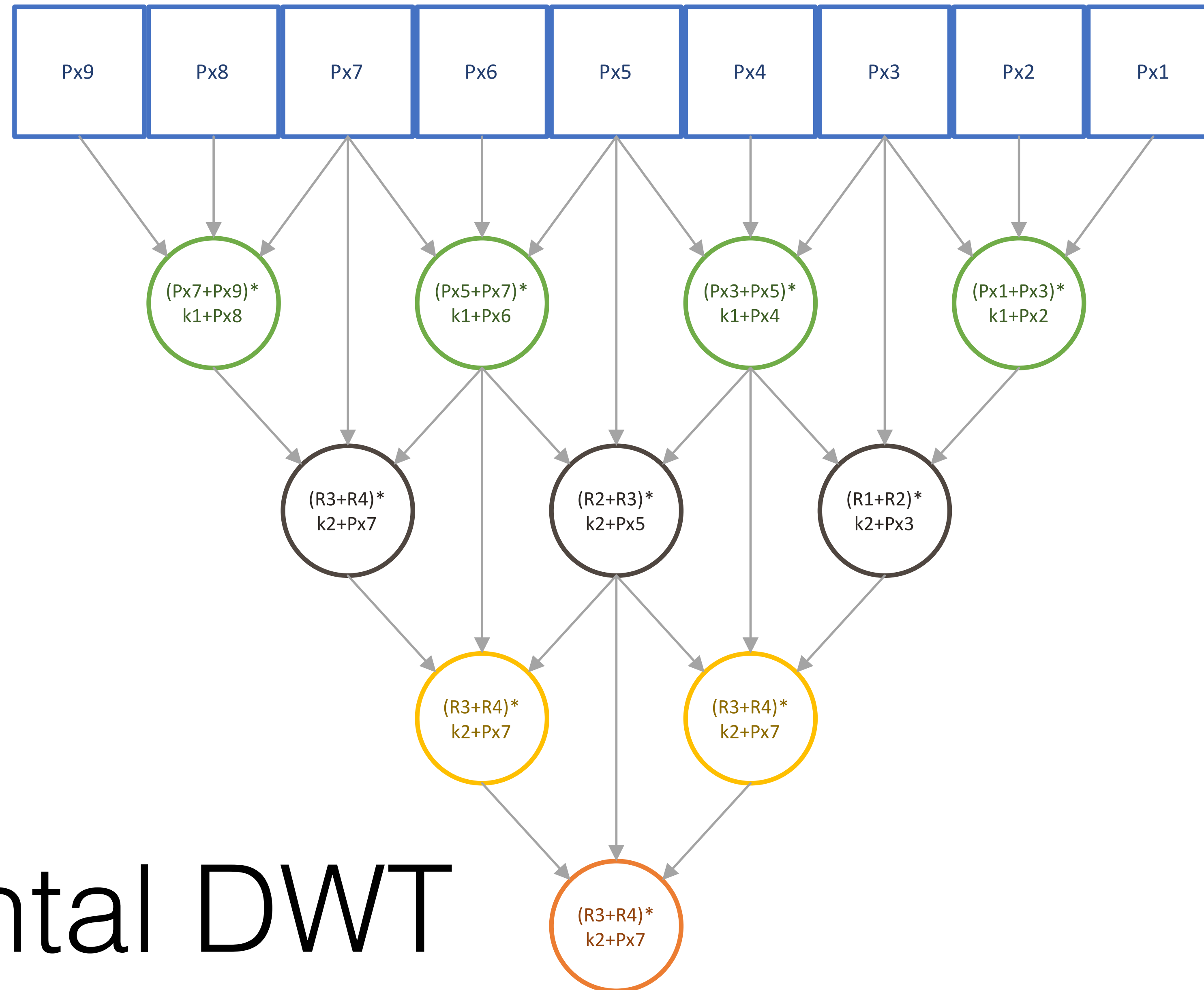
- Start with a pure SW implementation
- Measure performance / identify bottle-necks
- Translate expensive part to HDL

Discrete wavelet transformation

- Used by JPEG2000
- Applied to color channels of pictures
- 2D wavelet = 1D wavelet horizontal + 1D wavelet vertical

0

Cache Miss

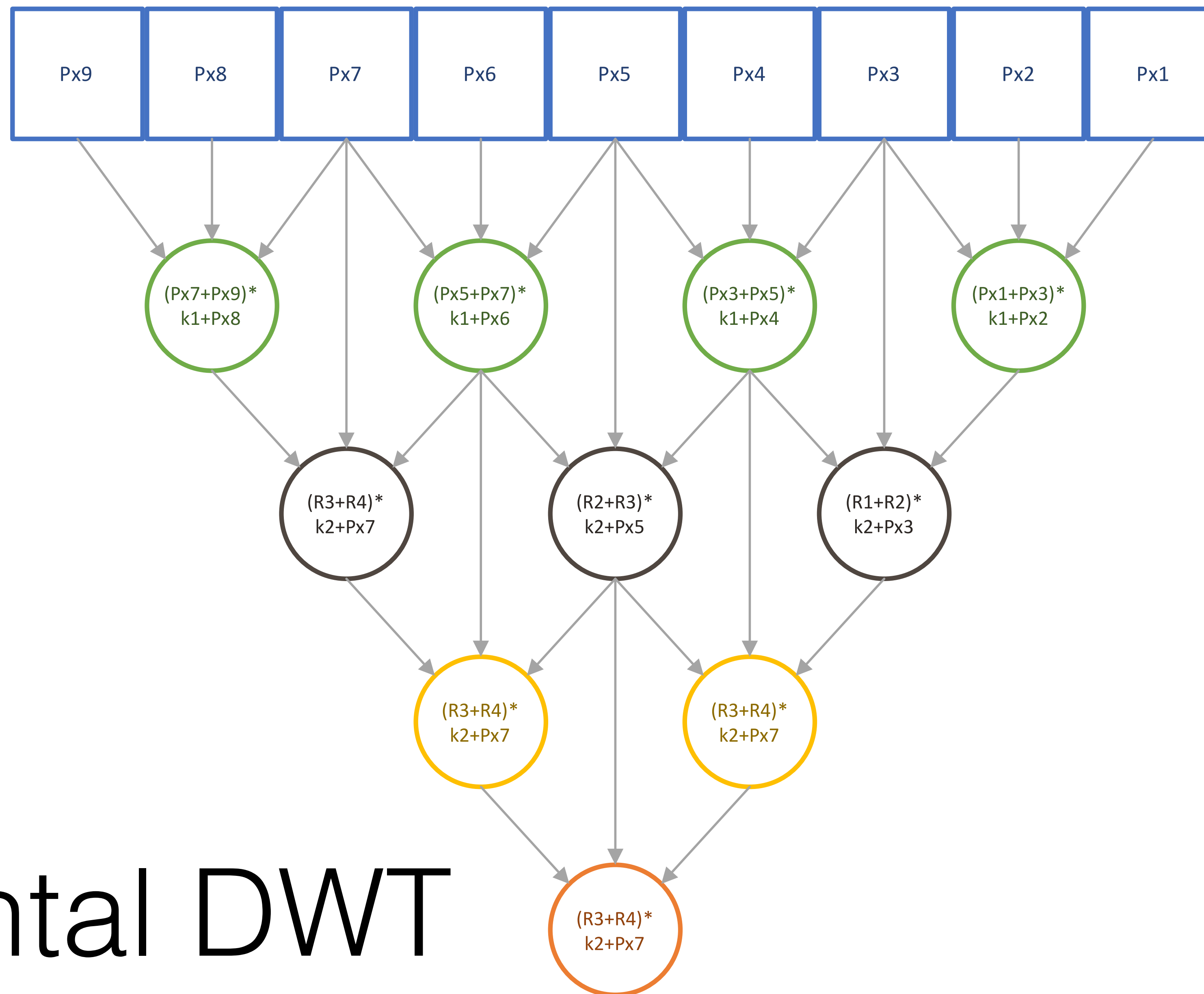


Horizontal DWT

0

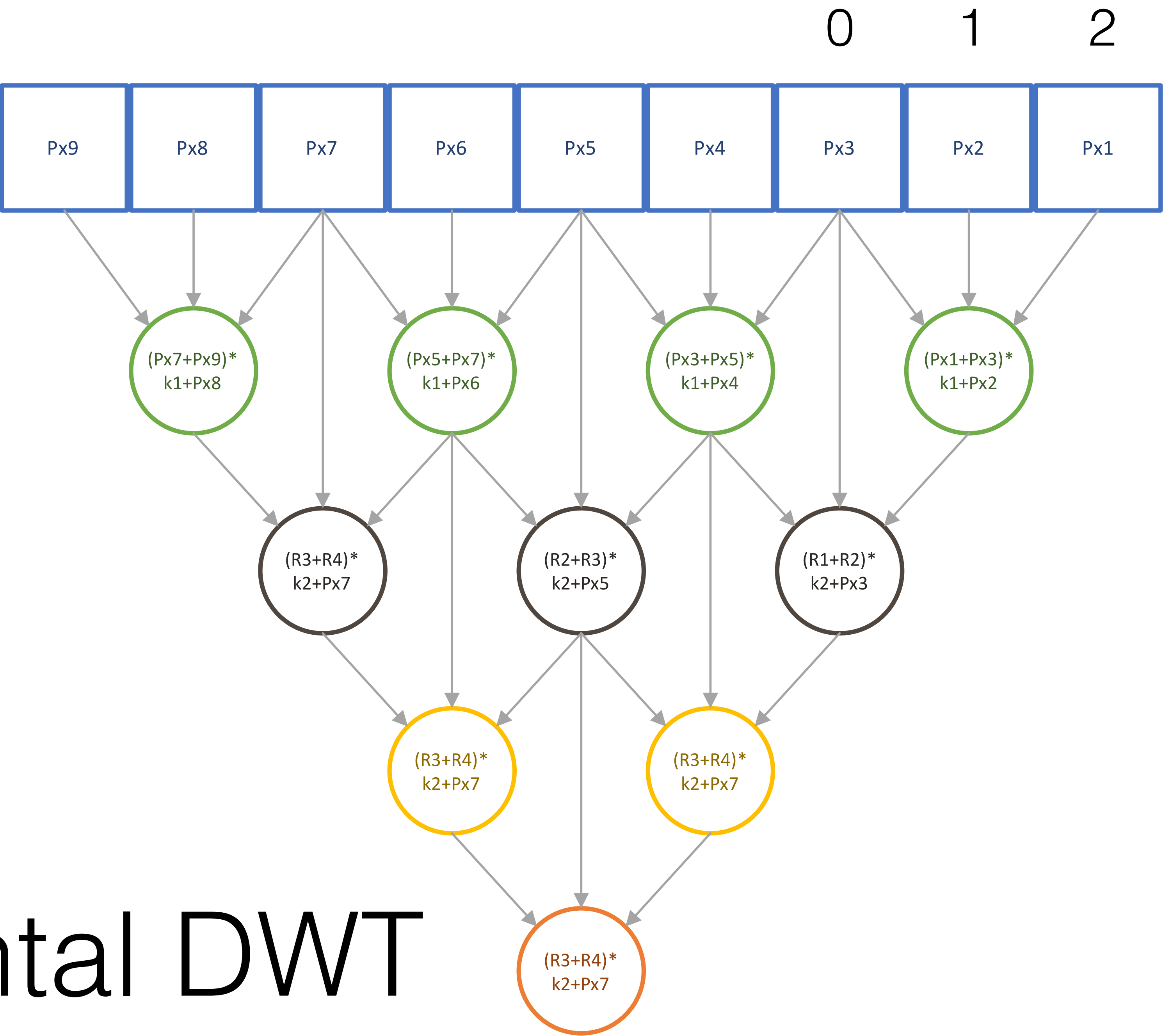
1

Cache Hit



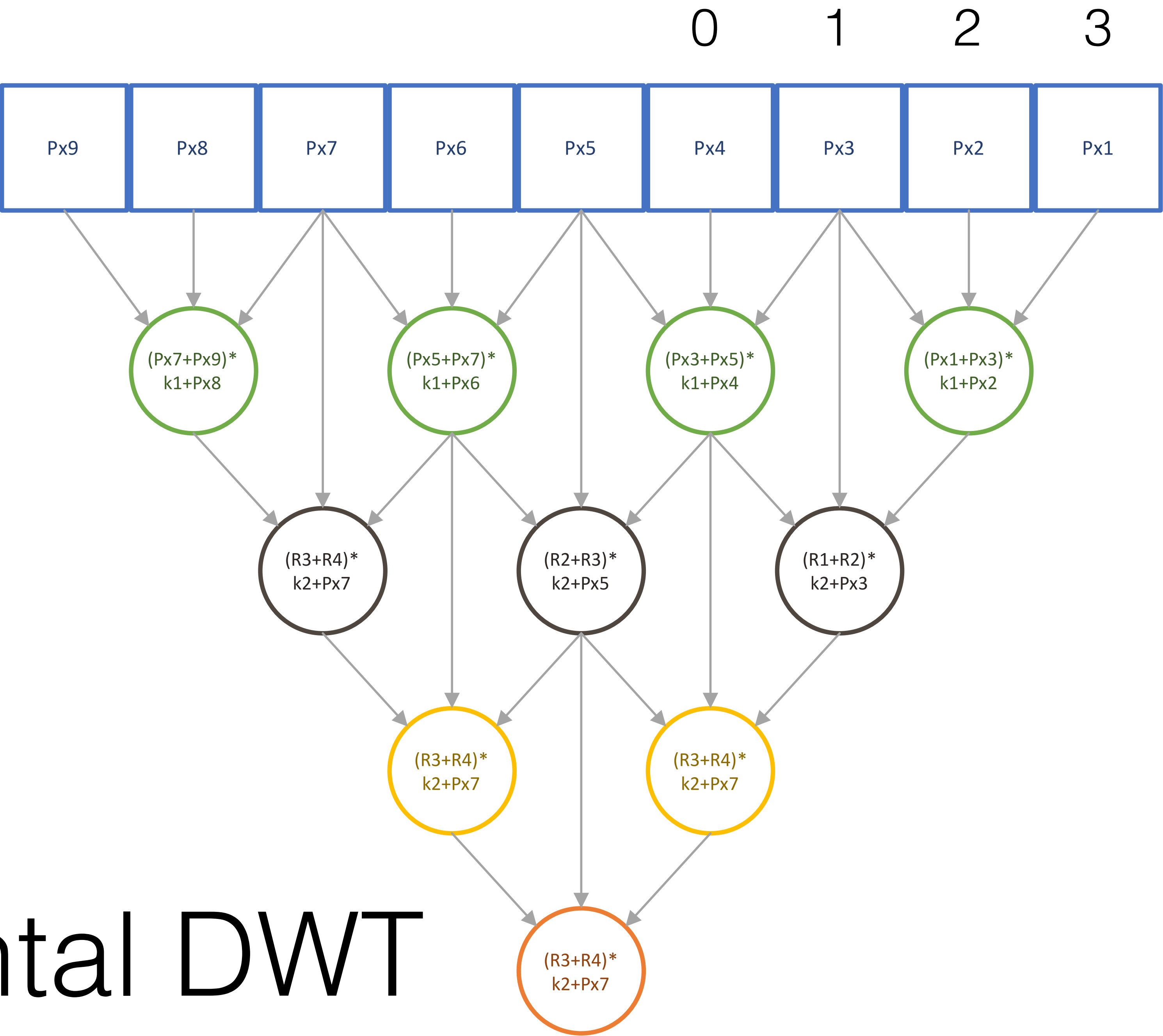
Horizontal DWT

Cache Hit



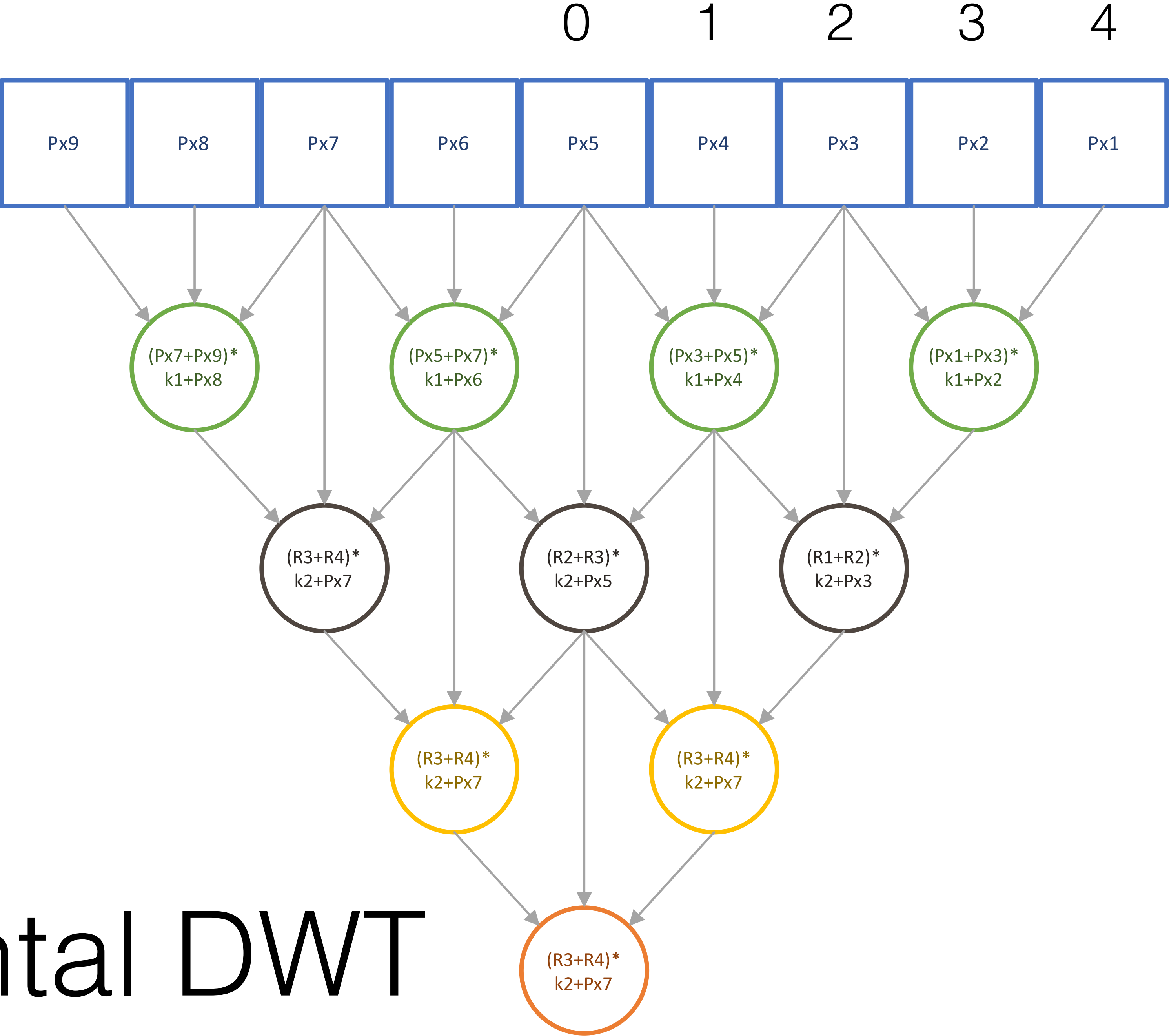
Horizontal DWT

Cache Hit



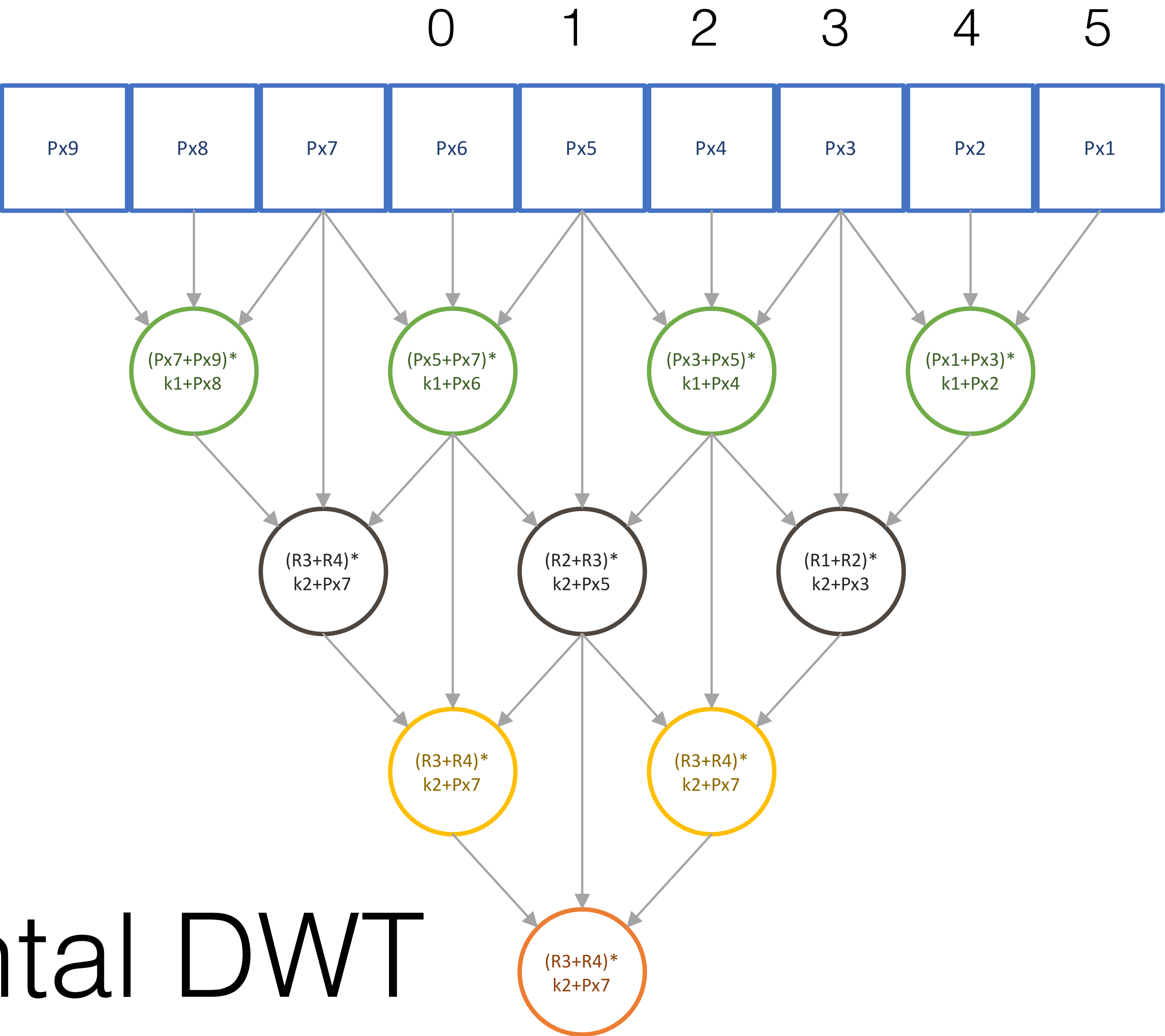
Horizontal DWT

Cache Hit



Horizontal DWT

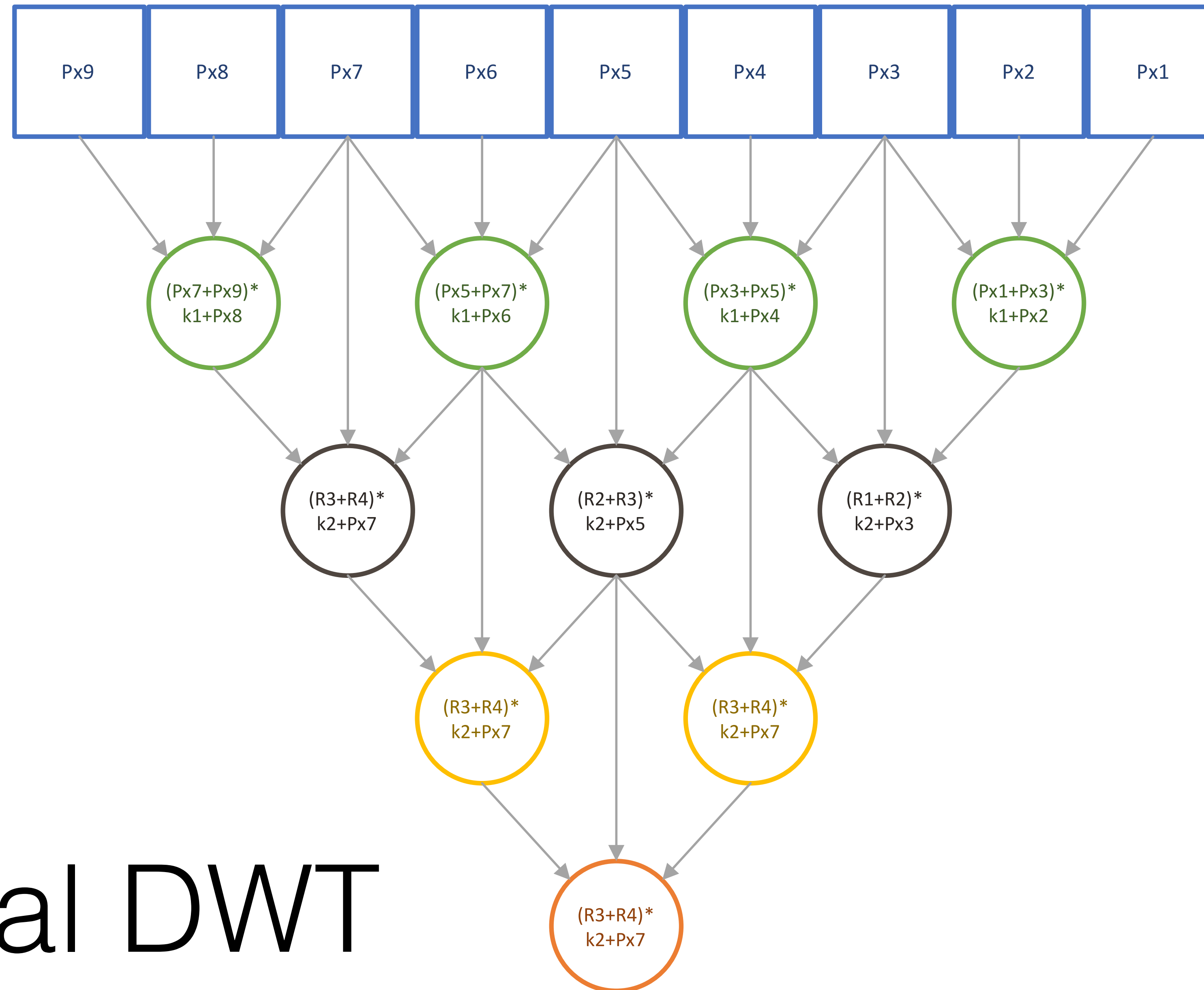
Cache Hit



Horizontal DWT

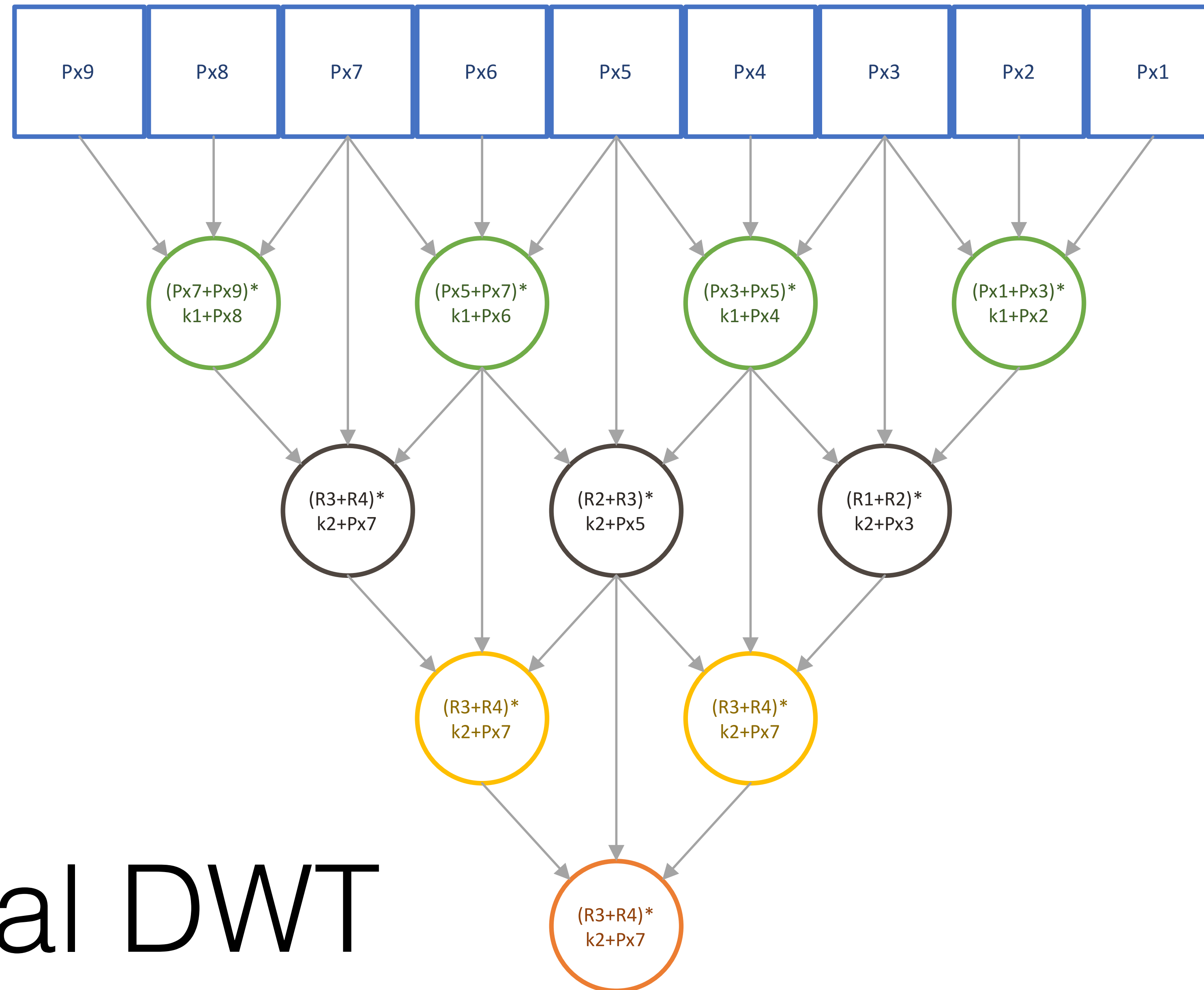
0

Cache Miss



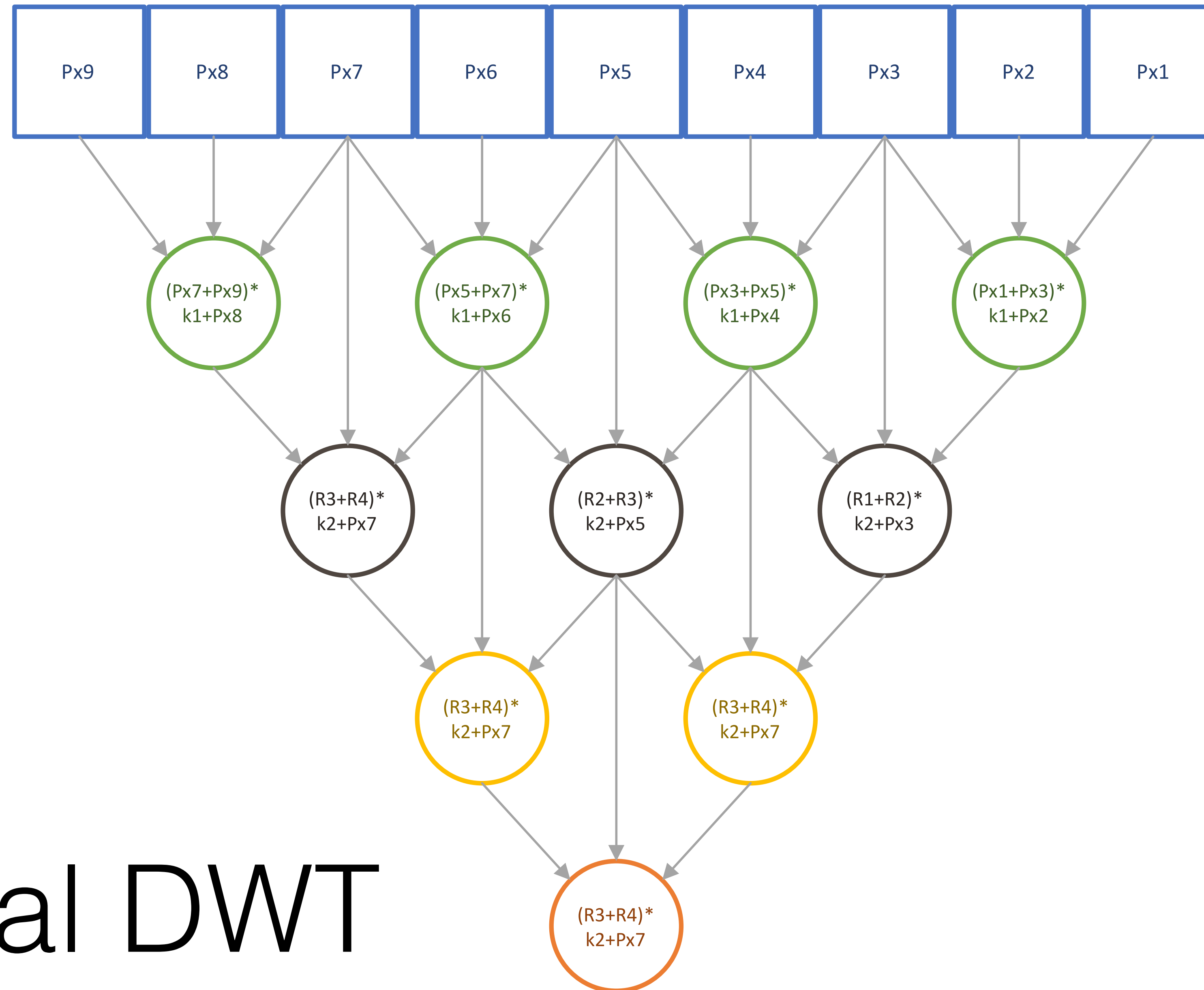
Vertical DWT

0 1024 **Cache Miss**



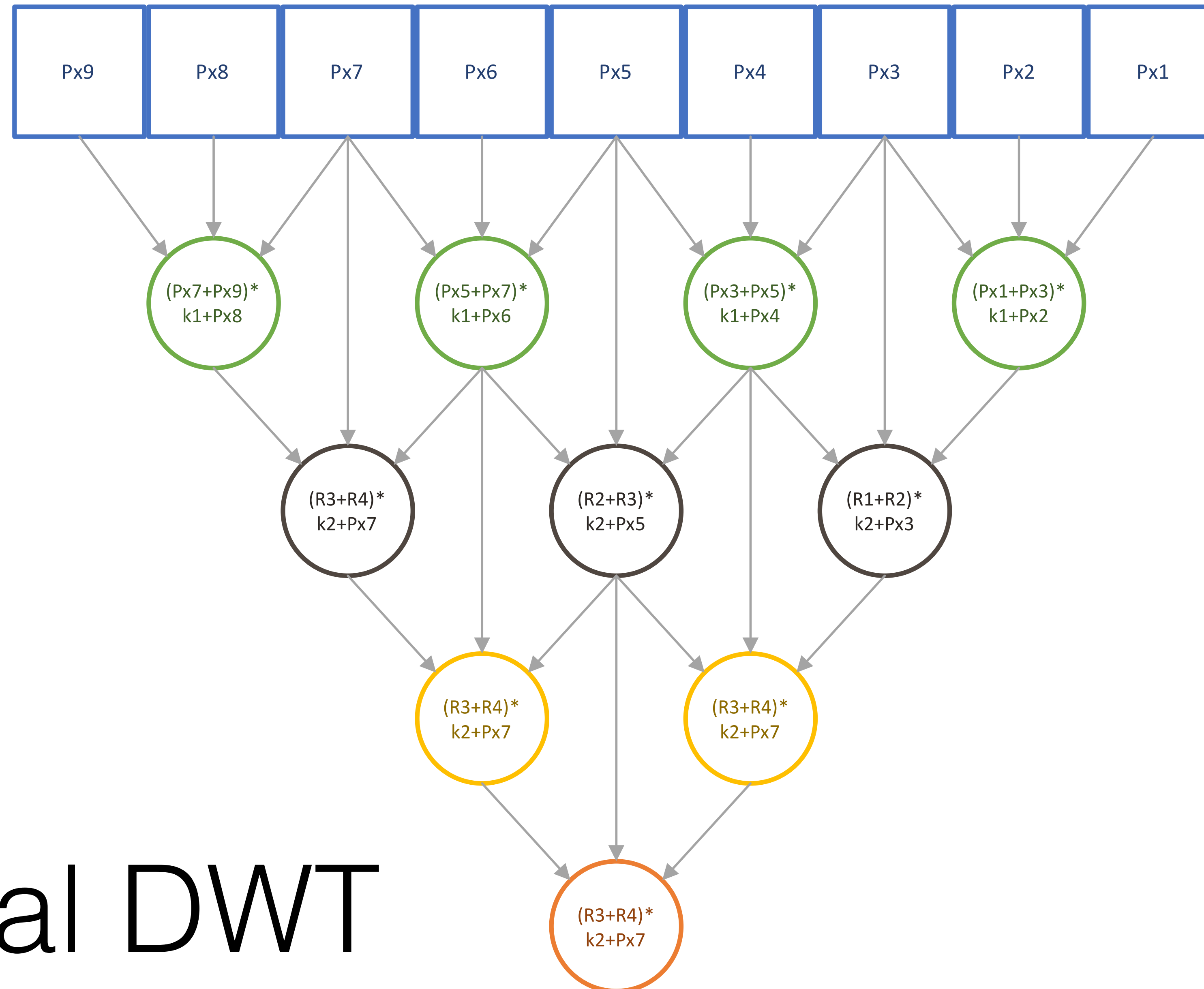
Vertical DWT

0 1024 2048 **Cache Miss**



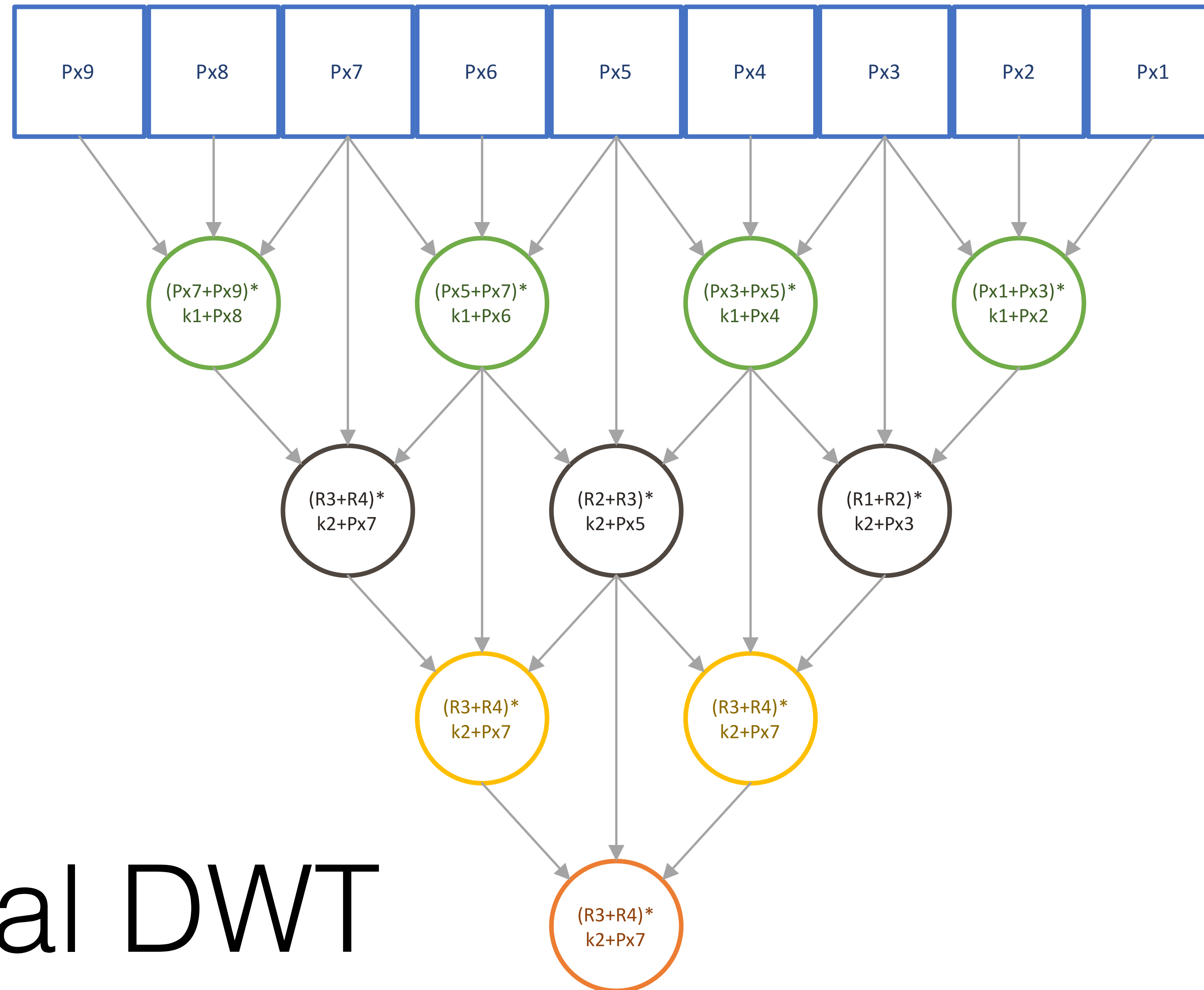
Vertical DWT

0 1024 2048 3072 **Cache Miss**



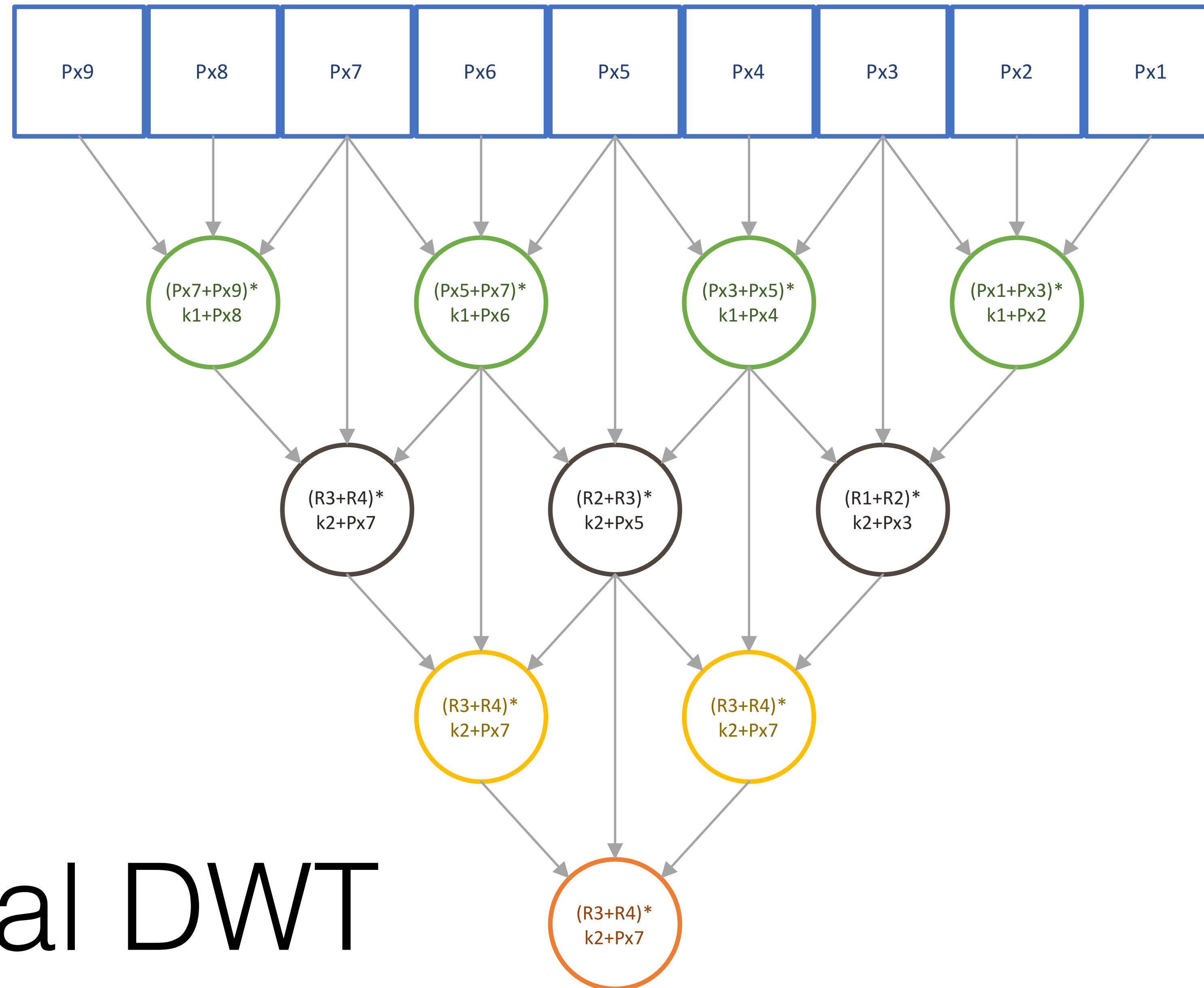
Vertical DWT

0 1024 2048 3072 4096 **Cache Miss**



Vertical DWT

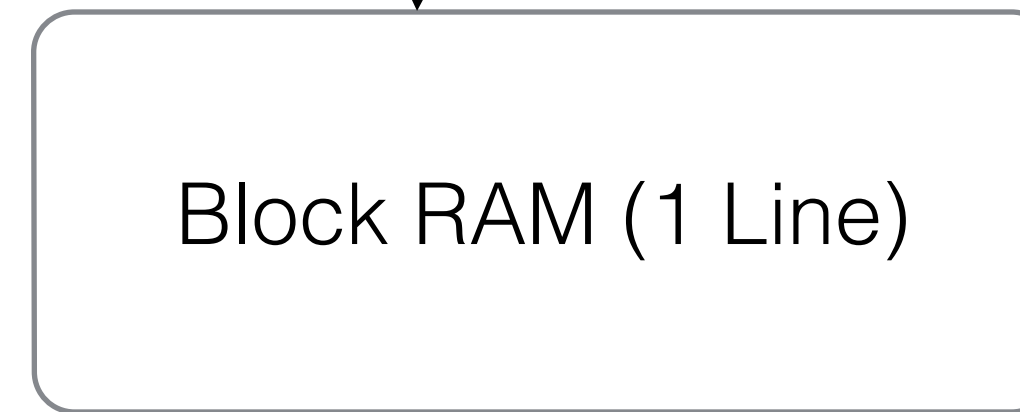
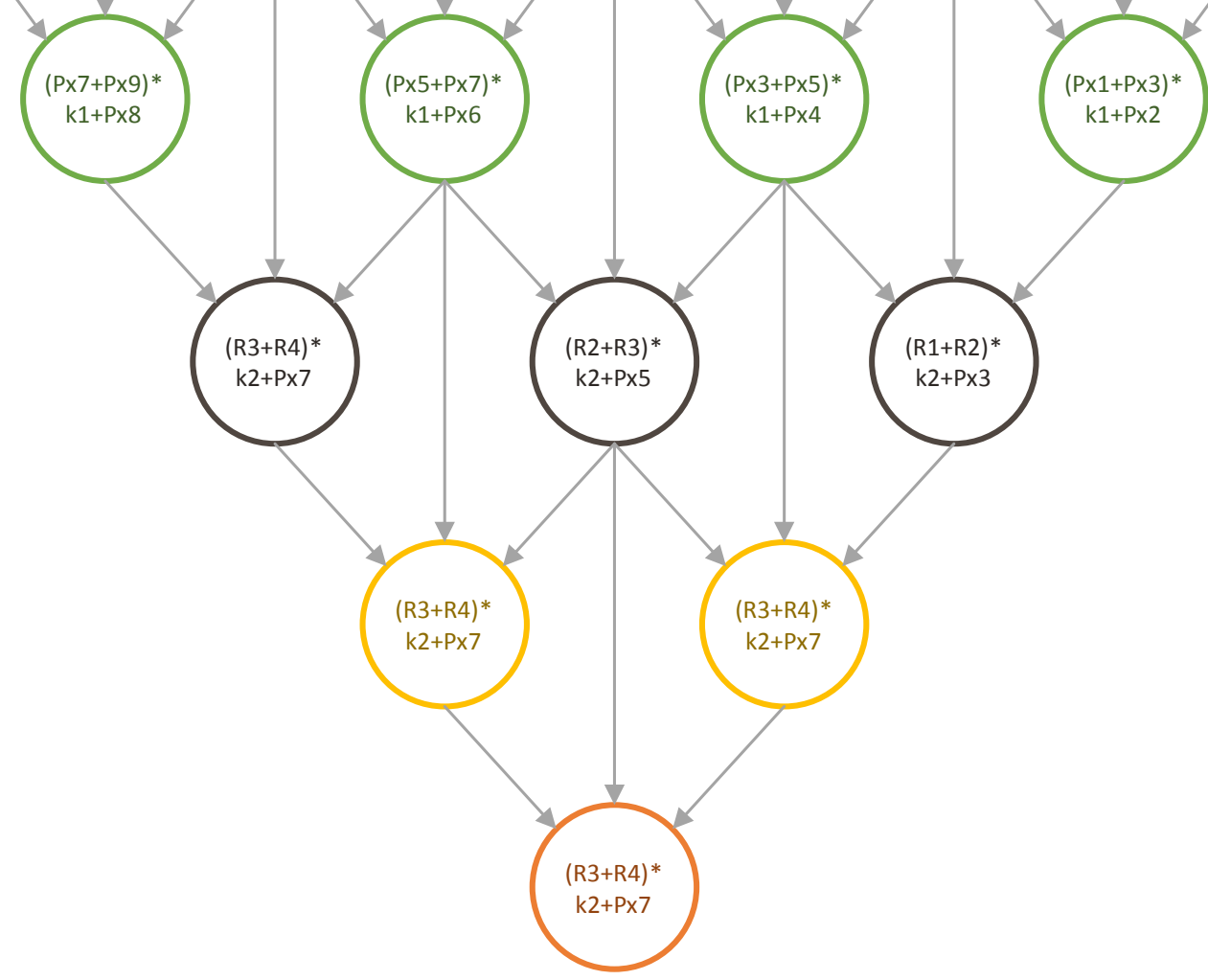
0 1024 2048 3072 4096 5120 **Cache Miss**



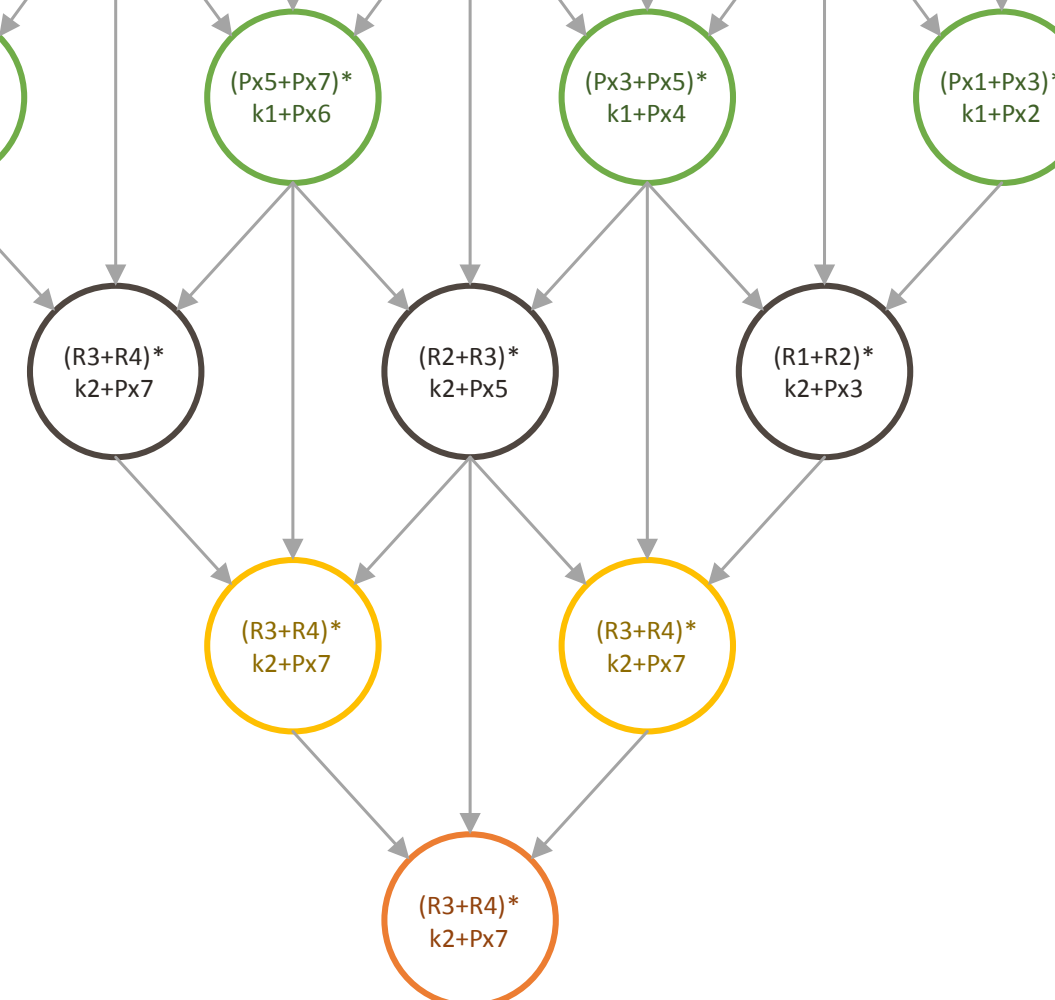
Vertical DWT



Horizontal DWT



Vertical DWT



FPGA

DWT Summary

- 1D Horizontal
 - CPU perfect cache hits due to prefetch -> very fast
- 1D Vertical
 - CPU near always cache miss -> significant slow down
- FPGA
 - Linear read -> 2 lines latency -> linear write @ full memory speed

Tech guide

	x86	Gpu	Dsp	Fpga	μ C
Embed	Maybe	Hard	Easy	Easy	Easy
Low power	Unusual	Nope	Sometimes	Sometimes	Yes
Float op	Good	Excellent	Excellent	Possible	Nope
Int op	Excellent	Excellent	Excellent	Excellent	Mediocre
Control flow	Excellent	Challenging	Fair	Challenging	Excellent
IO	Mediocre	Nope	Ok	Ginormous	Ok
Pipelining	Nope	Nope	Nope	Yes	Nope
Programmability	Easy	Medium	Medium	Challenging	Easy
Timing control	Medium	What?	Fair	Excellent	Fair

Vendor Overview

- Xilinx / Altera
 - Kind of the same
 - High Performance
 - High Bandwidth
 - Biggest 2 on the market
 - Decent tooling
- Actel (microsemi)
 - Low Power
 - High reliability
- Lattice
 - Low Power
 - Cheap

Vendor Overview

- Xilinx / Altera
 - Kind of the same
 - High Performance
 - High Bandwidth
 - Biggest 2 on the market
 - Decent tooling
- Actel (microsemi)
 - Low Power
 - High reliability
- Lattice
 - Low Power
 - Cheap

Most expensive Xilinx: 32.000 EUR

Most expensive Altera: 19.755 EUR (3 pieces min order)

Hacker friendly Boards

	Actel	Altera	Xilinx
Cheap	PSHDL Board ~3x €	DE0-Nano 70€ BEMICRO CV 40€	Papilio One 47€ (Spartan 3) Mojo V3 54€ (Spartan 6) XuLA2-LX25 103€
Powerful	Igloo 2 boards	Cyclone 5 Arria	Artix/Kintex/Virtex
SoC	SmartFusion2 Starter Kit 295€ SmartFusion Starter Kit 100€	EBV SoCrates 359€	MicroZedBoard ~200€ Parallela 85€
CPU +FPGA	Datenkrake ~95€		Logi (Kickstarter)

Thanks, questions?

For infos about the PSHDL board, visit:

boards.pshdl.org

Join the workshop on Sunday 9pm Hall **F**