

# NEXUS

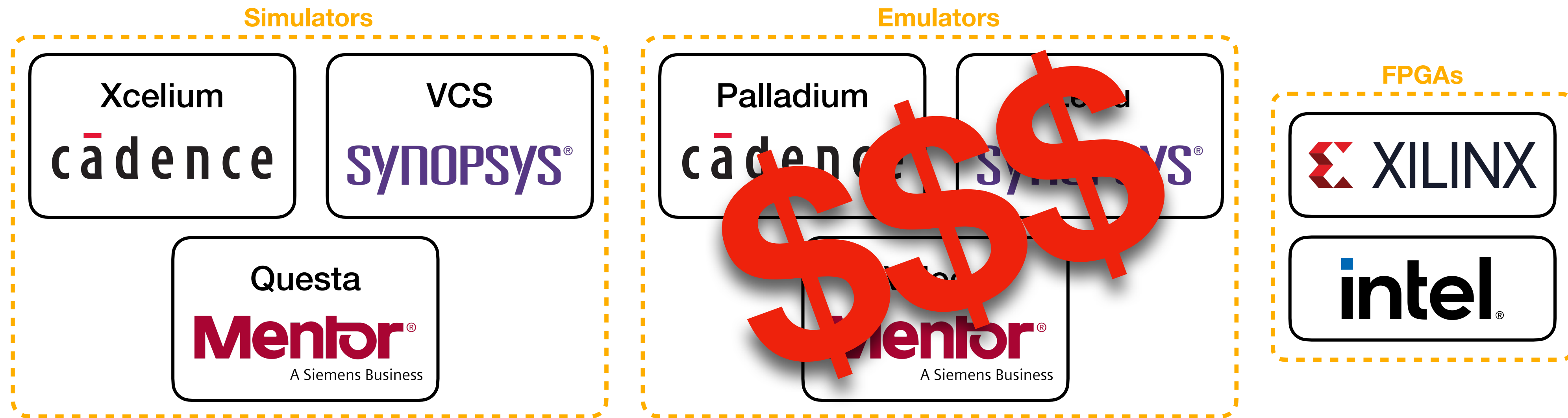
**Hardware Accelerated RTL Simulation**

**Peter Birch**  @intuity  intuity.io

# What is Nexus?

- Hobby project started in January 2021
- Hardware-accelerated RTL simulation
- For mid-sized FPGAs such as Xilinx Artix 7 200T

# Commercial Options



Increasing execution speed  
Increasing effort to setup  
Decreasing design visibility

# Open Source & Low-Cost Options

## Simulators



## Emulators



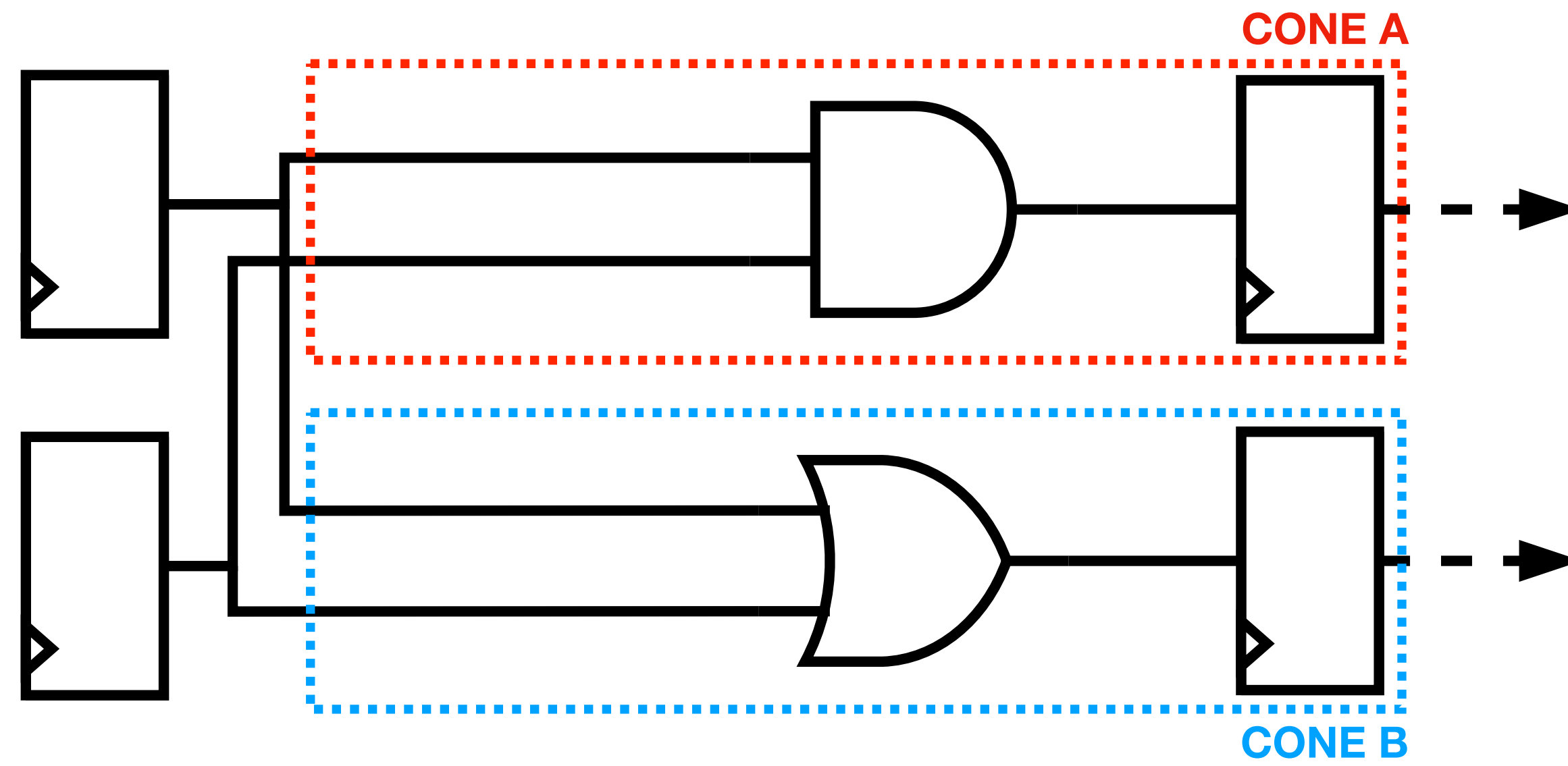
## FPGAs



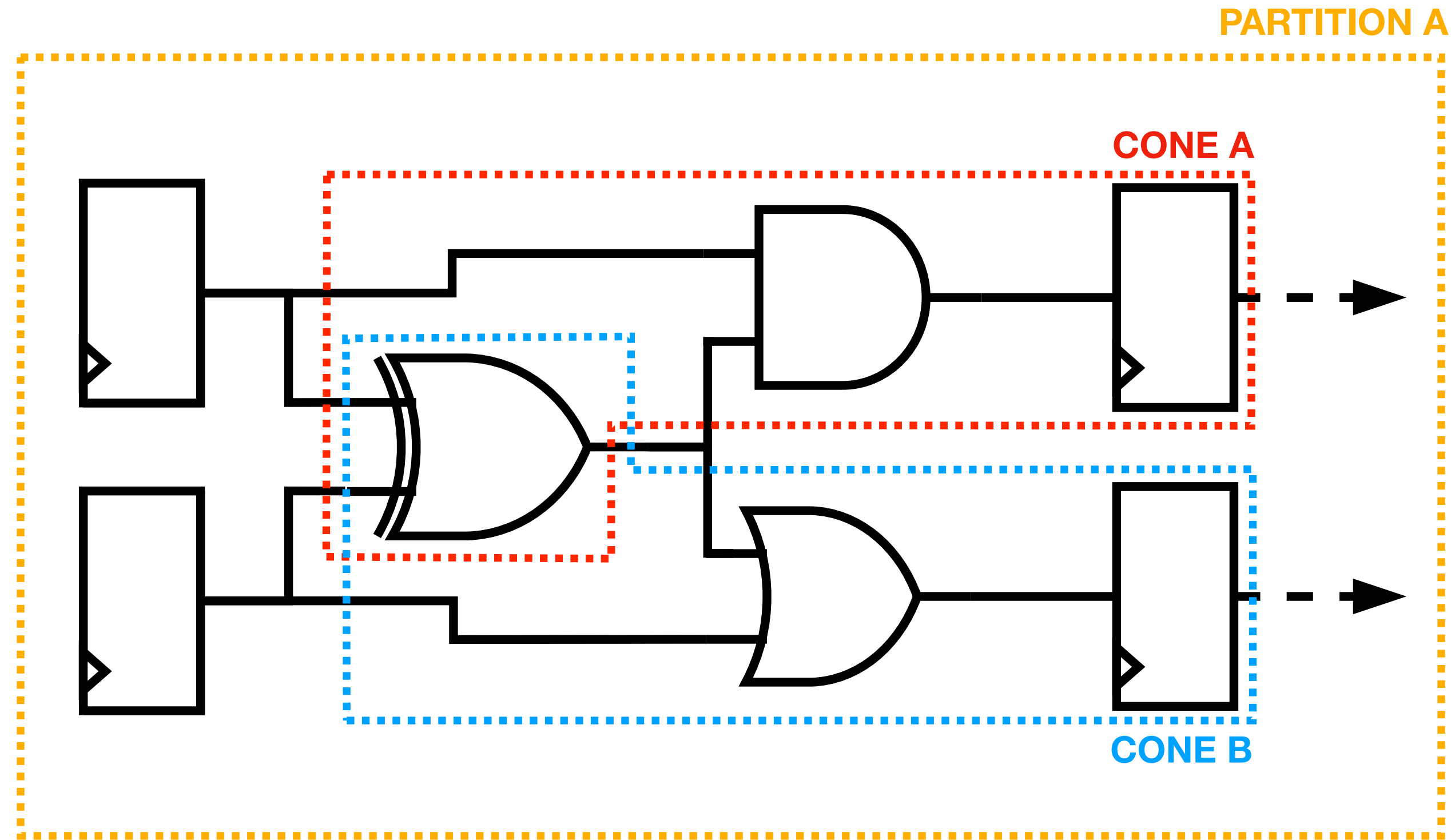
# How is Nexus Different?

- Free & open source
- Targets moderately sized FPGAs
- Design is simulated by many small programs running in parallel
- No timing constraints to meet, so no place-and-route flow
- Signal tracing can be added and removed on the fly

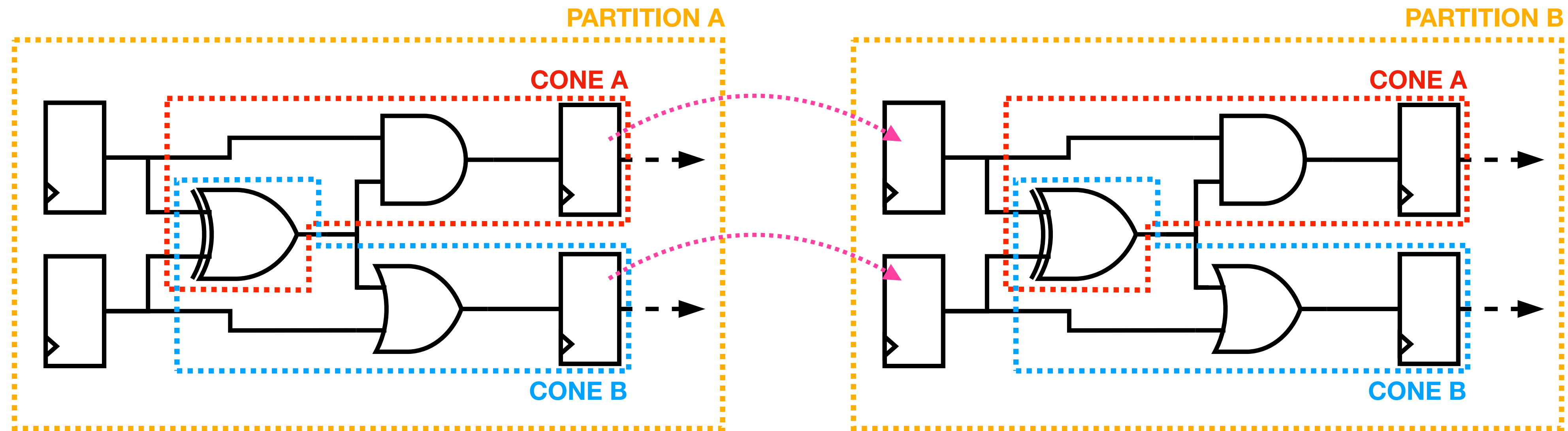
# Concept



# Concept

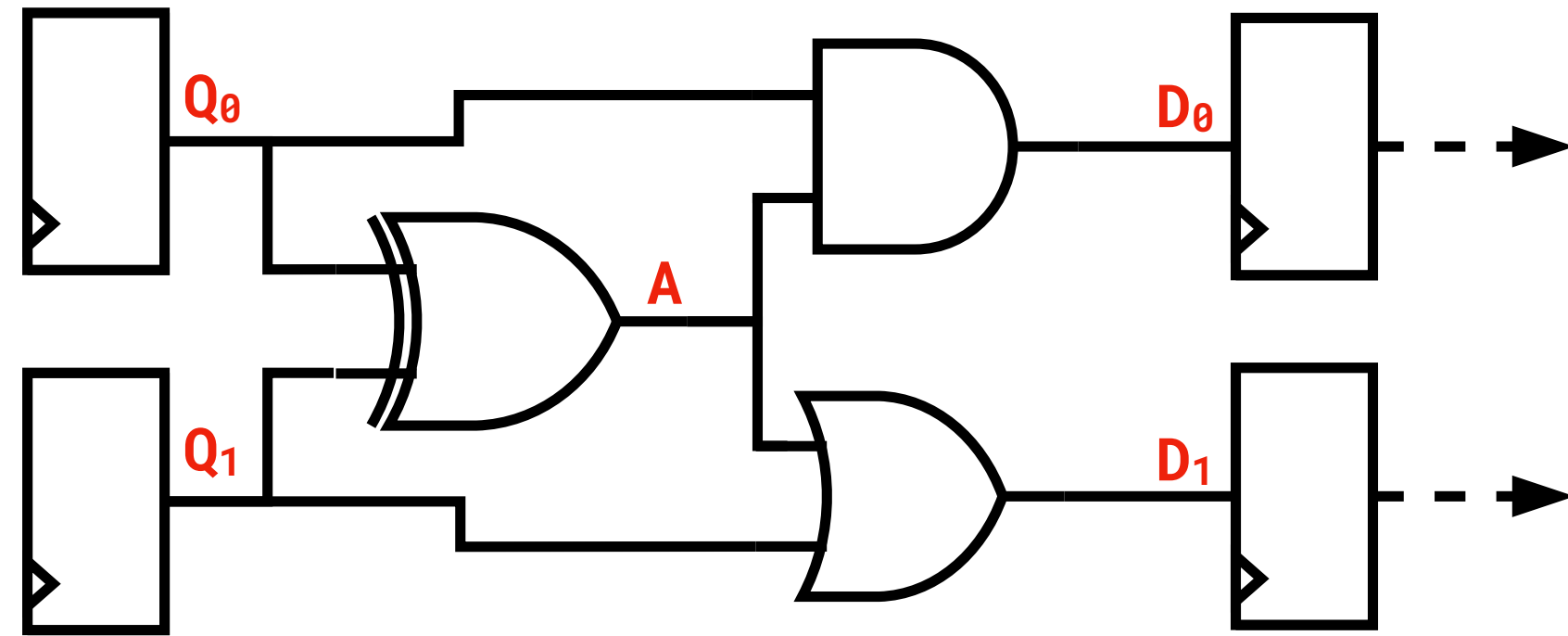


# Concept





# Concept

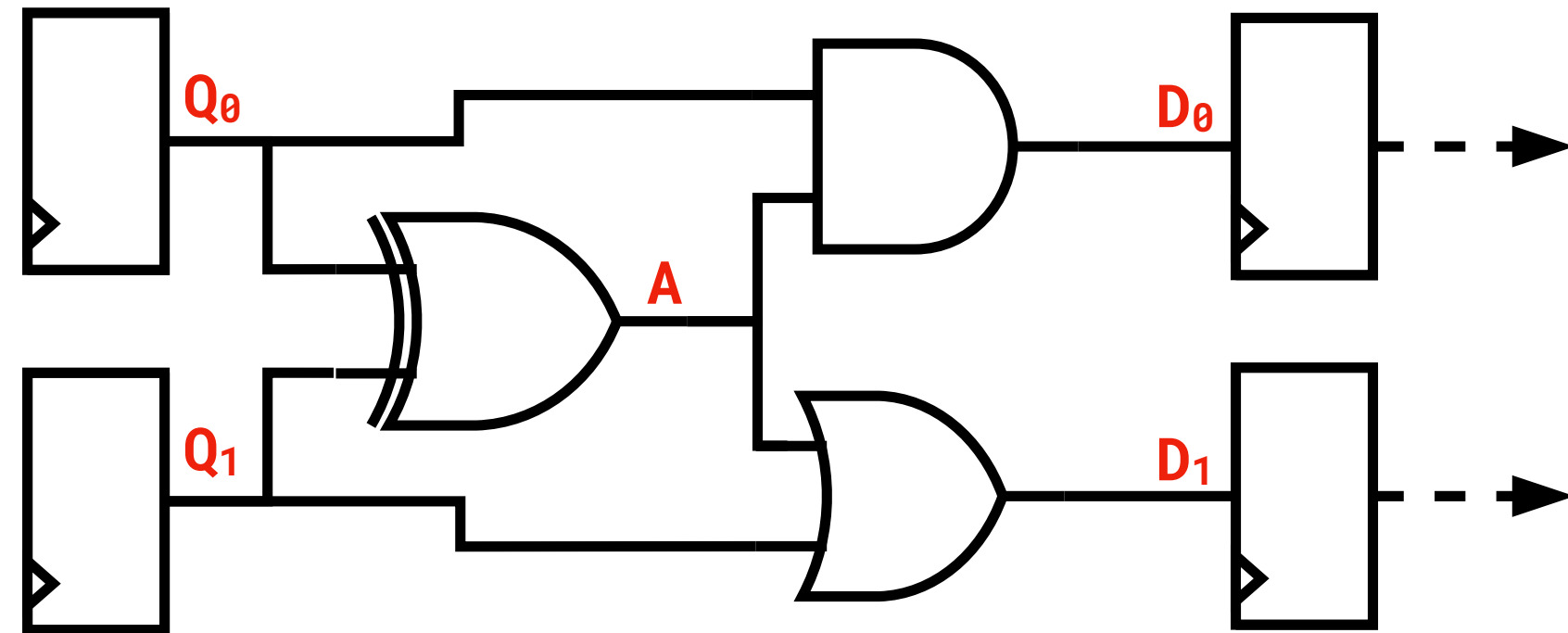


$$A = \text{XOR}(Q_0, Q_1)$$

$$D_0 = \text{AND}(Q_0, A)$$

$$D_1 = \text{OR}(Q_1, A)$$

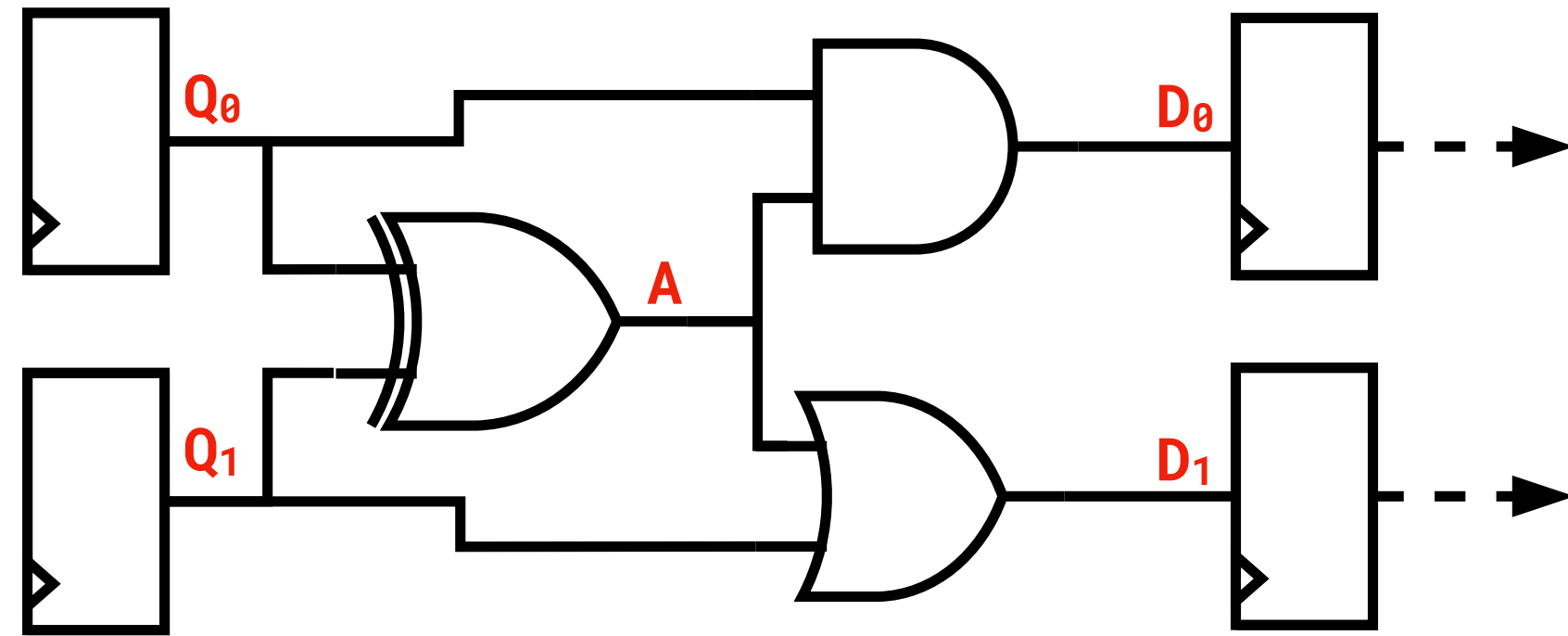
# Concept



$$D_0 = \text{AND}(Q_0, \text{XOR}(Q_0, Q_1))$$

$$D_1 = \text{OR}(Q_1, \text{XOR}(Q_0, Q_1))$$

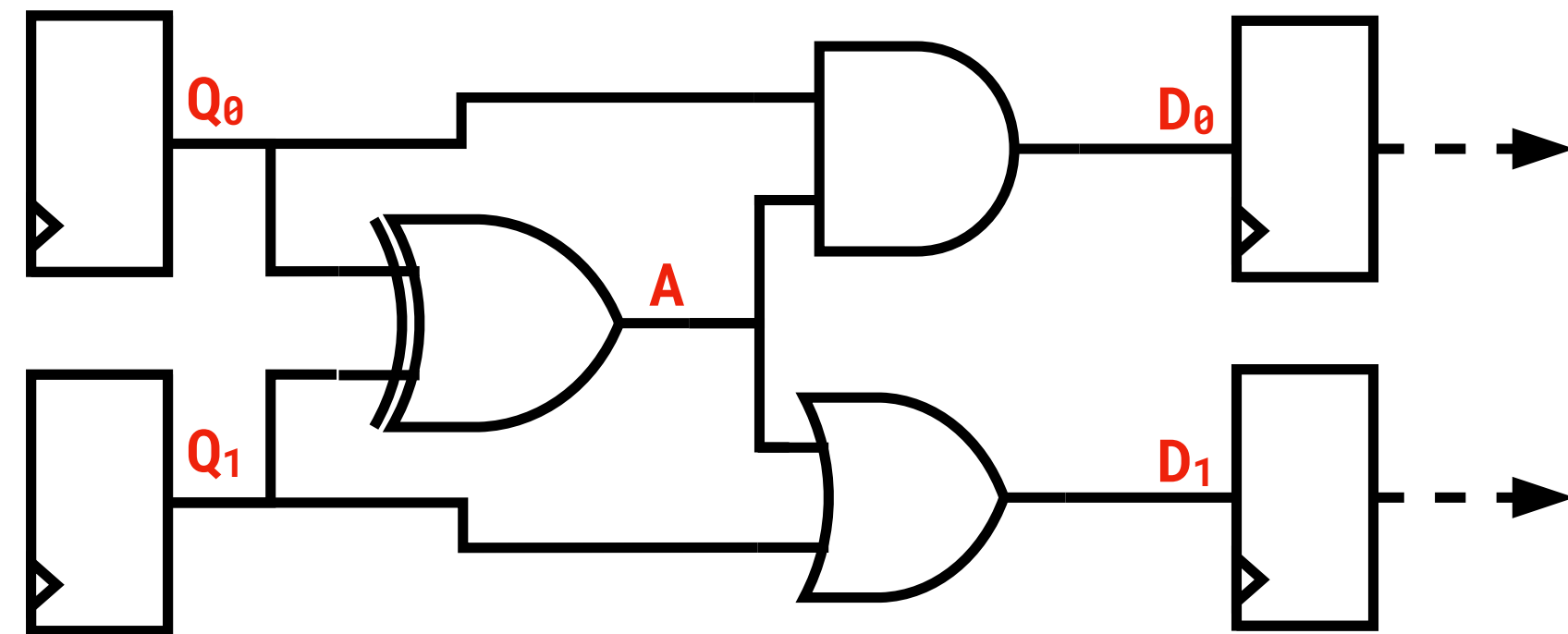
# Concept



$$D_1 = \text{OR}(Q_1, \text{XOR}(Q_0, Q_1))$$

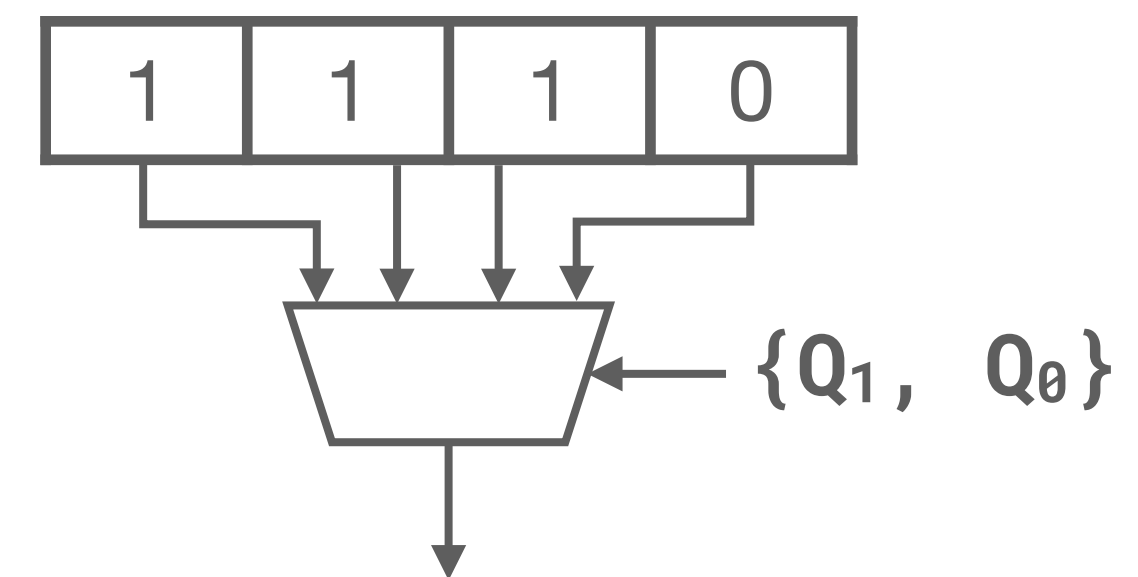
$Q_1, Q_0$	$D_0$
0, 0	0
0, 1	1
1, 0	1
1, 1	1

# Concept



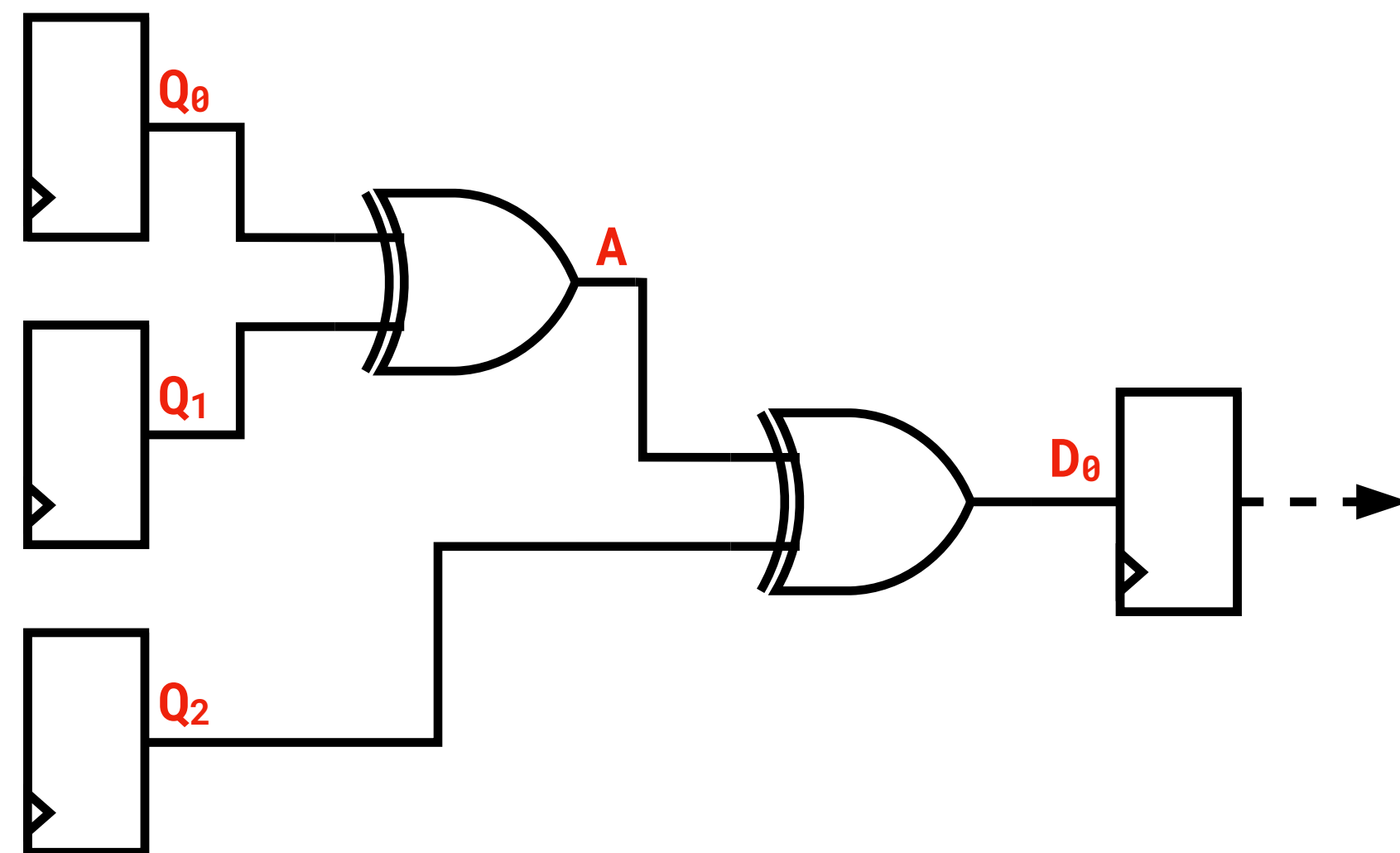
$$D_1 = \text{OR}(Q_1, \text{XOR}(Q_0, Q_1))$$

$Q_1, Q_0$	$D_0$
0, 0	0
0, 1	1
1, 0	1
1, 1	1

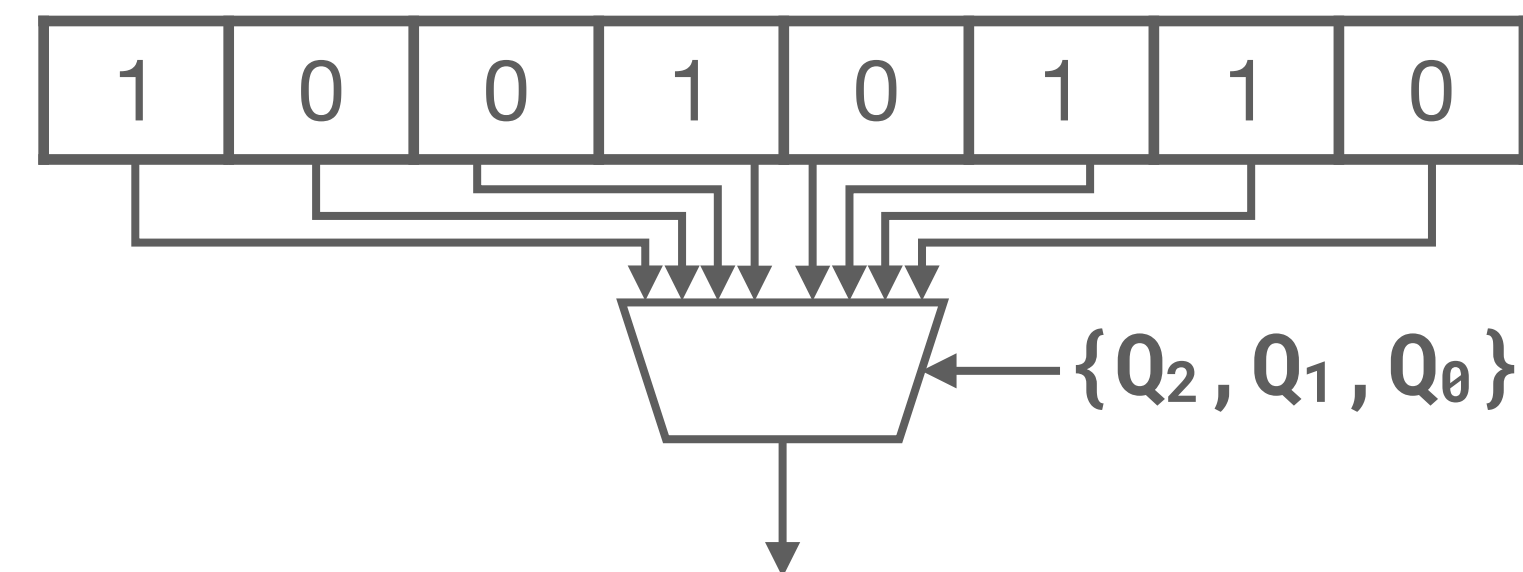


# Concept

$$D_0 = \text{XOR}(\text{XOR}(Q_0, Q_1), Q_2)$$



$Q_2, Q_1, Q_0$	$D_0$
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



# Concept



# Concept

Instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD	0	0	0	RSVD				OFFSET		ADDRESS								SLOT	RSVD				TGT		RSVD							
STORE	0	0	1	RSVD				OFFSET		ADDRESS								SLOT	MASK						SRC							
WAIT	0	1	0	RSVD																												
SEND	0	1	1	RSVD				OFFSET		ADDRESS								SLOT	ROW			COLUMN			SRC							
TRUTH	1	0	0	TABLE								MUX_C		MUX_B		MUX_A		SRC_C		SRC_B		RSVD		SRC_A								
SHUFFLE	1	1	BIT7		BIT6		BIT5		BIT4		BIT3		BIT2		BIT1		BIT0		TGT		SRC											

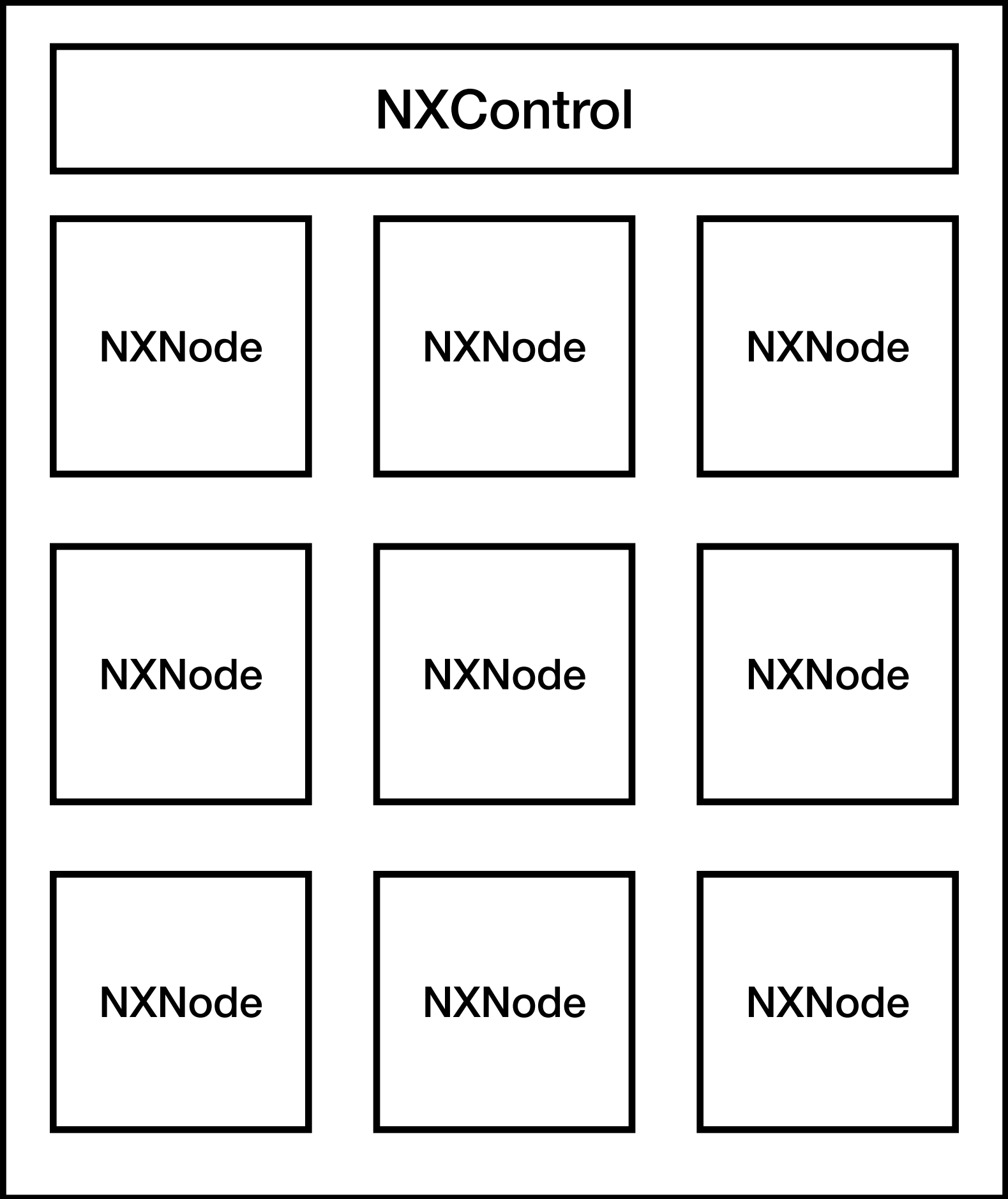


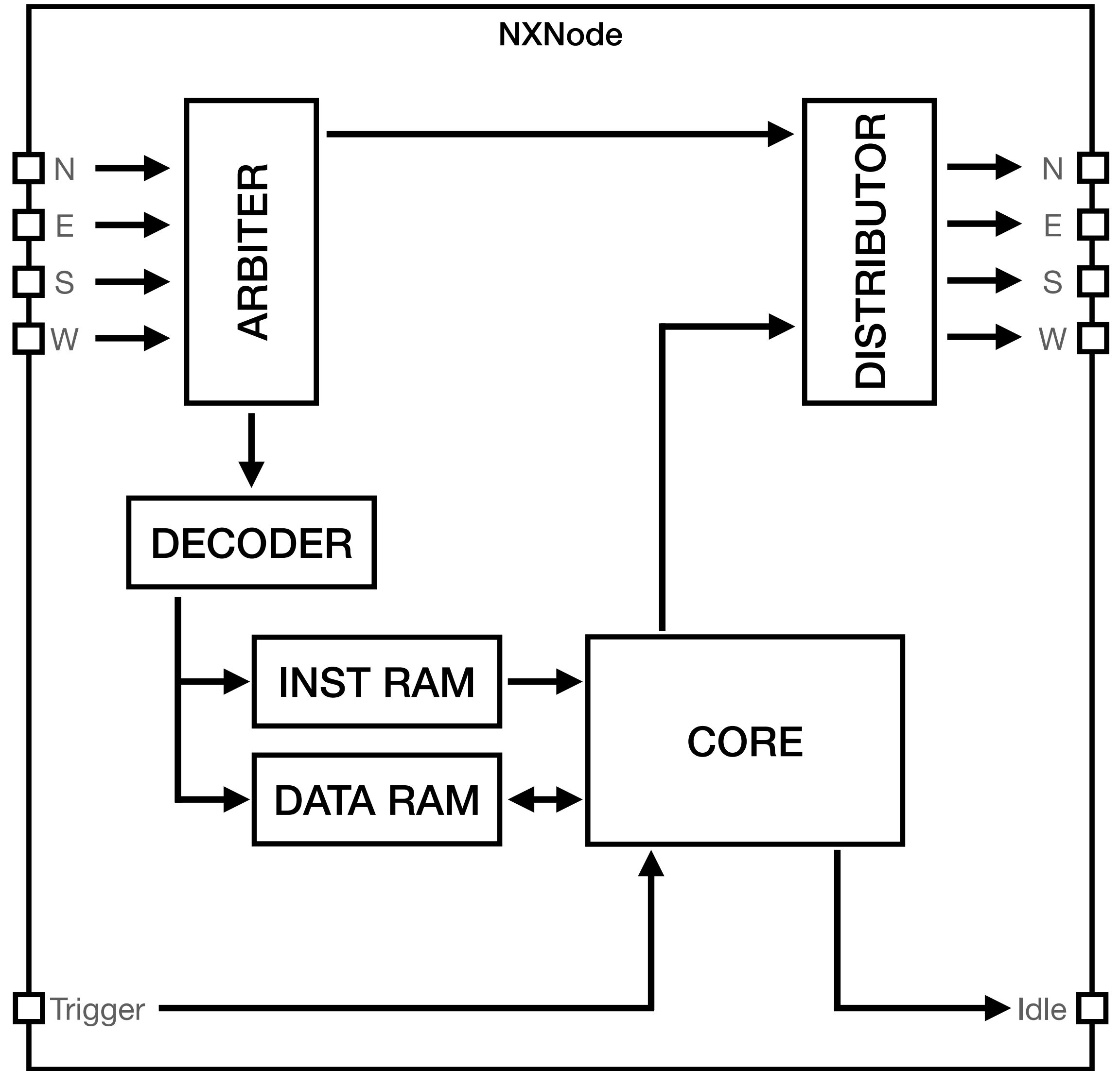


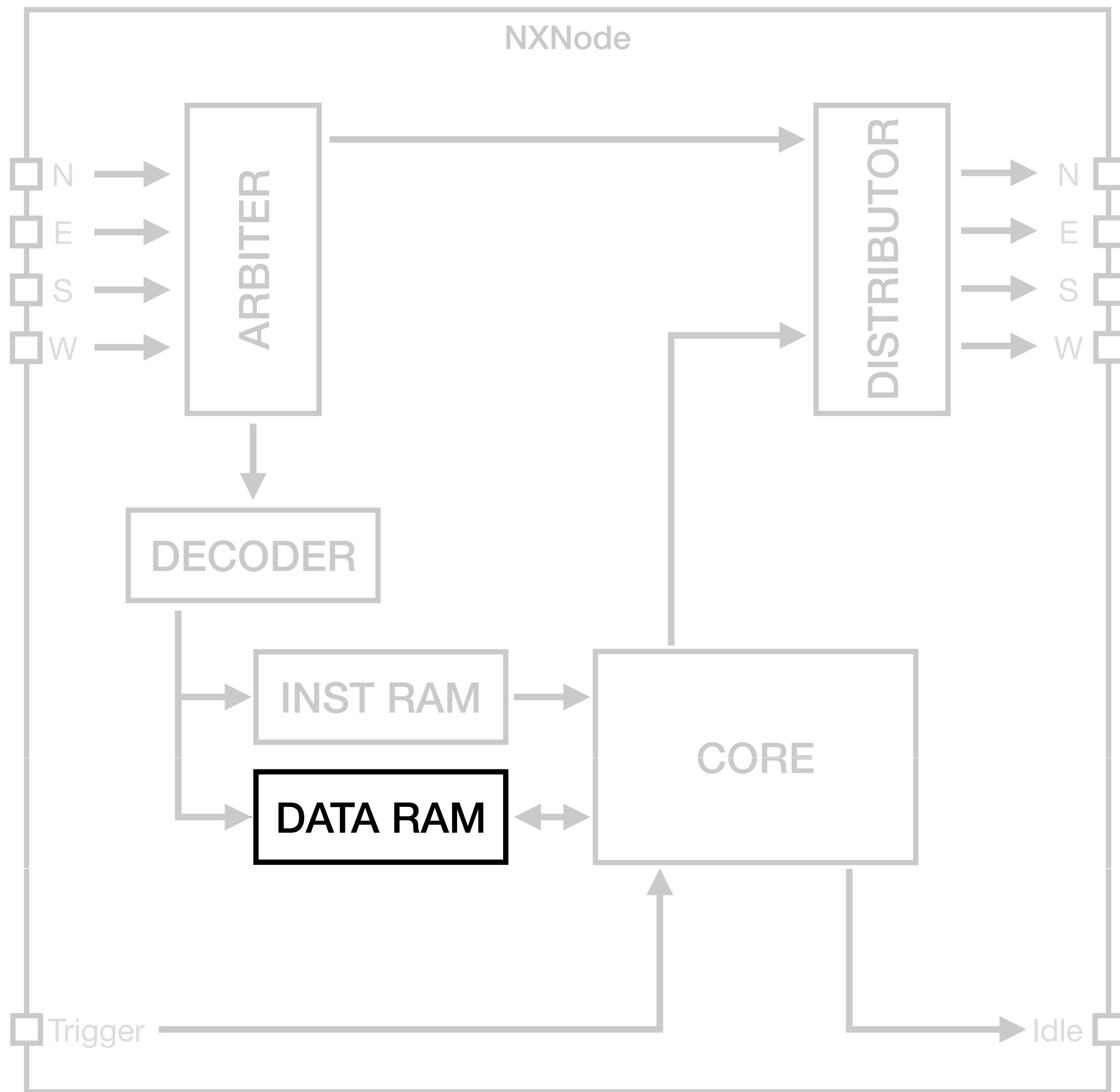
Host System



PCIe





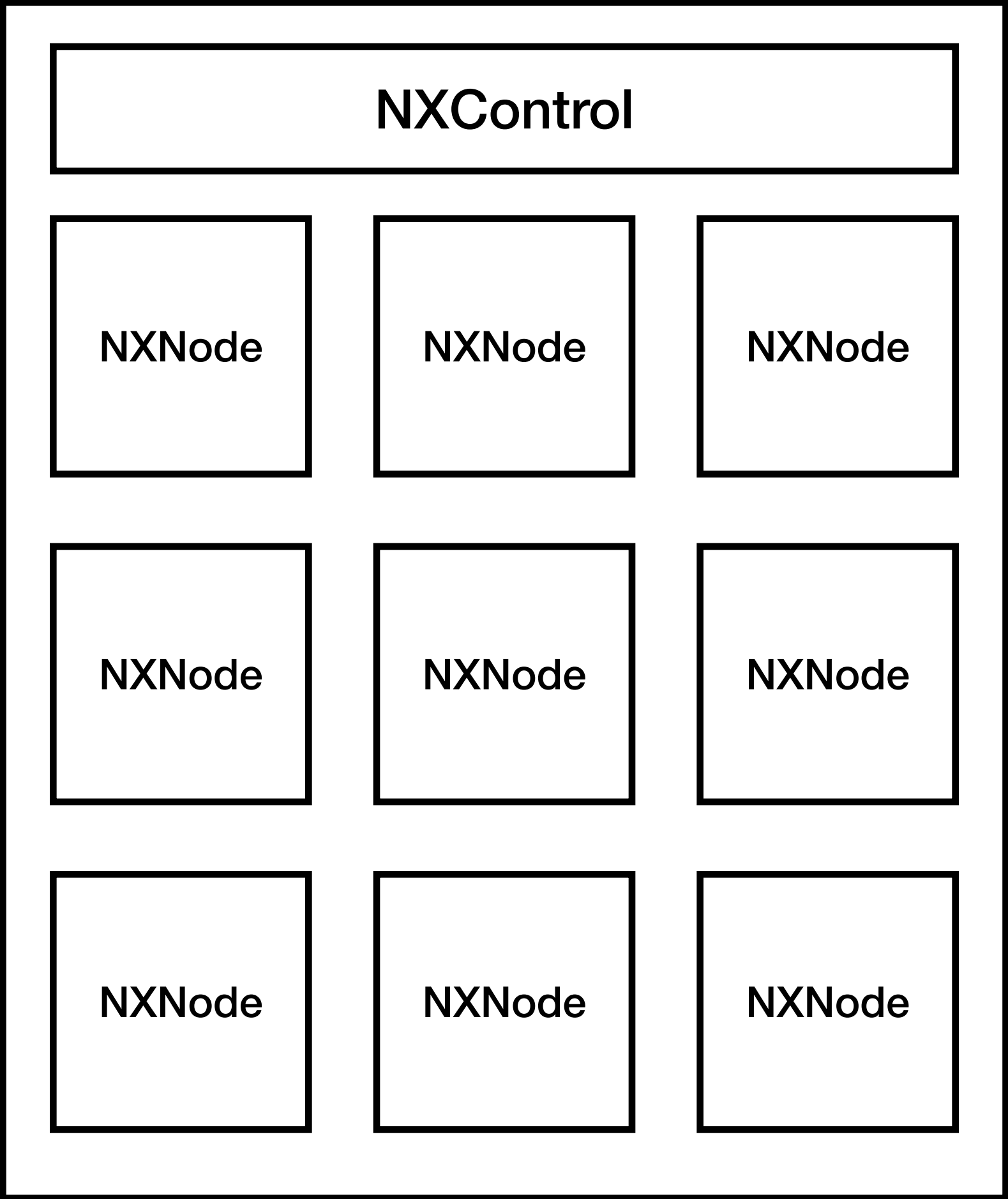


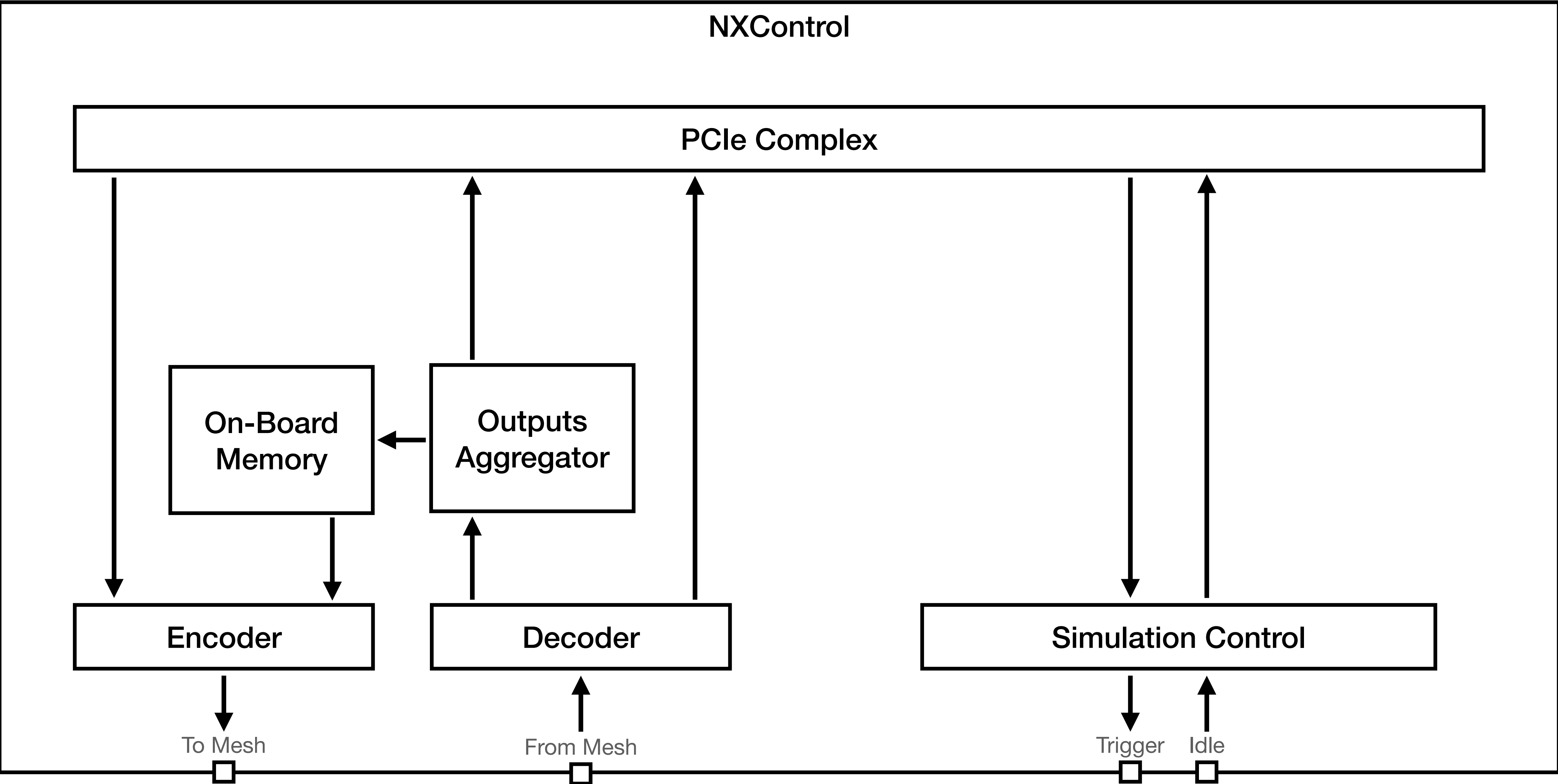
- **offset** register alternates every cycle
- Drives the least-significant address bit of the RAM
- **LOAD & STORE** can persist, invert, or force the offset bit
- Enables next state computation without corrupting the current state

Host System

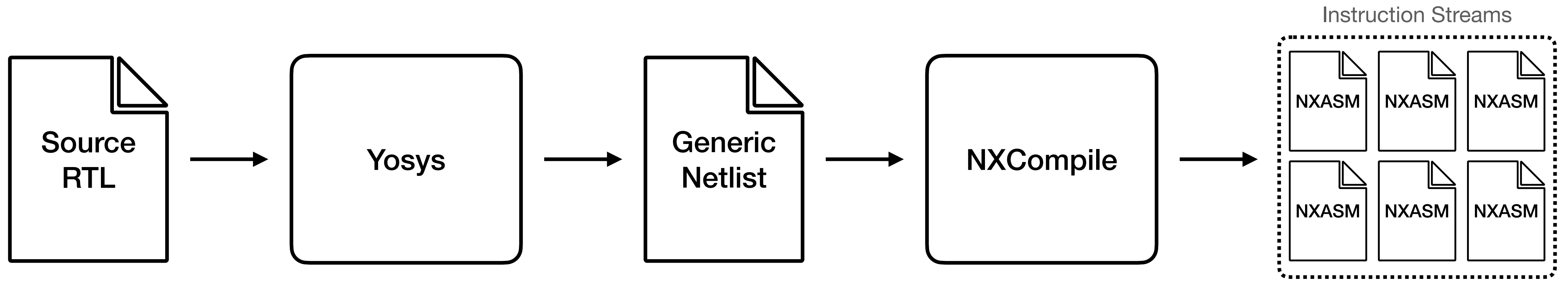


PCIe

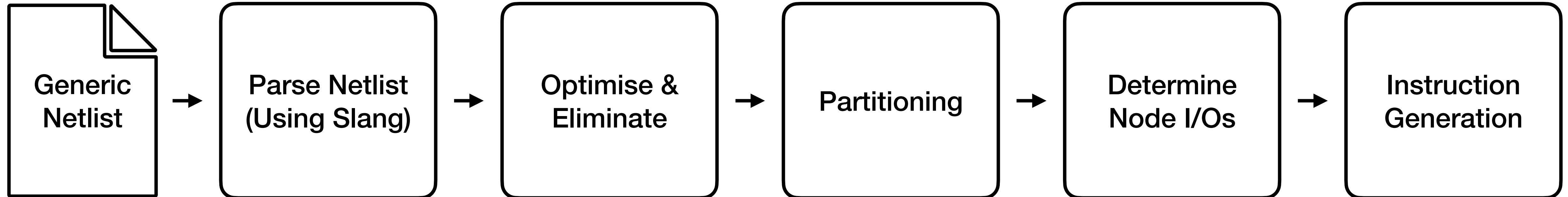




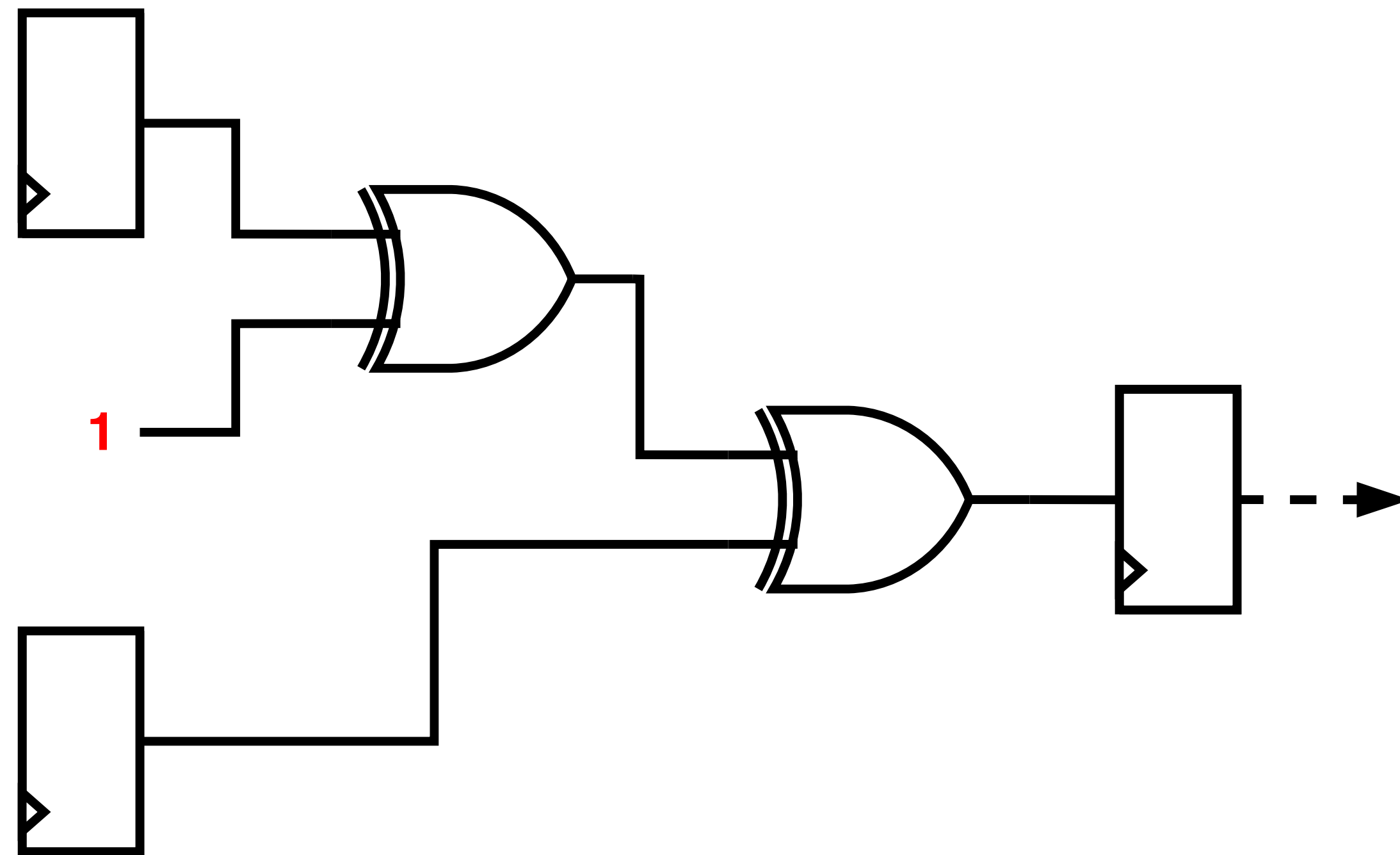
# Toolchain



# NXCompile

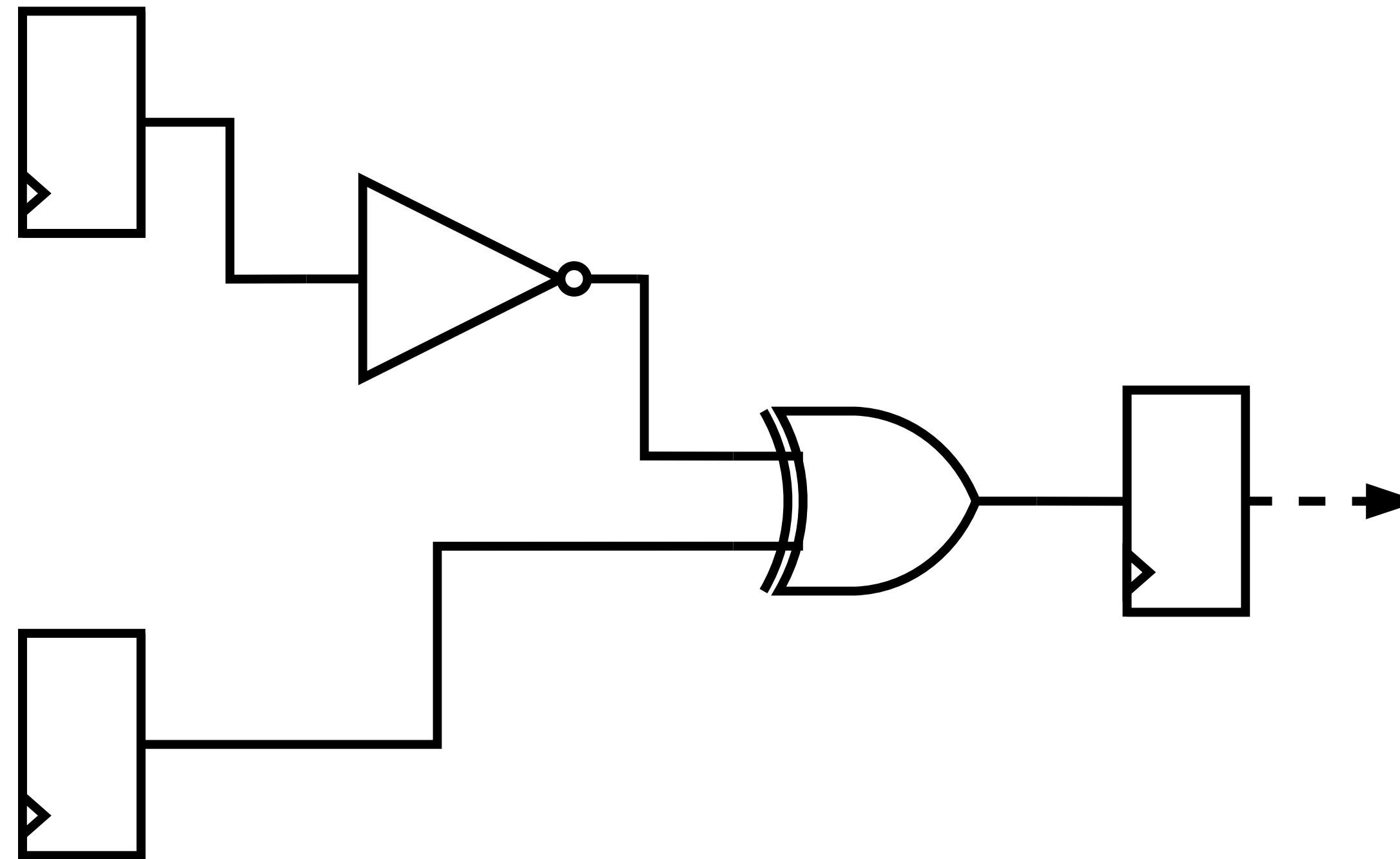


# NXCompile: Optimisation

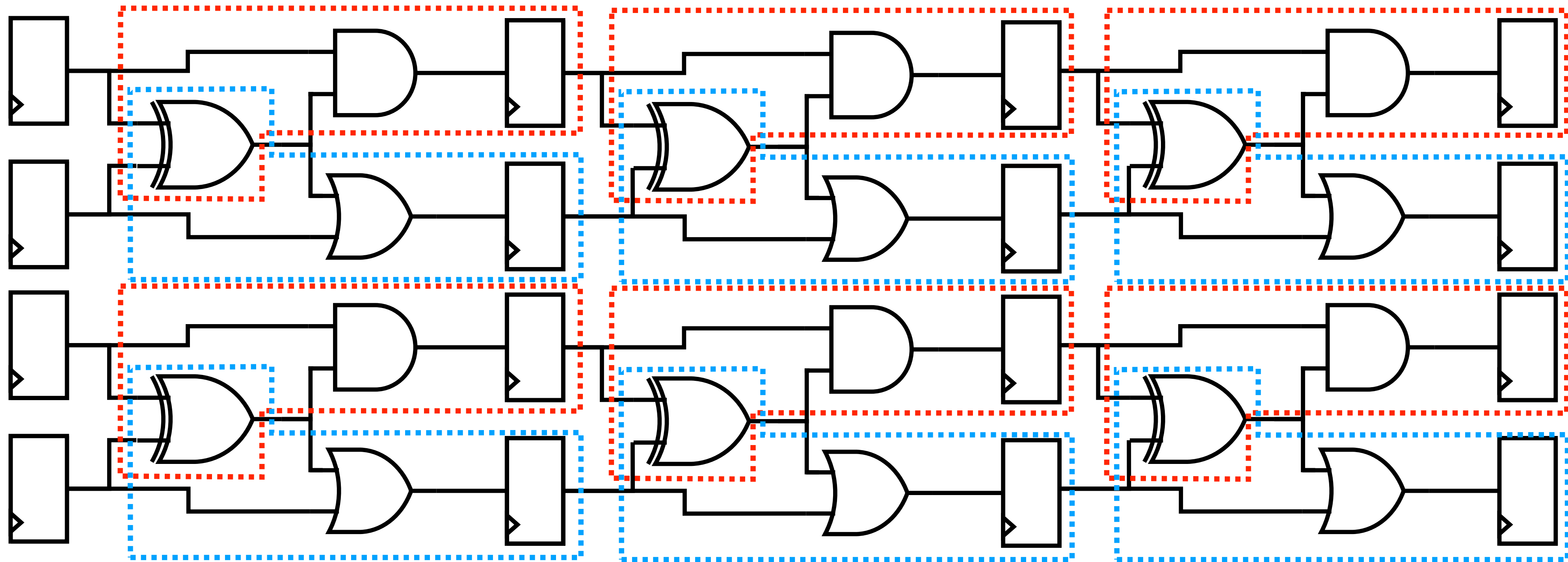




# NXCompile: Optimisation

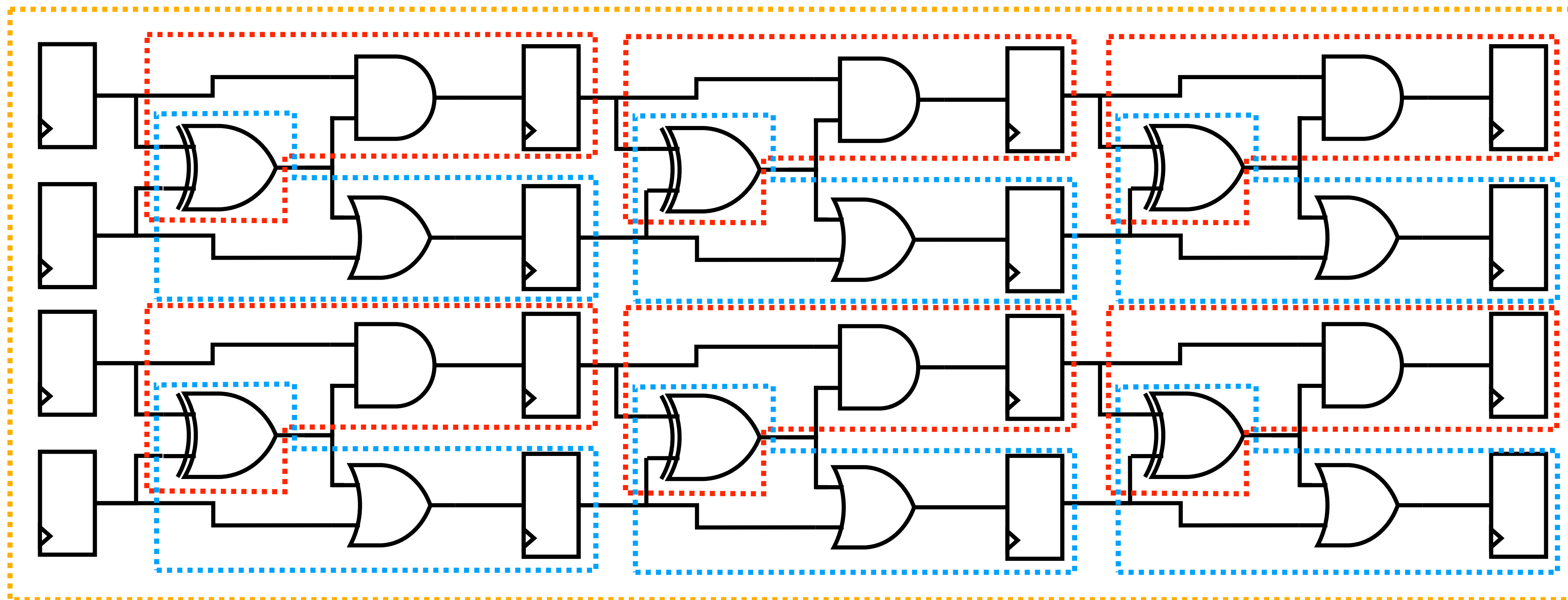


# NXCompile: Partitioning

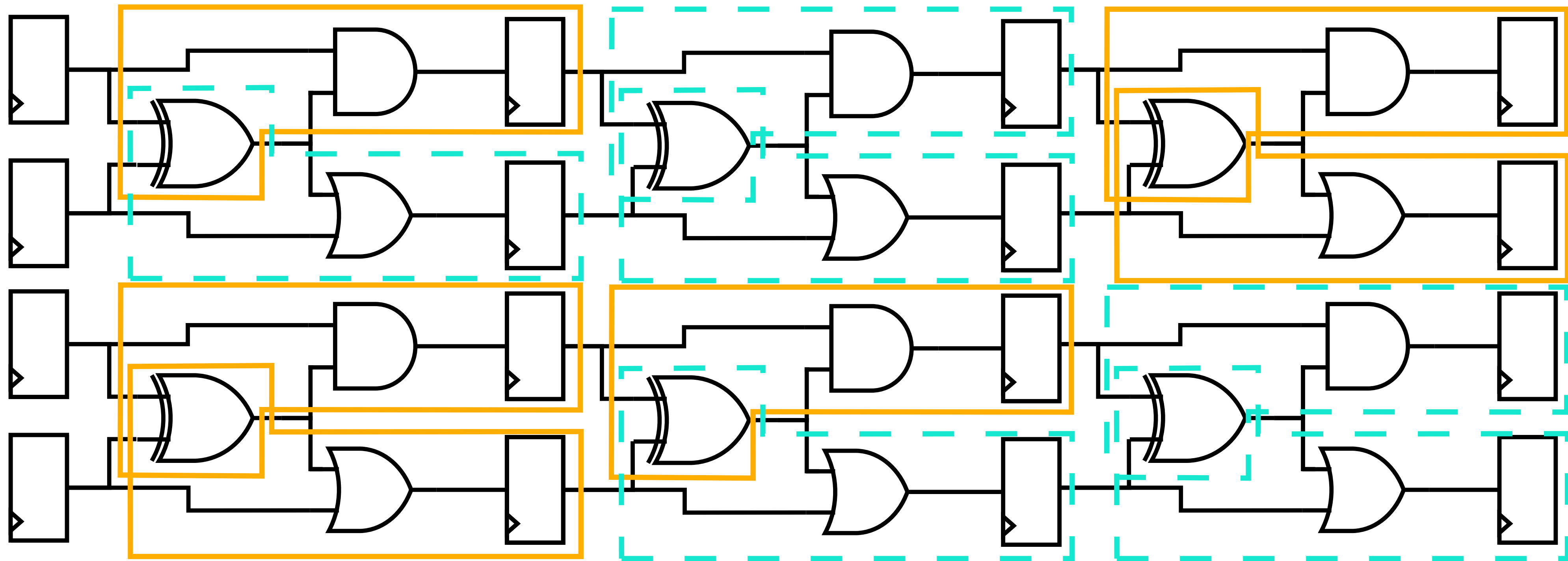


# NXCompile: Partitioning

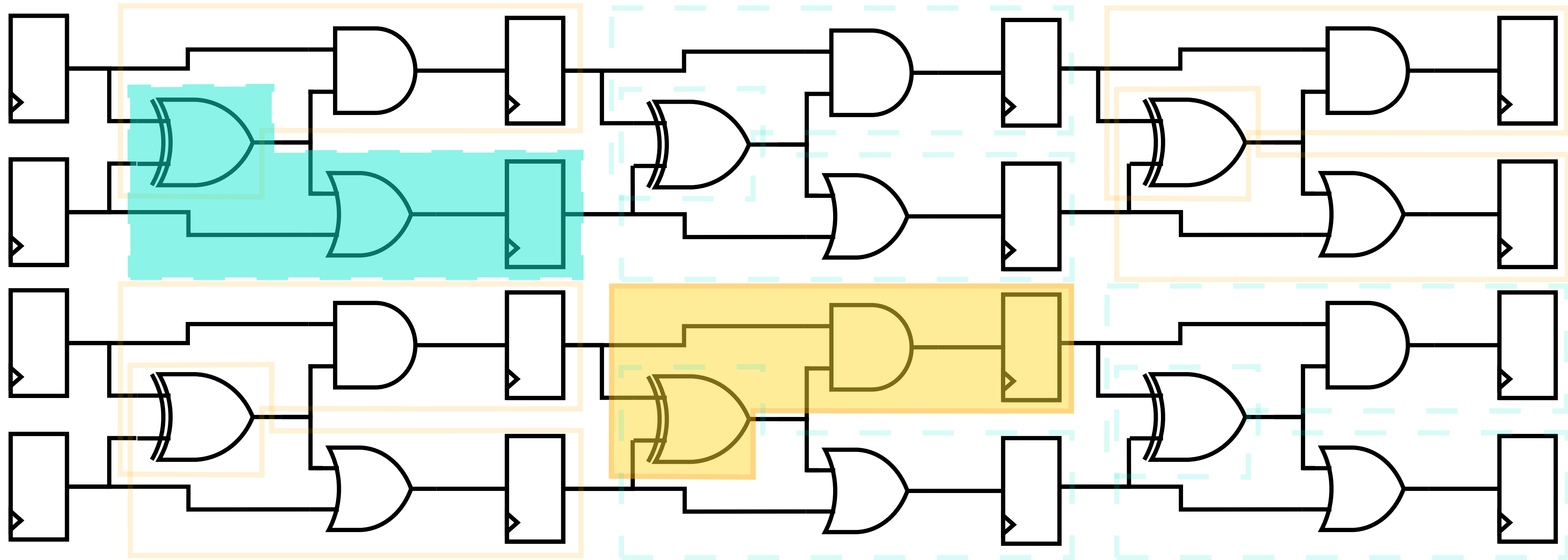
PARTITION A



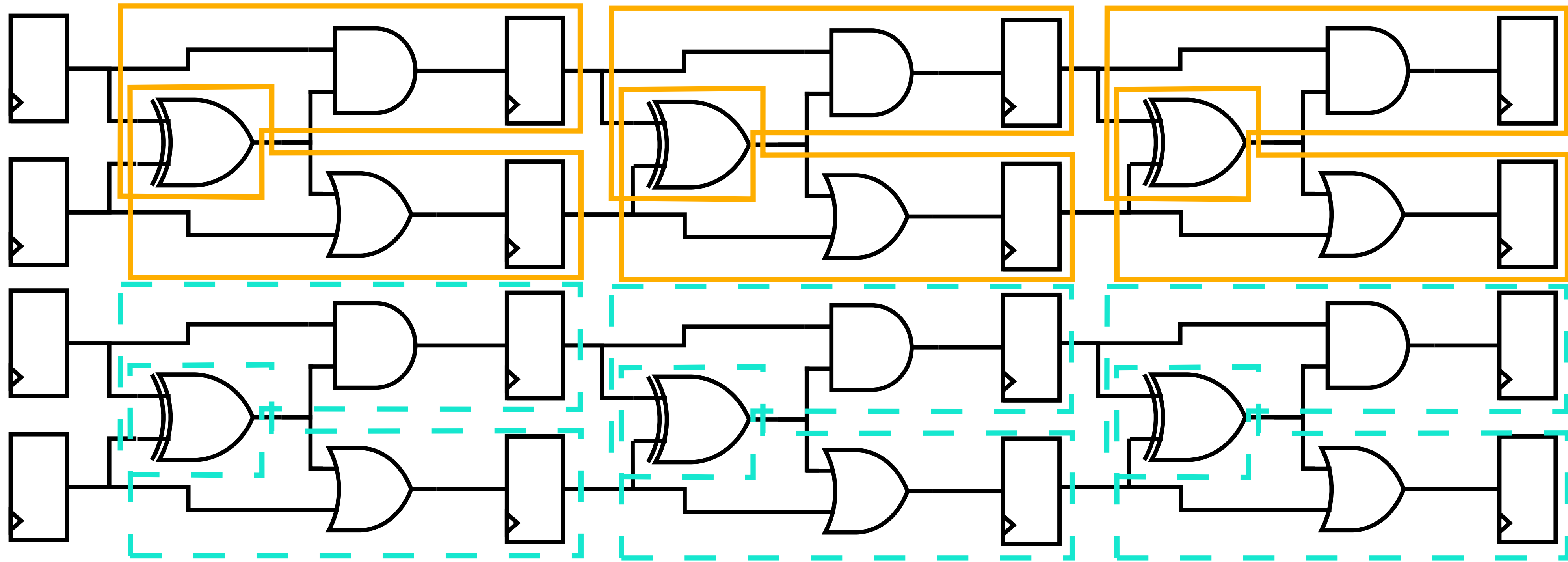
# NXCompile: Partitioning



# NXCompile: Partitioning



# NXCompile: Partitioning



# NXCompile: Instruction Generation

Result	Operation	Encoding
T0	<code>XOR(AND(A[2], A[1]), A[0])</code>	0x80
T1	<code>AND(T0, A[3])</code>	0x88
T2	<code>AND(T1, A[4])</code>	0x88
T3	<code>AND(A[5], T2)</code>	0x88
T4	<code>XOR(AND(A[6], A[7]), T3)</code>	0x80

# NXCompile: Instruction Generation

Result	Operation	Encoding
T0	<b>XOR</b> ( <b>AND</b> (A[2], A[1]), A[0])	0x80
T1	<b>AND</b> (T0, A[3])	0x88
T2	<b>AND</b> (T1, A[4])	0x88
T3	<b>AND</b> (A[5], T2)	0x88
T4	<b>XOR</b> ( <b>AND</b> (A[6], A[7]), T3)	0x80

PC	Instruction
0	<b>LOAD</b> R0, 0x0
1	<b>LOAD</b> R1, 0x2
2	<b>TRUTH</b> 0x80 2 1 0 R0 R0 R0 → T0
3	<b>TRUTH</b> 0x88 0 0 3 R0 R7 R0 → T1
...	
5	<b>STORE</b> R7, 0x4, ...



# Predicted Performance

- 10x10 mesh fits on a Xilinx Artix-7 200T
- PicoRV32 with 10,231 gates and 1,624 flops requires ~60 nodes
- 250 MHz FPGA fabric clock → simulation clock of ~550 kHz (predicted)
- For comparison (on an Intel i5-1038NG7):
  - Icarus Verilog - 36 kHz
  - Verilator - **240 MHz**


# How to go faster?

- Verilator achieves a simulated clock tick every 10 CPU cycles
- Leverages much wider scalar arithmetic
- Possible optimisations:
  - Perform scalar addition in a single cycle
  - Evaluate multiple smaller logical operations in a single cycle
  - Reduce cycles spent on data manipulation

# What's Next

- Toolchain:
  - Partitioning working
  - 'Next-state' instruction sequences work
  - State update instruction sequences in progress
- Fast C++ model being used to prove architecture
- Aiming for RTL complete by the end of the year

# Conclusion

- Nexus is close to achieving hardware-accelerated simulation
- The architecture and compiler are reaching a point where small designs like PicoRV32 are viable
- Plenty of scope for further optimisations
- Project is entirely open source:
  -  [github.com/intuity/nexus](https://github.com/intuity/nexus)
  - See the [nxcompile\\_slang](#) branch for the latest progress