

Arbolta

Fault Tolerance Study using Minimal Hardware Simulation

Motivation

Hardware Faults are Real and Inevitable



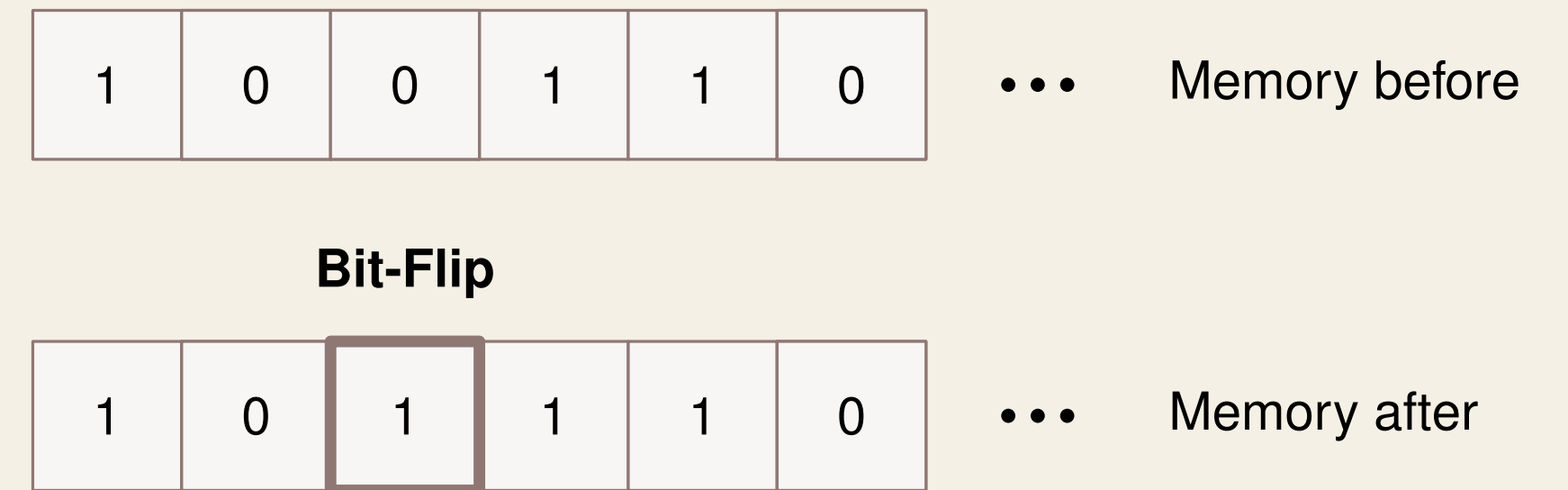
- When physical components malfunction or behave abnormally
- In the data center
 - Manufacturing defects noticed in large-scale systems
 - Examples of silent data corruption from faulty CPUs observed across industry
 - Last year, 78% of unexpected training interruptions during 54-day period of Llama 3 pre-training attributed to hardware issues (Meta [1])
- At the edge
 - Single event upsets from radiation (e.g., space, energy physics deployments)
 - Power fluctuations in low-power, energy harvesting sensors

[1] <https://arxiv.org/pdf/2407.21783>

Motivation

HW Faults in Terms of Bit Flips

- Transient faults
 - Temporary, non-recurring bit flip ($0 \rightarrow 1$, $1 \rightarrow 0$)
 - Typically caused externally (e.g., radiation, power fluctuations)
- Permanent faults
 - Bit irreversibly and persistently stuck at 0 or 1
 - Typically caused by hardware defects or wear-out
- Intermittent faults
 - Recurring faults which appear and disappear intermittently
 - Typically caused by unstable hardware conditions or defects



Motivation

Impact of HW Faults on Machine Learning

- Data or instruction memory
 - Wrong branch predictions, unintended flow control
 - Corrupted model parameters
- Arithmetic logic
 - Incorrect gradient calculations during training
 - Compromised convergence and accuracy
- Silent Data Corruption (SDC)
 - Undetected errors that propagate through system

Motivation

Reliability Assessment of Neural Networks

- Most works use HW-independent fault injection simulation [2]
 - Typically involves bit flip injection into weights and activations during inference
 - Model weights → fault in DRAM
 - Activations → fault in arithmetic logic
 - High-level data science approach
 - Allows for re-use of optimized ML frameworks (e.g., PyTorch)
 - Development of mitigation methods like flip-aware training

Motivation

Limits of Software-Based Fault Models

- Real world hardware is more complicated
 - >40% of Llama 3 training interruptions were due to GPU hardware issues **not** including DRAM (e.g., processor, SRAM) [3]
- Existing HW-independent frameworks abstract underlying HW
 - Cannot capture low-level microarchitectural behavior
 - E.g., faults in control logic

Motivation

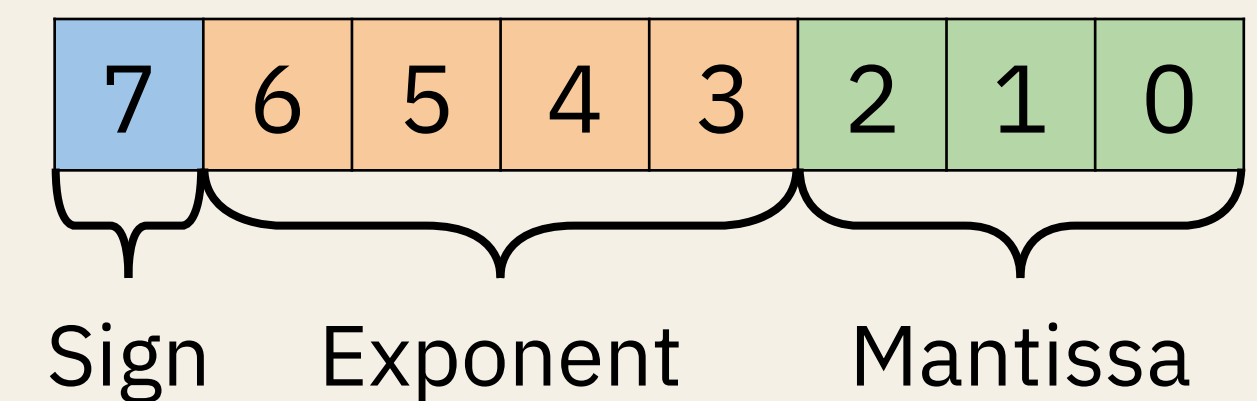
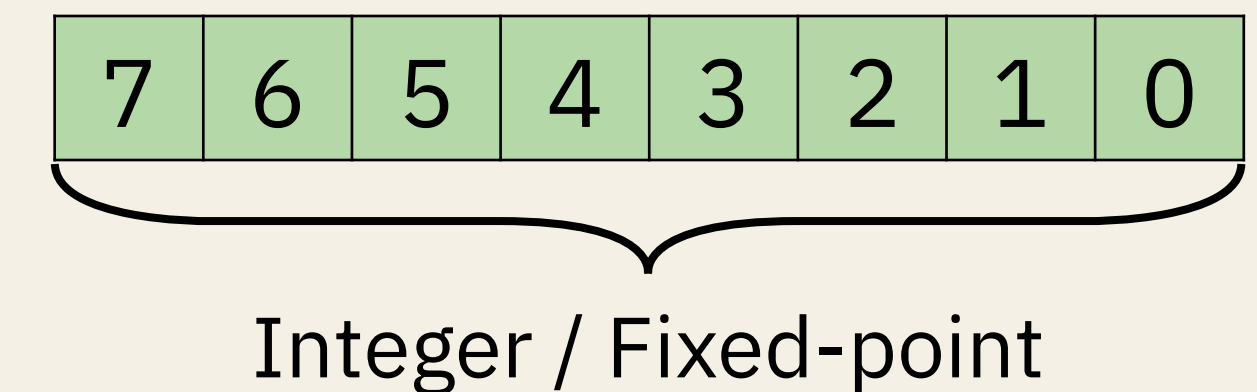
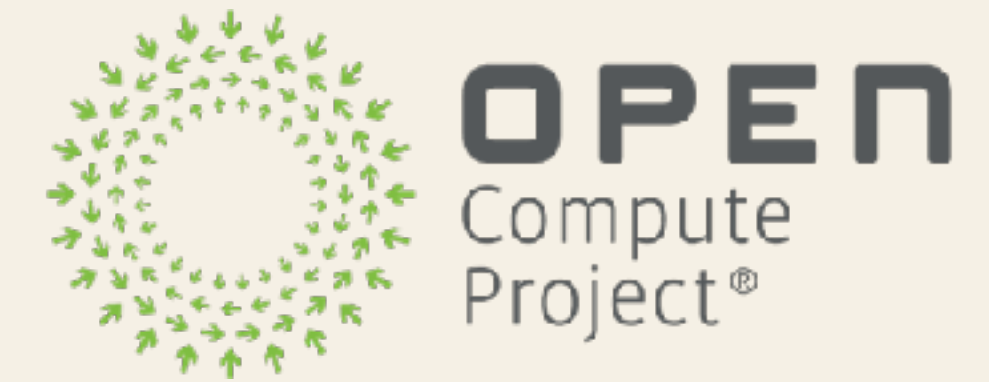
HW-Aware Fault Models

- Software model of high-level accelerator microarchitecture
 - Understanding how faults in accelerator HW affect performance
 - Typically based on systolic arrays
 - One variable per register in data path
- Limiting assumptions
 - Specific accelerator architecture
 - Standard datatype implementation (e.g., IEEE 754 FP32)
 - Faults in control logic *always* result in detectable failure

Motivation

Number Formats

- Industry adoption of emerging datatypes (FP8, MX-types)
 - Promise of reduced training and inference costs
 - Existing format specs don't require reproducibility
- Lack of standardized HW implementations
 - Ex, many ways of performing FP8 multiply-accumulate
 - Simulate with existing HW, cast operands to FP16
 - Dedicated FP8 multiply hardware, accumulate in fixed-point
- Understanding microarchitectural effects of faults is critical



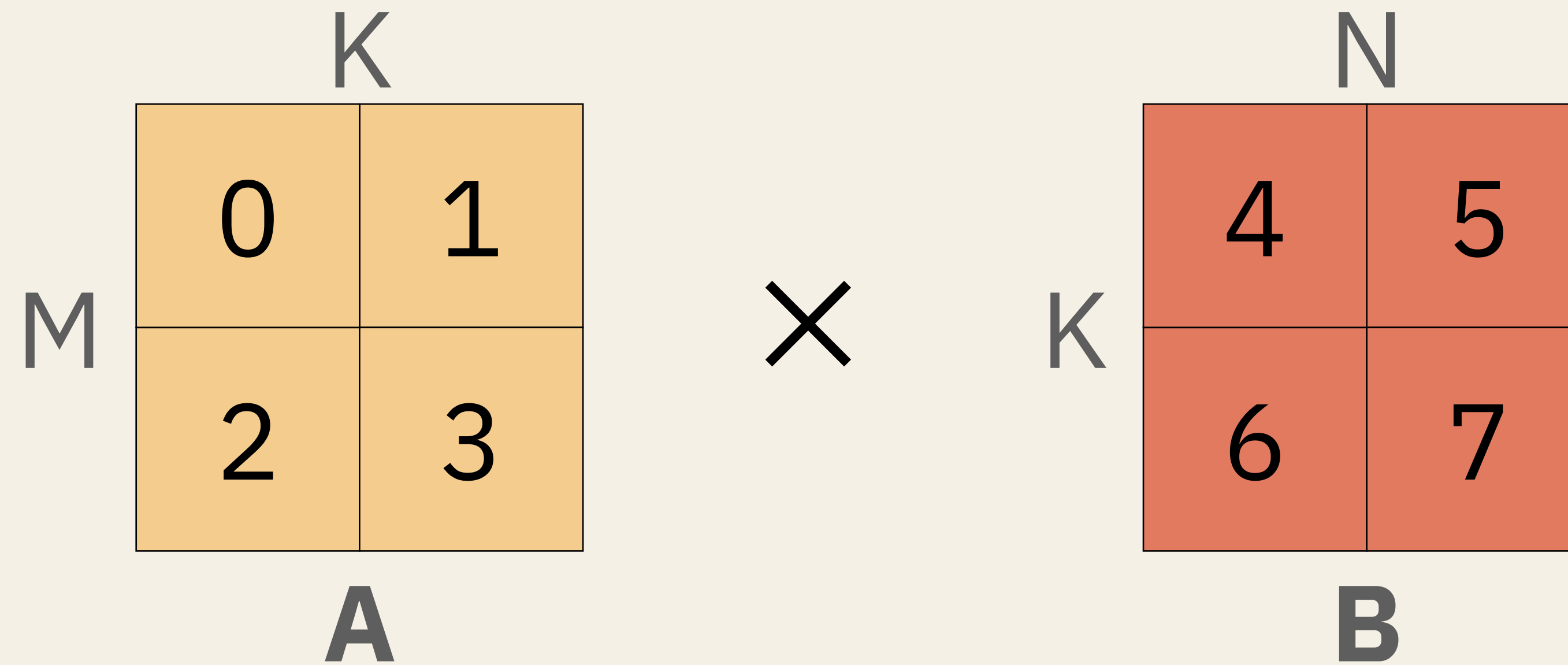
OCP Microscaling

Shared Scale

Fault Example

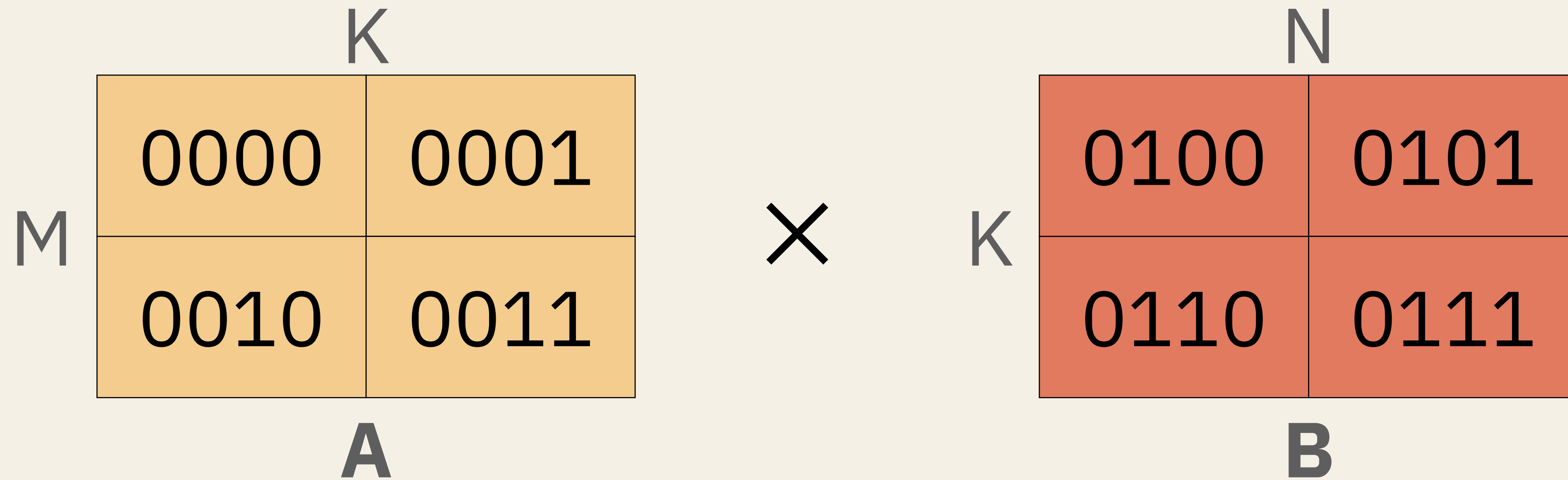
Fault Example

2x2 Matmul in Software



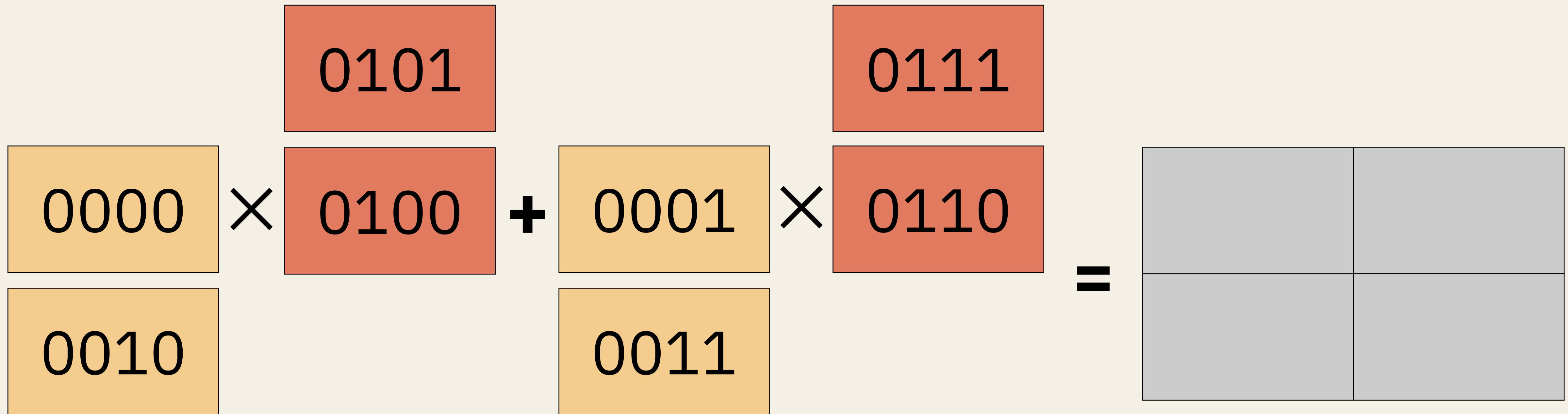
Fault Example

2x2 Matmul in Software



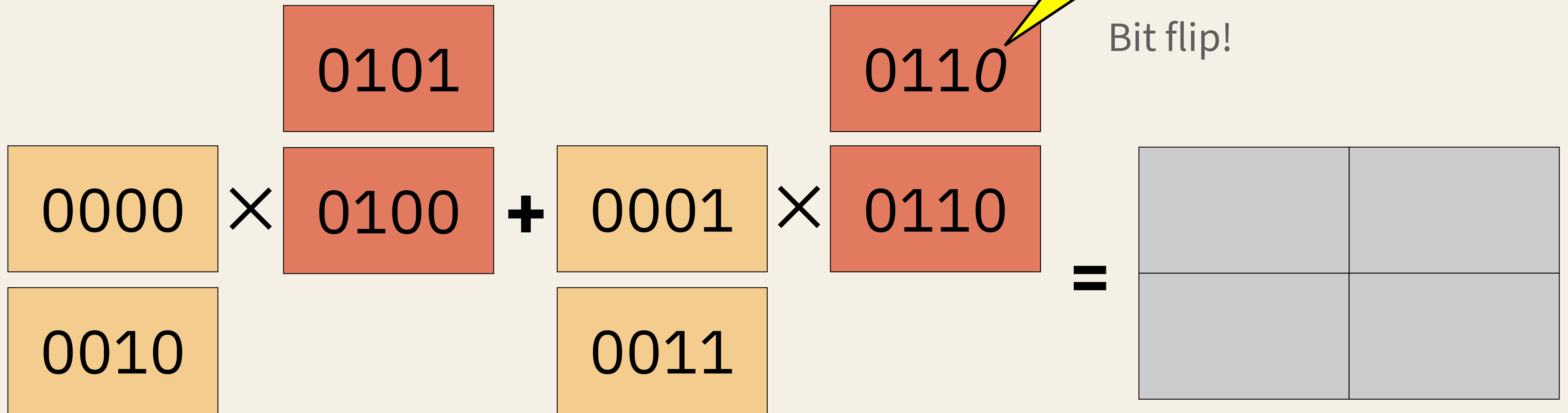
Fault Example

2x2 Matmul in Software



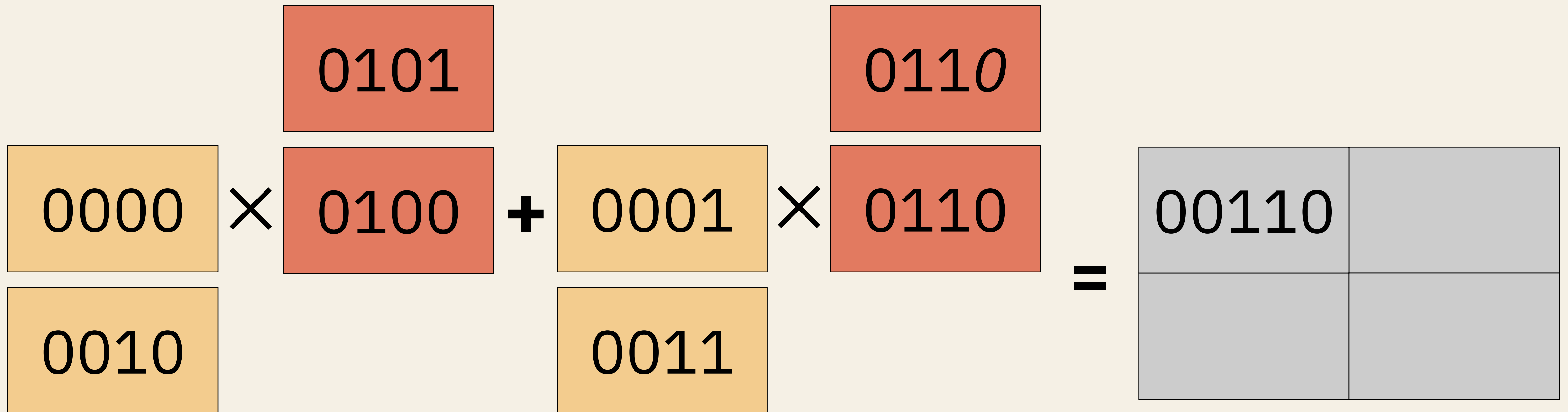
Fault Example

2x2 Matmul in Software



Fault Example

2x2 Matmul in Software



Fault Example

2x2 Matmul in Software

$$\begin{array}{cc} \boxed{0000} \times \boxed{0101} & + & \boxed{0001} \times \boxed{0110} \\ \boxed{0010} \times \boxed{0100} & + & \boxed{0011} \times \boxed{0110} \end{array} = \begin{array}{|c|c|} \hline 00110 & 00110 \\ \hline 11010 & \\ \hline \end{array}$$

Error!
7 → 6

Fault Example

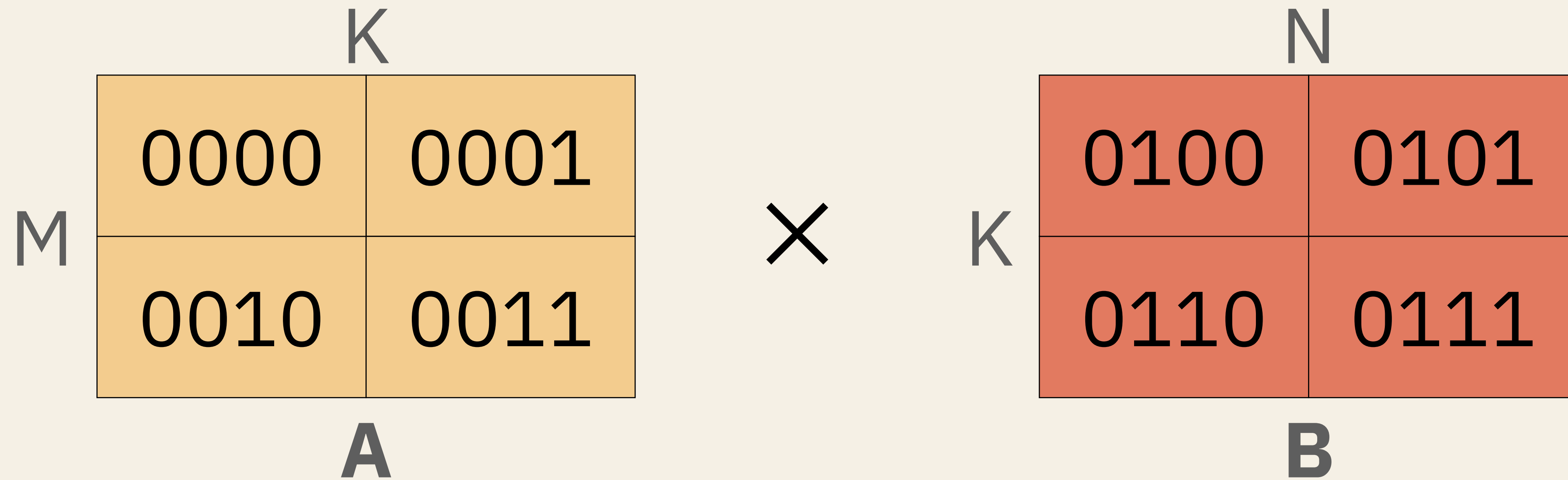
2x2 Matmul in Software

$$\begin{array}{c} 0000 \\ 0010 \end{array} \times \begin{array}{c} 0101 \\ 0100 \end{array} + \begin{array}{c} 0001 \\ 0011 \end{array} \times \begin{array}{c} 0110 \\ 0110 \end{array} = \begin{array}{|c|c|} \hline 00110 & 00110 \\ \hline 11010 & 11100 \\ \hline \end{array}$$

Error!
31 → 28

Fault Example

2x2 Matmul in Hardware



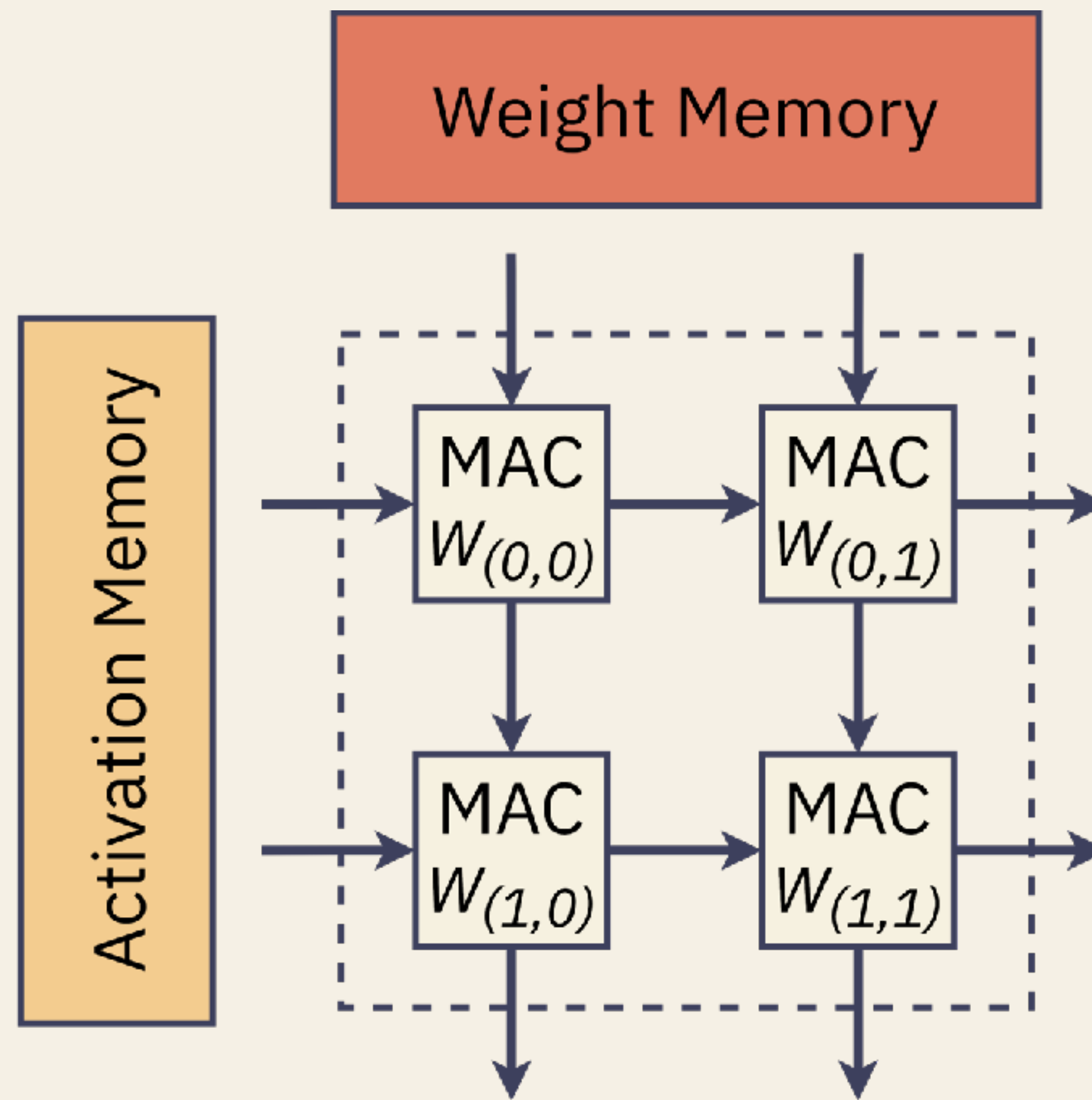
Fault Example

2x2 Matmul in Hardware

0000	0001
0010	0011

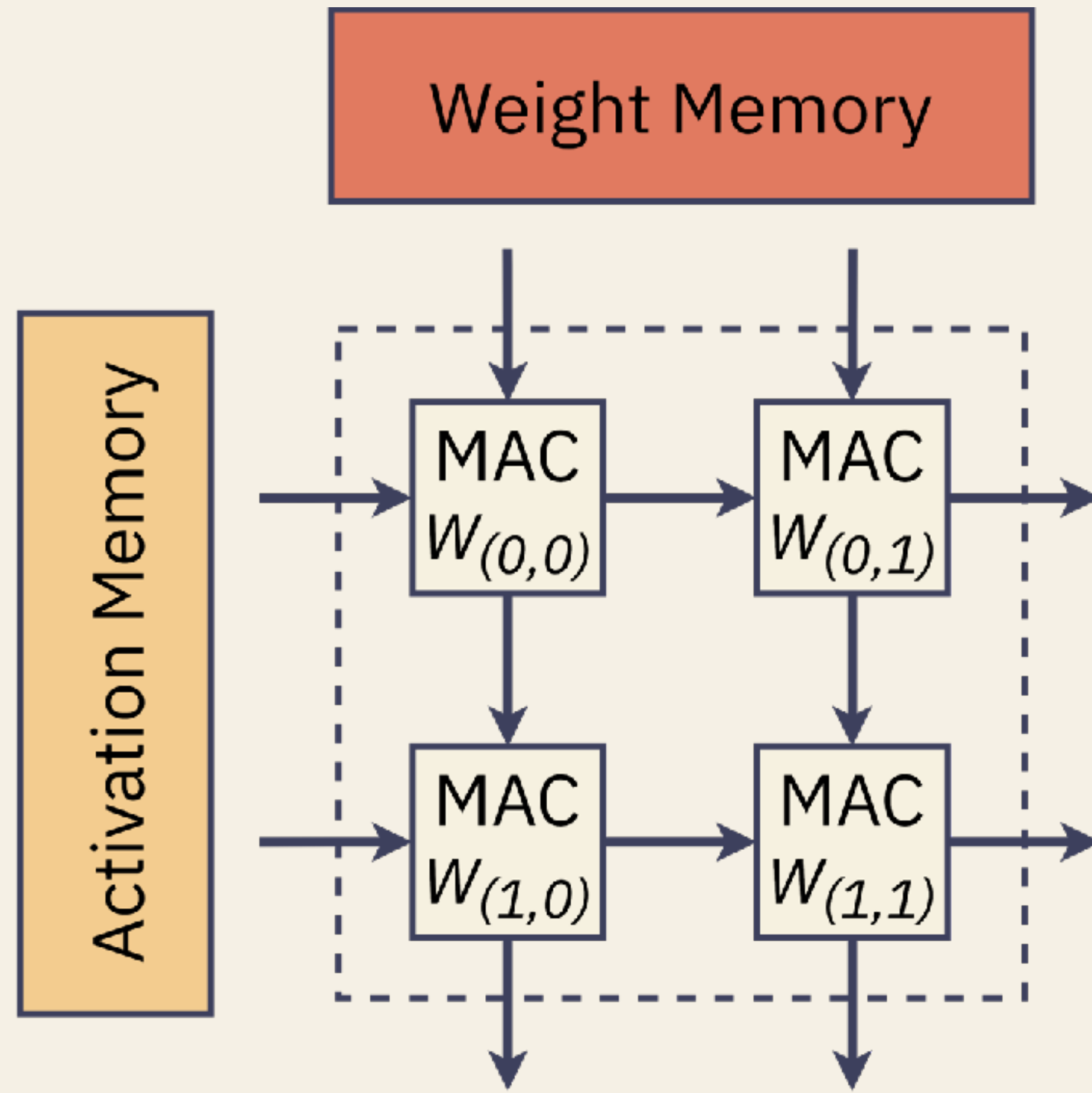
×

0100	0101
0110	0111



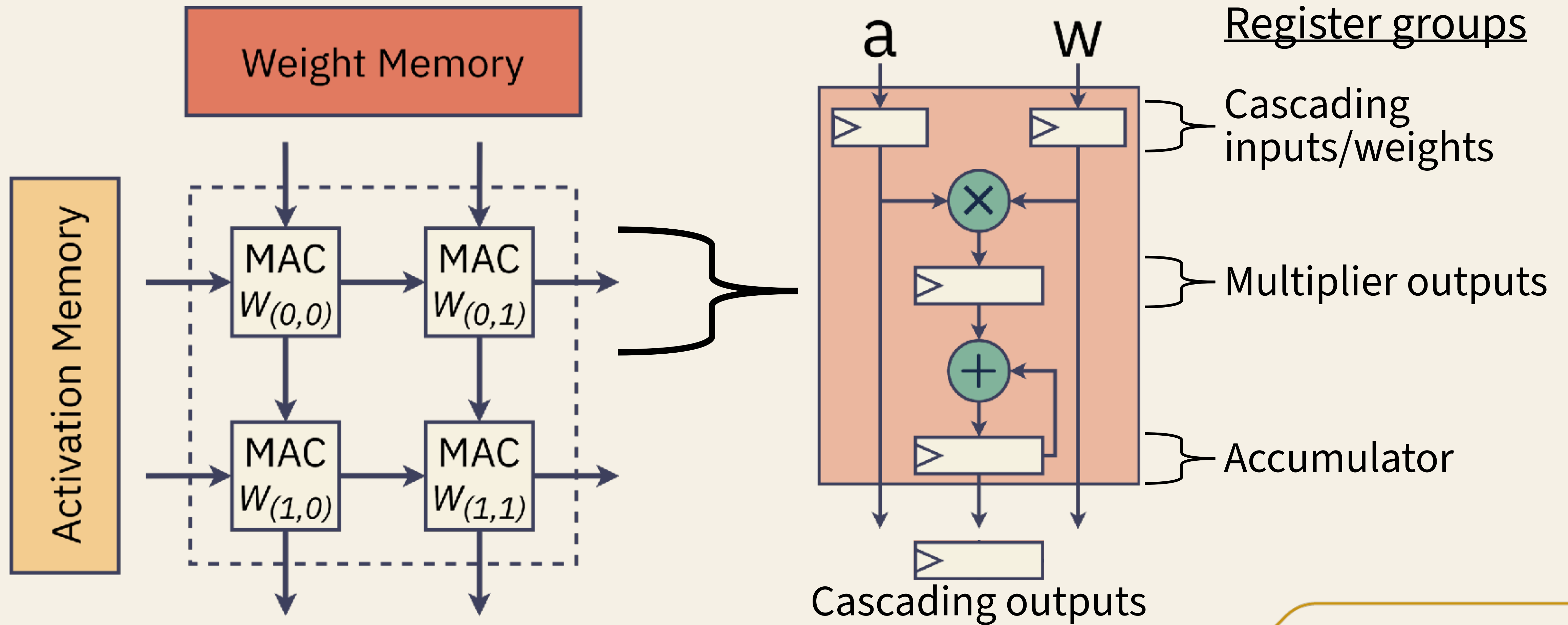
Fault Example

2x2 Matmul in Hardware



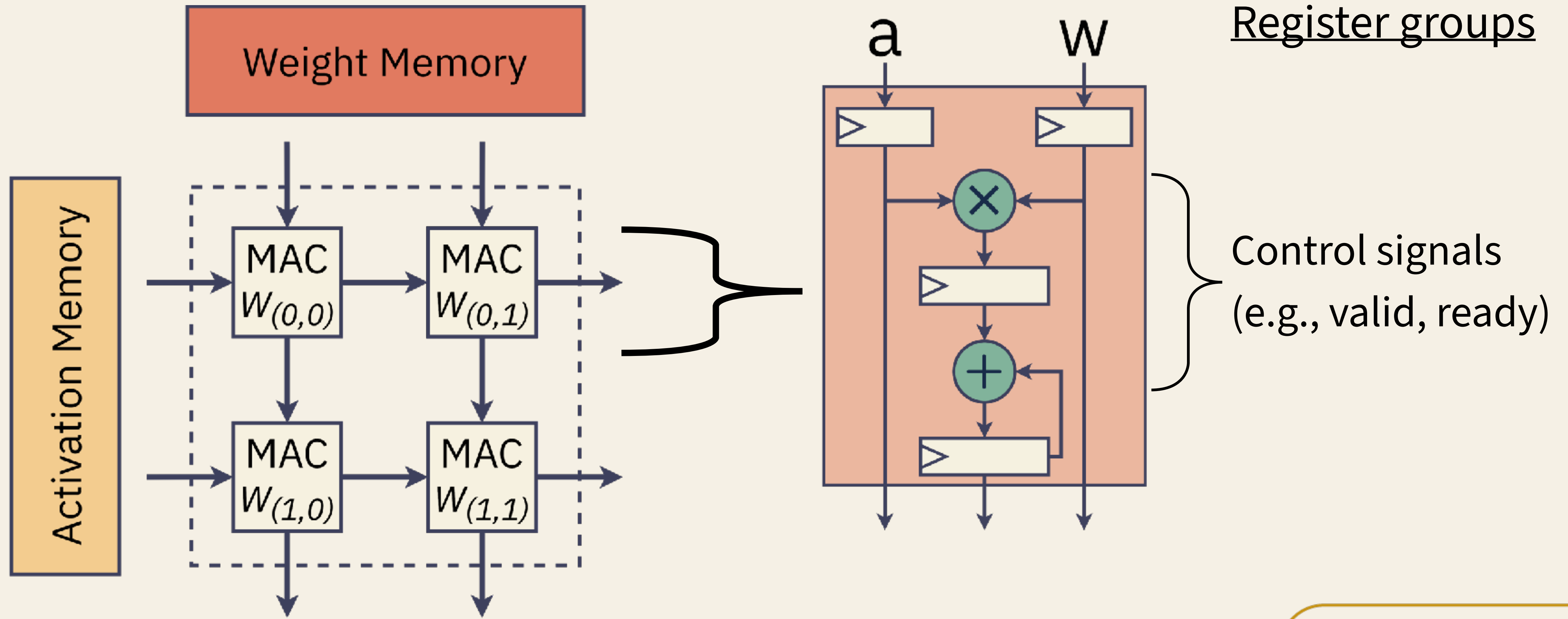
Fault Example

2x2 Matmul in Hardware



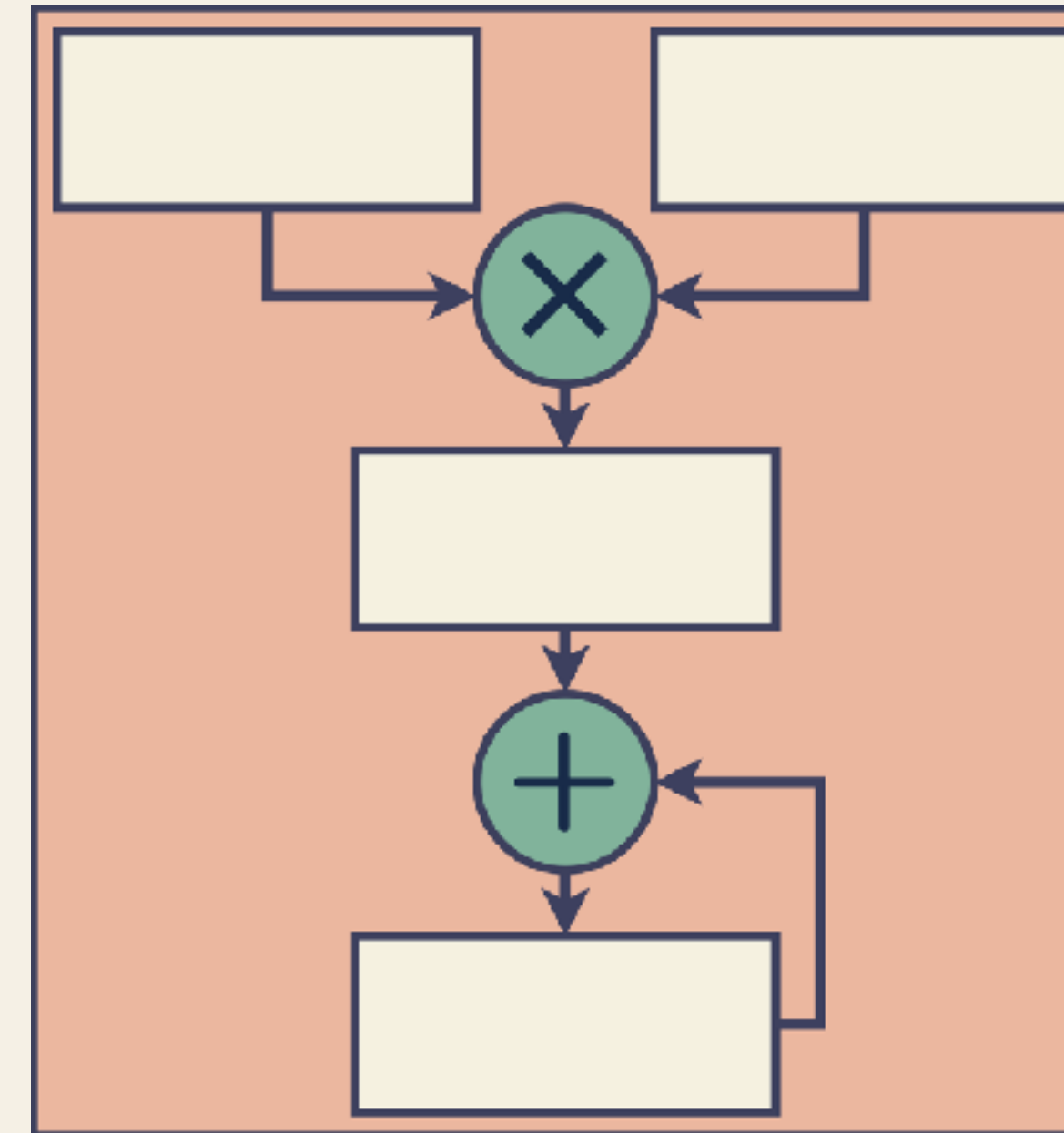
Fault Example

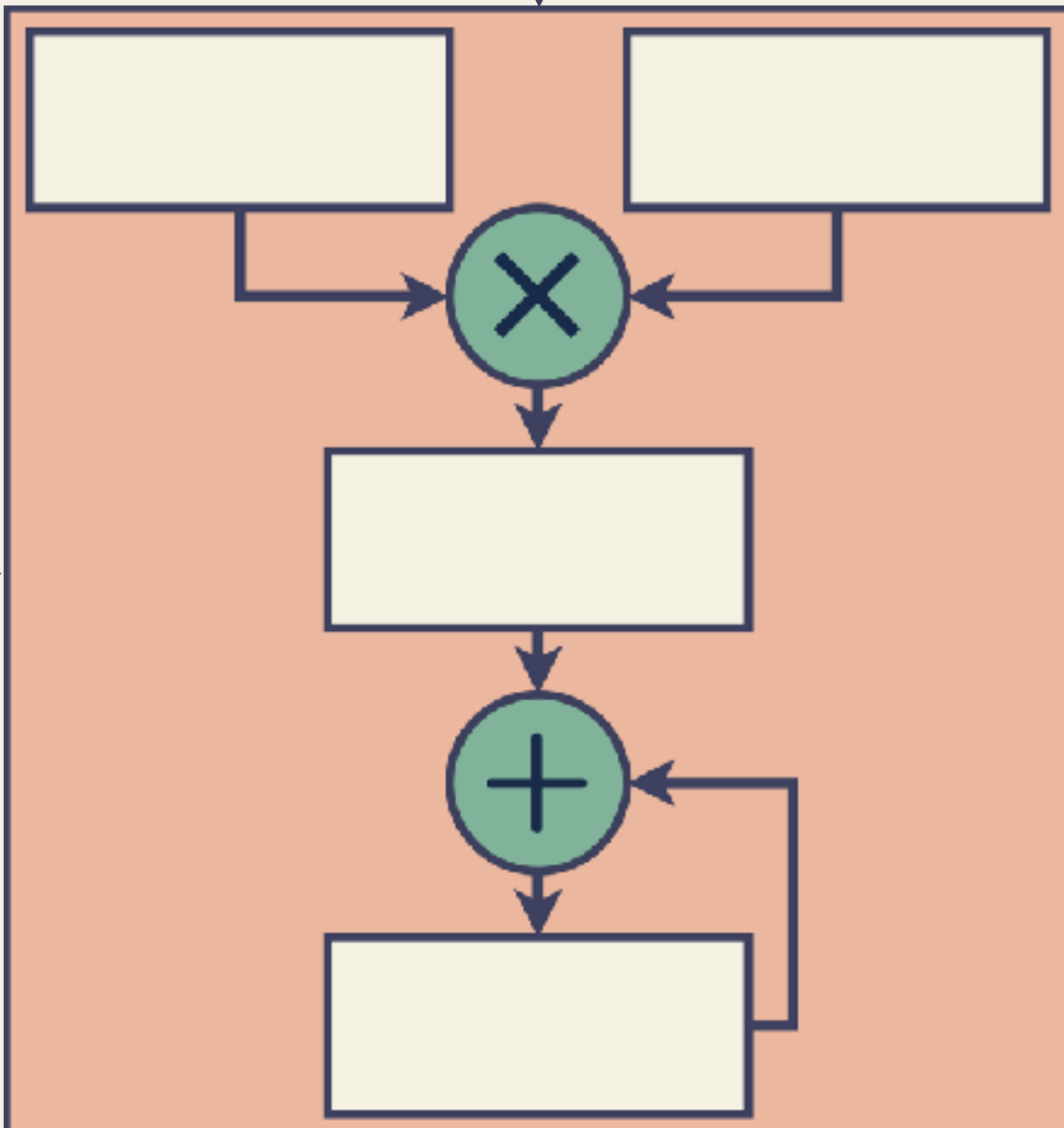
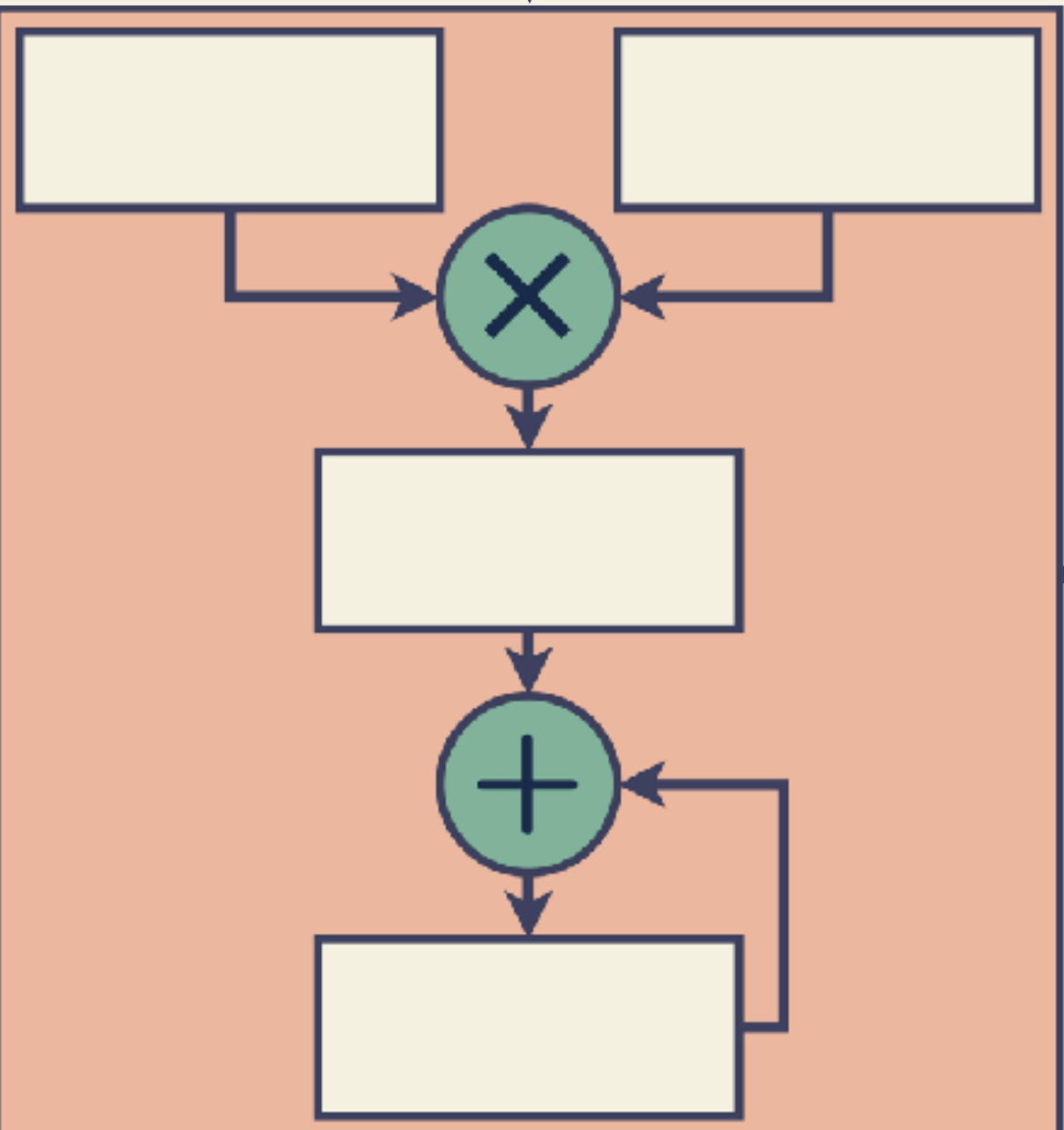
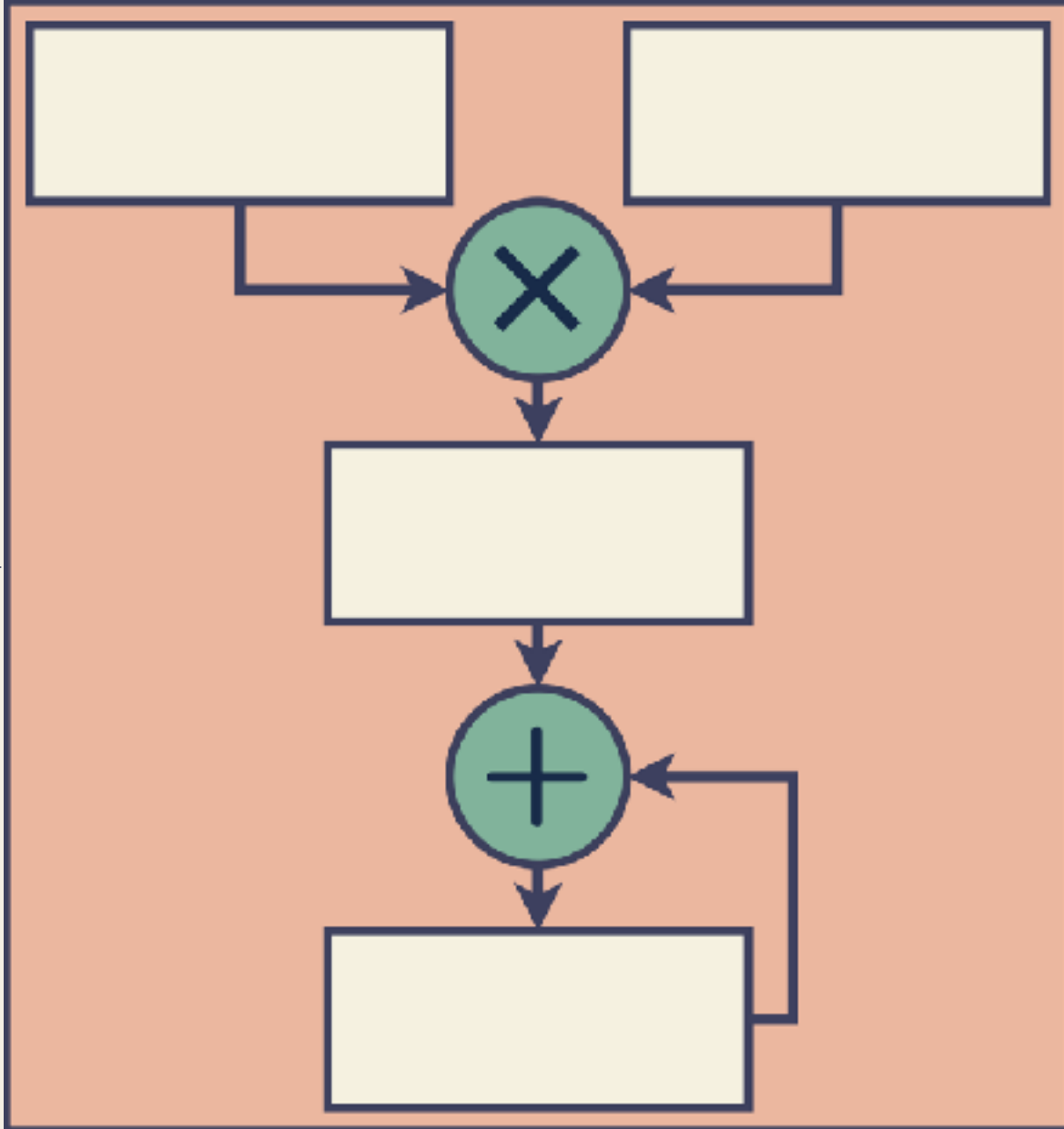
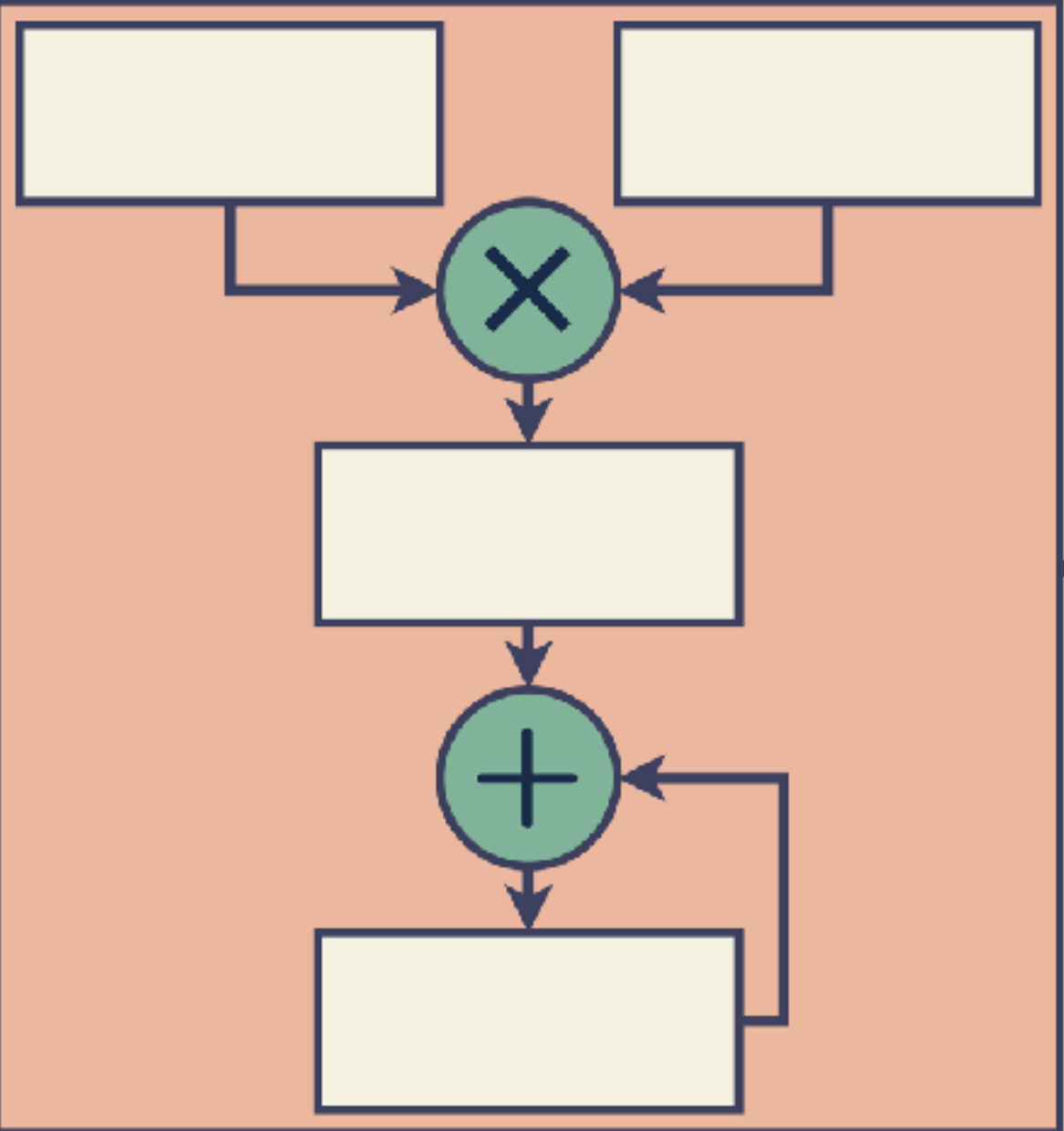
2x2 Matmul in Hardware



Fault Example

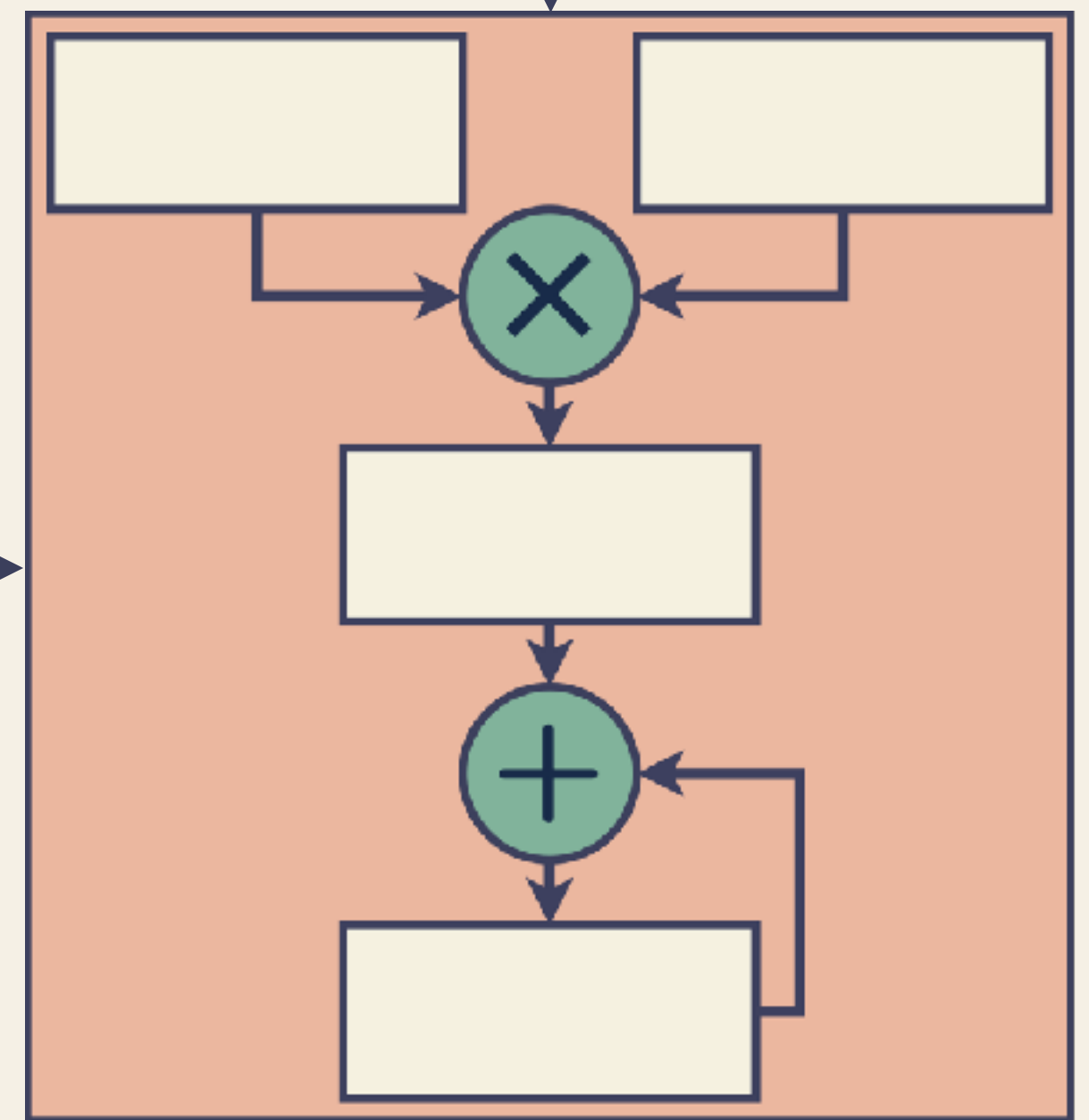
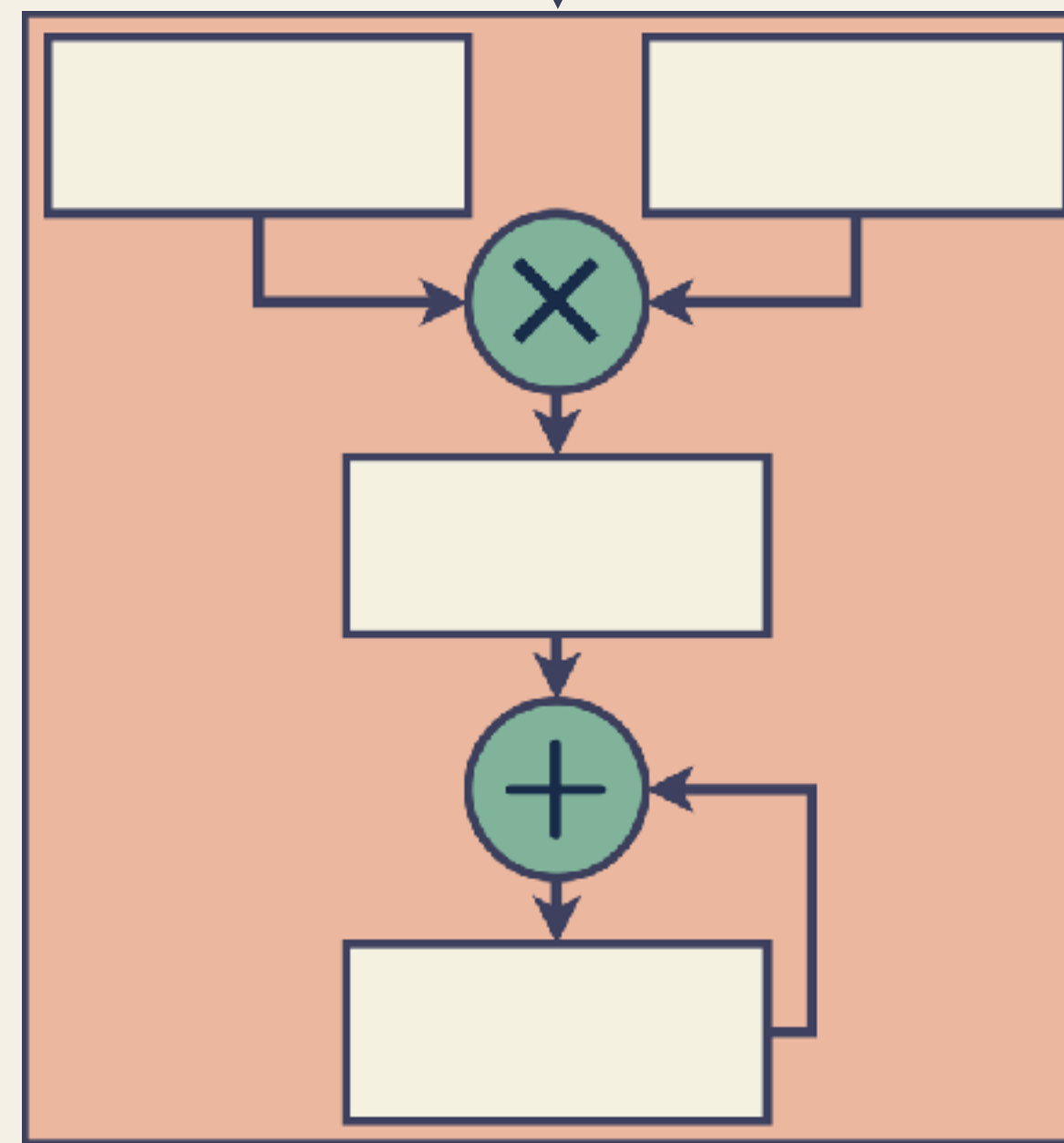
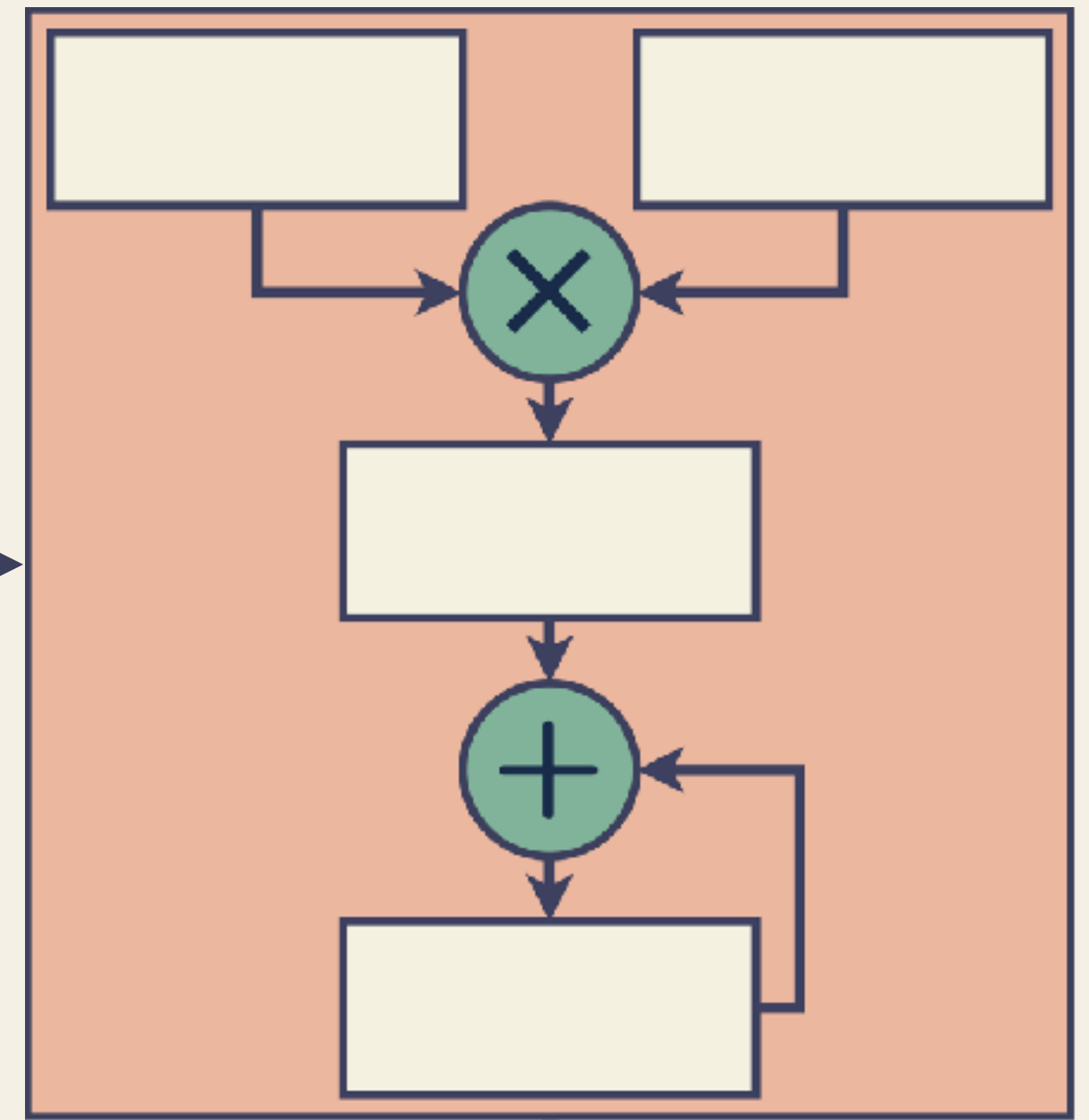
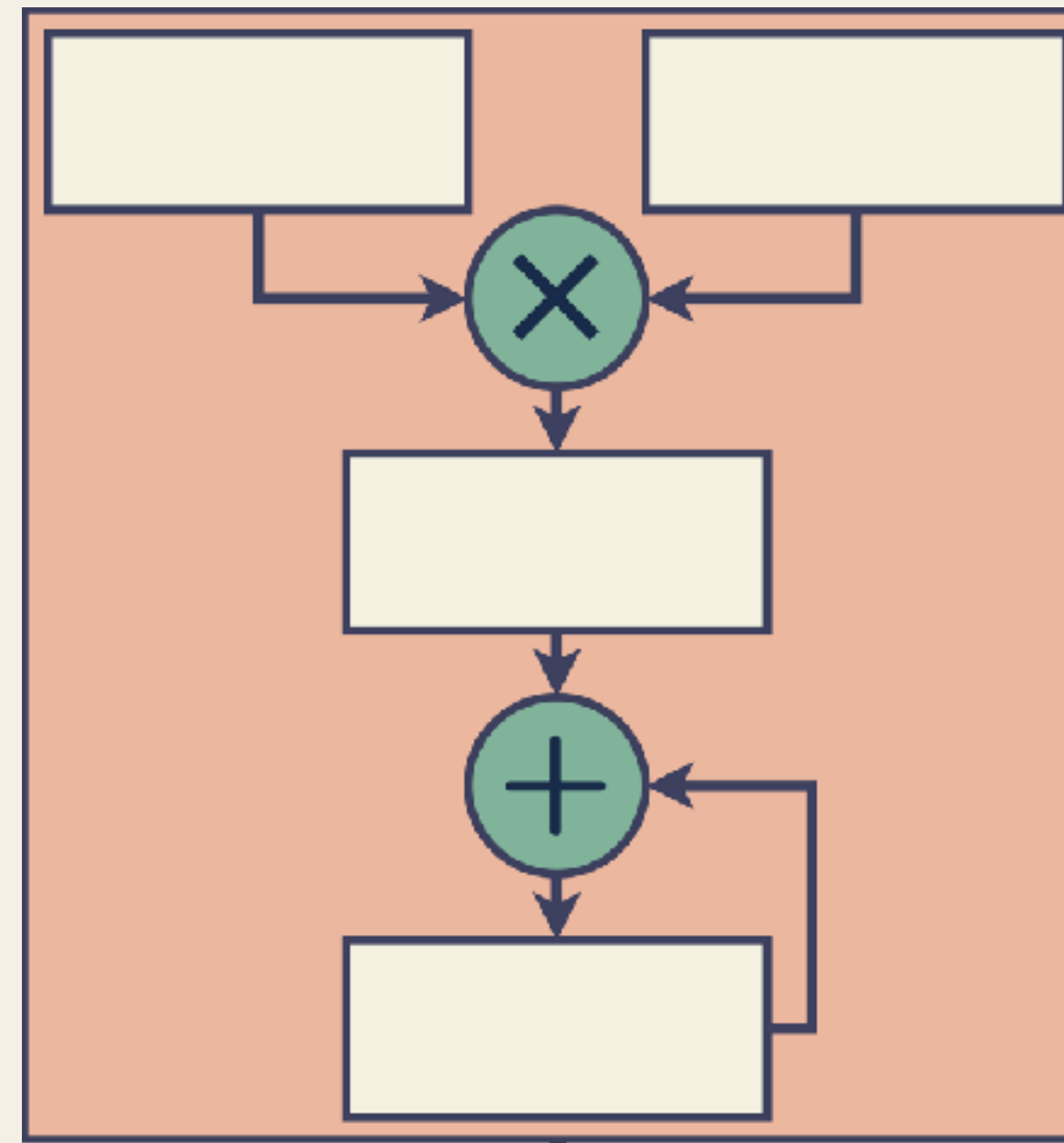
2x2 Matmul in Hardware

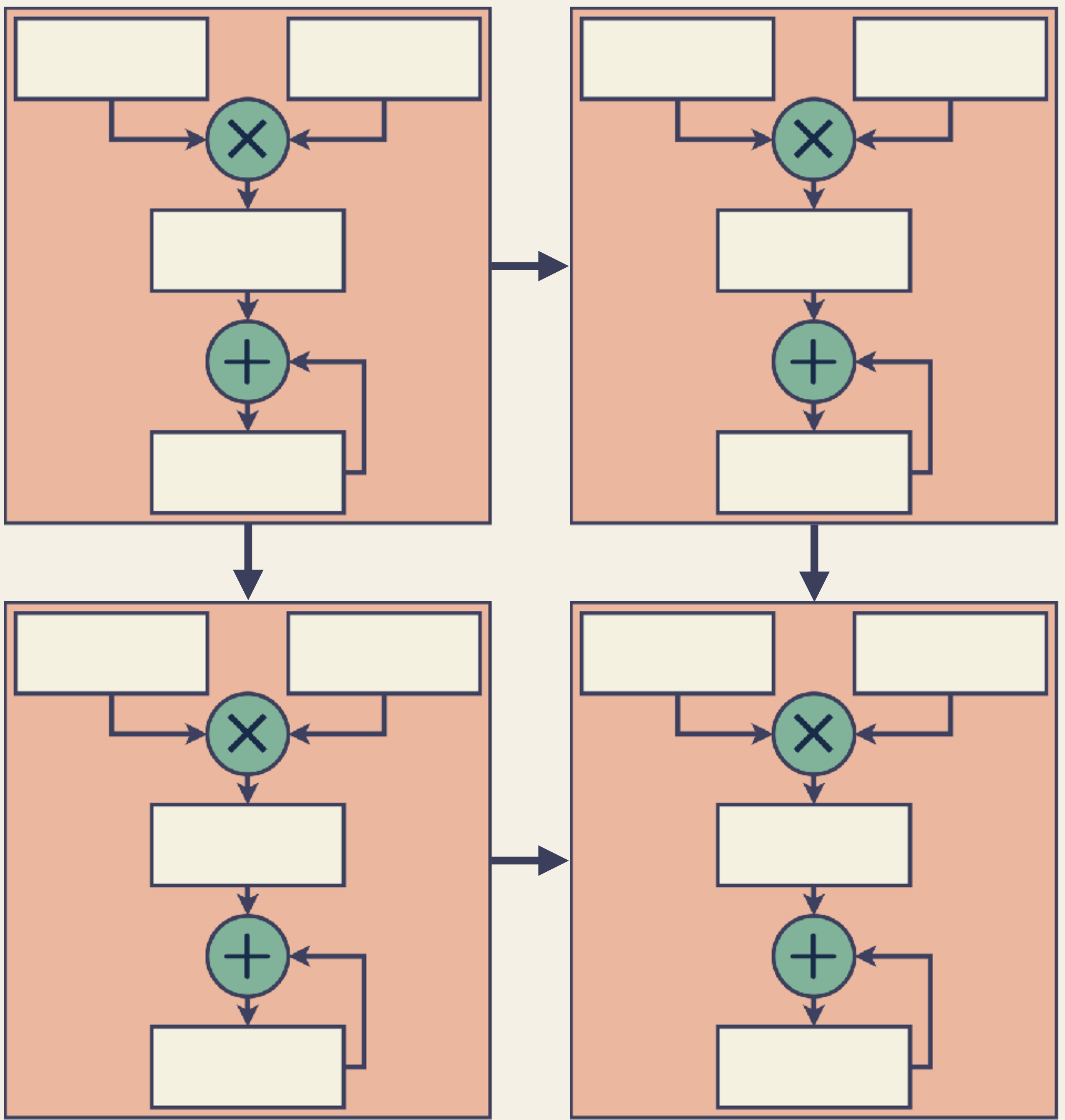
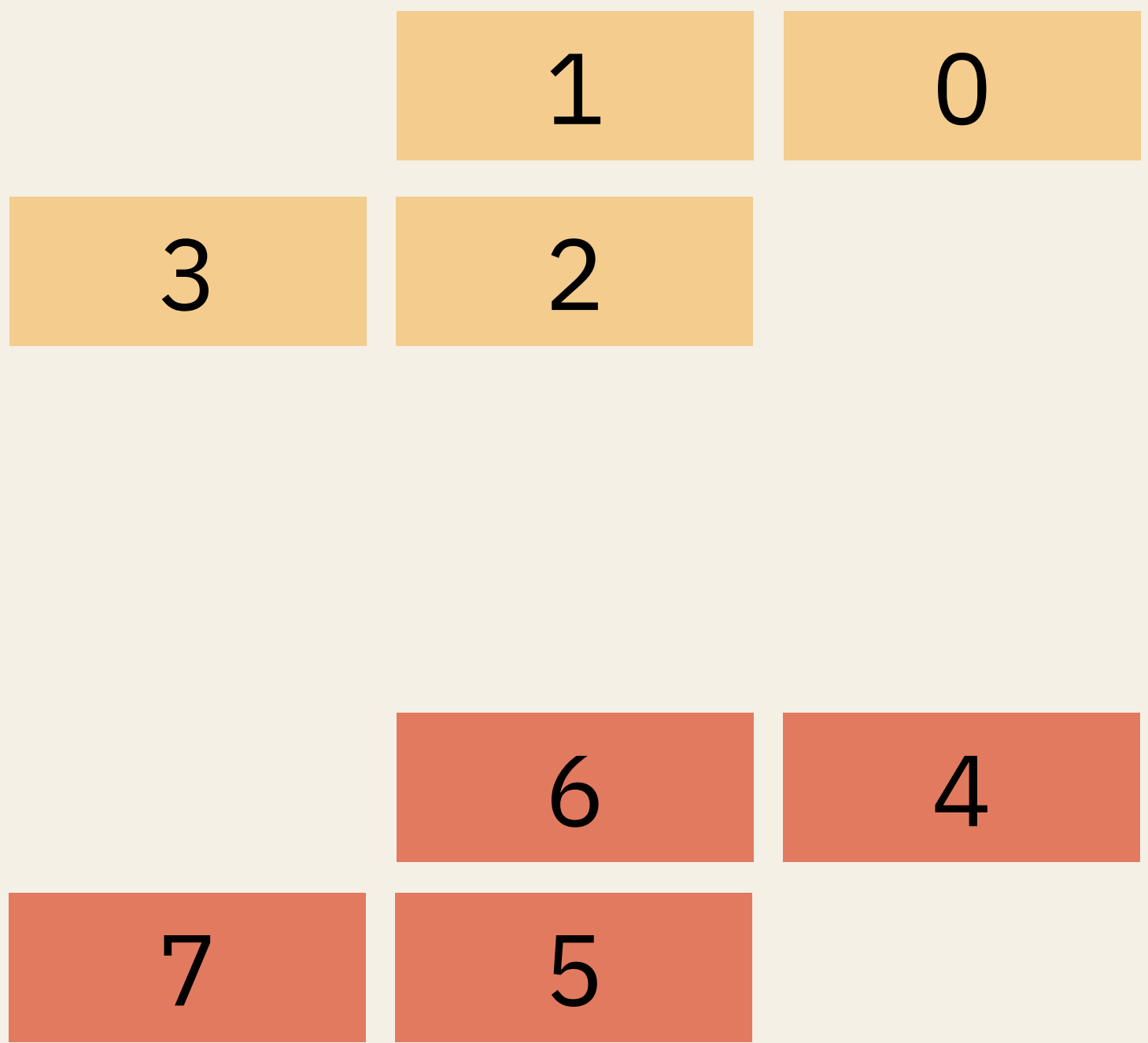




0	1
2	3

4	5
6	7





Moves left → right

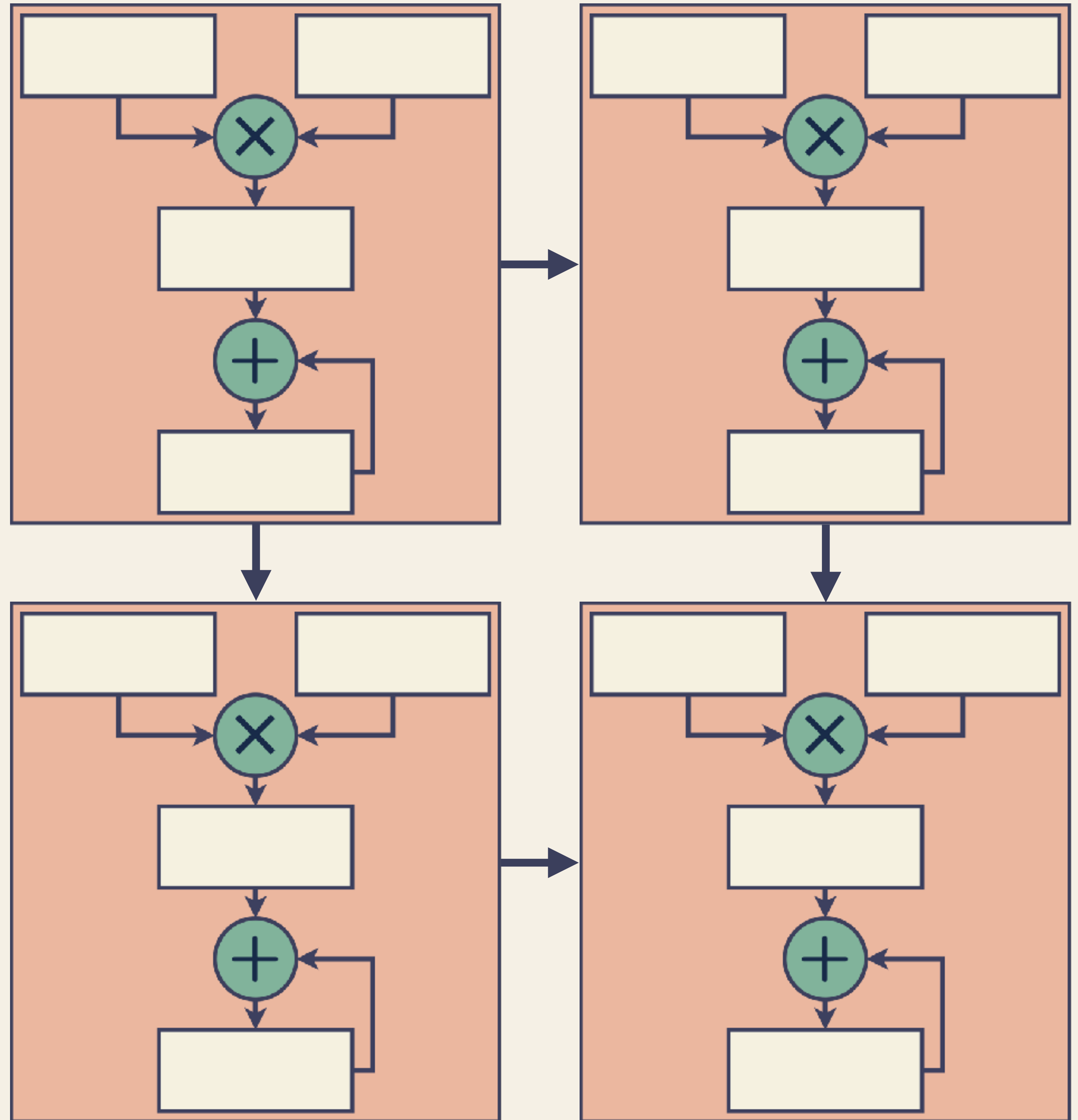
0001 0000

0011 0010

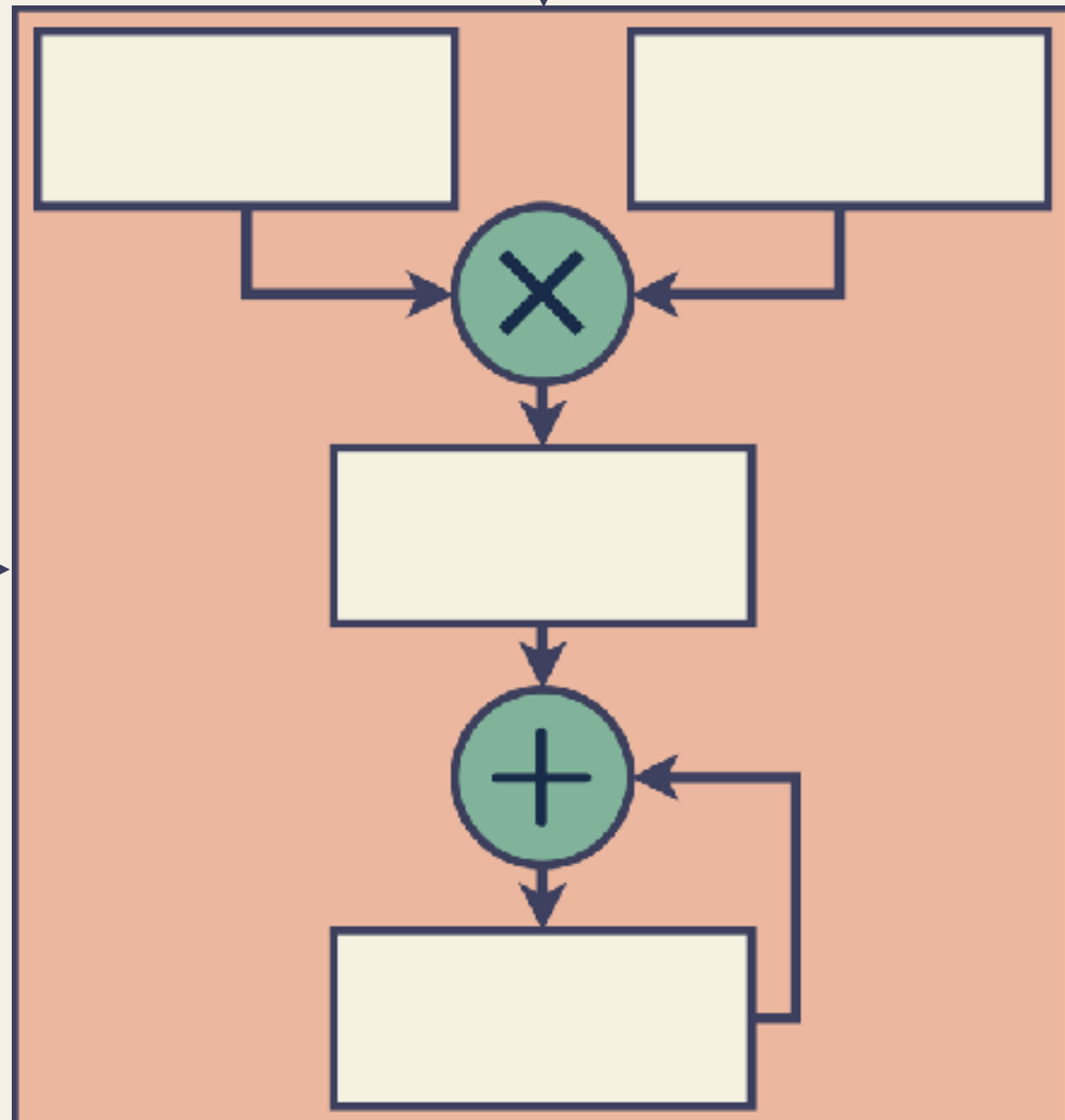
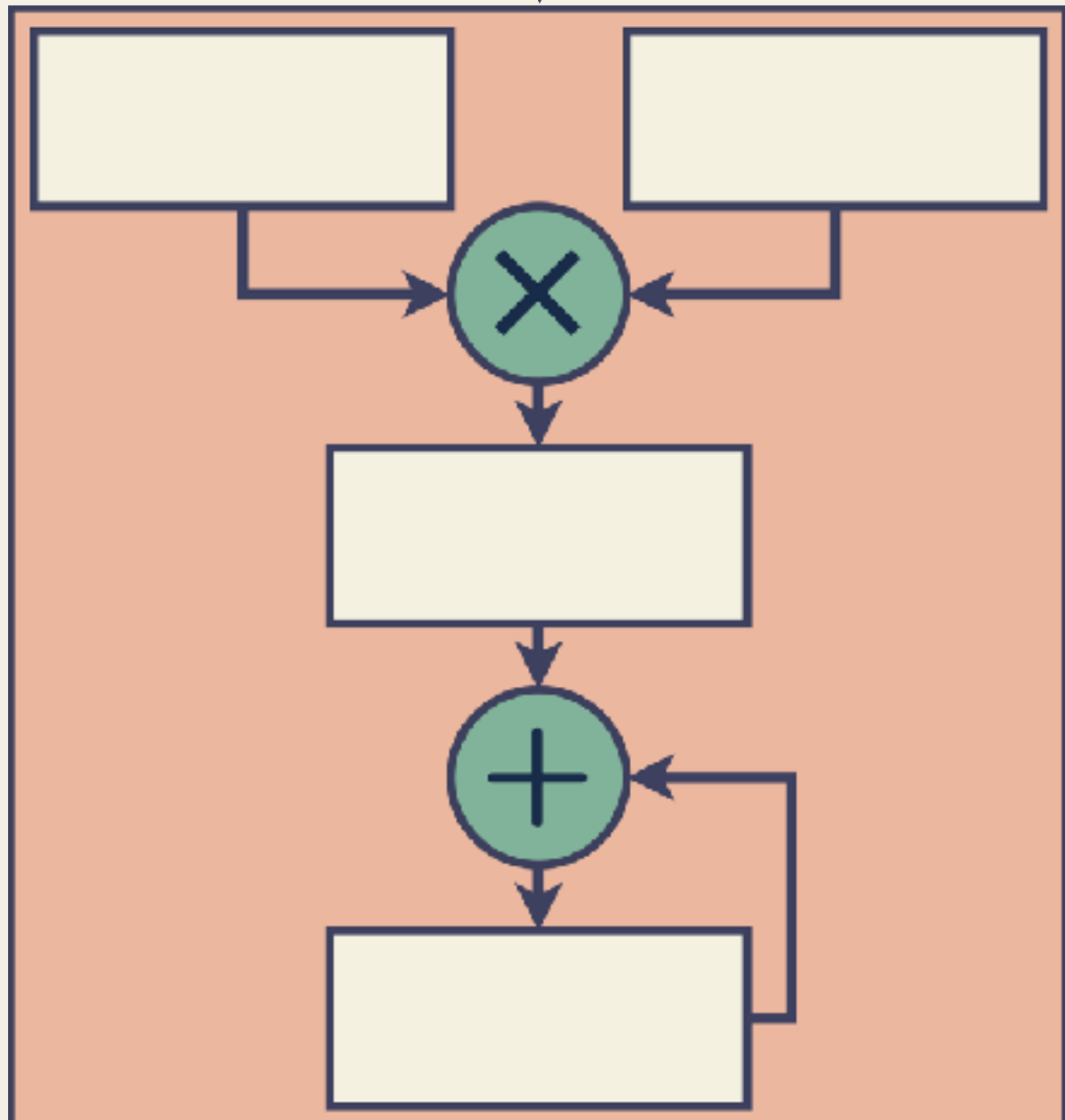
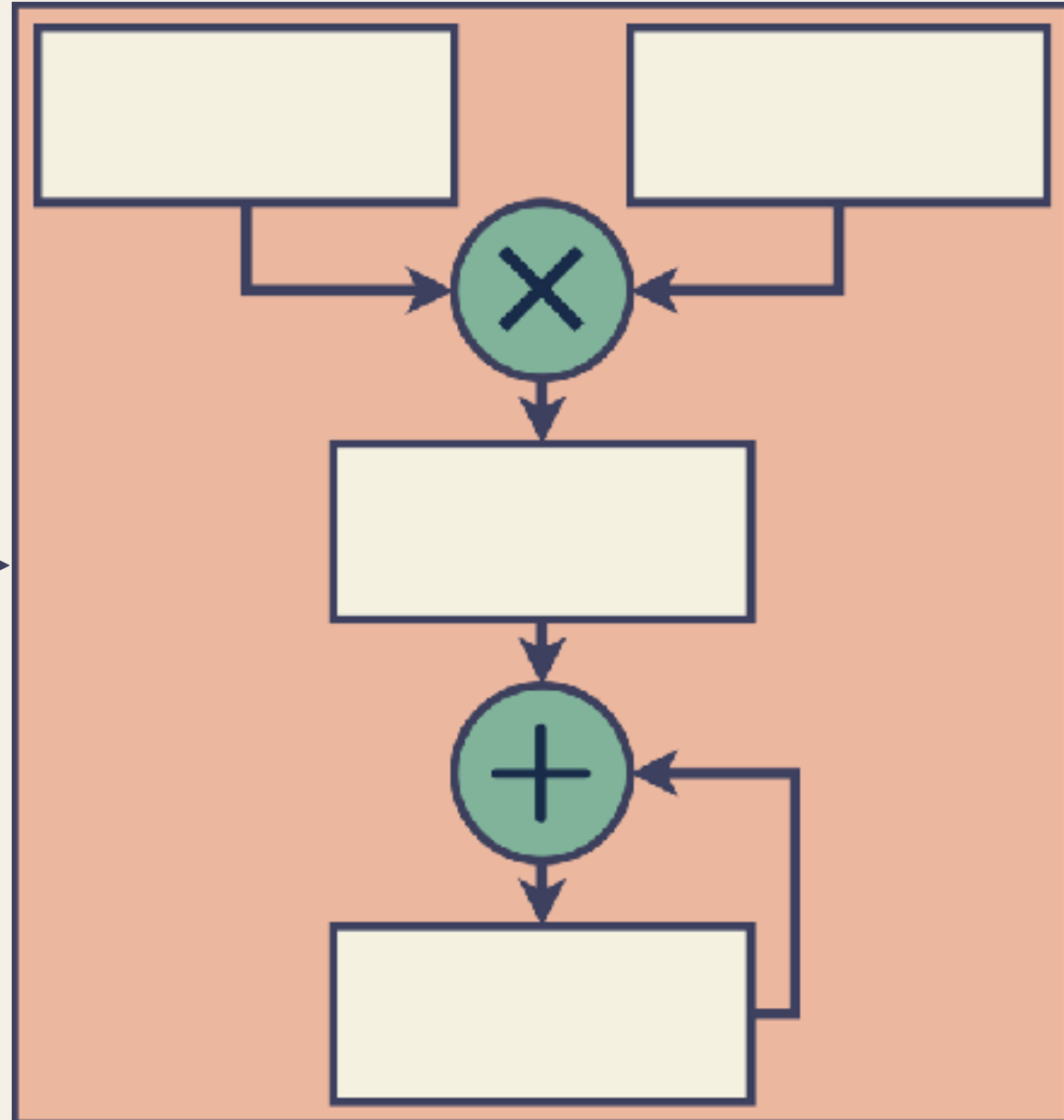
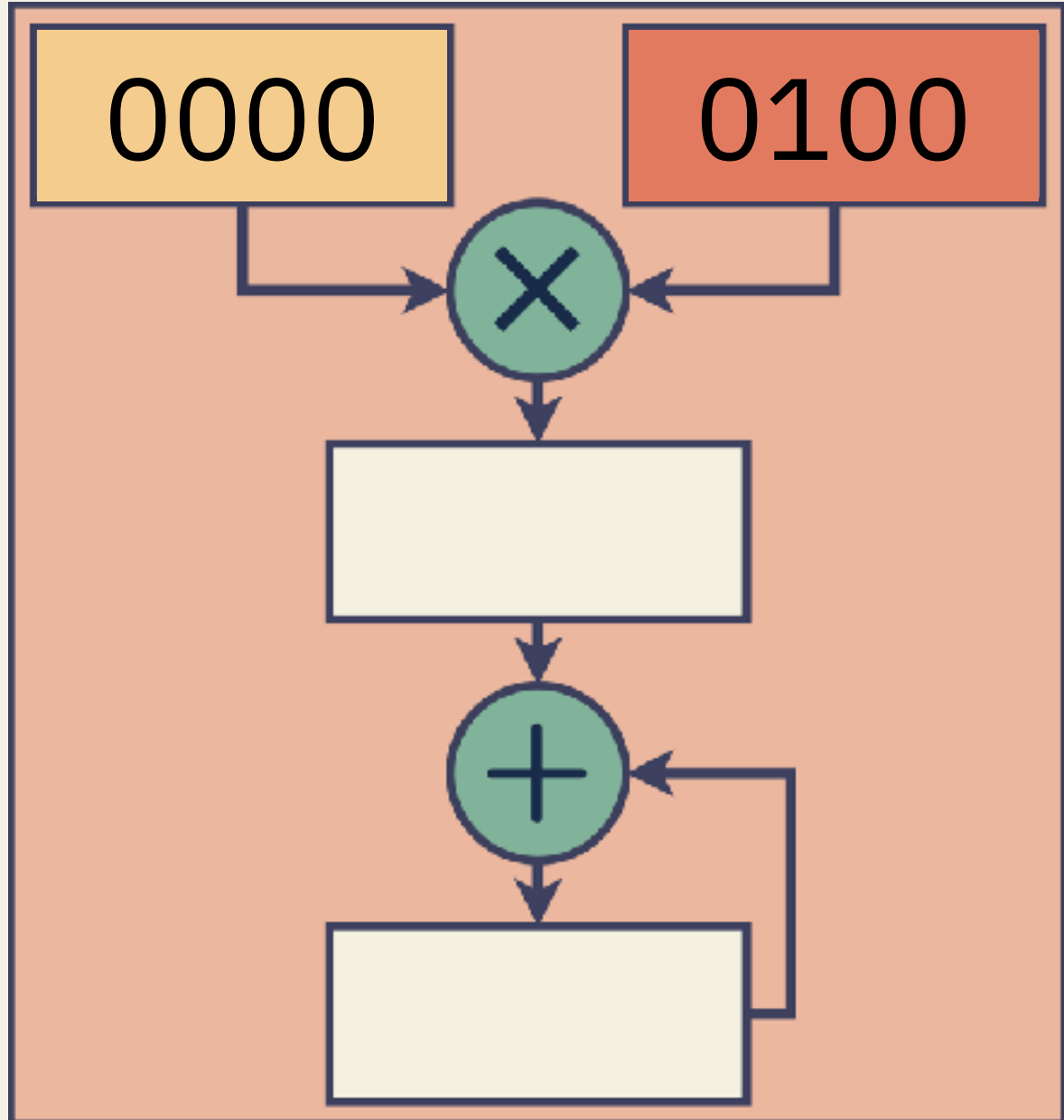
Moves top → down

0110 0100

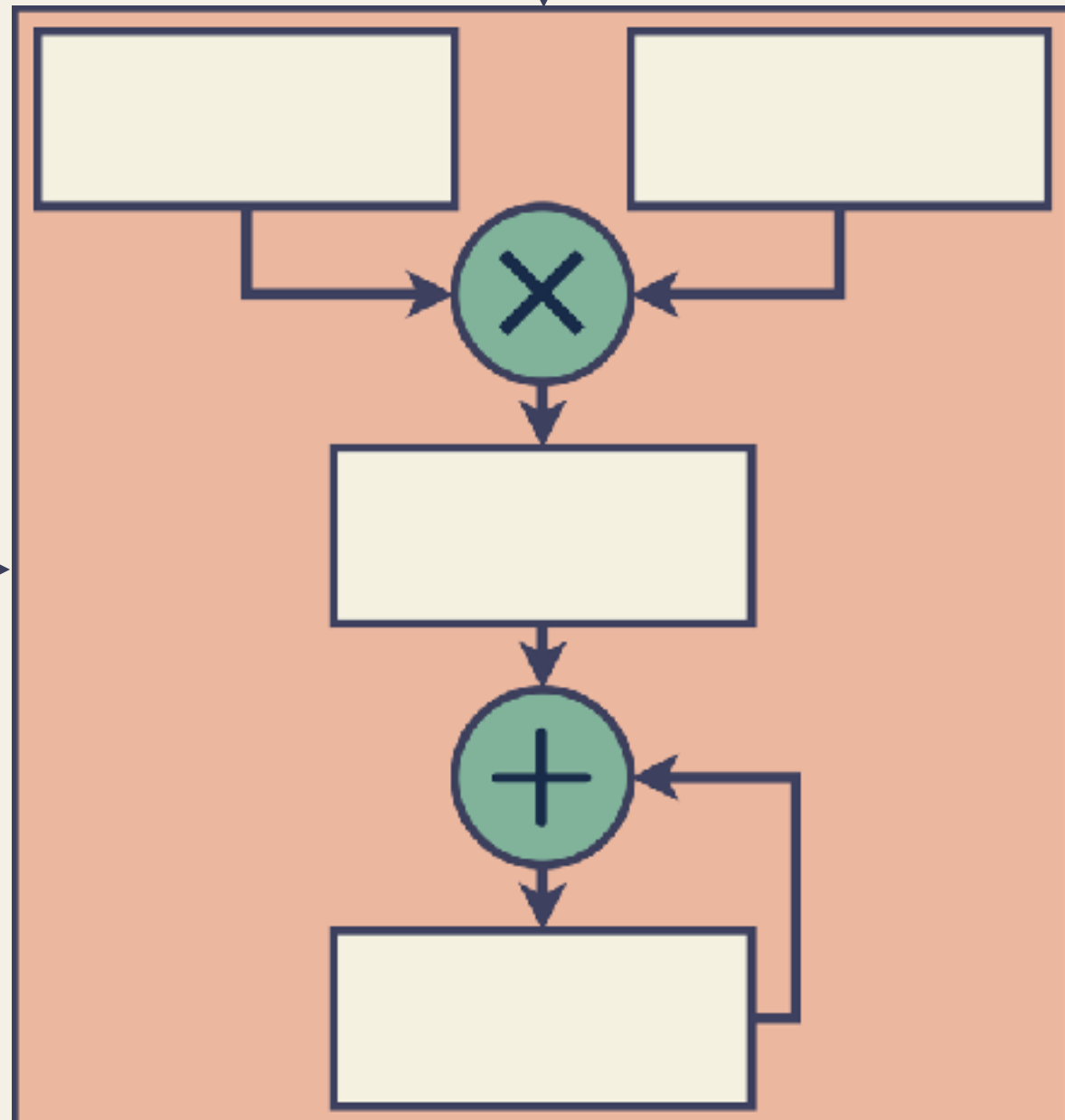
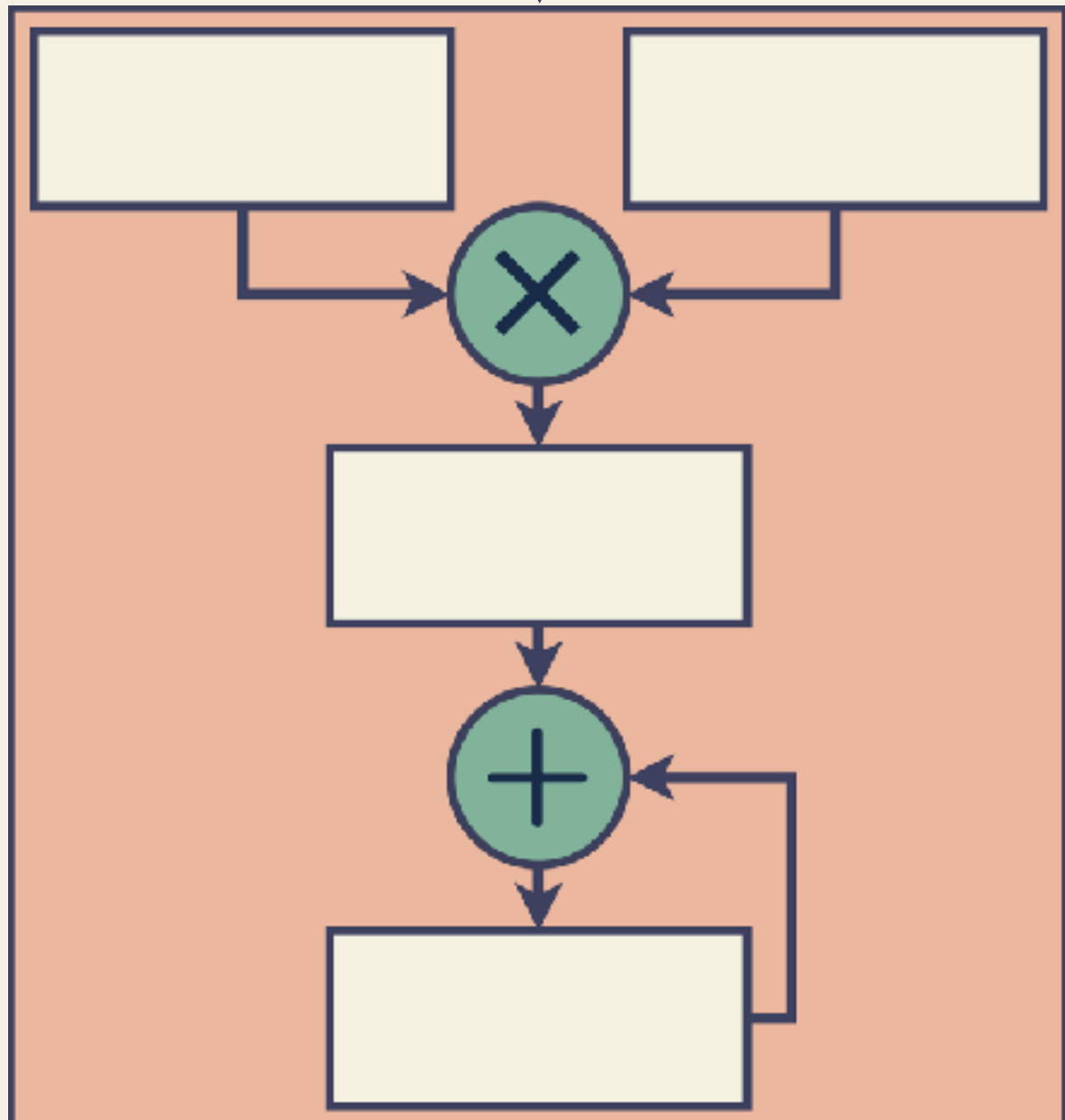
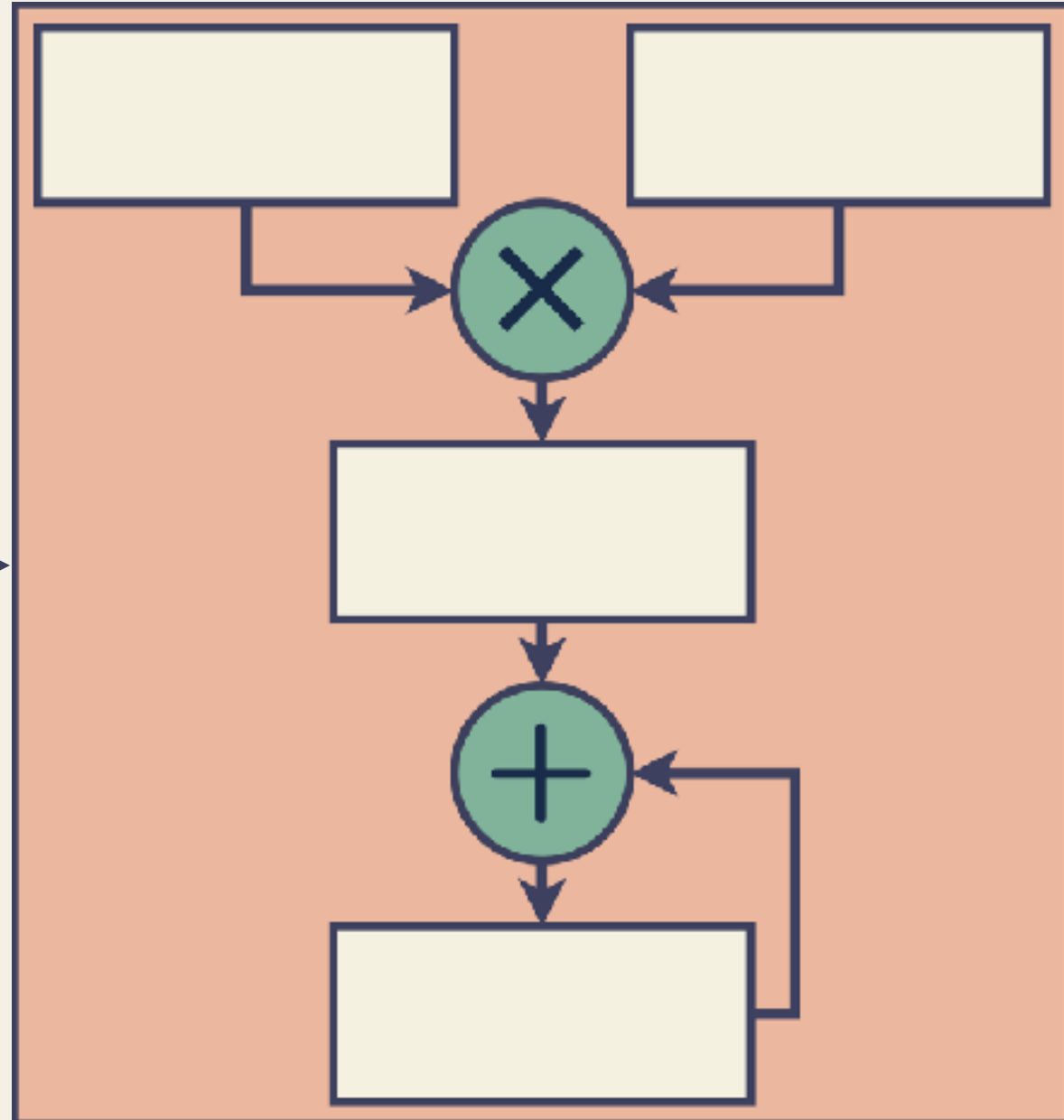
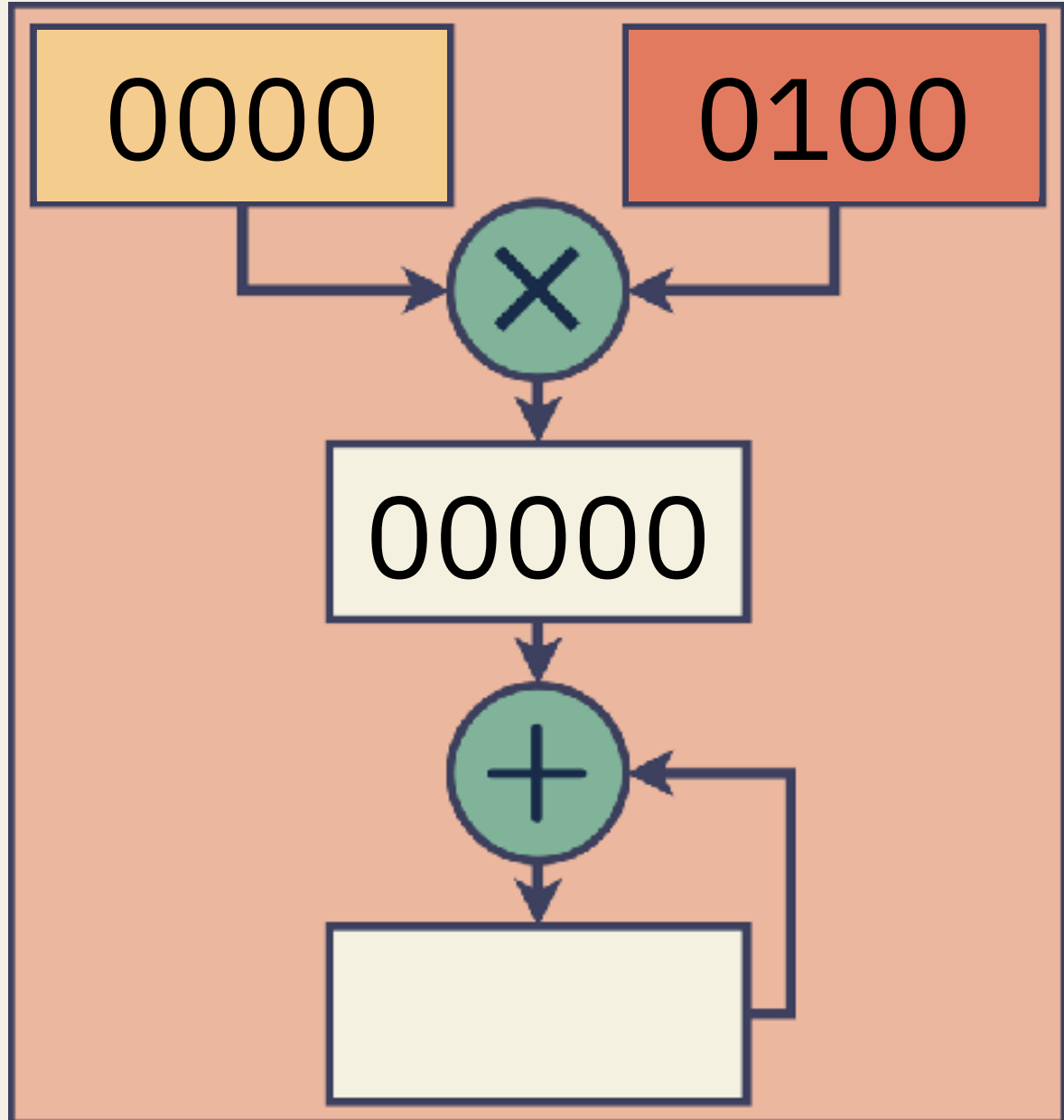
0111 0101



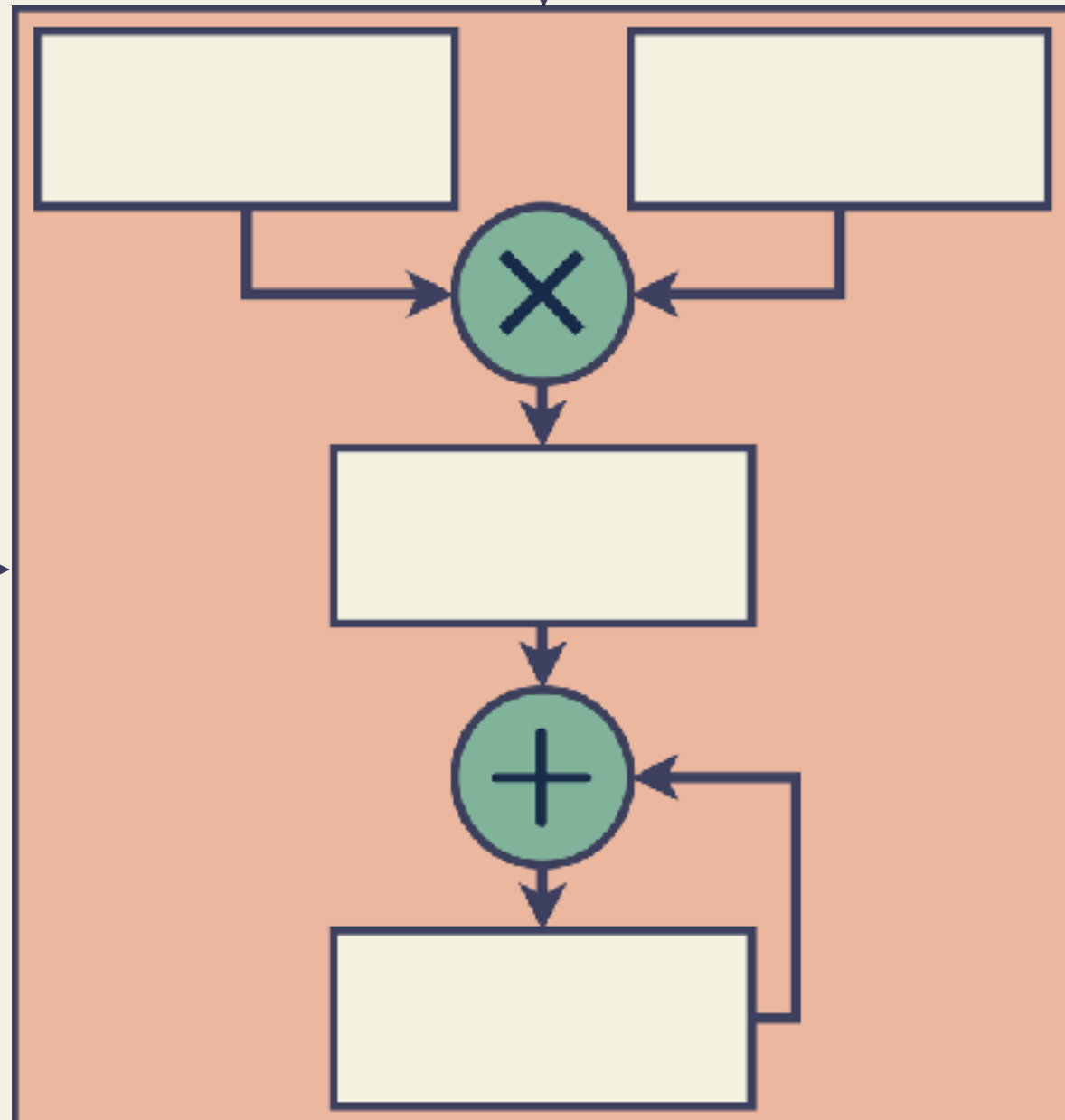
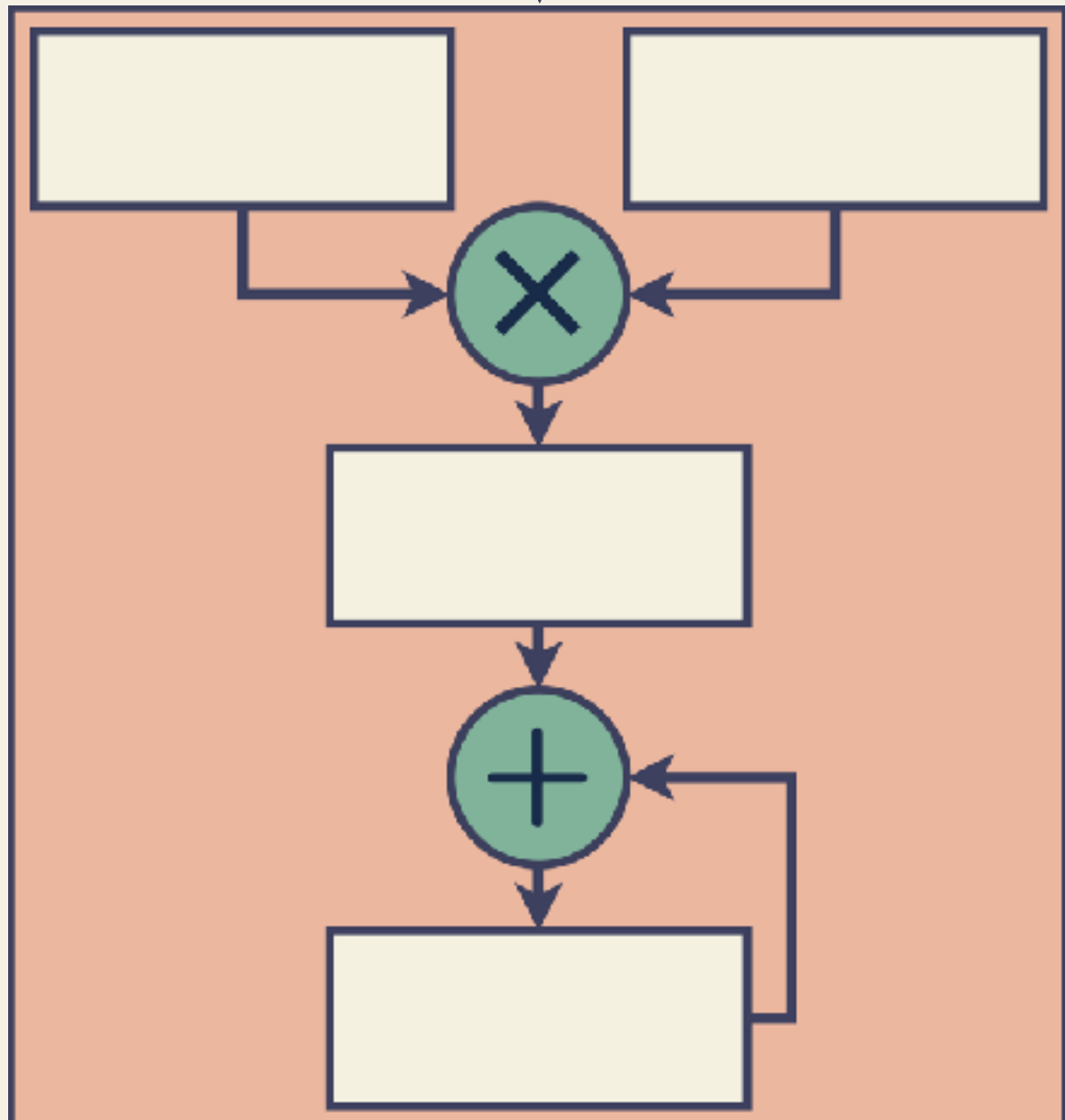
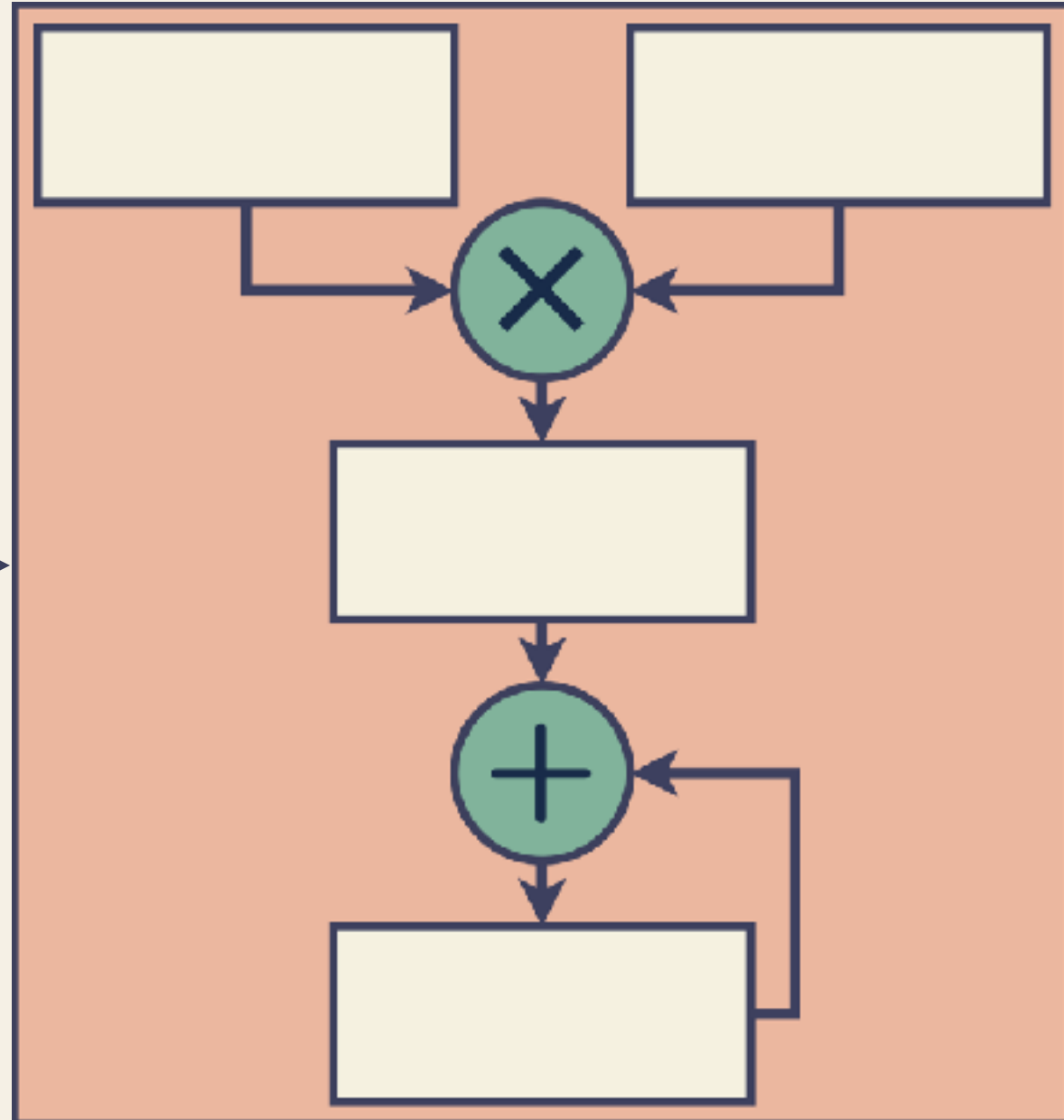
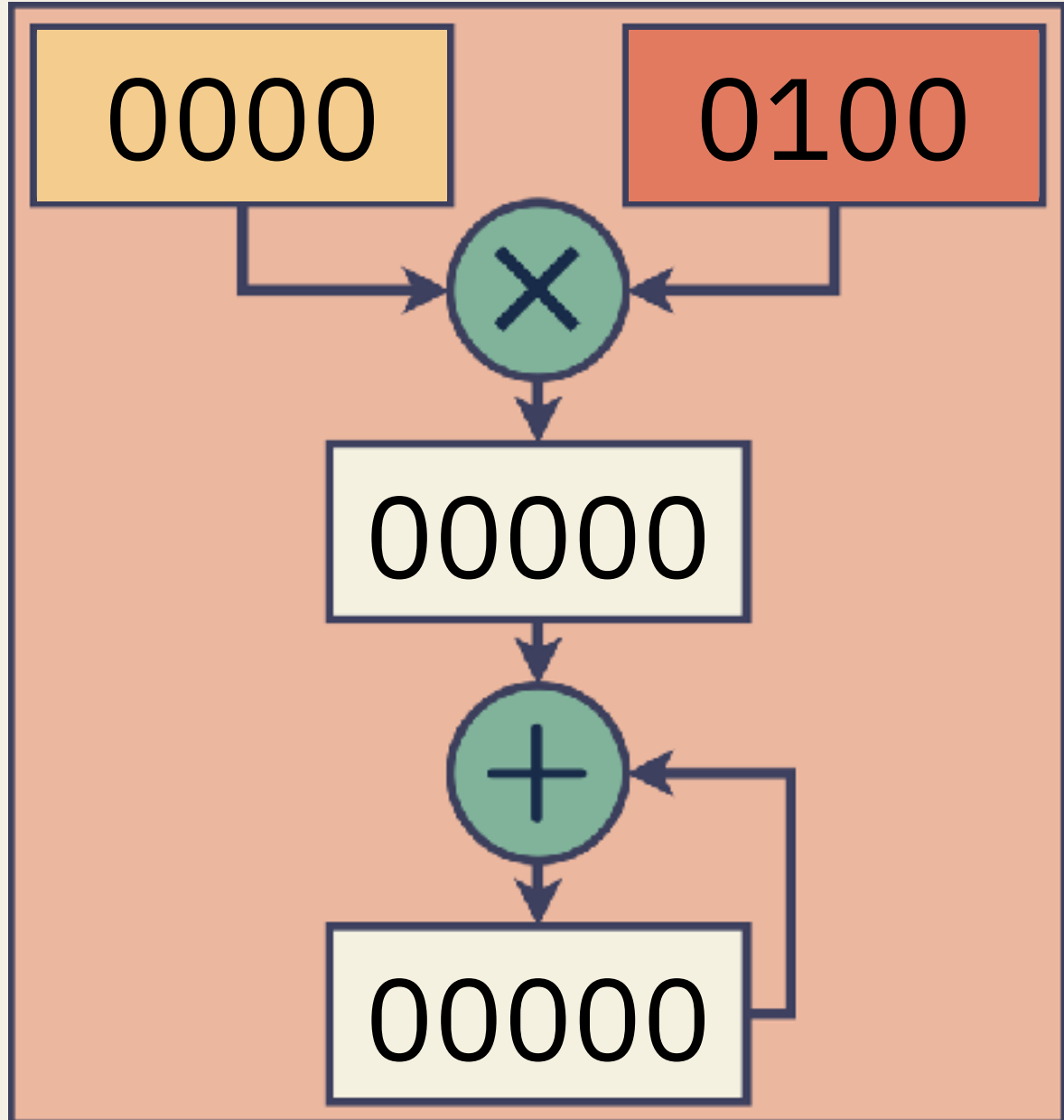
	0001
0011	0010
	0110
0111	0101



	0001
0011	0010
	0110
0111	0101

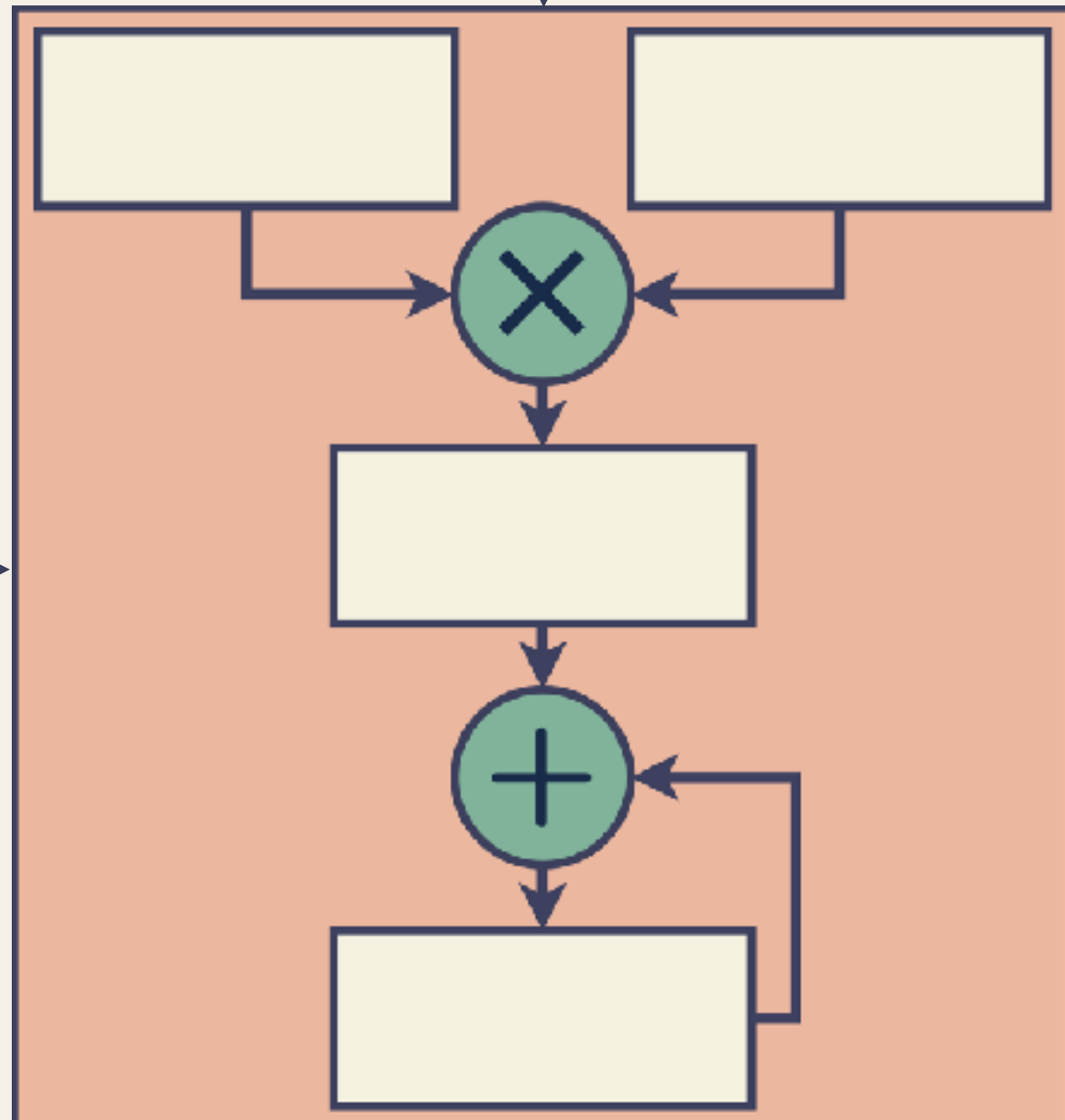
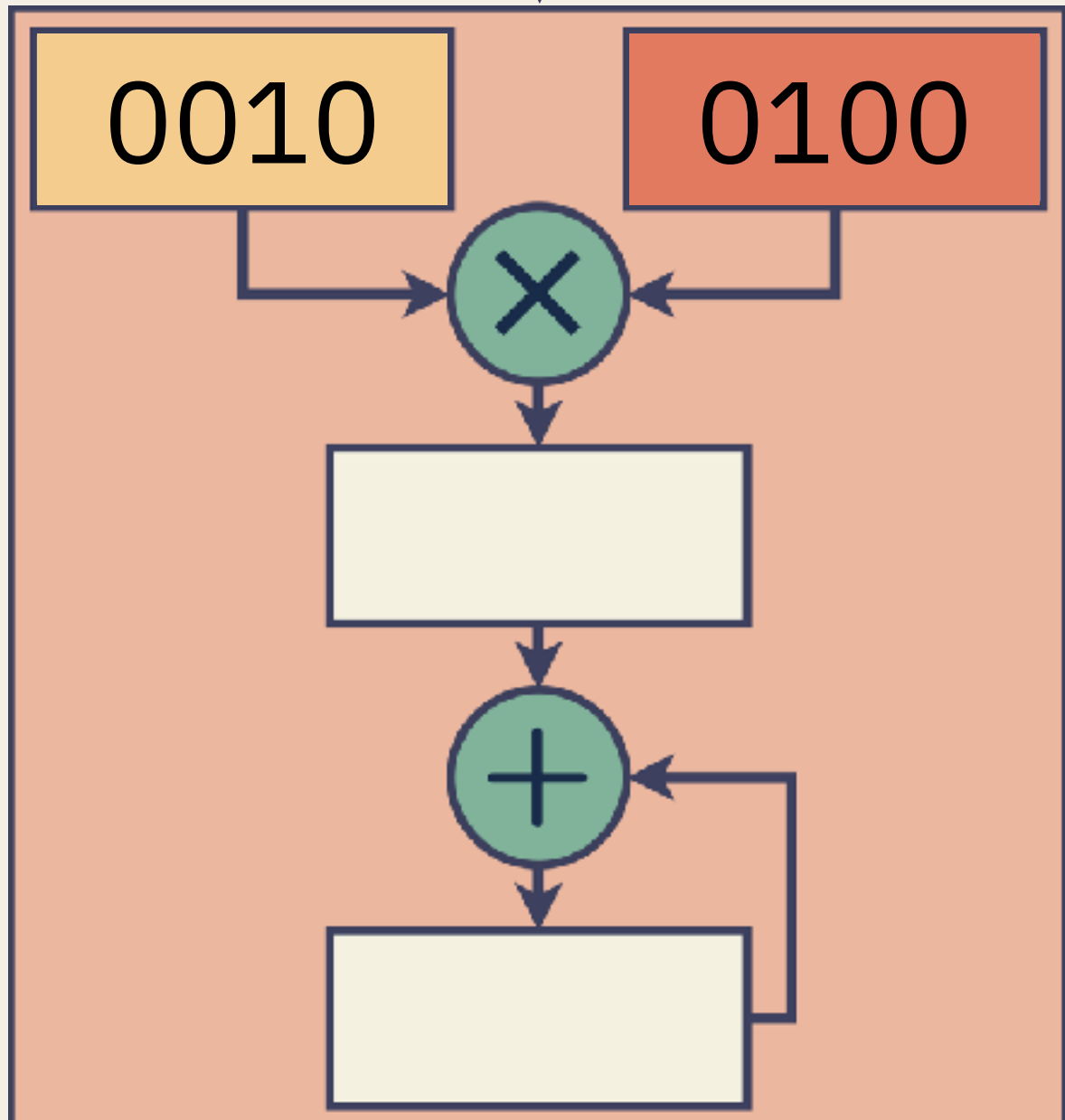
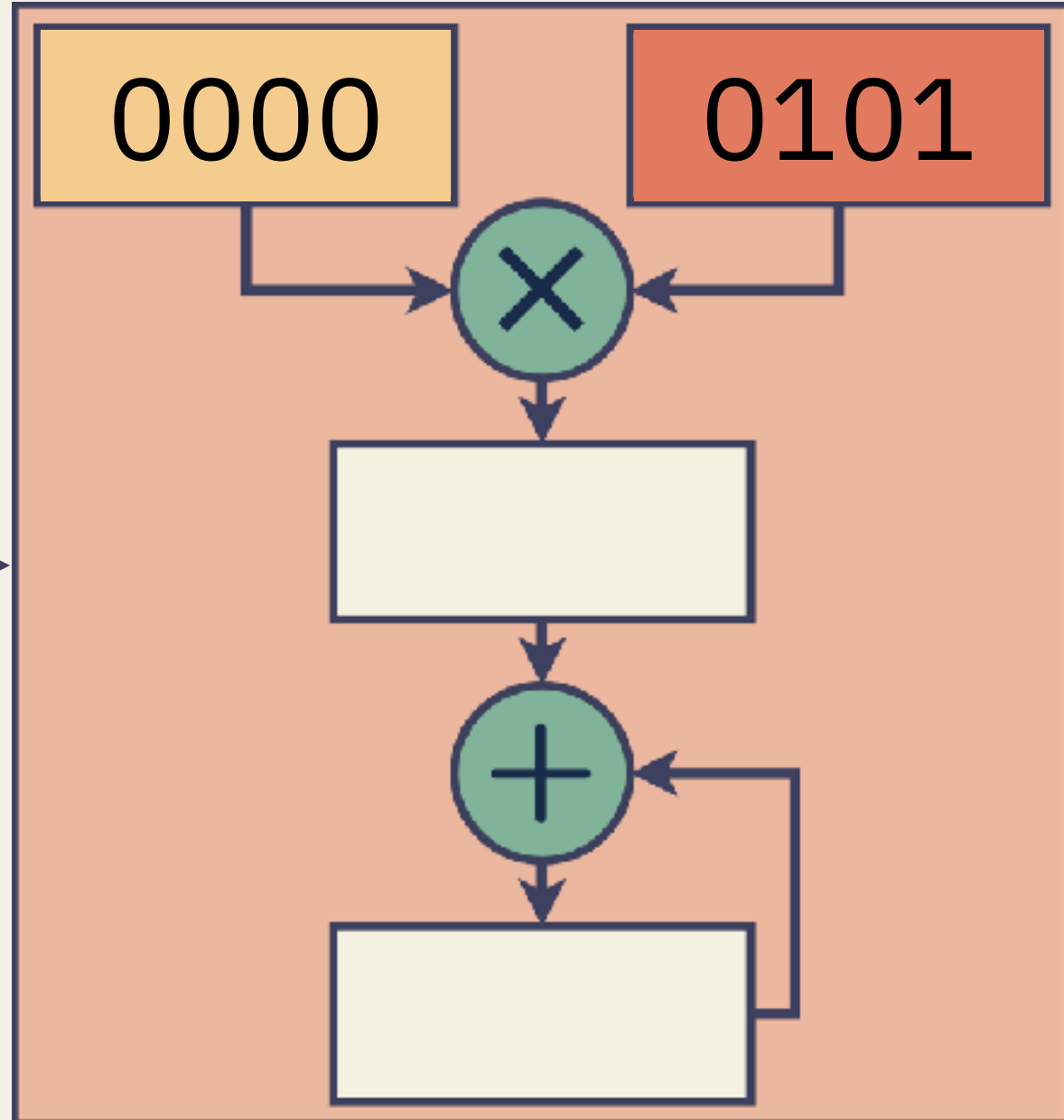
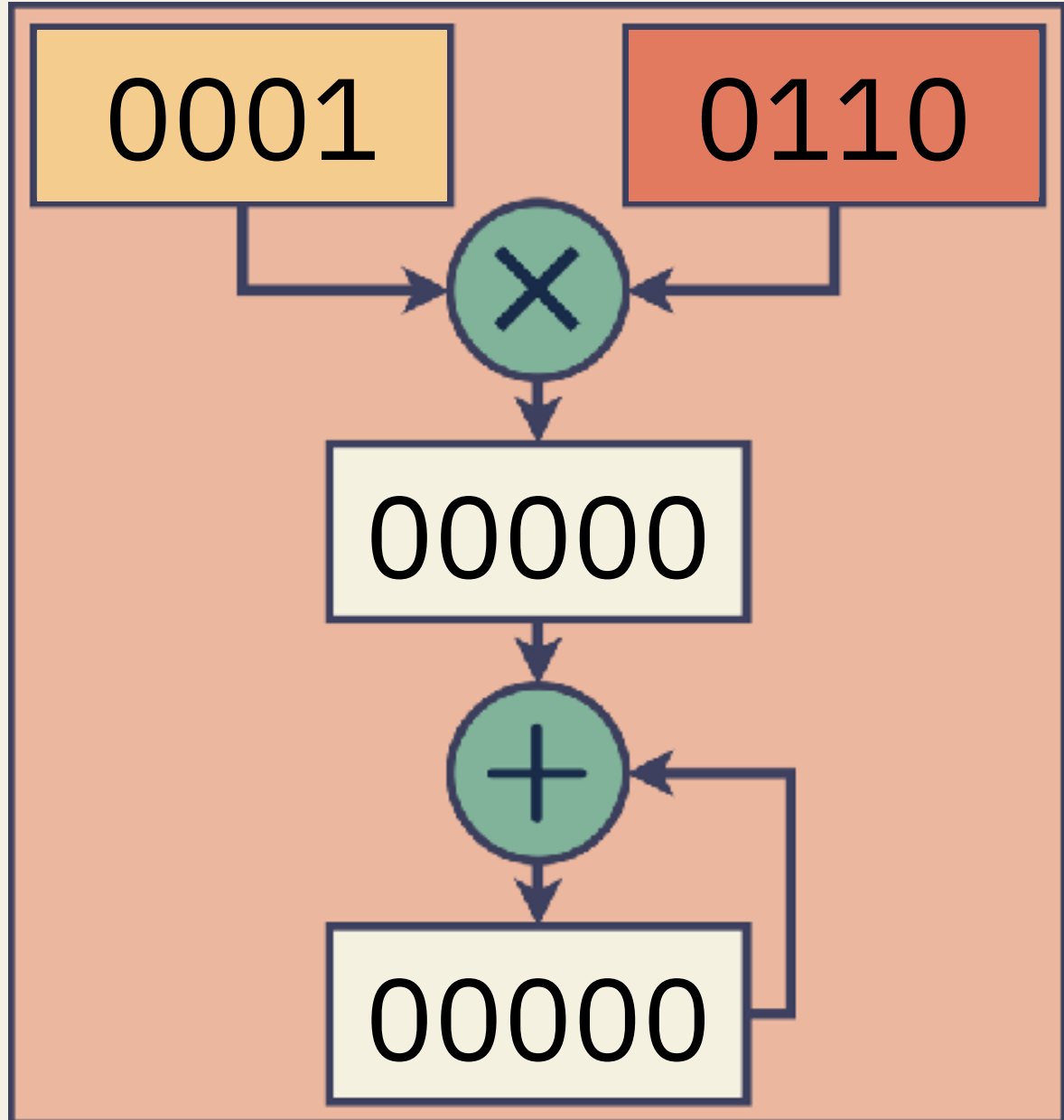


	0001
0011	0010
	0110
0111	0101



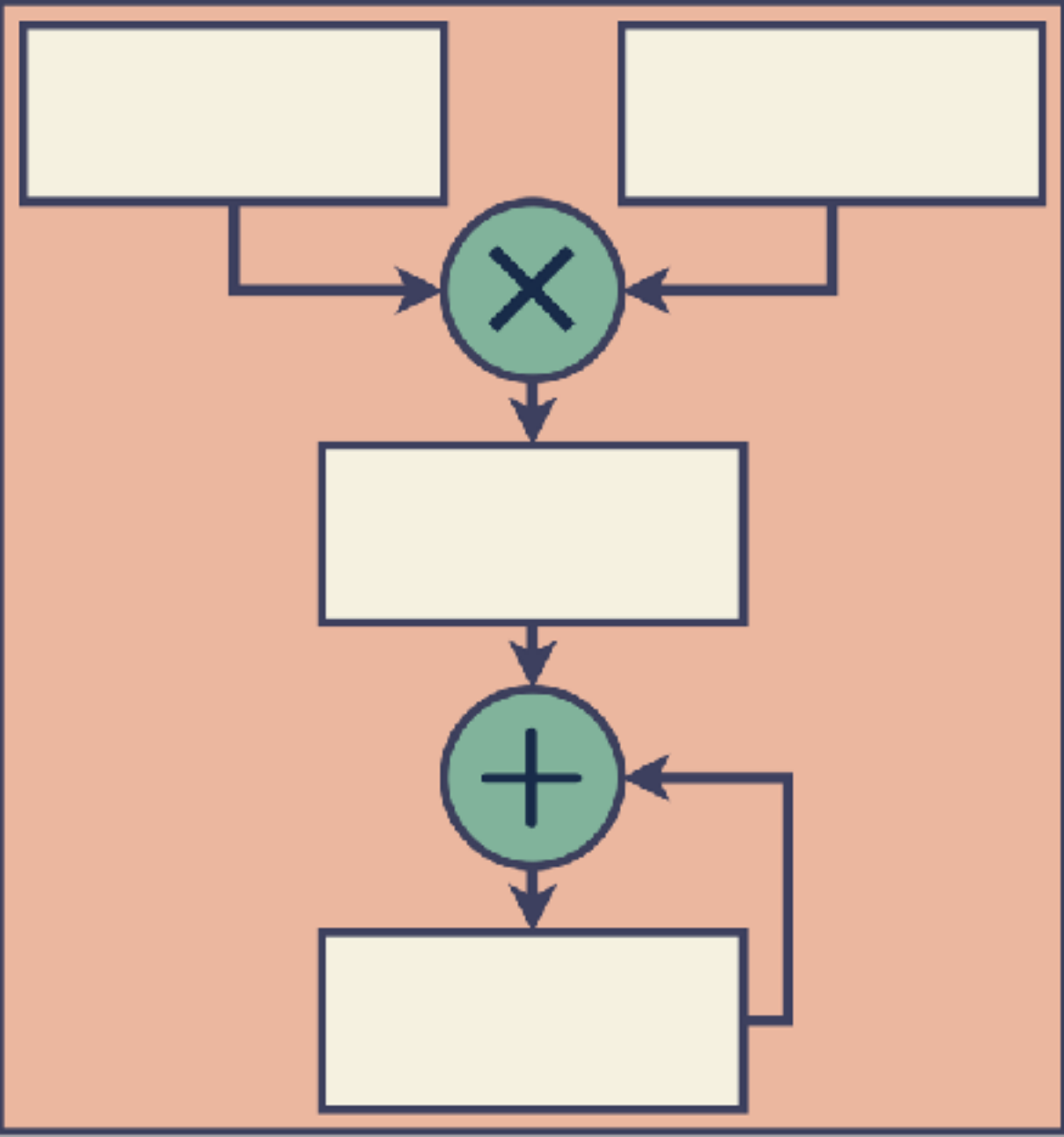
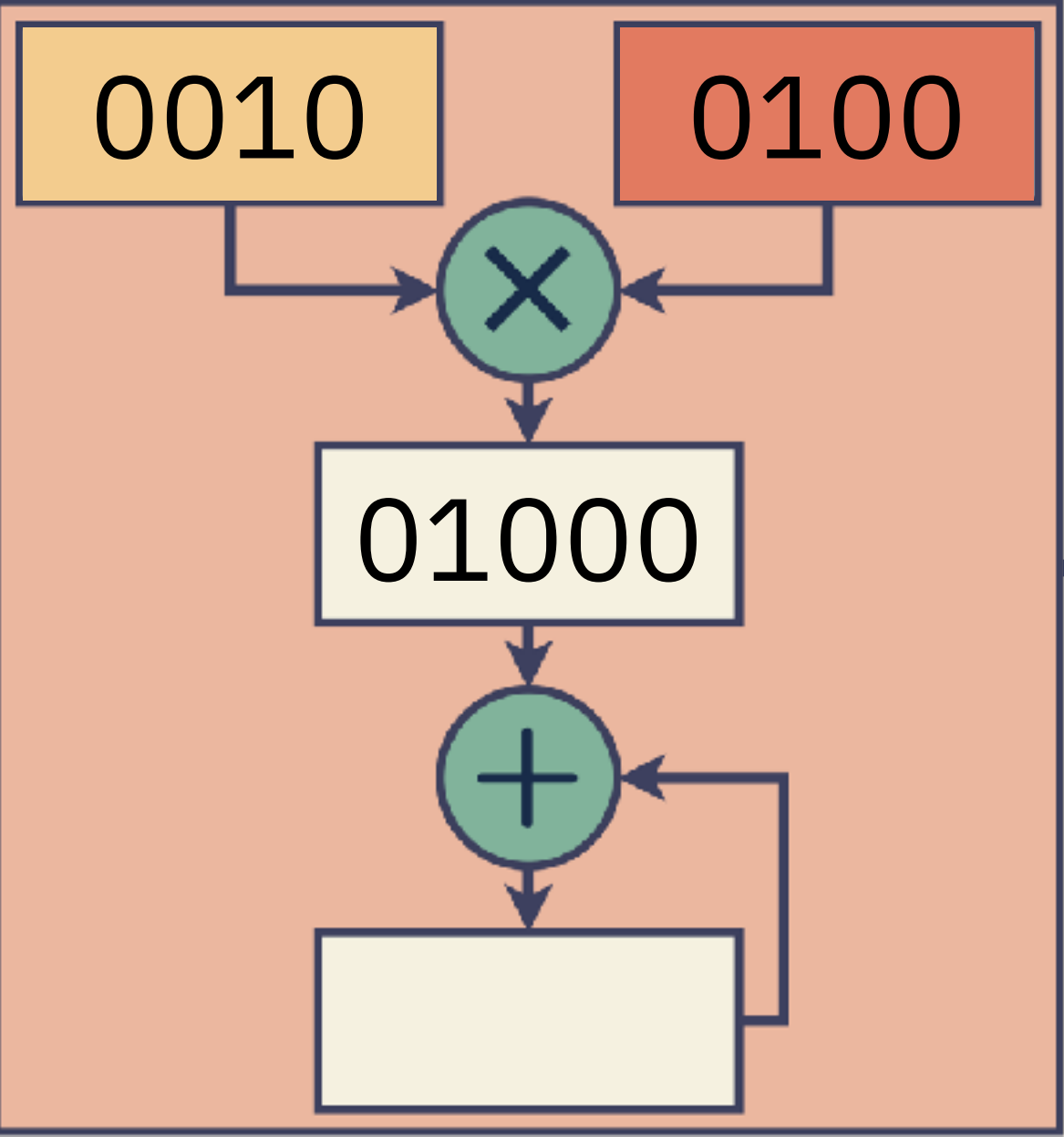
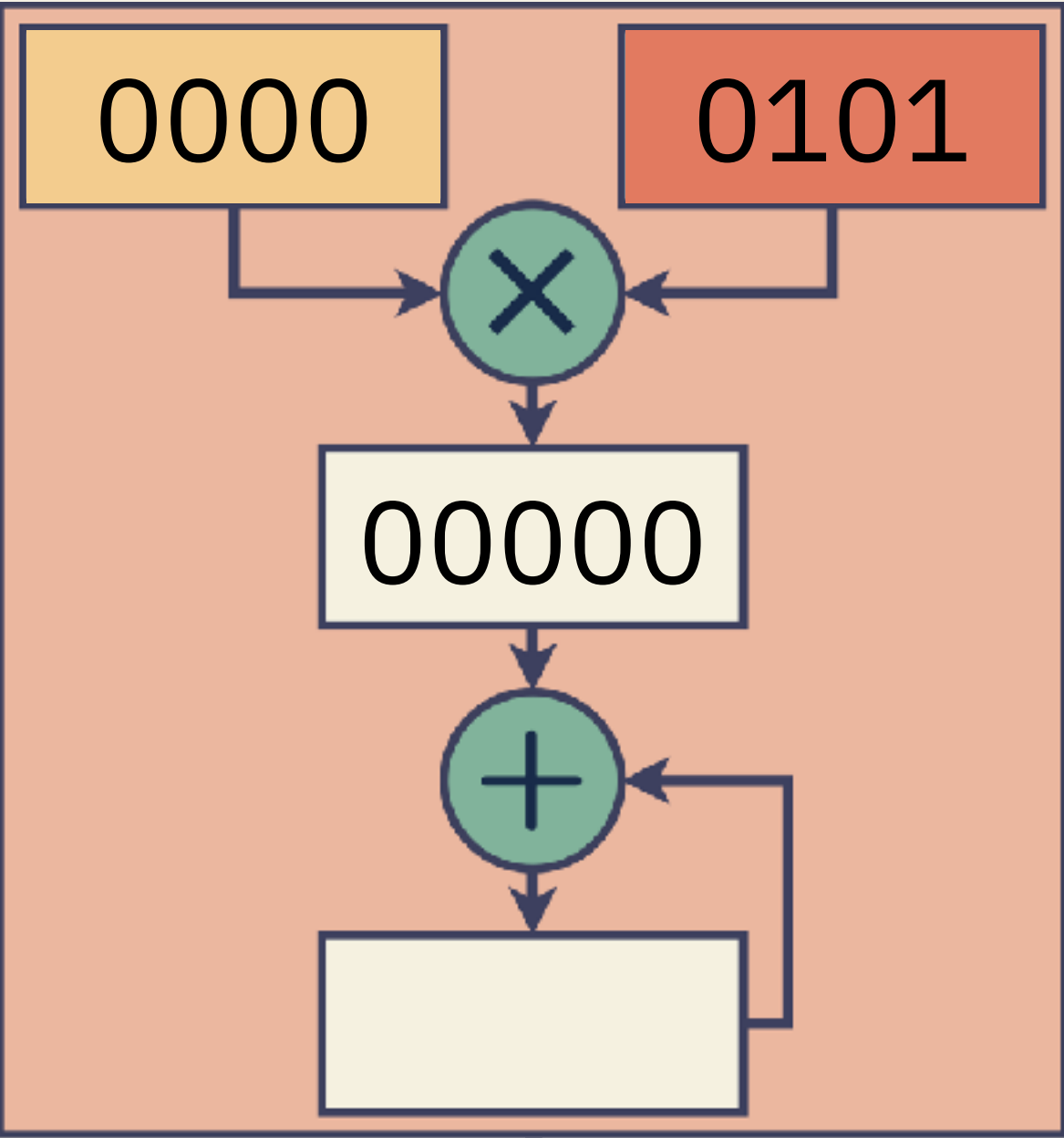
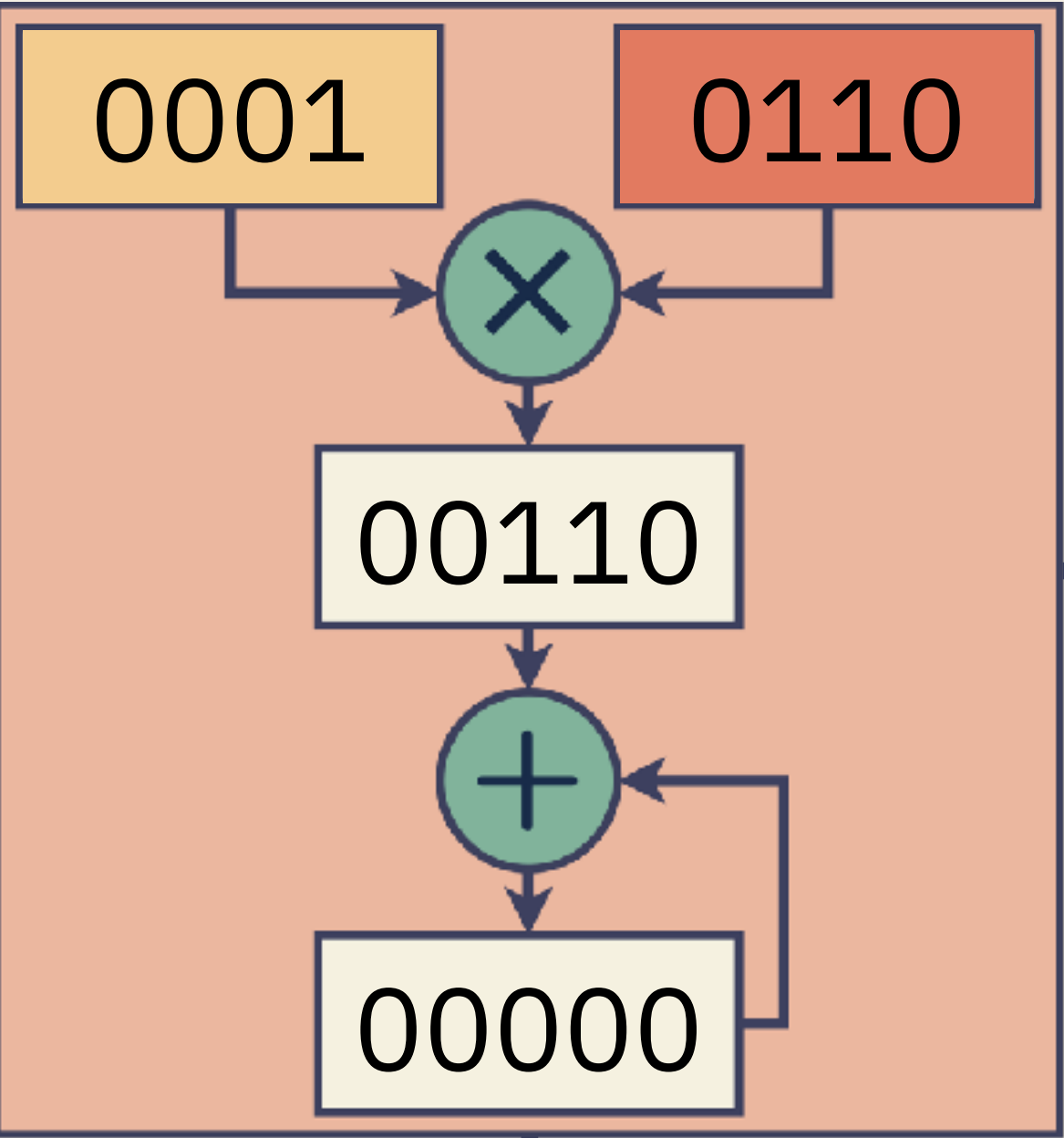
0011

0111



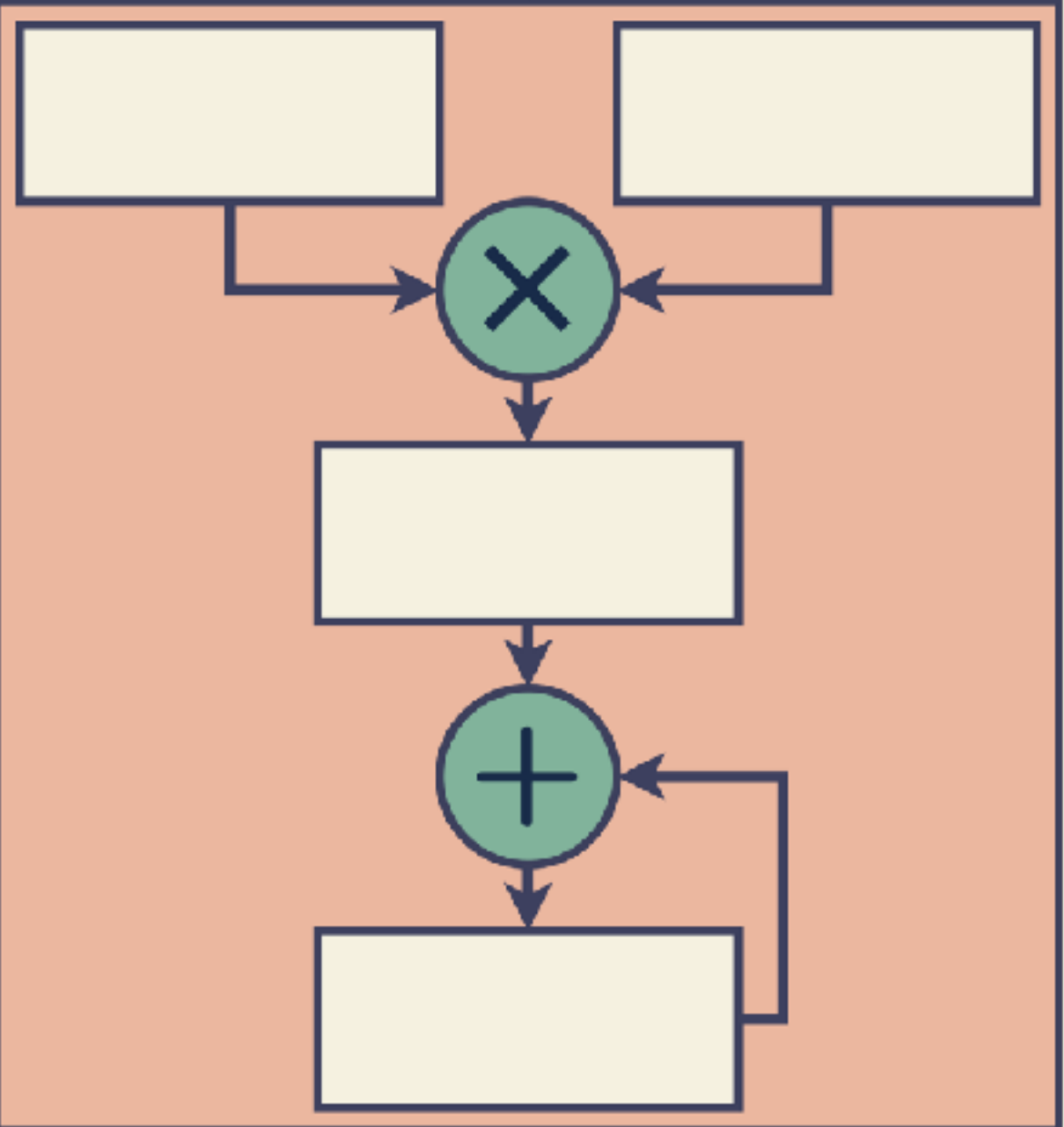
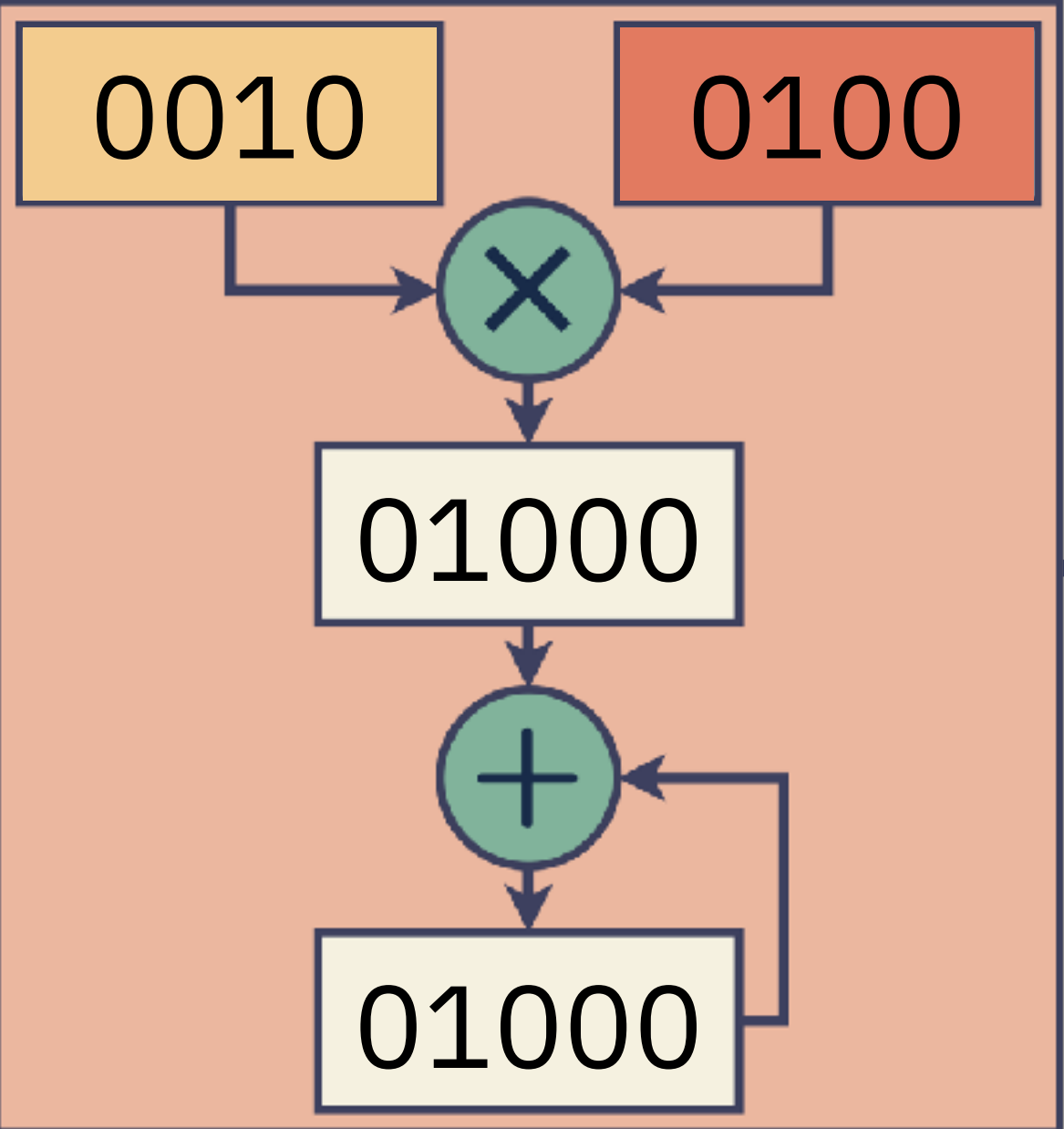
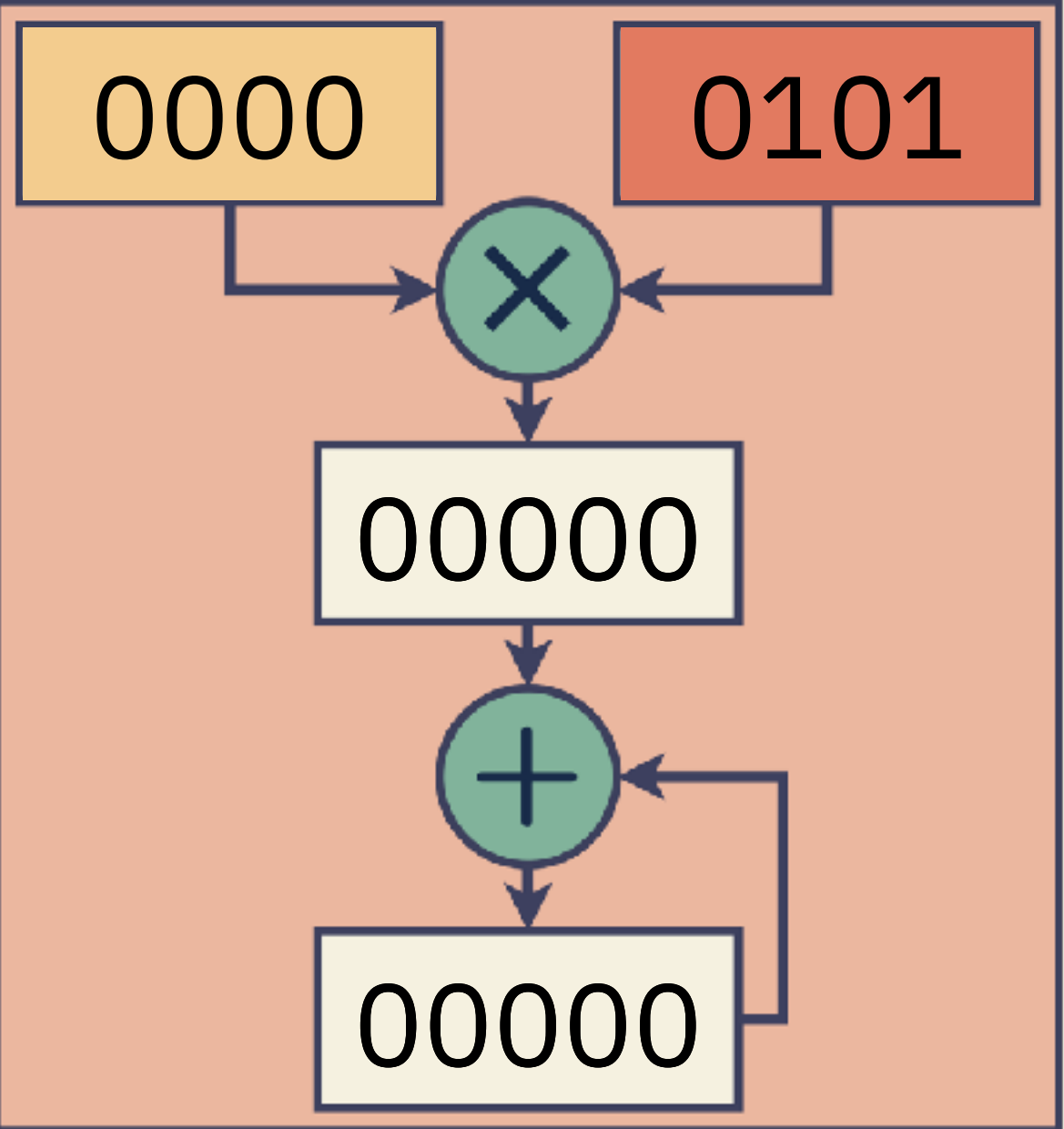
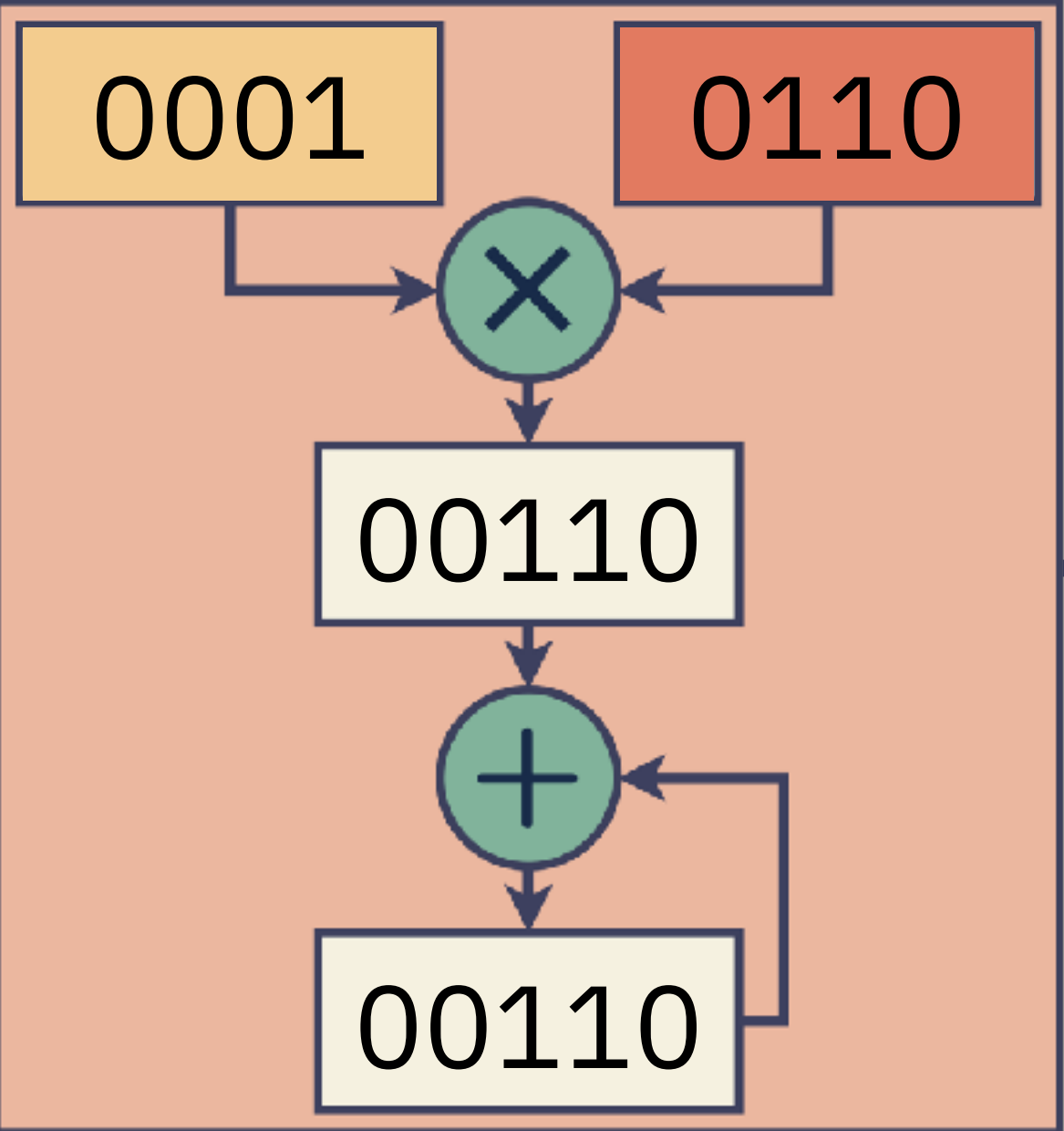
0011

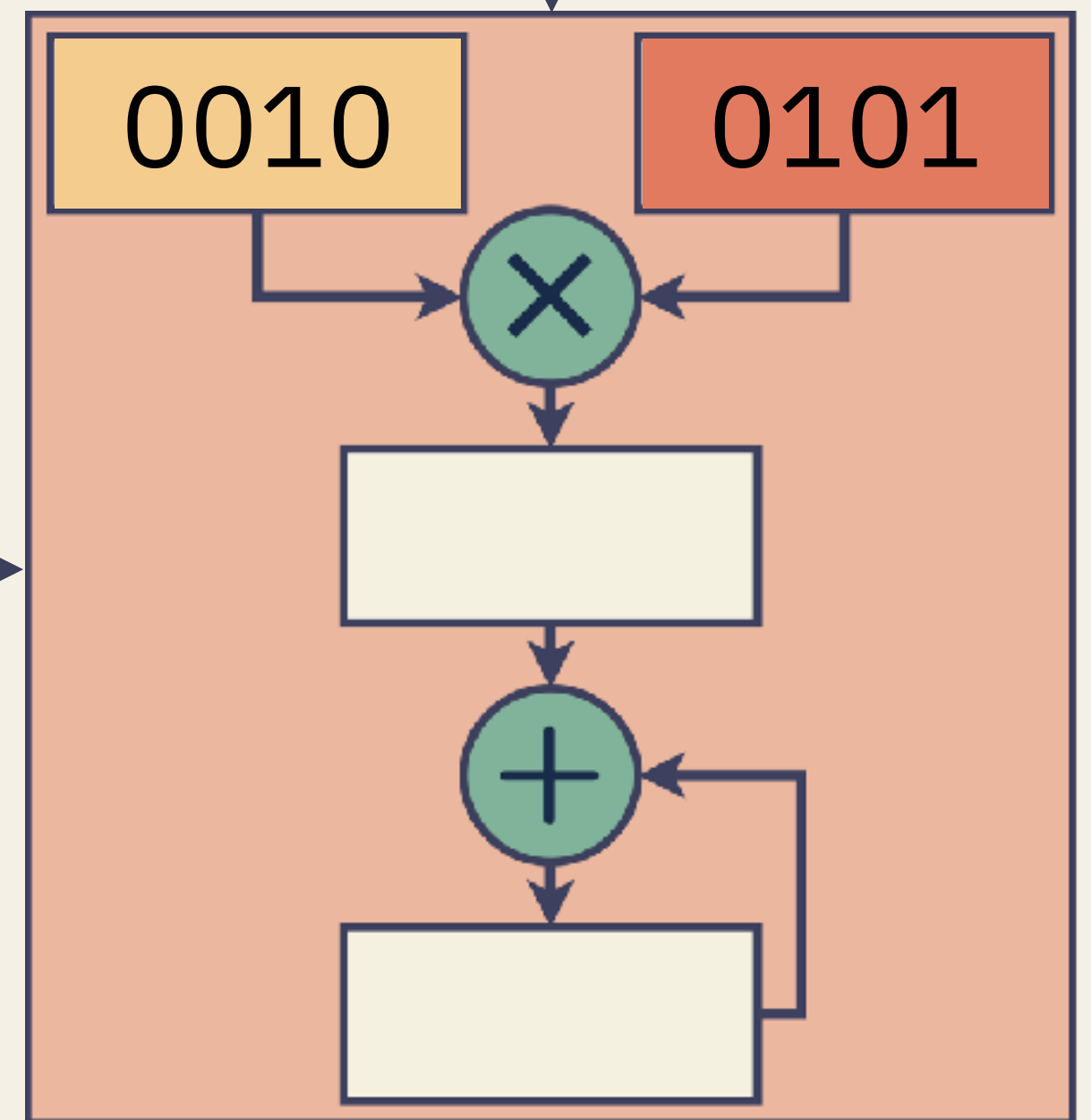
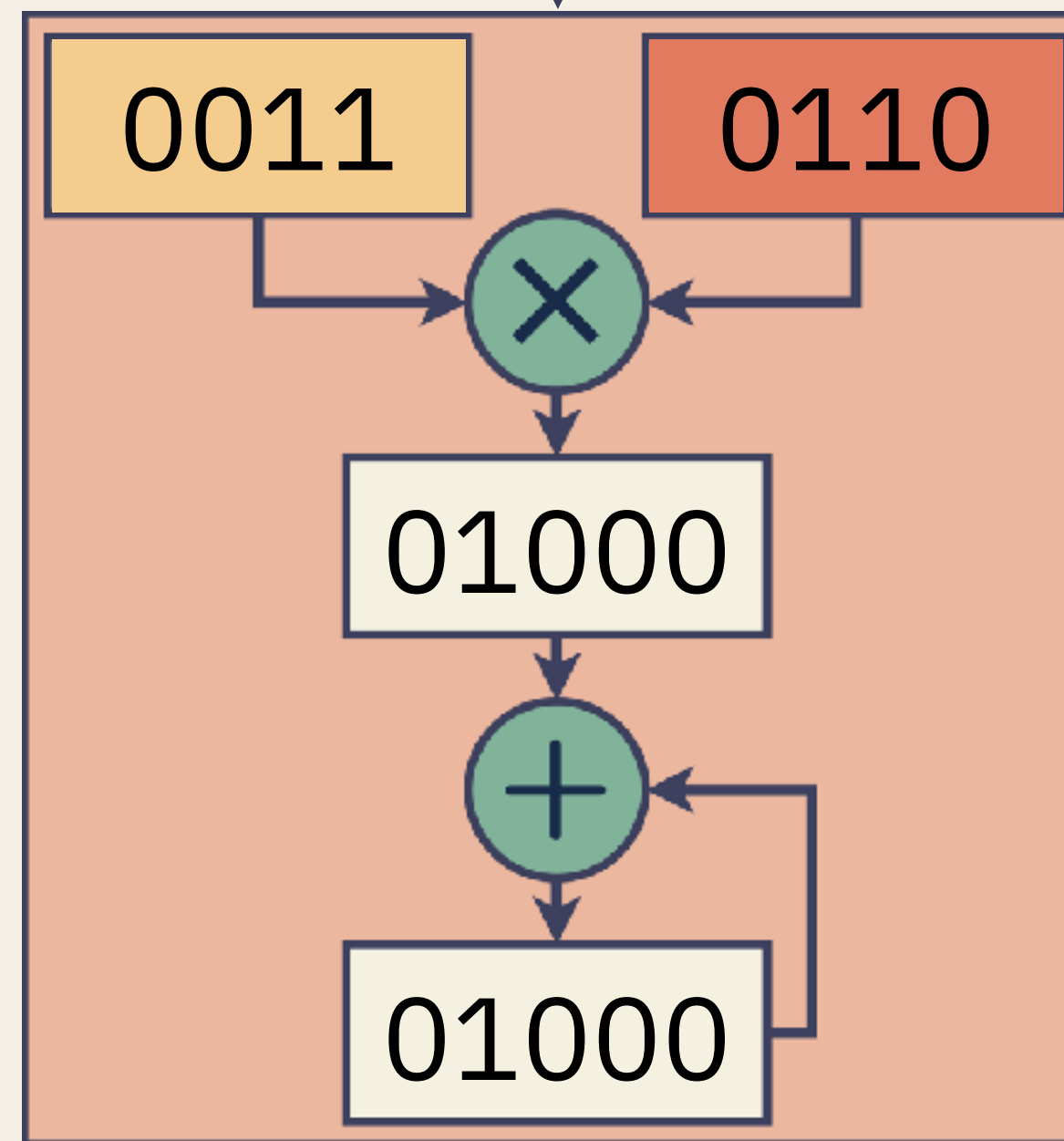
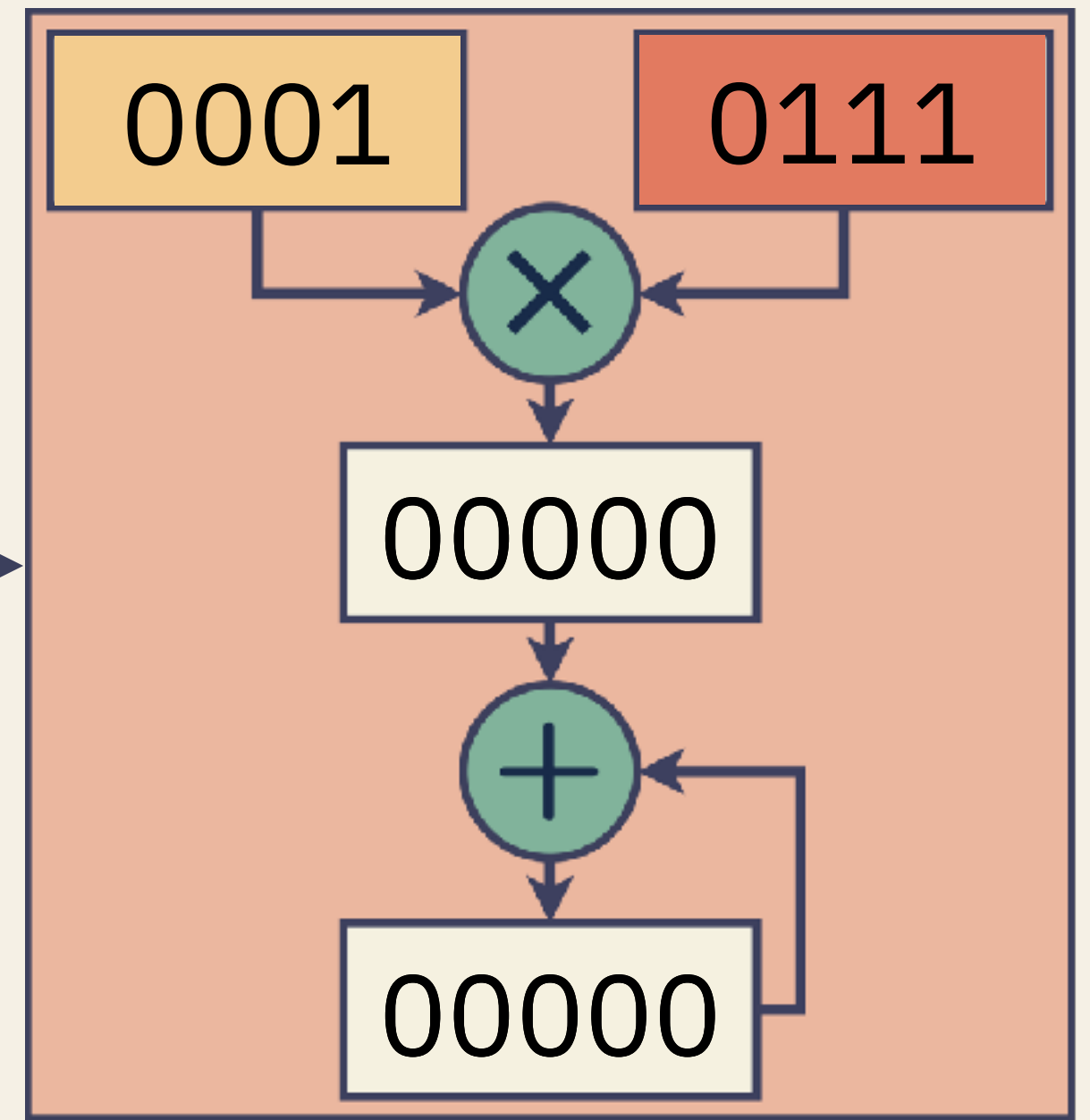
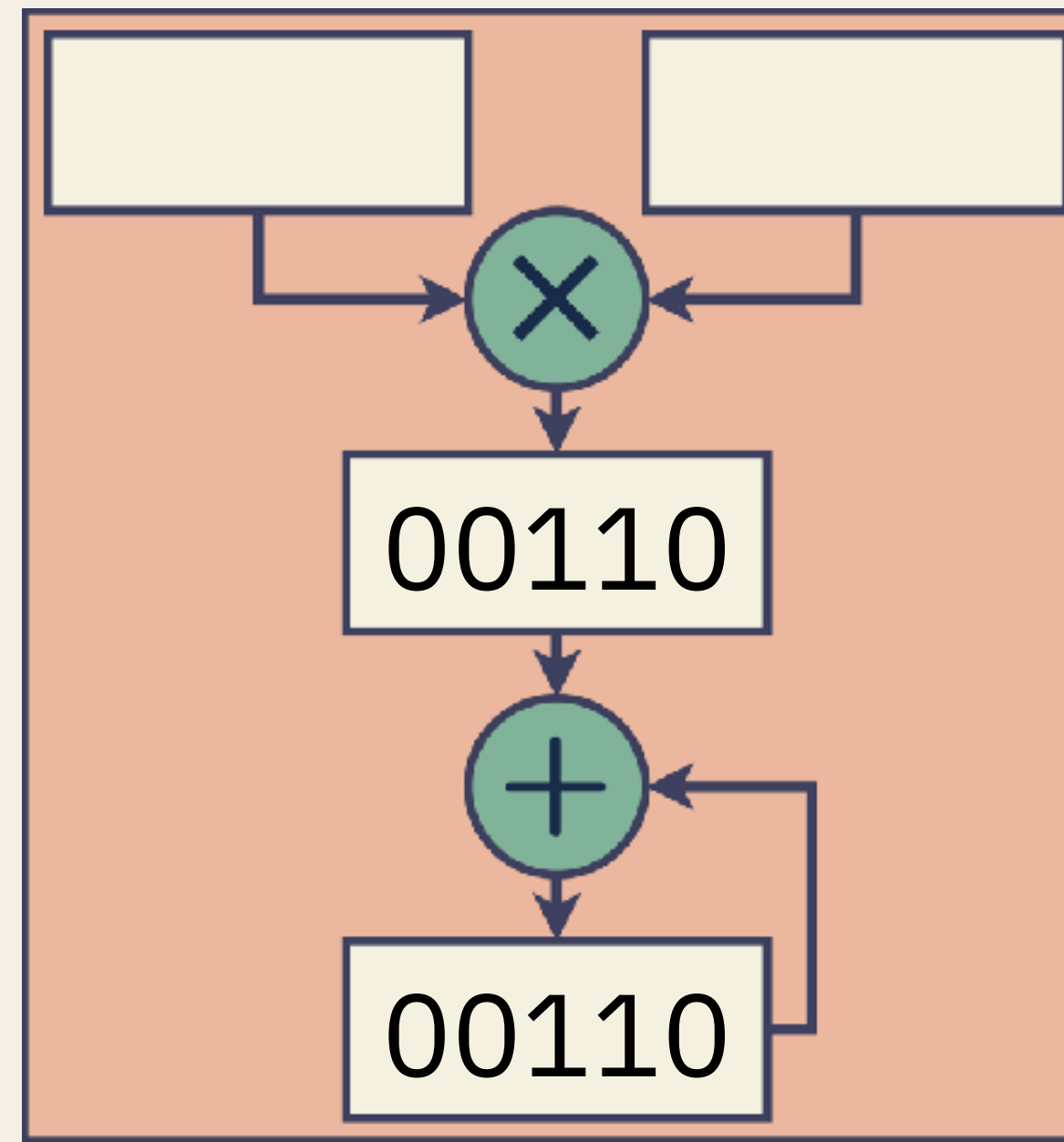
0111

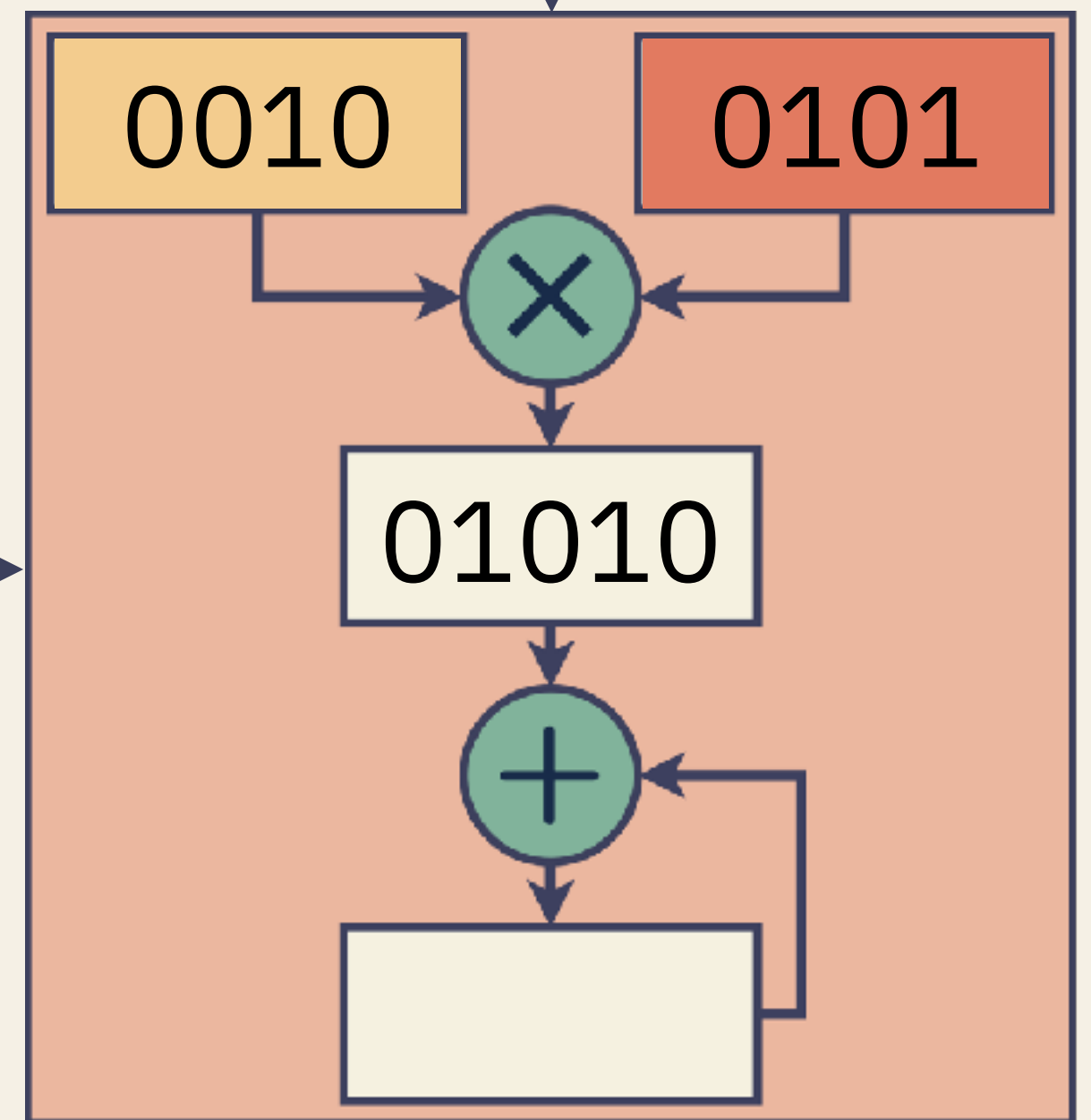
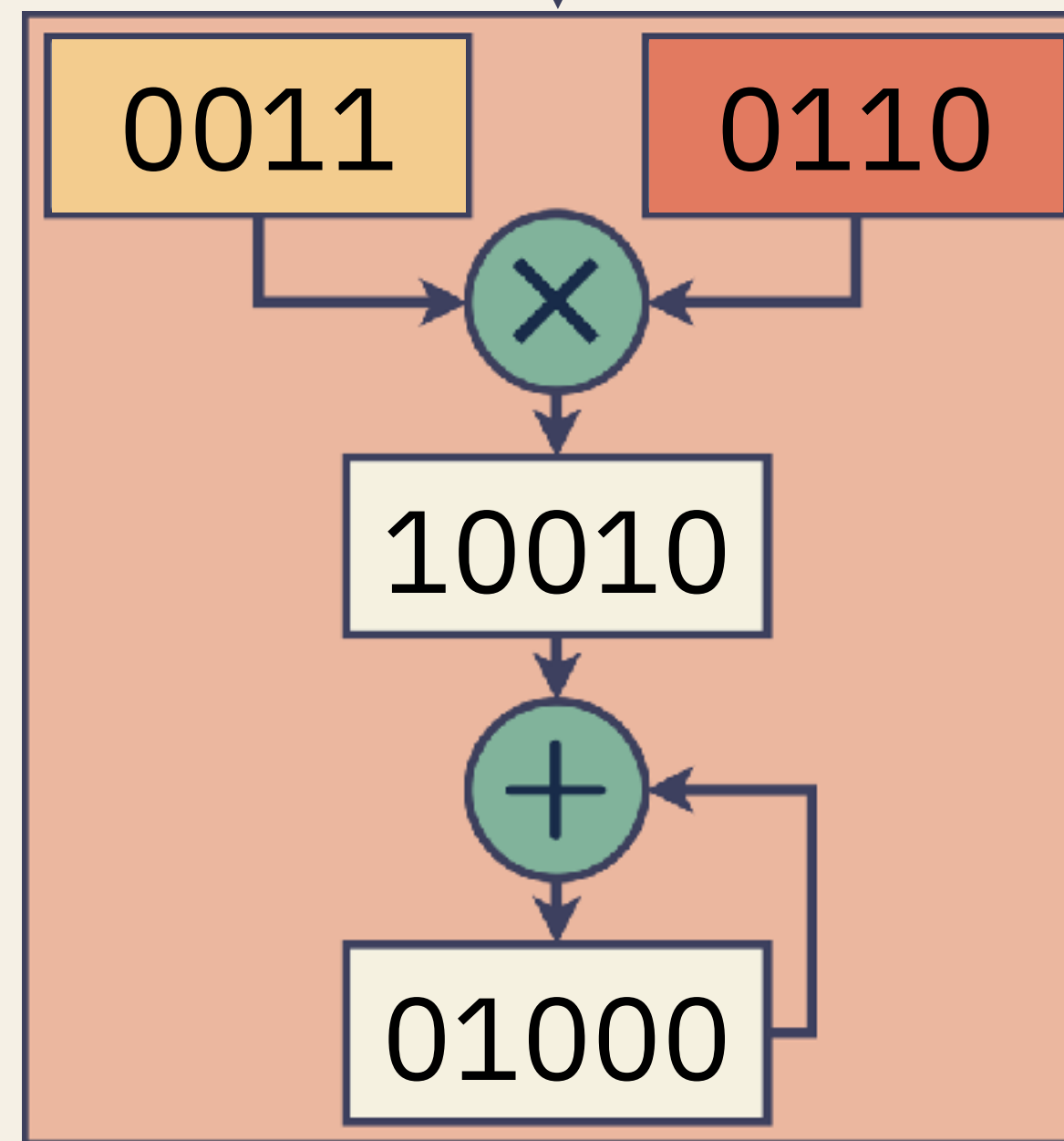
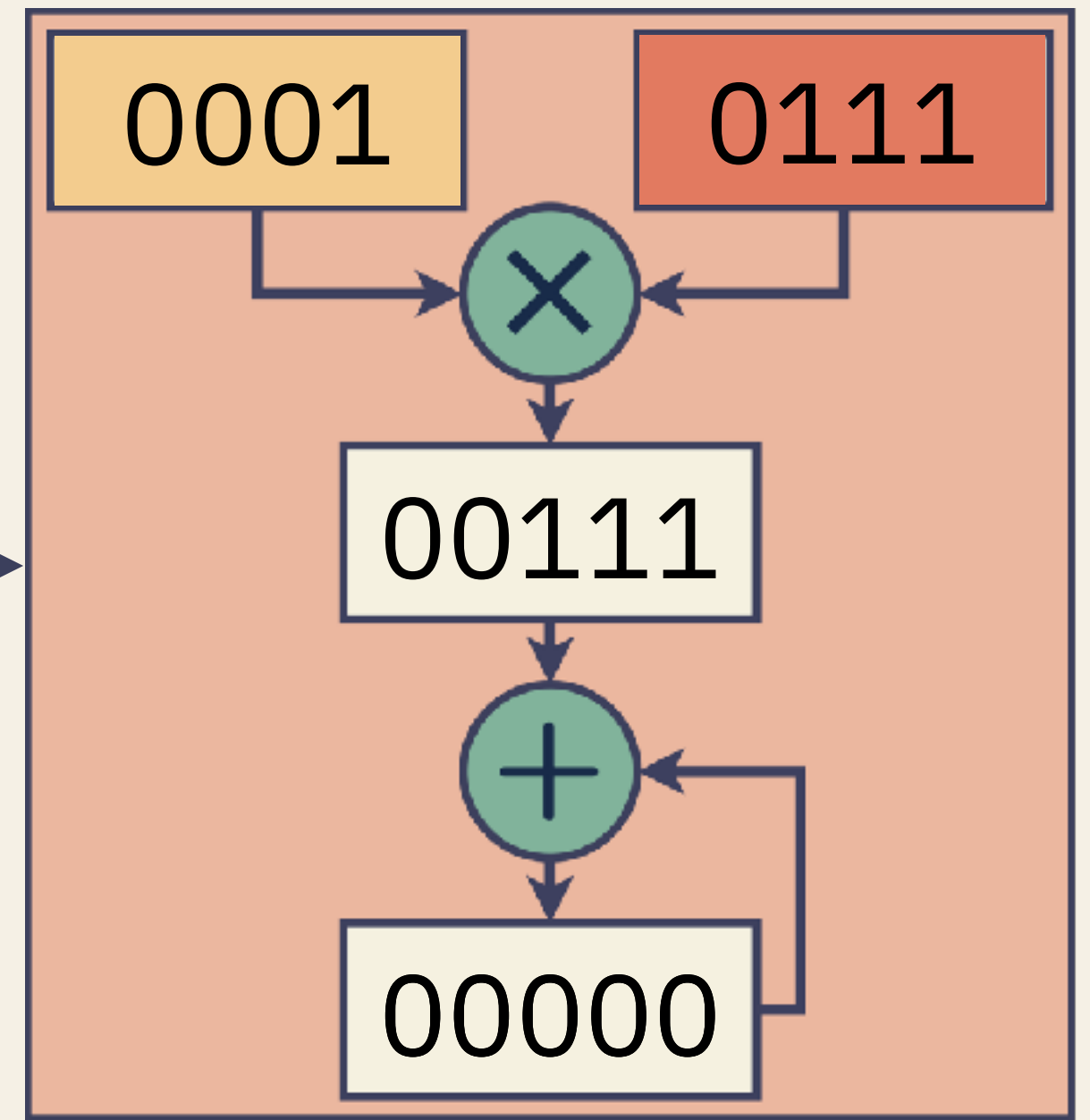
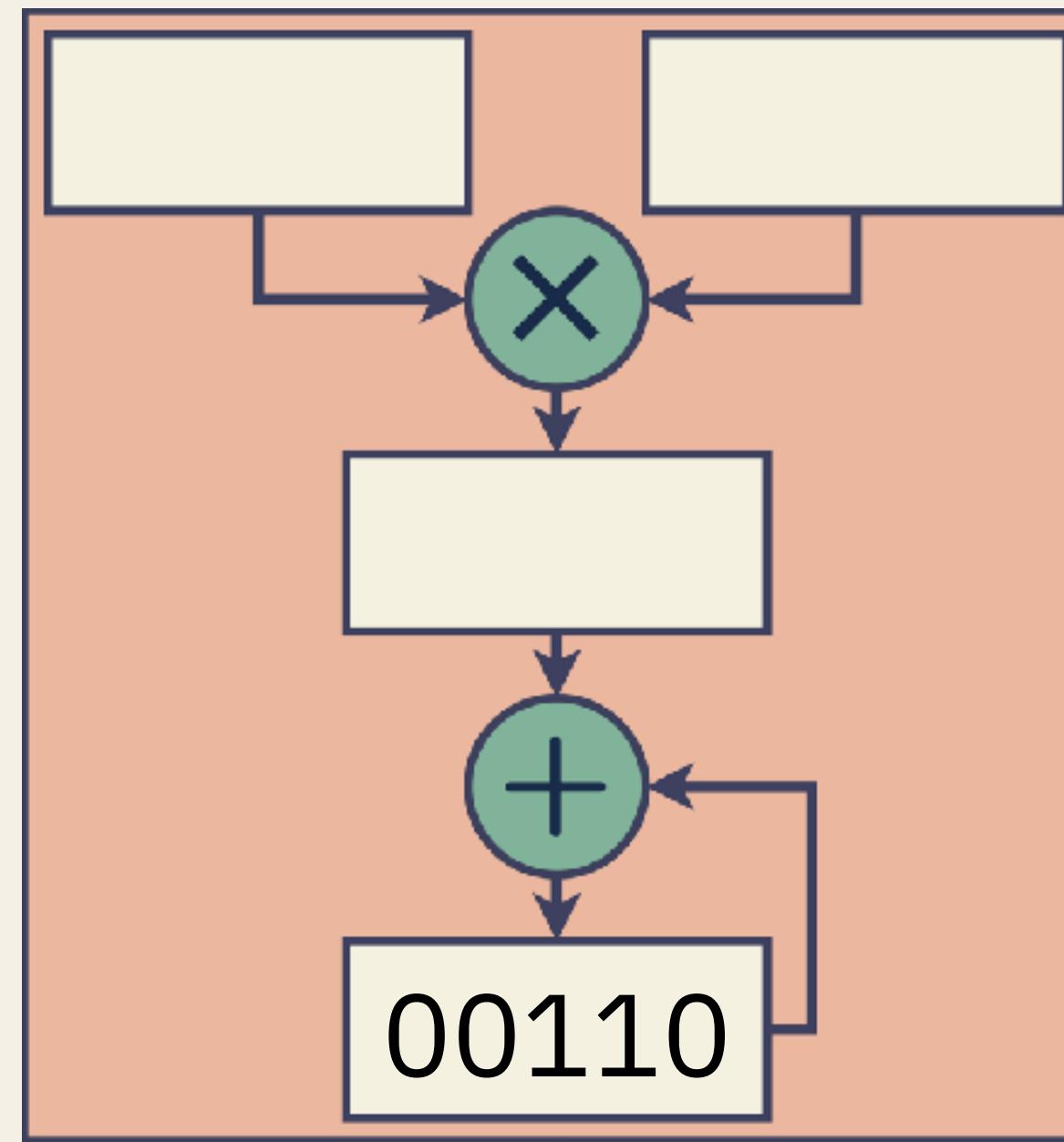


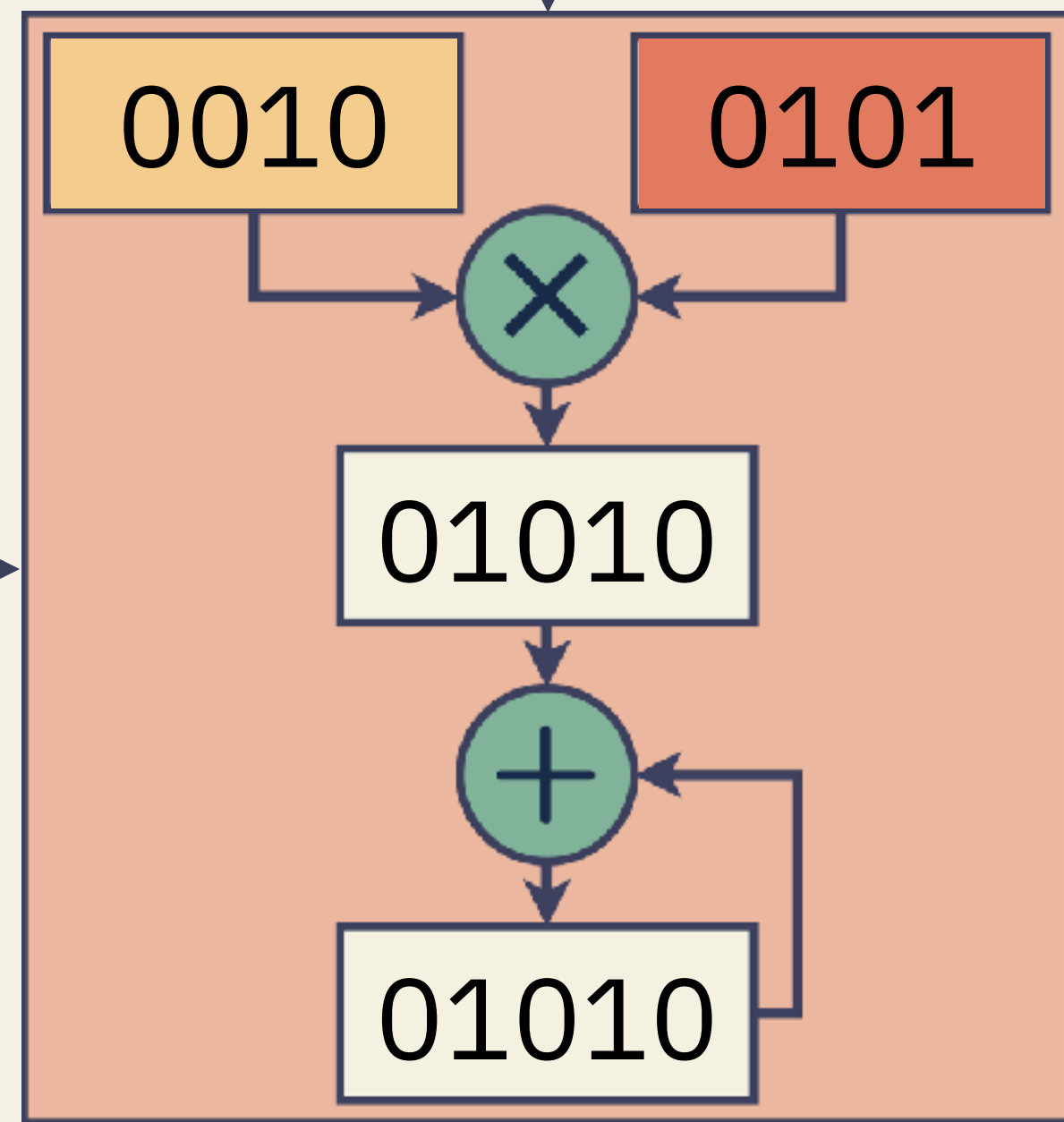
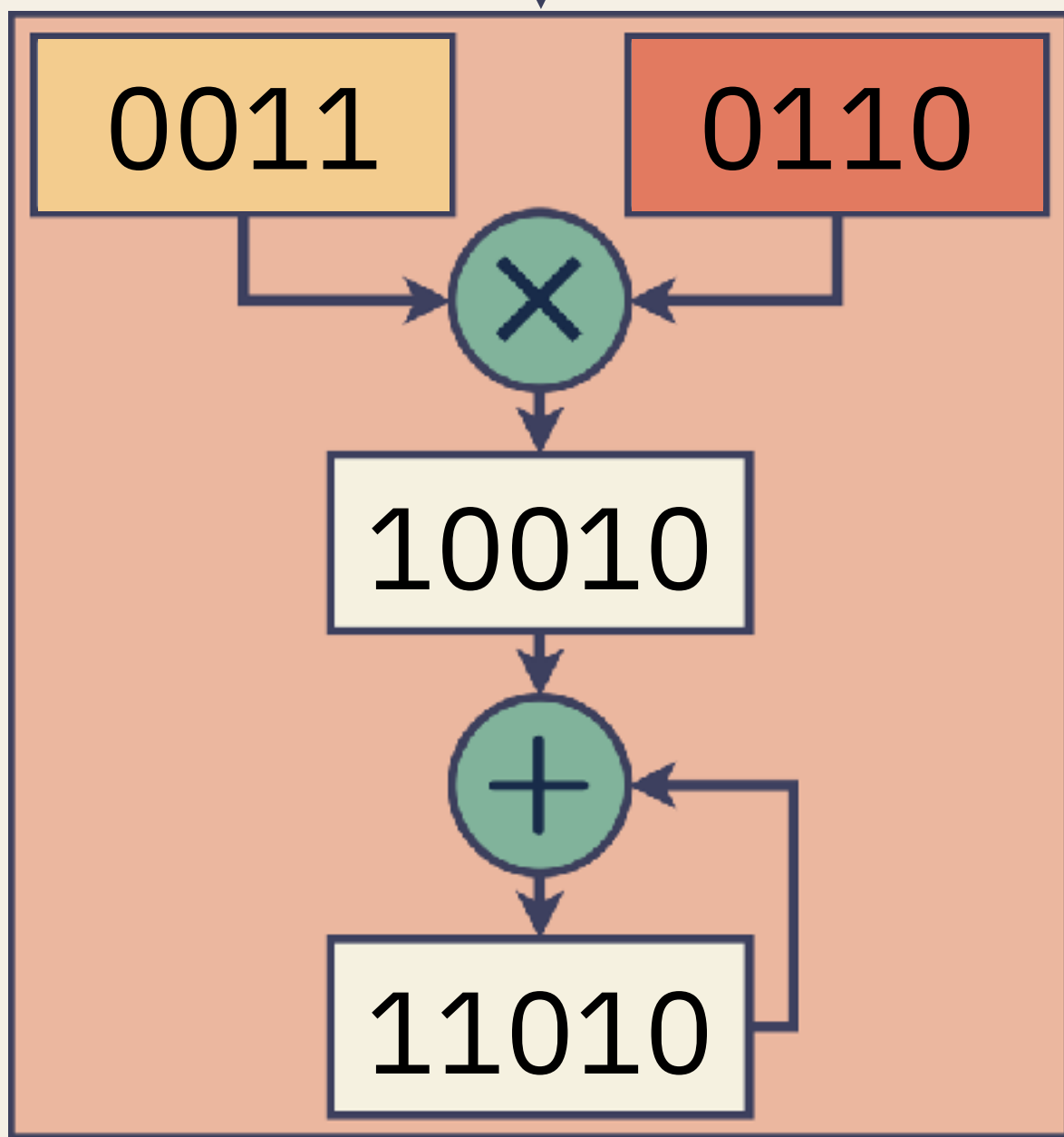
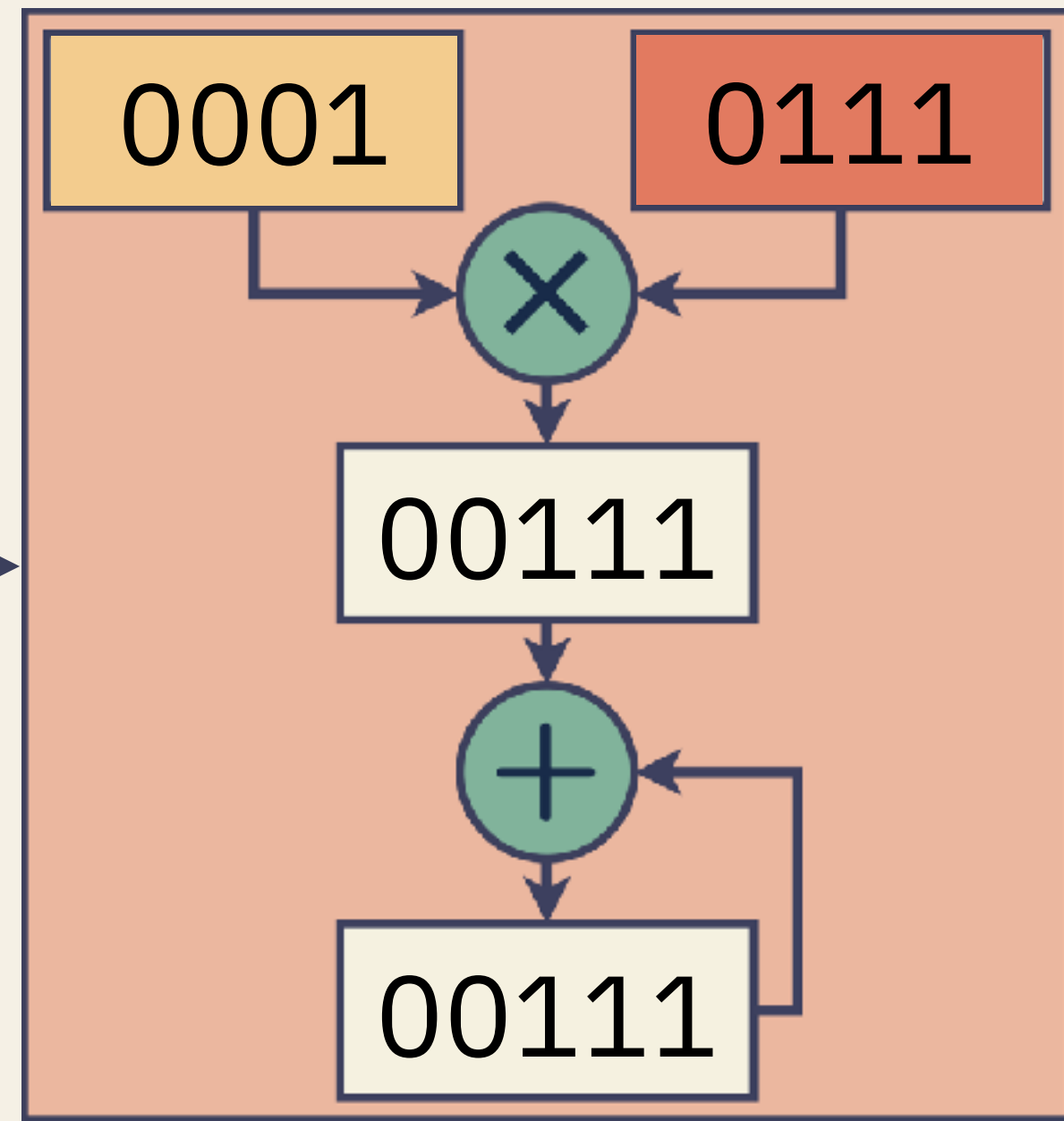
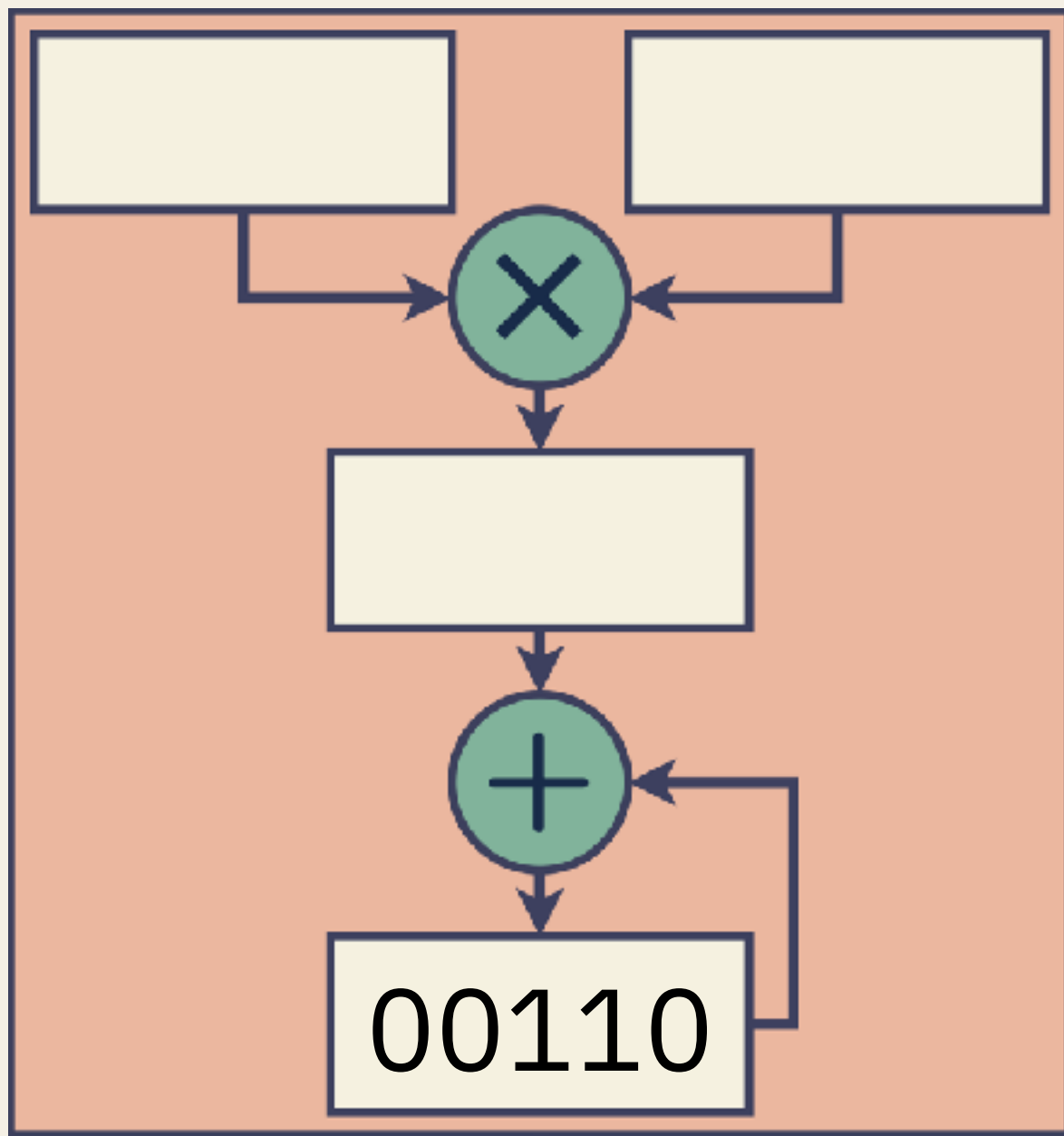
0011

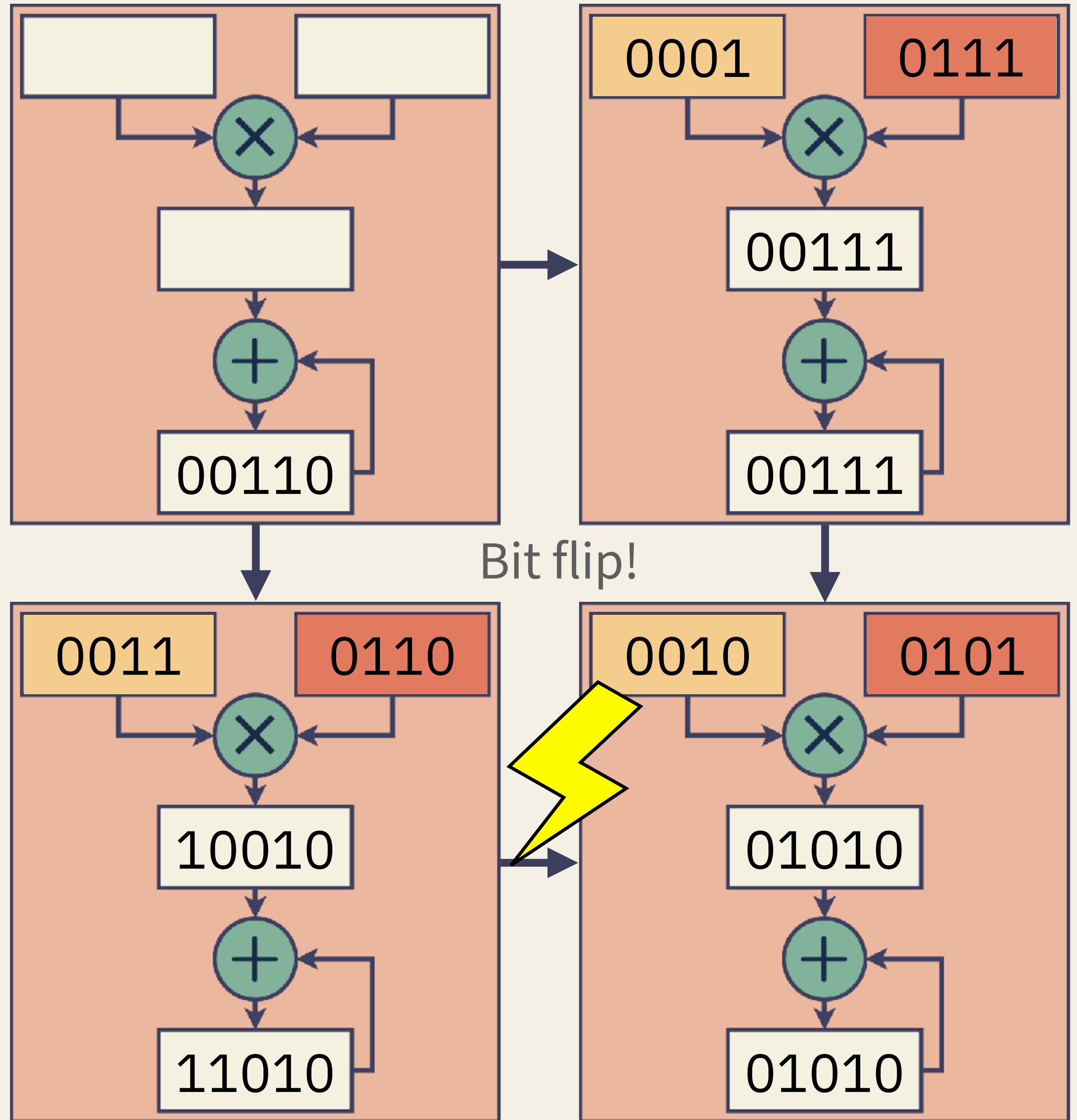
0111

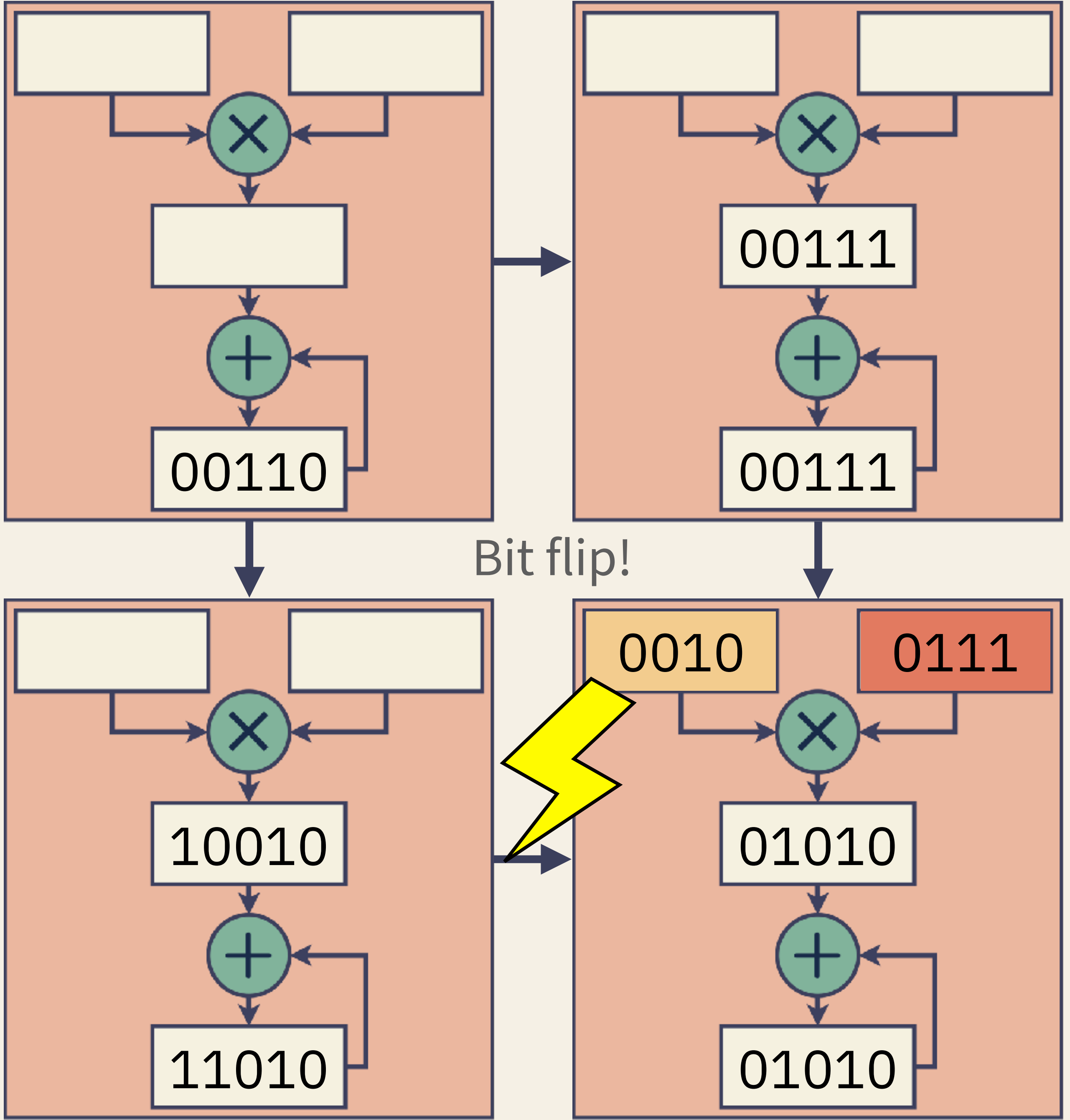




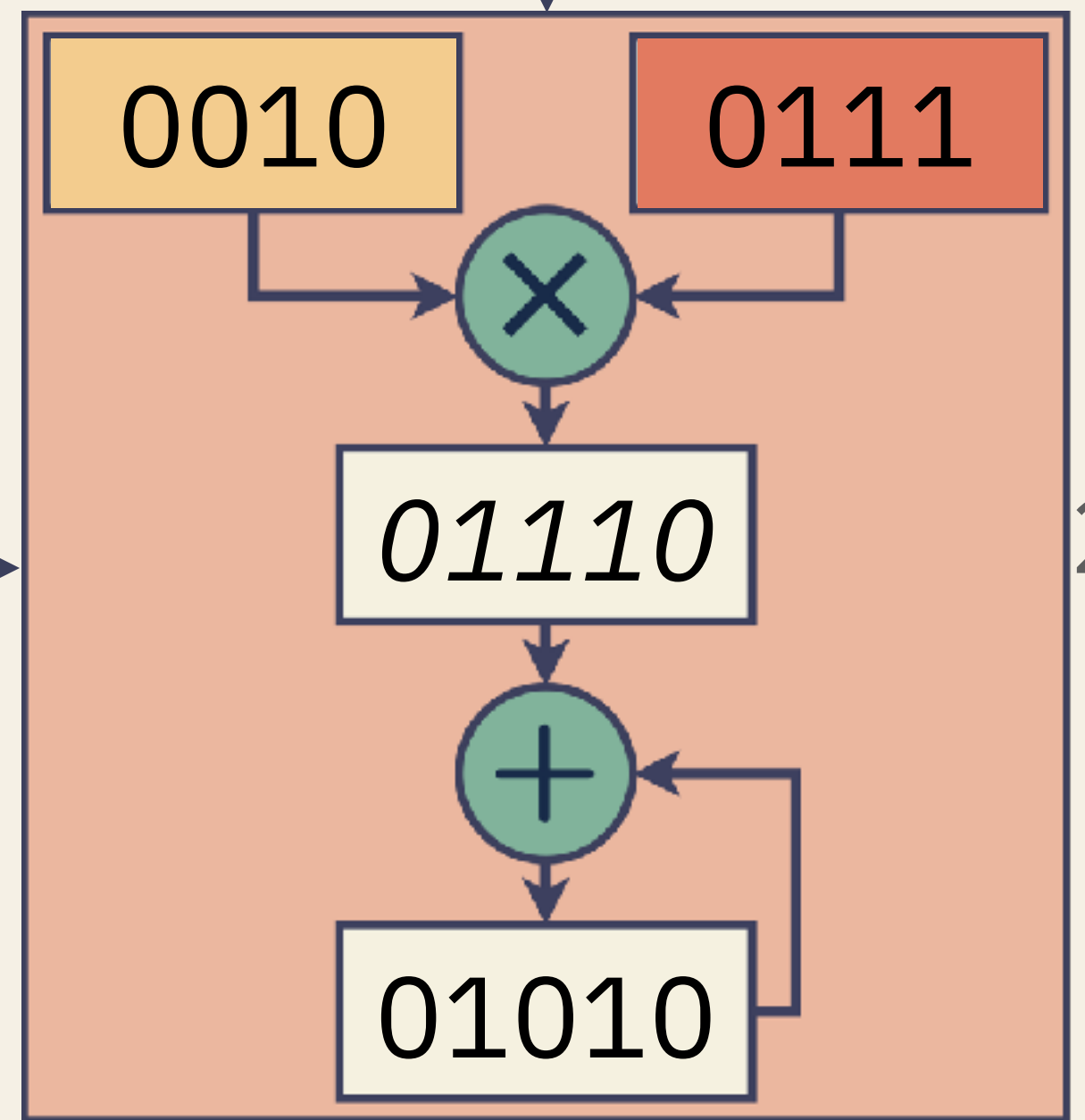
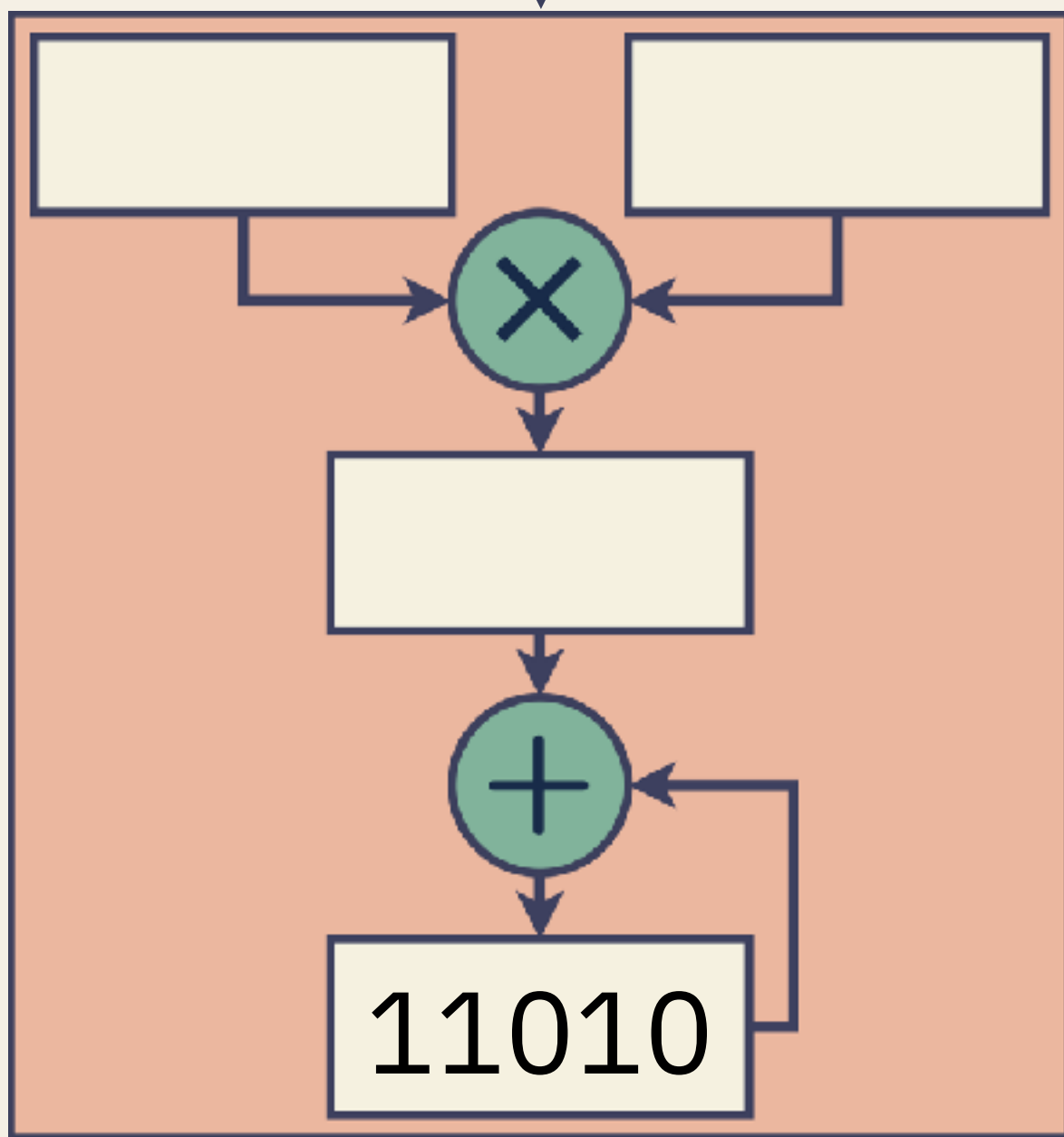
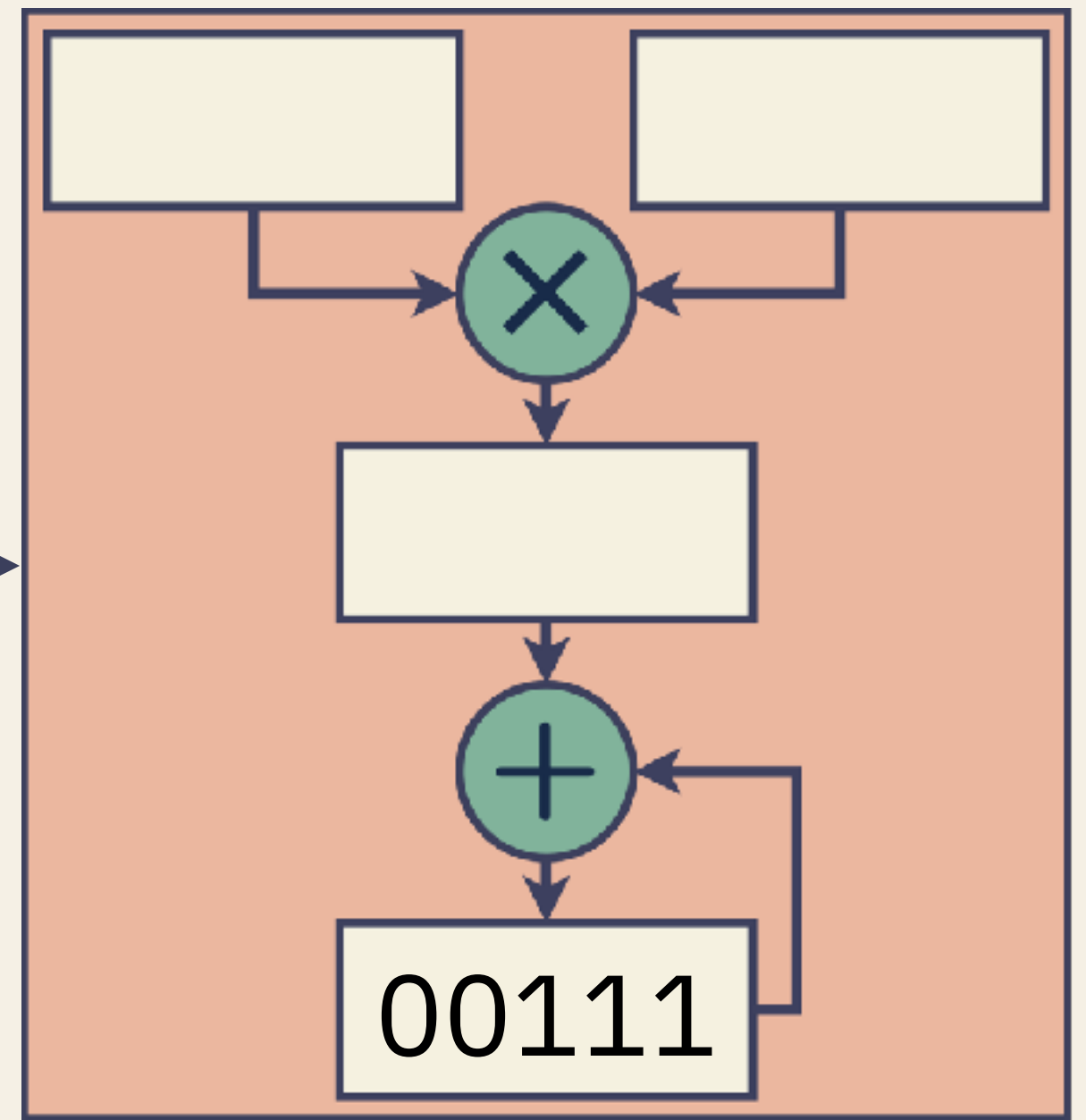
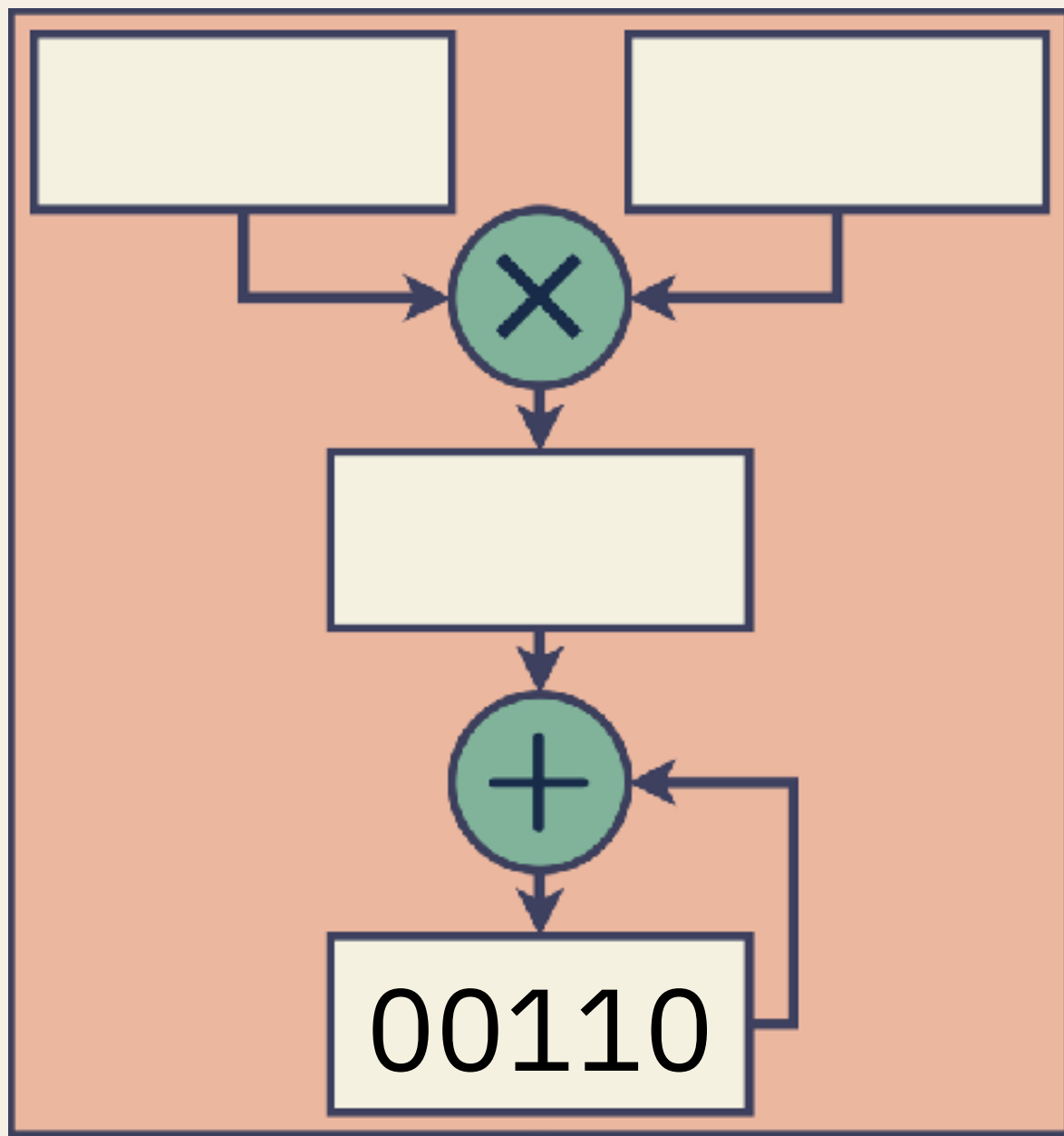




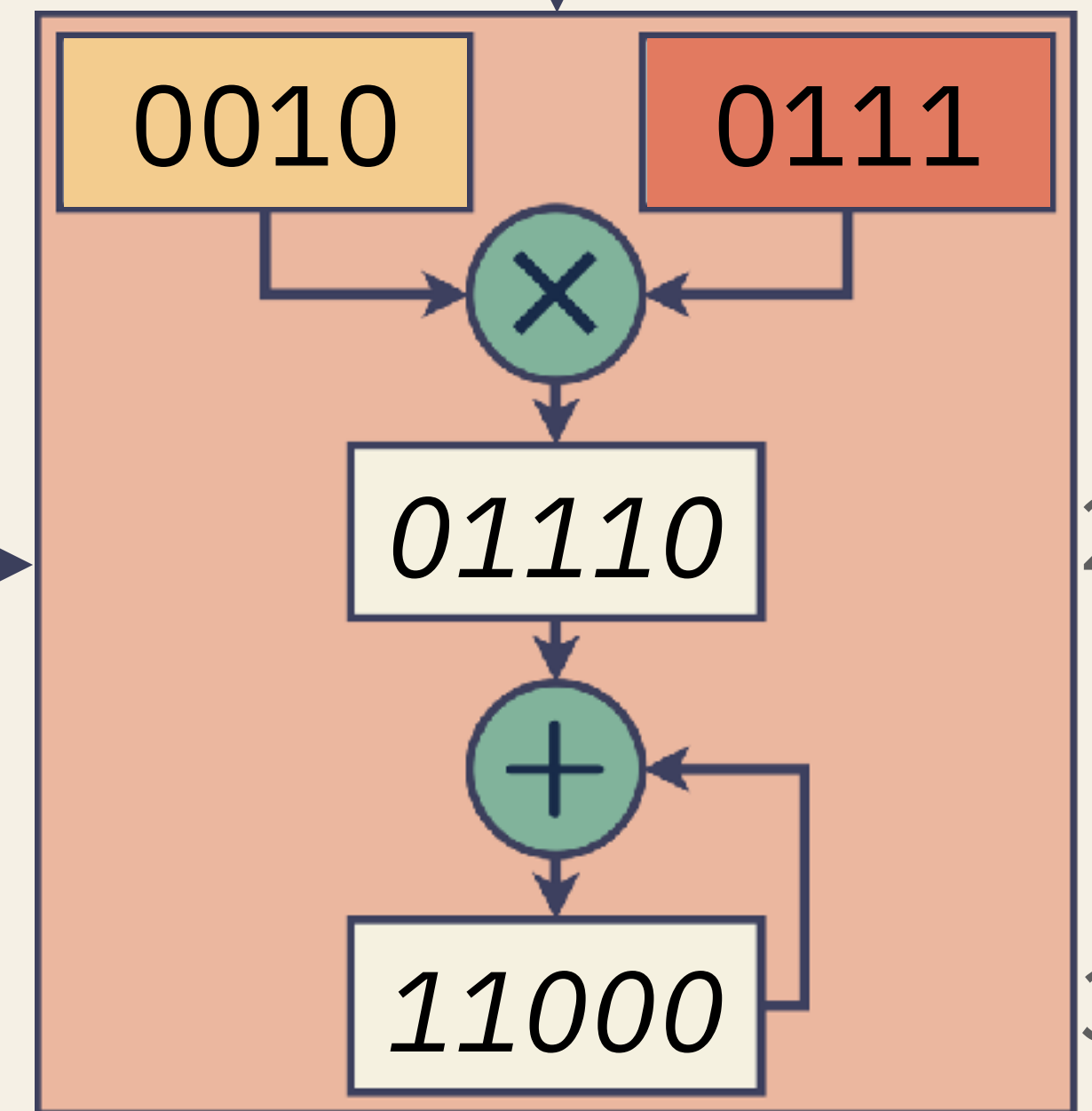
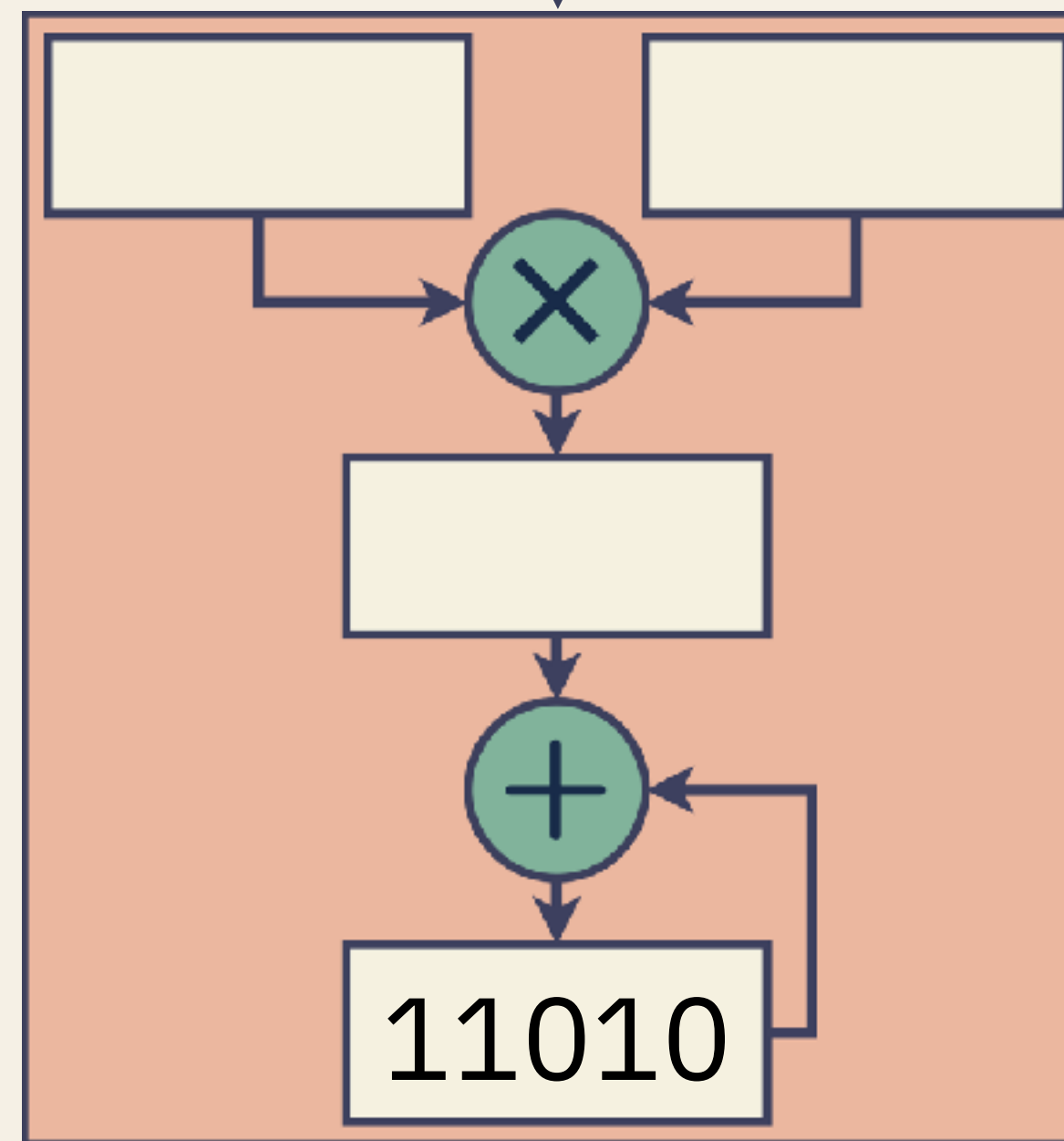
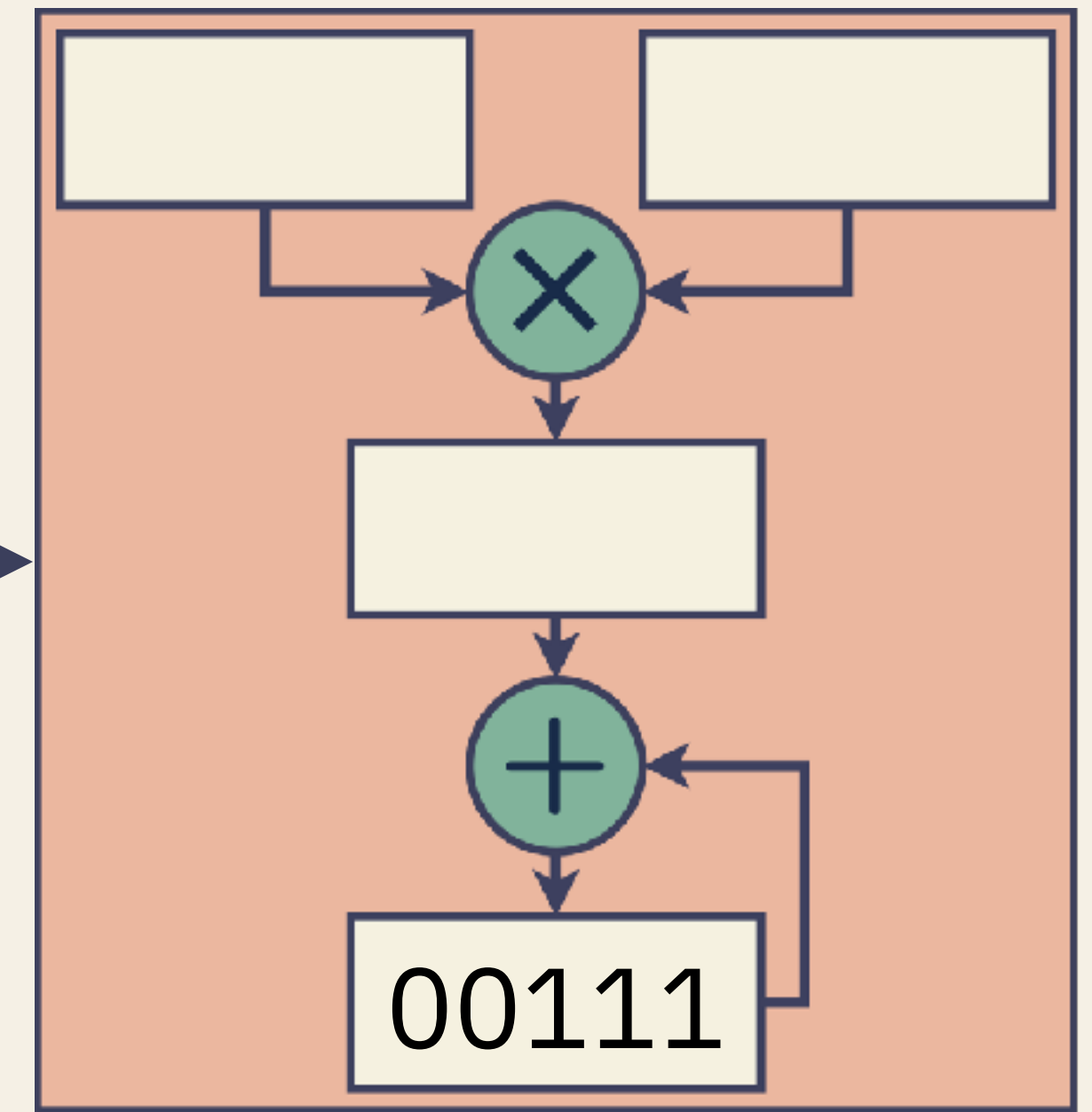
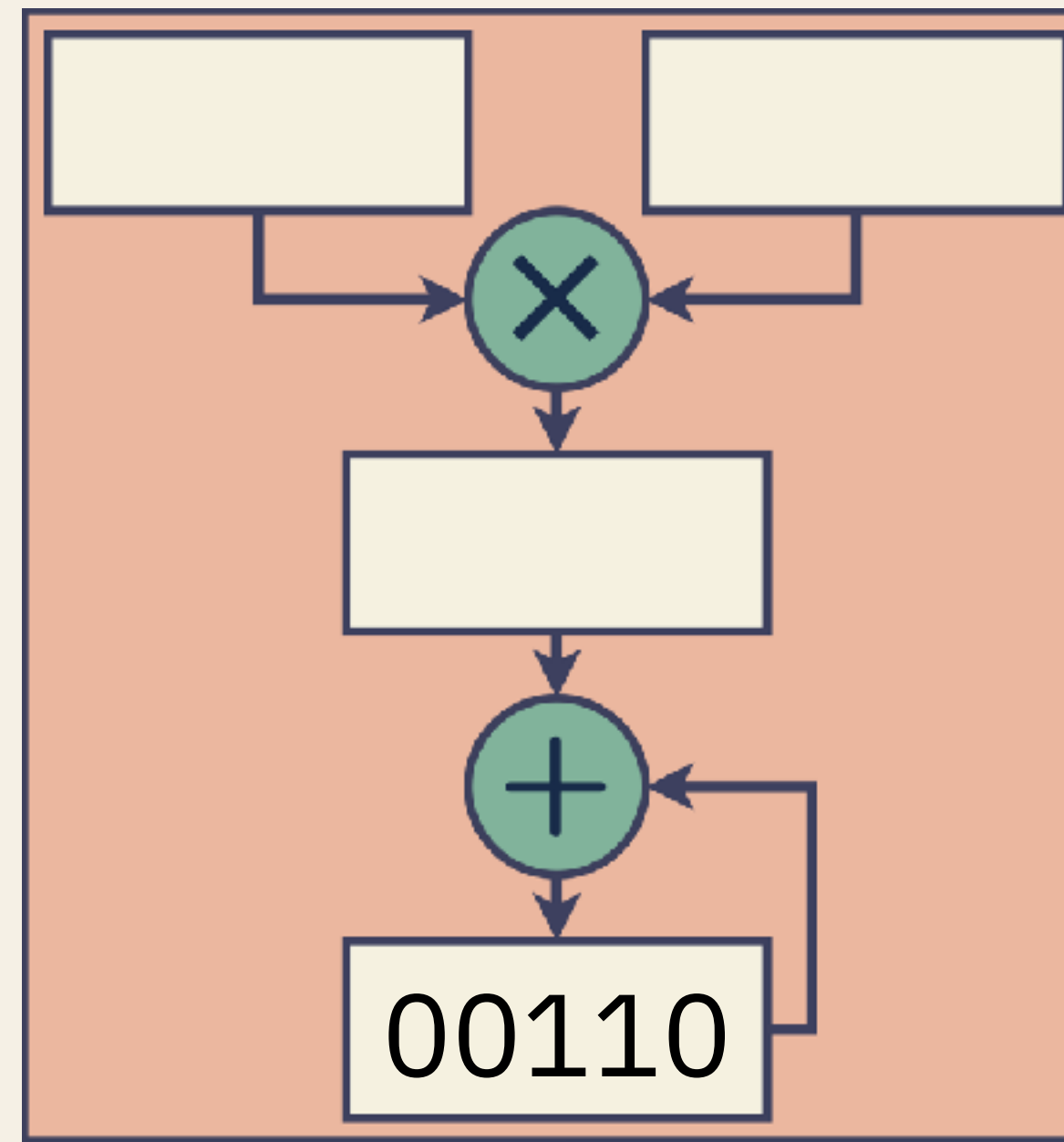




Bit flip!



Error!
21 → 14



Error!
21 → 14

31 → 24

Reasoning About Hardware Faults

Background

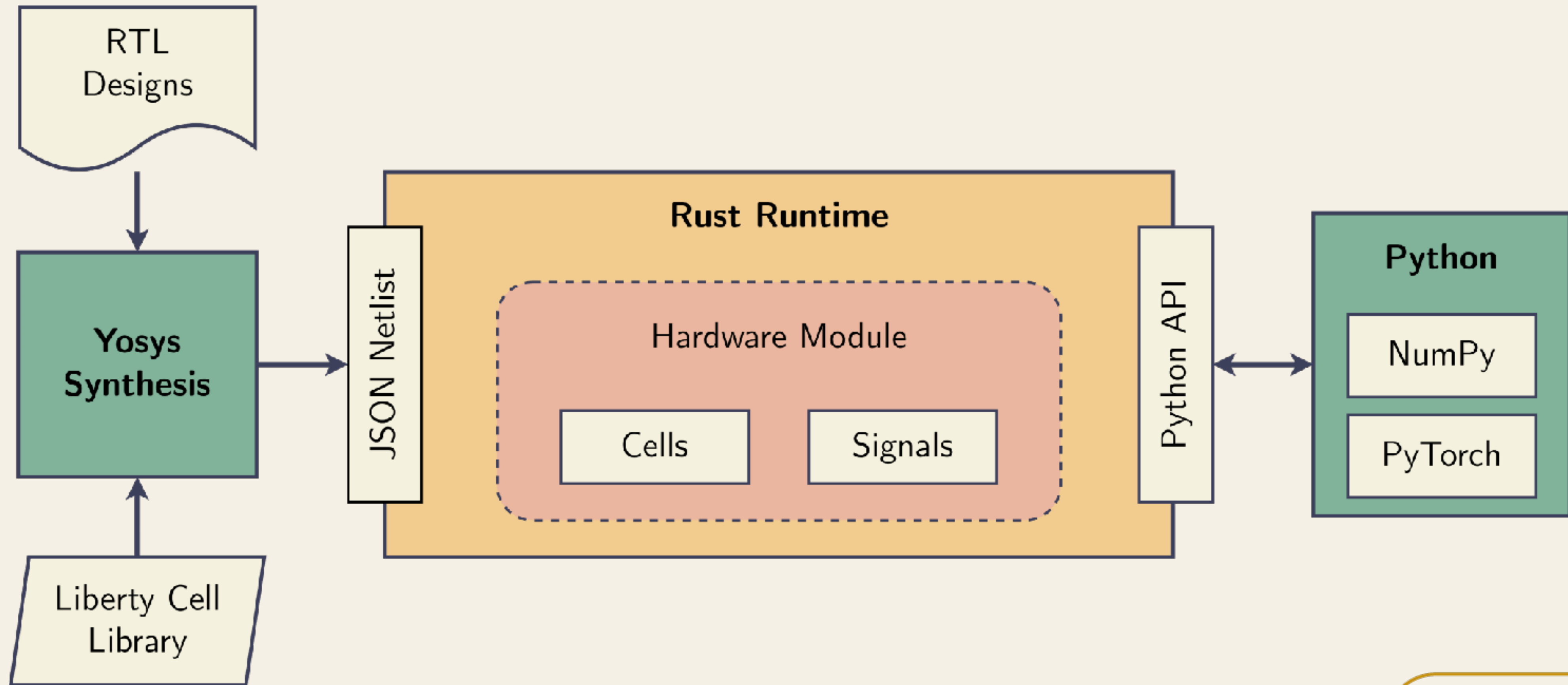
- Minimal gate-level simulator for HW/SW co-design
 - Higher-level process cells, down to liberty cells
 - Muxes, adders, shifters, ... → NOR, NAND, ...
- Originally for high-level power analysis of ML hardware
 - Interact with simulated HW from Python
 - Use real data from ML/DS frameworks (NumPy, PyTorch)
 - Design space-exploration for emerging low-precision datatypes
- Currently extending runtime for simulating fault-injection

Arbolta 

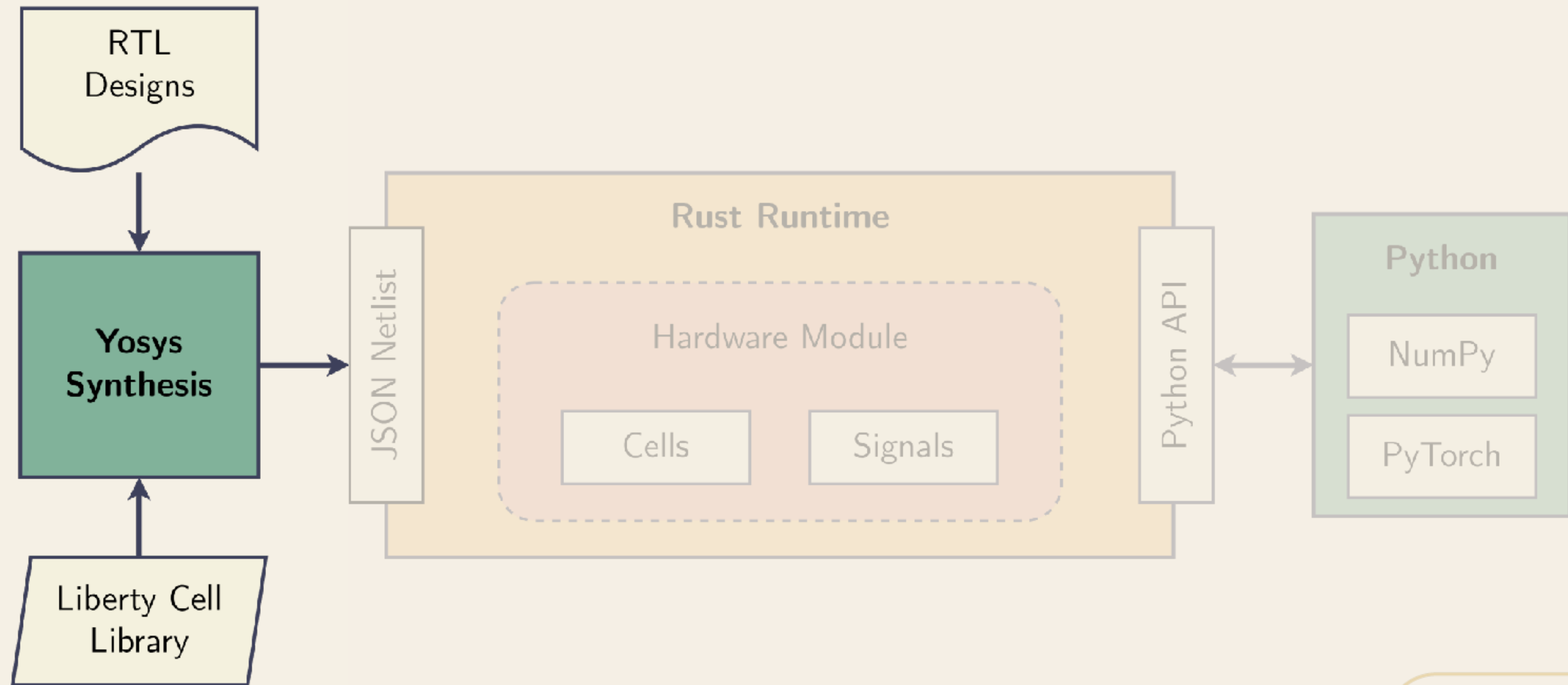
Main Idea

- Instead of dedicated framework, just do minimal simulation

Design

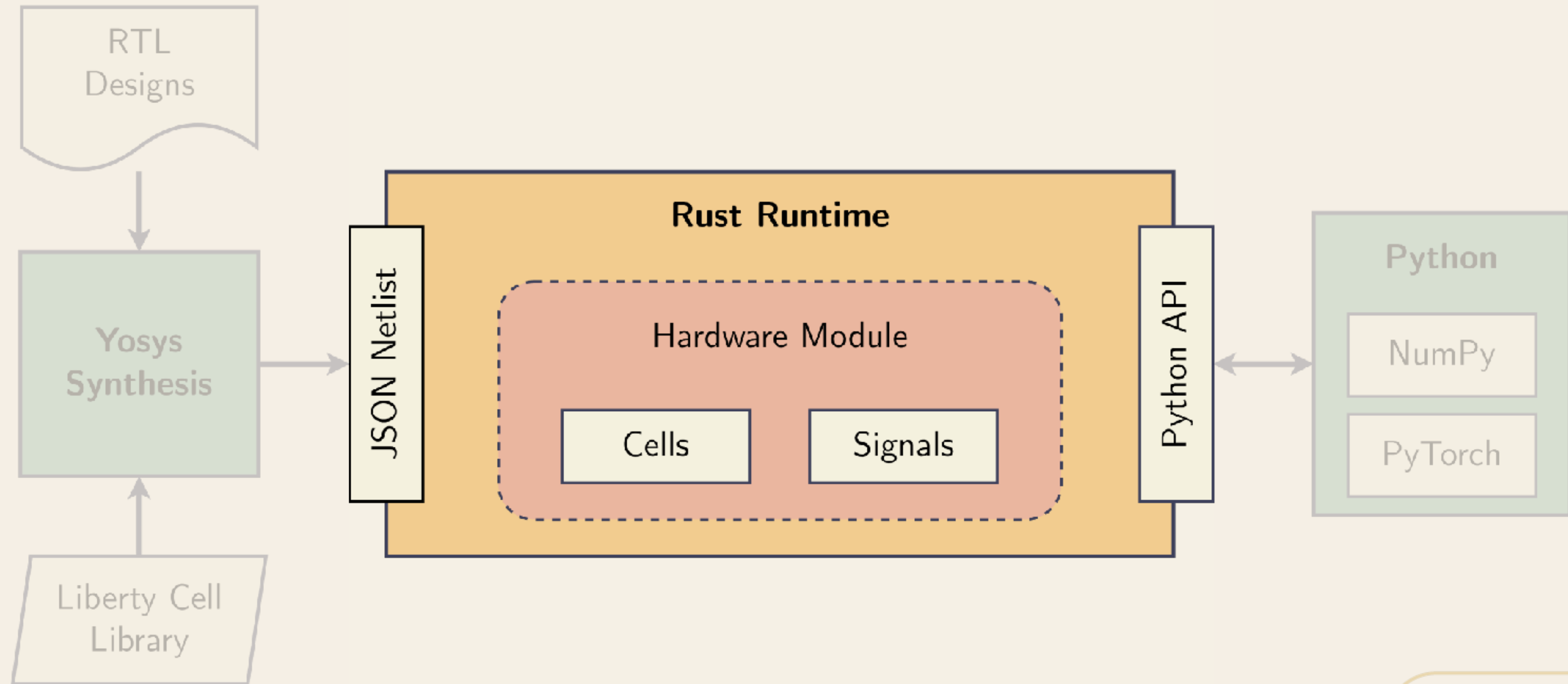


Design



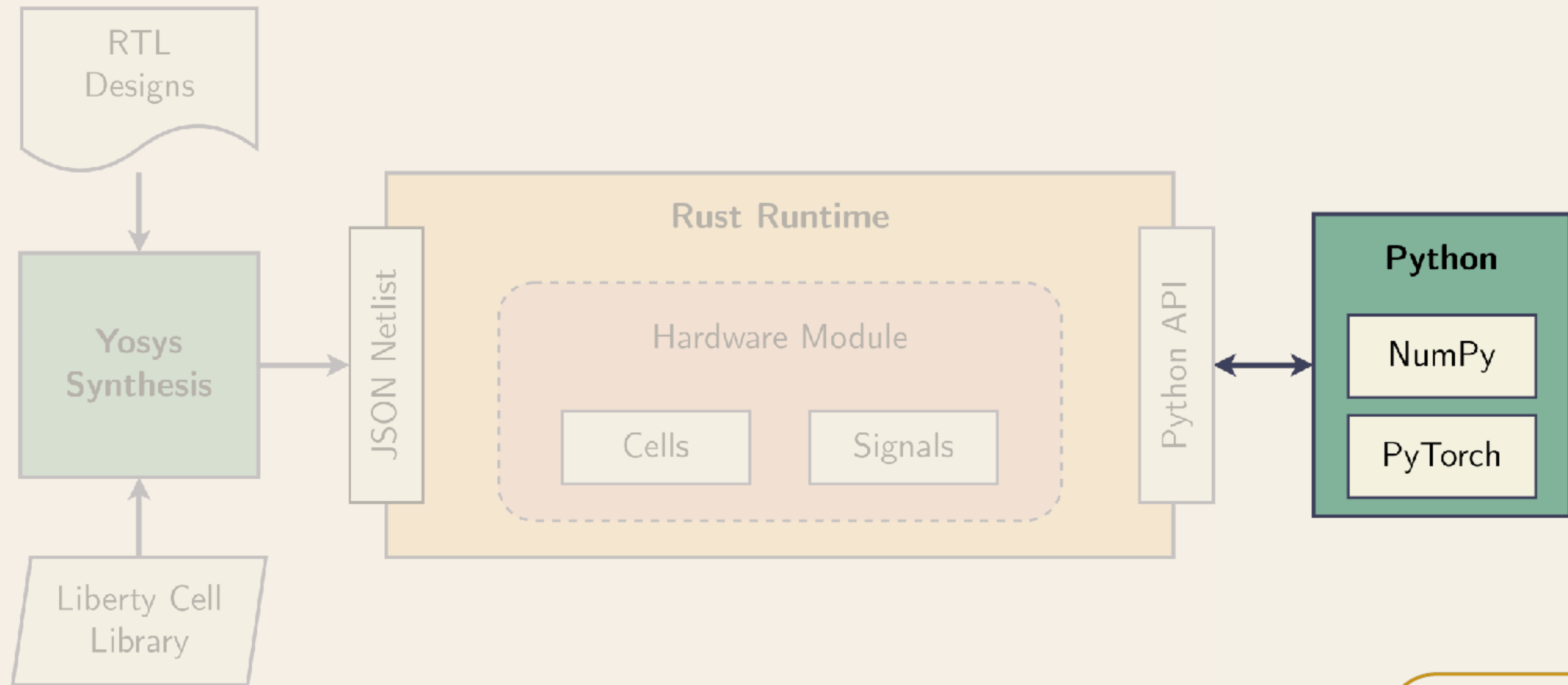
Arbolta

Design



Arbolta

Design



Arbolta 

Design

- PyTorch forward hooks for use with Brevitas
 - Pass quantized weights + inputs through simulated hardware
 - Arbitrary datatype formats (e.g. minifloats)
 - Currently using systolic array for linear and convolution layers

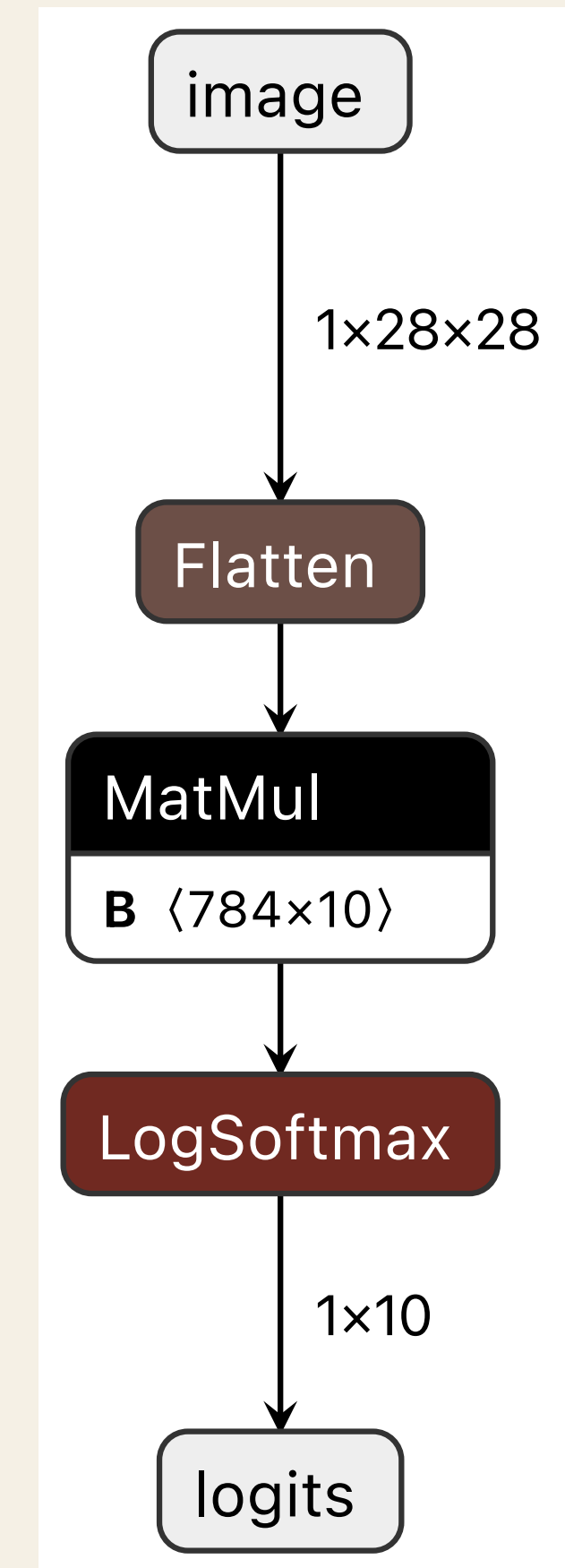
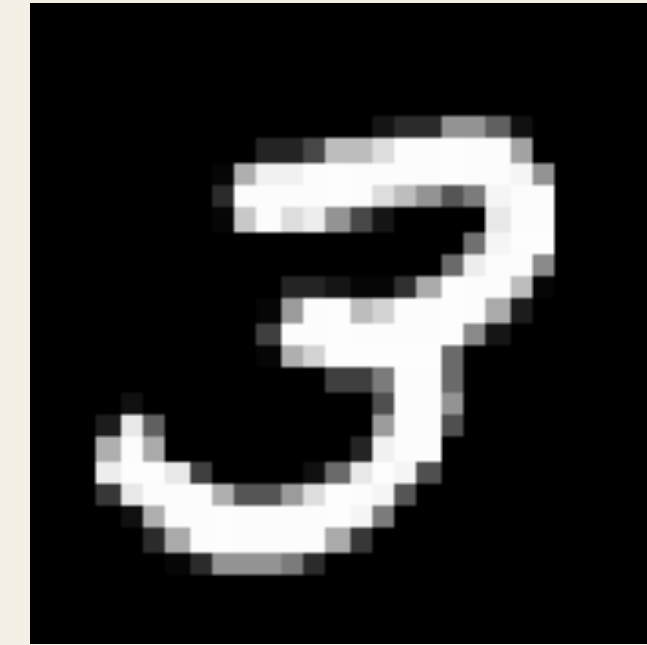


Arbolta Fault Injection Example

Example

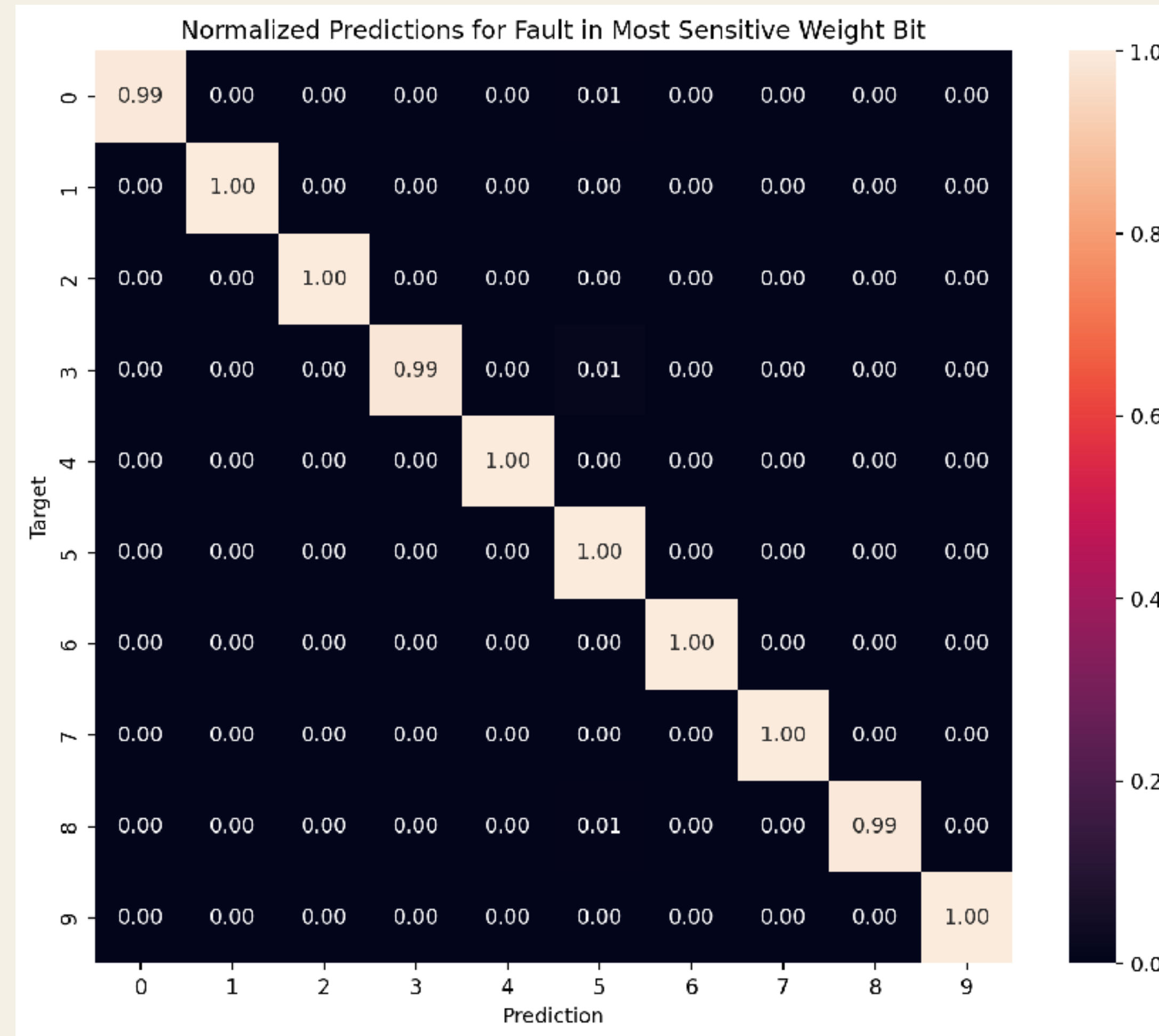
Int4 MNIST

- Experimental setup
 - Single linear layer, quantized
 - 8x8 systolic array for batched inference
 - Can run 8 images at a time
 - Known correctly predicted inputs
 - Two's complement int4 quantized weights + inputs
 - Fault injection on control signals (e.g., valid, ready)
 - Permanent faults (worst case)
 - Infinite loops = failure



Int4 MNIST Batch Example

Worst Case from Weight Faults (Baseline)



Int4 MNIST Batch Example

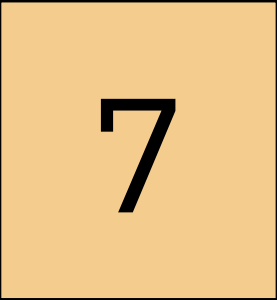
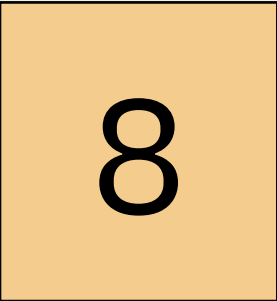
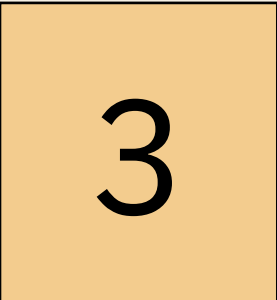
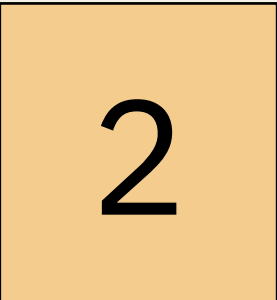
1D Confusion Matrix

Images	Possible Predictions										
	0	1	2	3	4	5	6	7	8	9	Fail
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	
8	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.99	0.00	
3	0.00	0.00	0.00	0.99	0.00	0.01	0.00	0.00	0.00	0.00	
2	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
...											

Weight faults

Int4 MNIST Batch Example

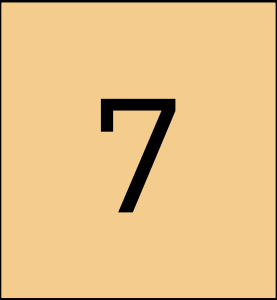
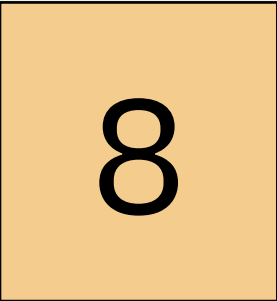
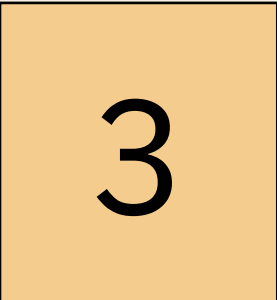
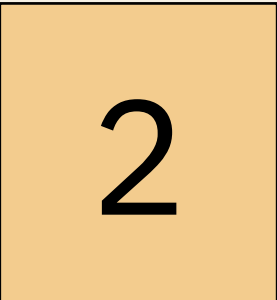
1D Confusion Matrix

Images	Possible Predictions										
	0	1	2	3	4	5	6	7	8	9	Fail
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.43	0.00	0.01	0.55
	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.99	0.00	
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.43	0.01	0.55
	0.00	0.00	0.00	0.99	0.00	0.01	0.00	0.00	0.00	0.00	
	0.05	0.00	0.00	0.39	0.00	0.00	0.00	0.00	0.01	0.00	0.55
	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.03	0.00	0.41	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.55
...											

HW faults

Int4 MNIST Batch Example

1D Confusion Matrix

Images	Possible Predictions										
	0	1	2	3	4	5	6	7	8	9	Fail
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.43	0.00	0.01	0.55
	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.99	0.00	
	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.43	0.01	0.55
	0.00	0.00	0.00	0.99	0.00	0.01	0.00	0.00	0.00	0.00	
	0.05	0.00	0.00	0.39	0.00	0.00	0.00	0.00	0.01	0.00	0.55
	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.03	0.00	0.41	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.55
...											

Silent Data
Corruption

FP4 E2M1 MNIST Batch Example

Hardware Setup

- Accelerator setup
 - Exact floating-point multiplication
 - Kulisch accumulation

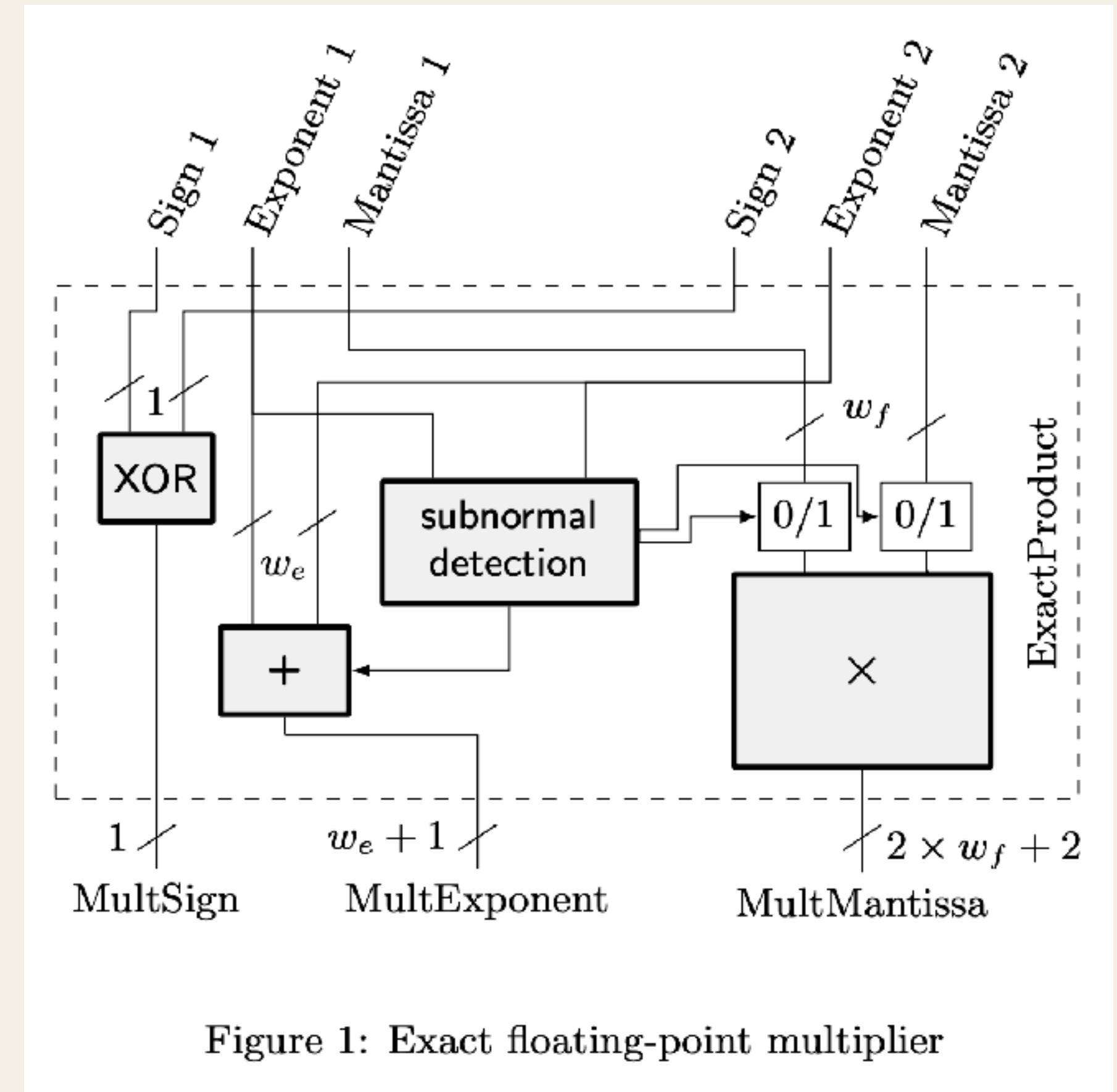
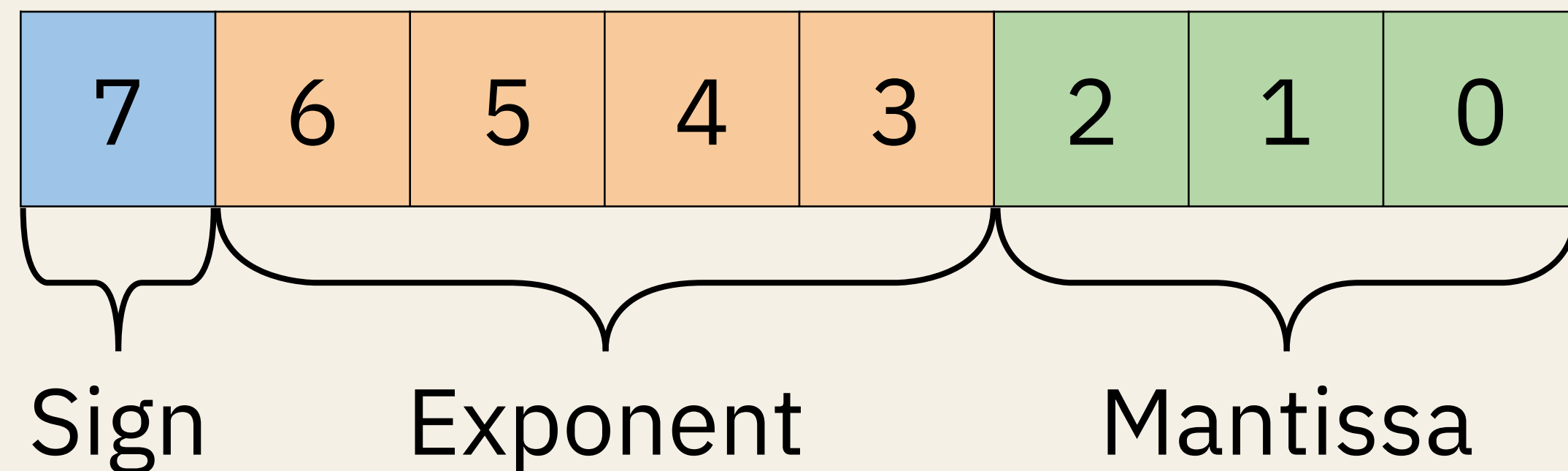
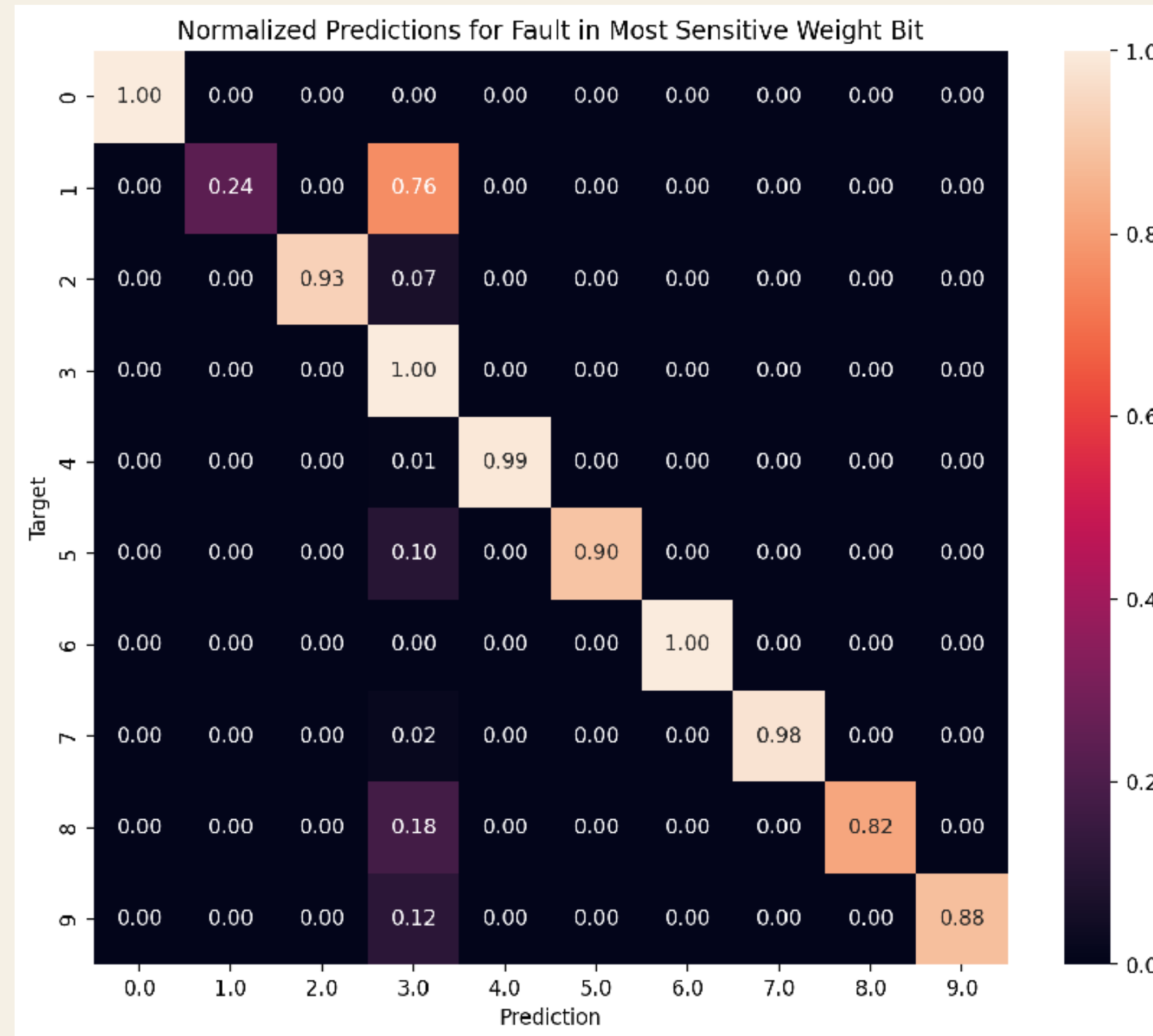


Figure 1: Exact floating-point multiplier

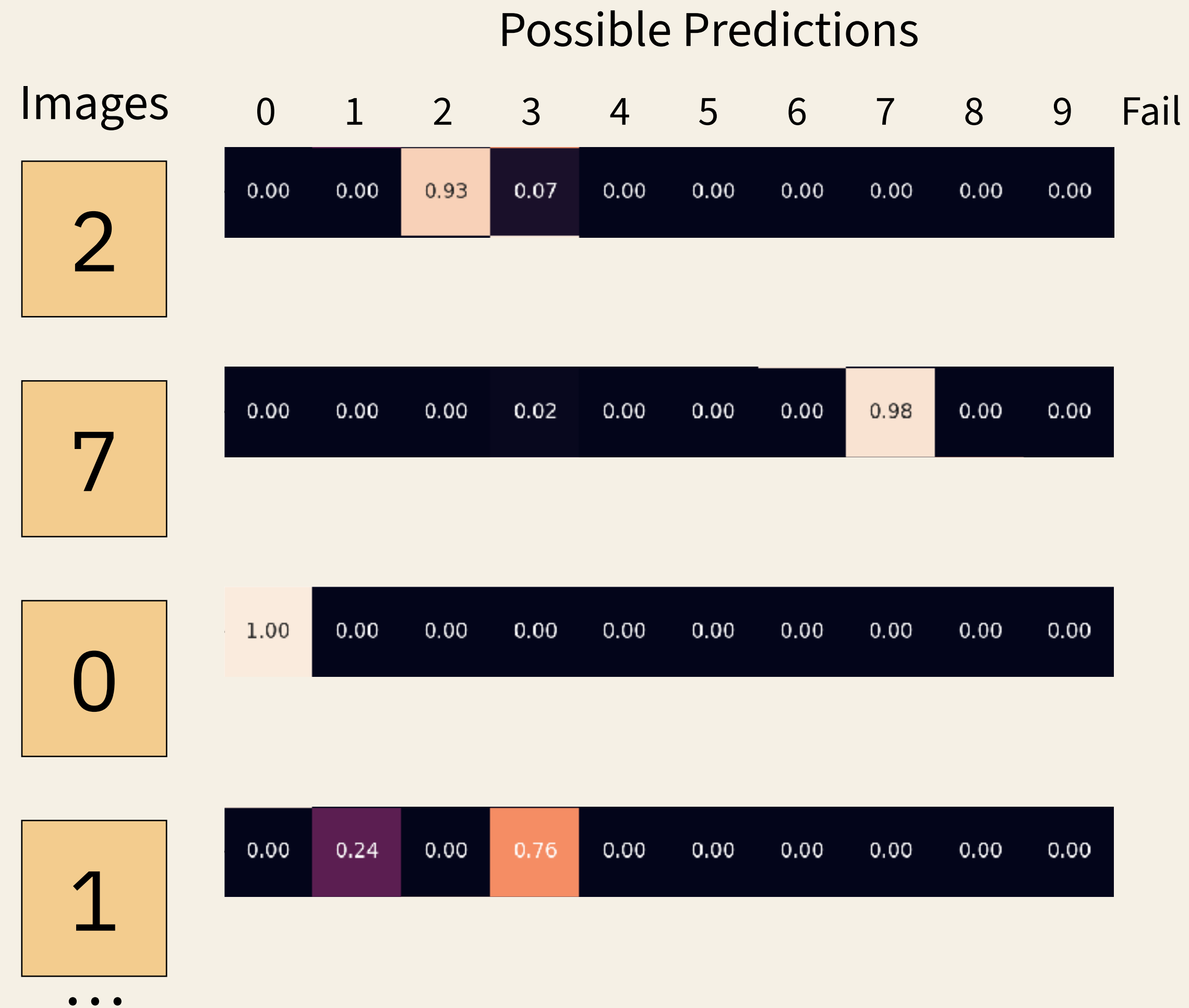
FP4 E2M1 MNIST Batch Example

Worst Case from Weight Faults (Baseline)



FP4 E2M1 MNIST Batch Example

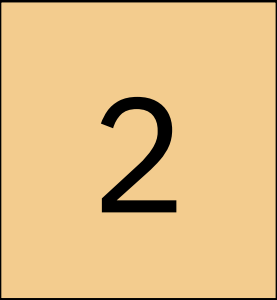
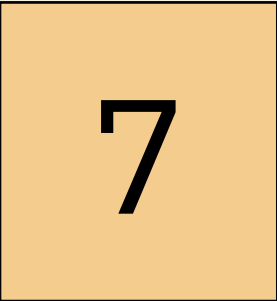
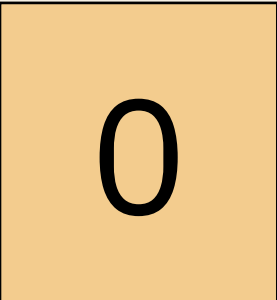
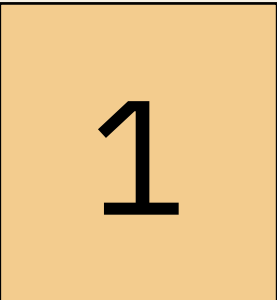
1D Confusion Matrix



Weight faults

FP4 E2M1 MNIST Batch Example

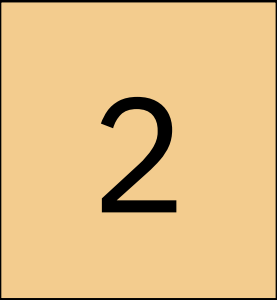
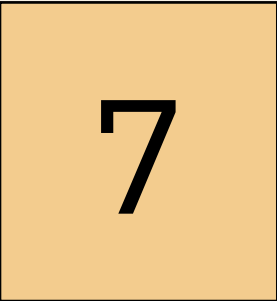
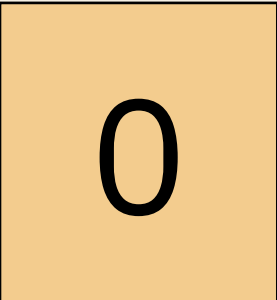
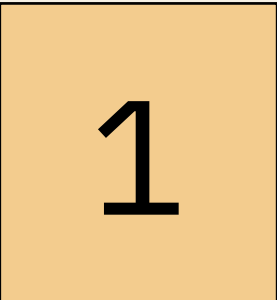
1D Confusion Matrix

Images	Possible Predictions										
	0	1	2	3	4	5	6	7	8	9	Fail
	0.00	0.00	0.93	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.05	0.00	0.39	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.55
	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.98	0.00	0.00	0.00
	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.43	0.00	0.00	0.55
	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.44	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.55
	0.00	0.24	0.00	0.76	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.03	0.41	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.55
...											

HW faults

FP4 E2M1 MNIST Batch Example

1D Confusion Matrix

Images	Possible Predictions										
	0	1	2	3	4	5	6	7	8	9	Fail
	0.00	0.00	0.93	0.07	0.00	0.00	0.00	0.00	0.00	0.00	
	0.05	0.00	0.39	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.55
	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.98	0.00	0.00	
	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.43	0.00	0.00	0.55
	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	0.44	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.55
	0.00	0.24	0.00	0.76	0.00	0.00	0.00	0.00	0.00	0.00	
	0.03	0.41	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.55
...											

Silent Data
Corruption

Ongoing Work

- Large scale fault injection campaigns on larger models
- Looking at transient faults
 - Using data from stuck-at faults to find 'sensitive' nets
- Considering dataflow
 - Weight/output stationary

