An FPGA Primer

for Software Engineers

Dr Matthew Grosvenor
Principal Engineer, Cisco


High Frequency Trading


Network Offload / Acceleration


Computational Storage


High Performance Compute/ AI / ML

```
int a = 1;
int b = 0;

int out_1;
int out_2;

out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
int a = 1;
int b = 0;

int out_1;
int out_2;

out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

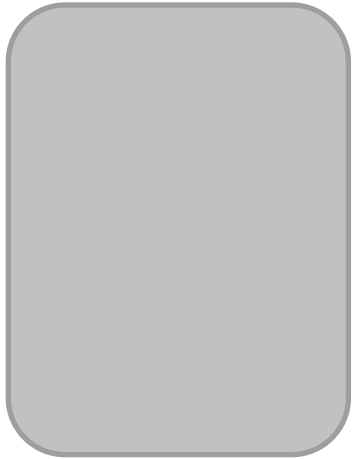
```
int a = 1;
int b = 0;

int out_1;
int out_2;

out_1 = ~(a & b);
Out_2 = out_1 & b;
```

Compiler

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

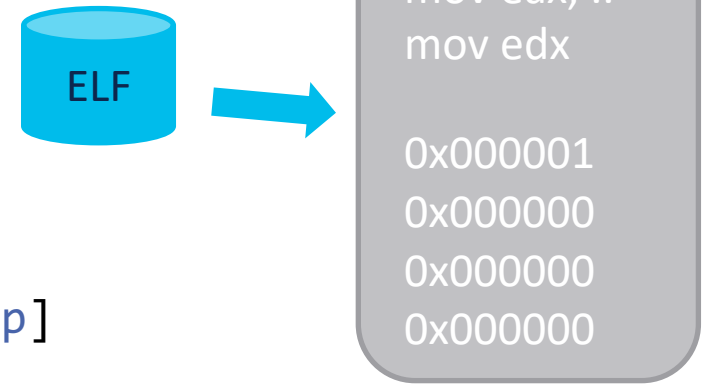
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



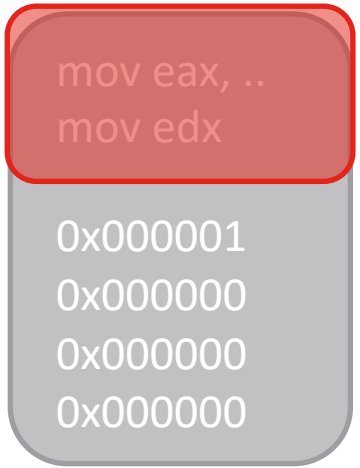
RAM

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4
```

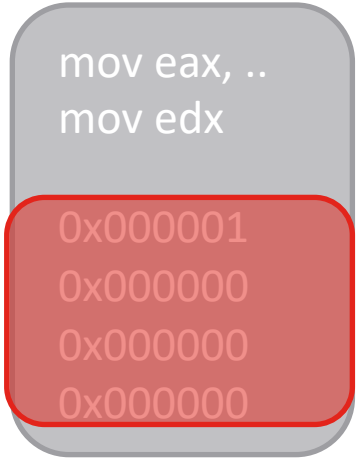
```
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



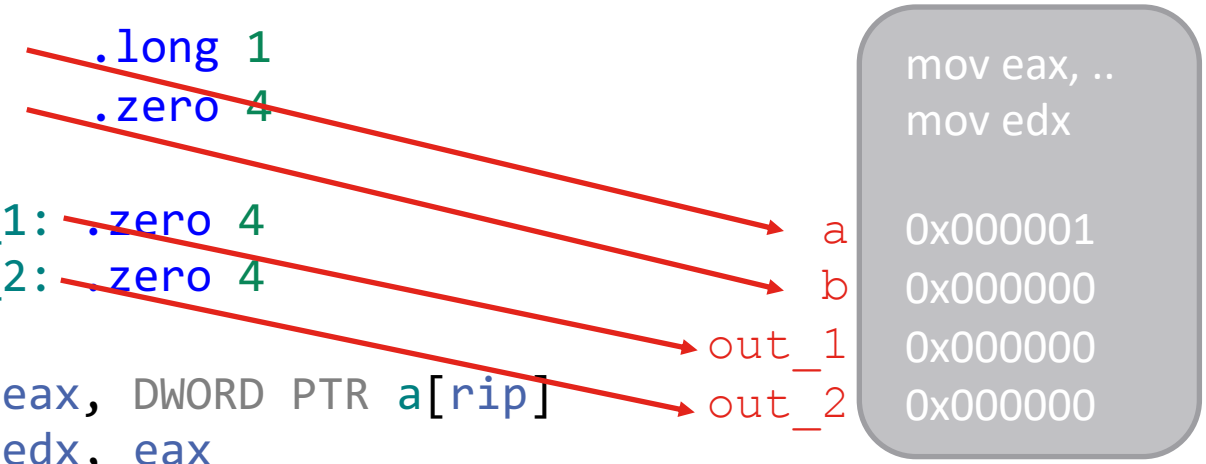
RAM

Code section

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



```
a:      .long 1
b:      .zero 4

out_1:  .zero 4
Out_2:  .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



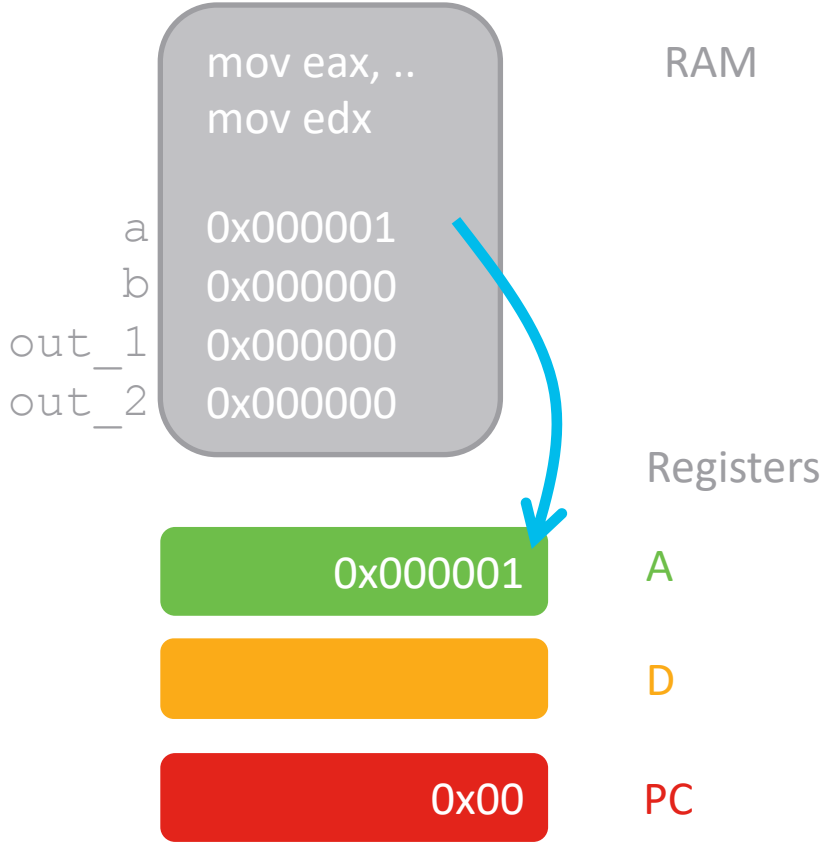
RAM

Data section

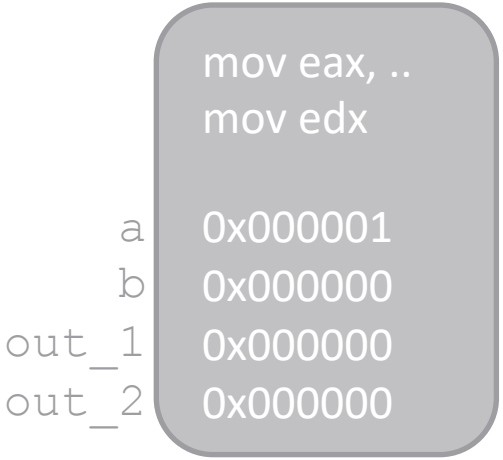
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

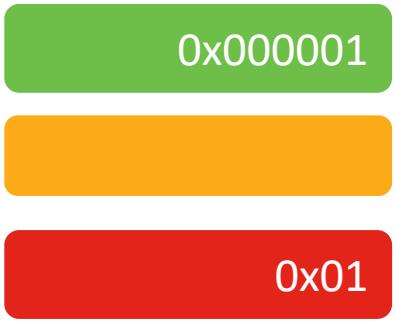


```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



RAM

Registers



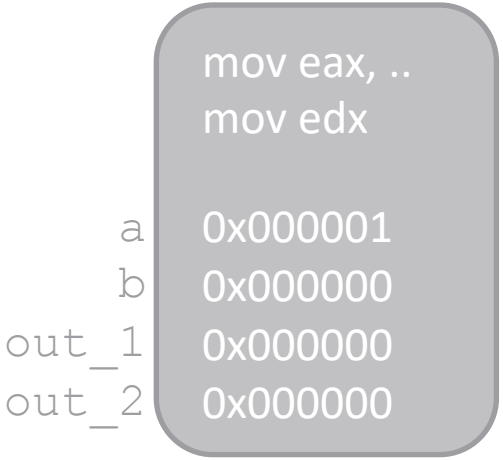
A

D

PC

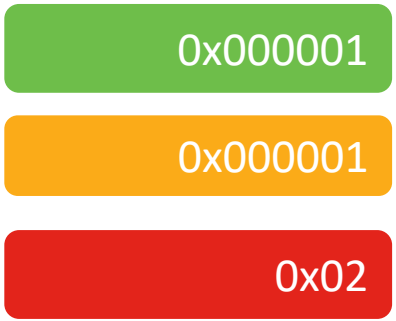
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

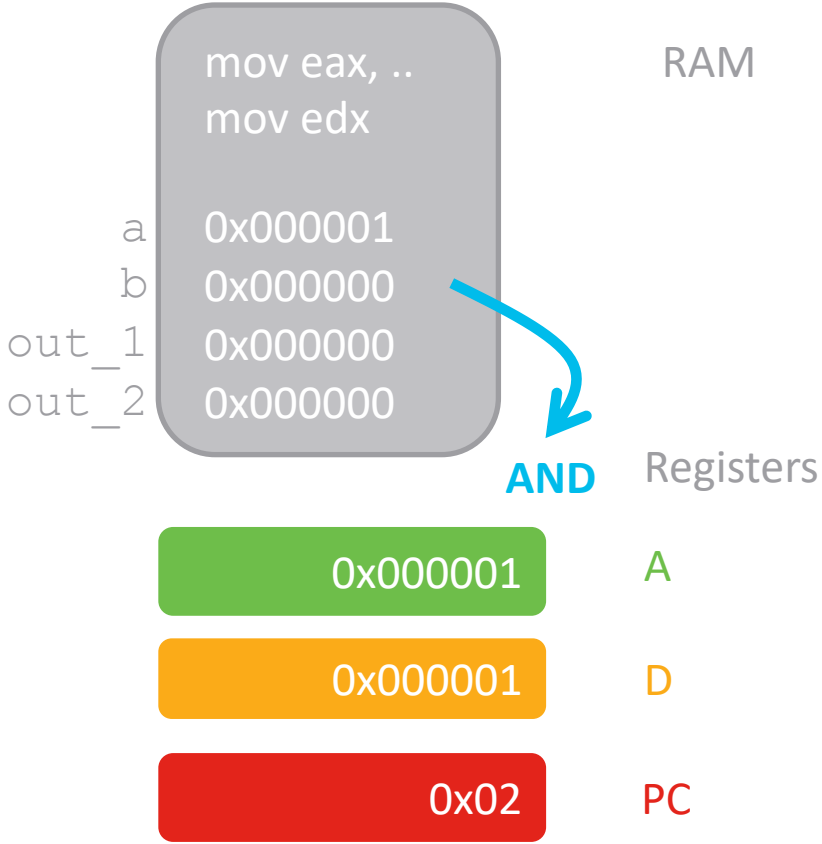


RAM

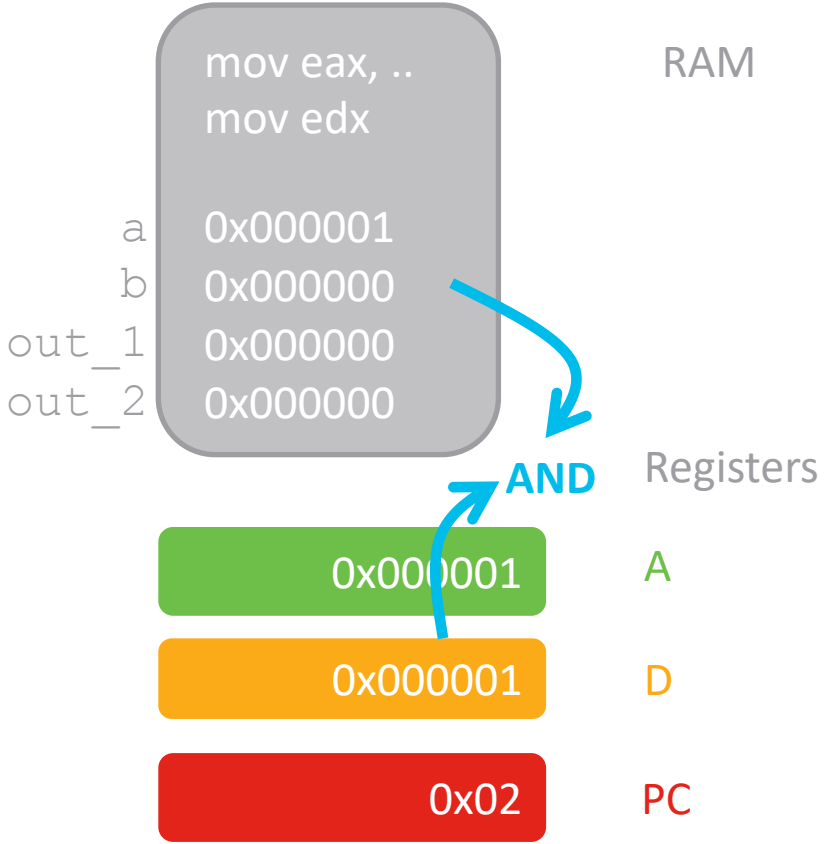
Registers



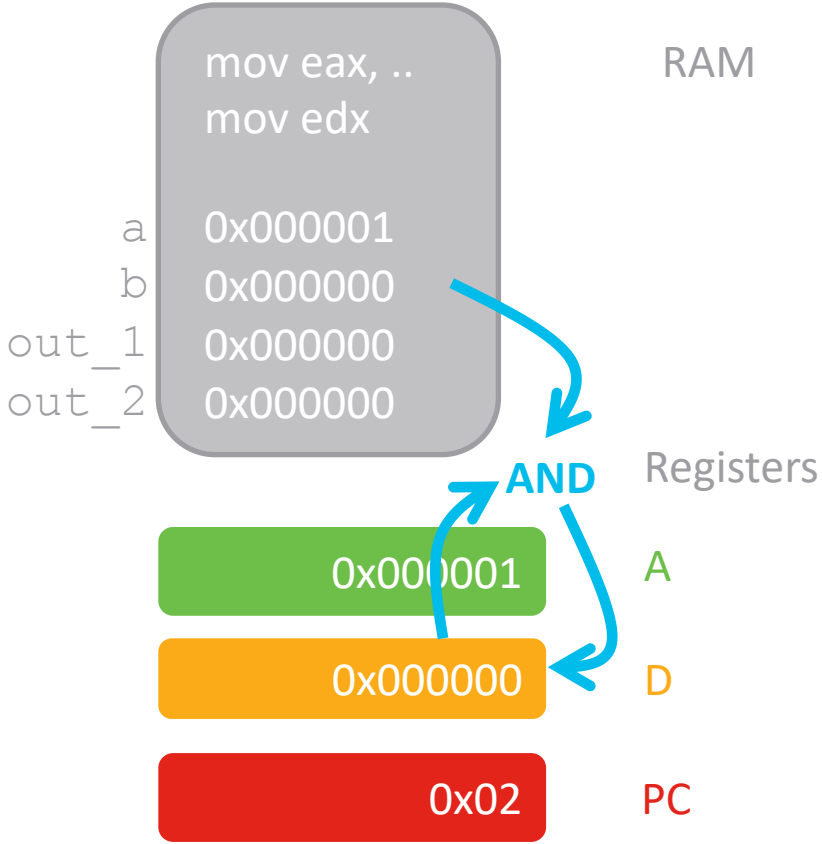
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



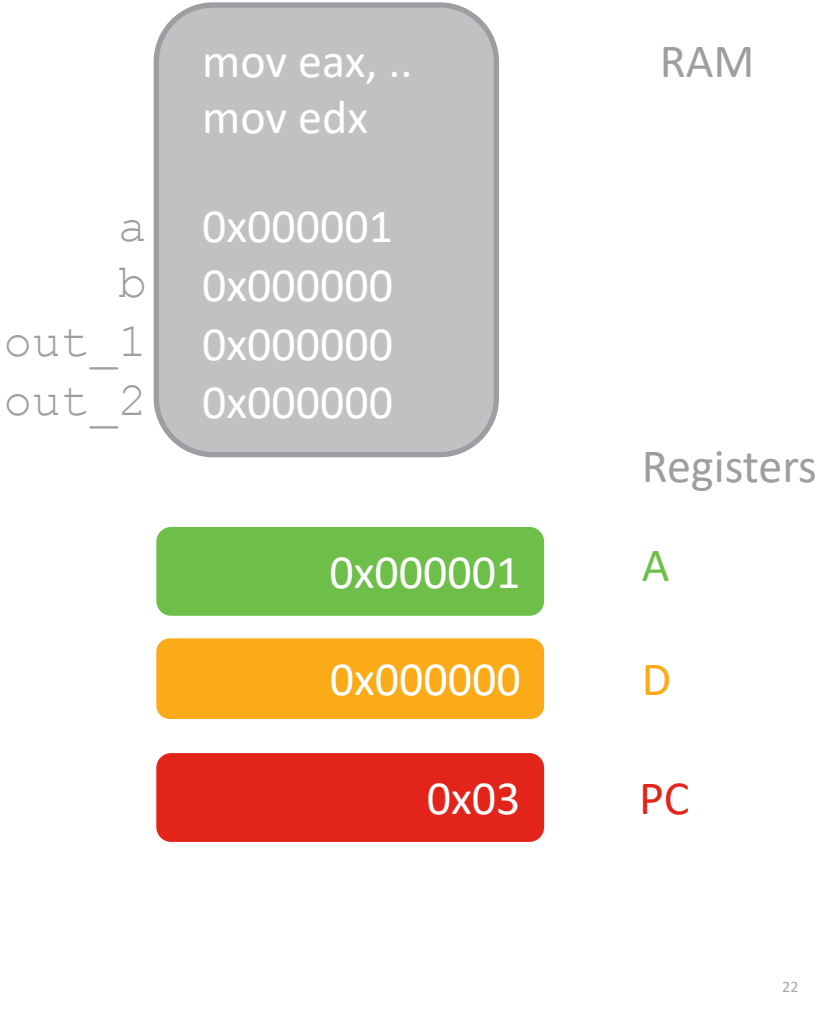
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



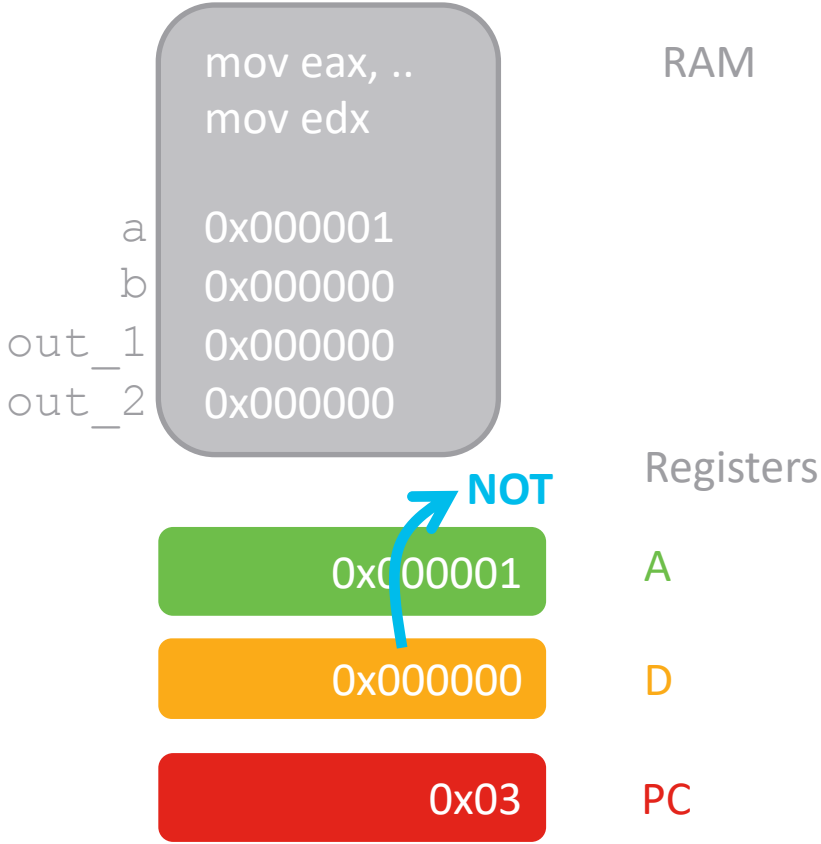
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



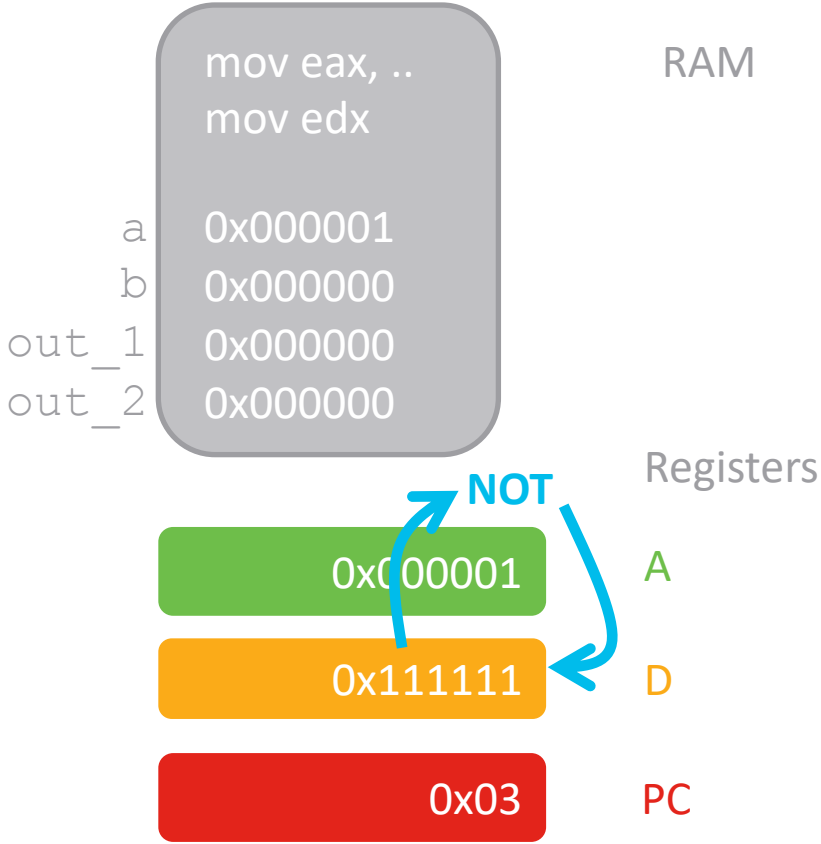
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



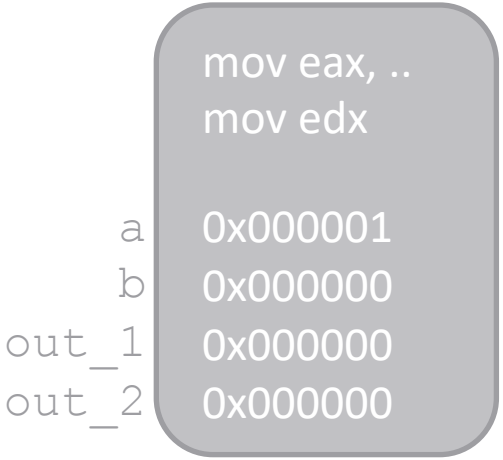
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```

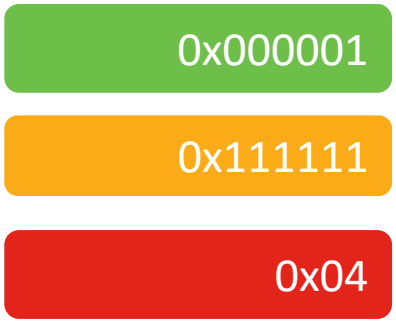


```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



RAM

Registers

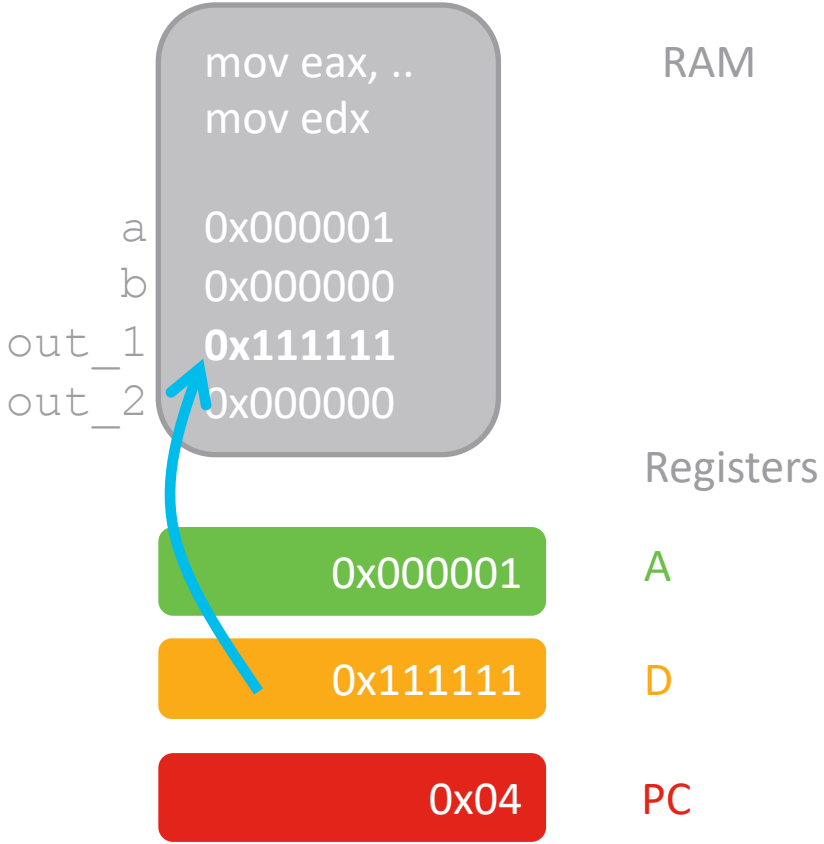


A

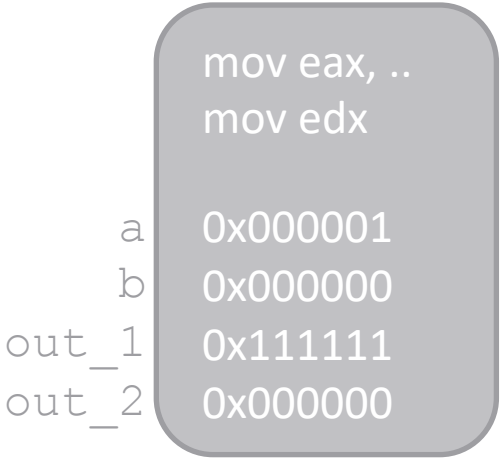
D

PC

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



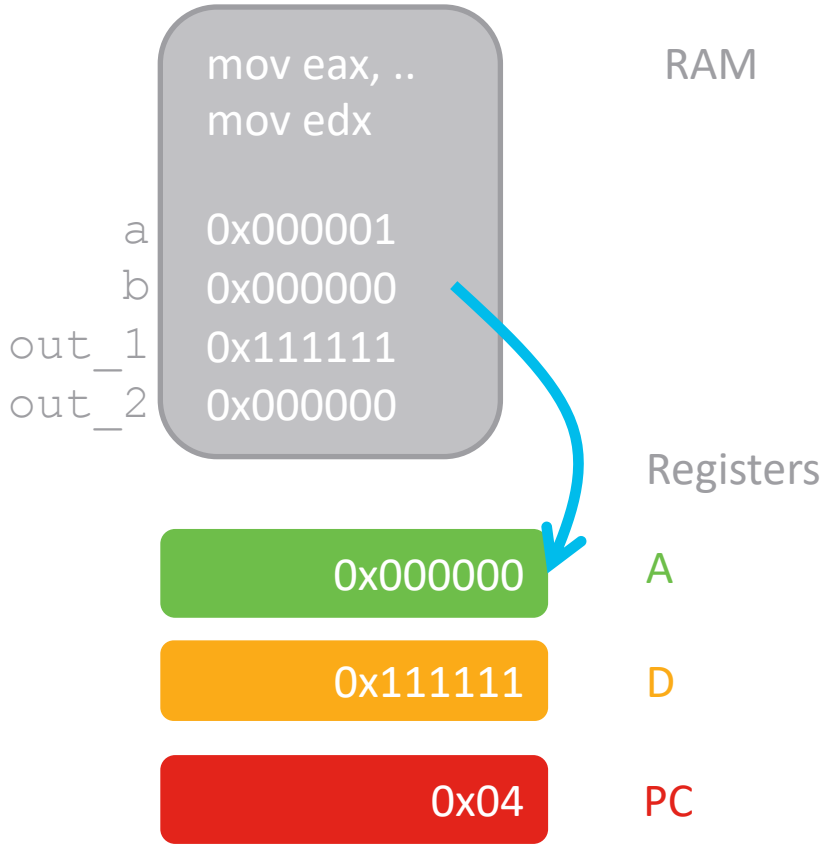
```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



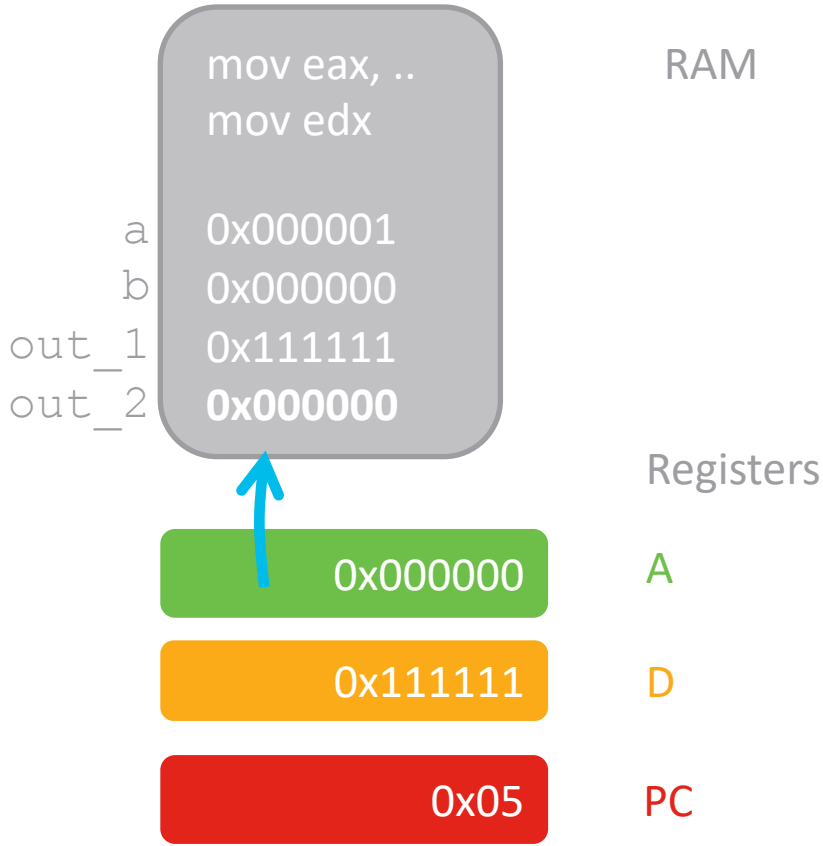
RAM

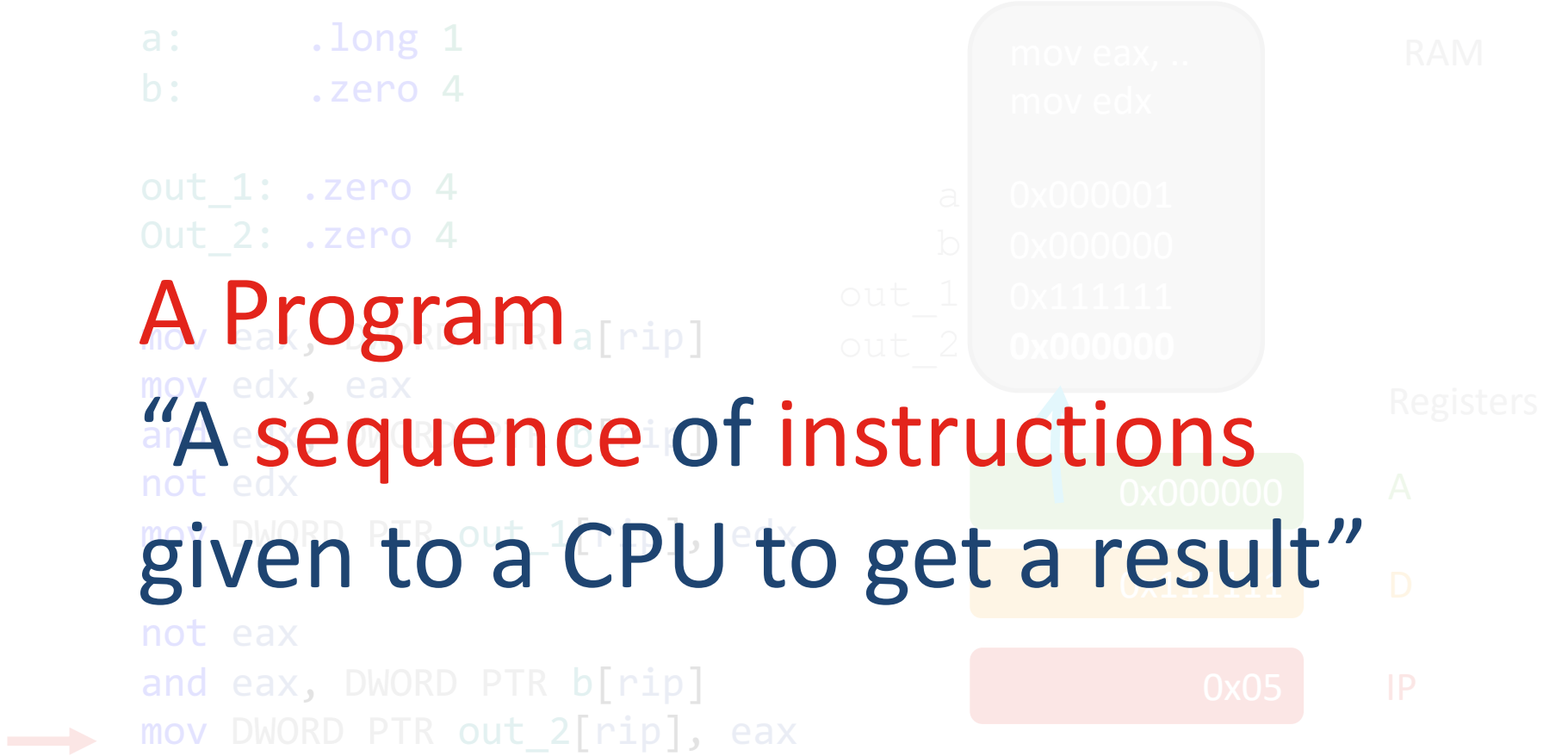
Registers

```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```



```
a:      .long 1
b:      .zero 4

out_1: .zero 4
Out_2: .zero 4

mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip], edx

not eax
and eax, DWORD PTR b[rip]
mov DWORD PTR out_2[rip], eax
```





A Program

"A sequence of instructions given to a CPU to get a result"

```
int a = 1;
int b = 0;

int out_1;
int out_2;

out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
int a = 1;
int b = 0;

int out_1;
int out_2;

out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
input a, b, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1,
wire out2;

assign out1 = ~(a & b);
assign out2 = out1 & b;

always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

```
int a = 1;
int b = 0;

int out_1;
int out_2;
```

Some declarations

```
out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
input a, b, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1,
wire out2;

assign out1 = ~(a & b);
assign out2 = out1 & b;

always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

```
int a = 1;
int b = 0;

int out_1;
int out_2;
```

Some declarations

```
input a, b, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1,
wire out2;
```

```
out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
assign out1 = ~(a & b);
assign out2 = out1 & b;

always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

```
int a = 1;
int b = 0;

int out_1;
int out_2;
```

Do the calculation

```
out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
input a, b, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1,
wire out2;

assign out1 = ~(a & b);
assign out2 = out1 & b;

always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

```
int a = 1;
int b = 0;

int out_1;
int out_2;
```

Do the calculation

```
out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
input a, b, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1,
wire out2;

assign out1 = ~(a & b);
assign out2 = out1 & b;

always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

```
int a = 1;
int b = 0;

int out_1;
int out_2;

out_1 = ~(a & b);
out_2 = out_1 & b;
```

```
input a, b, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1,
wire out2;

assign out1 = ~(a & b);
assign out2 = out1 & b;
```

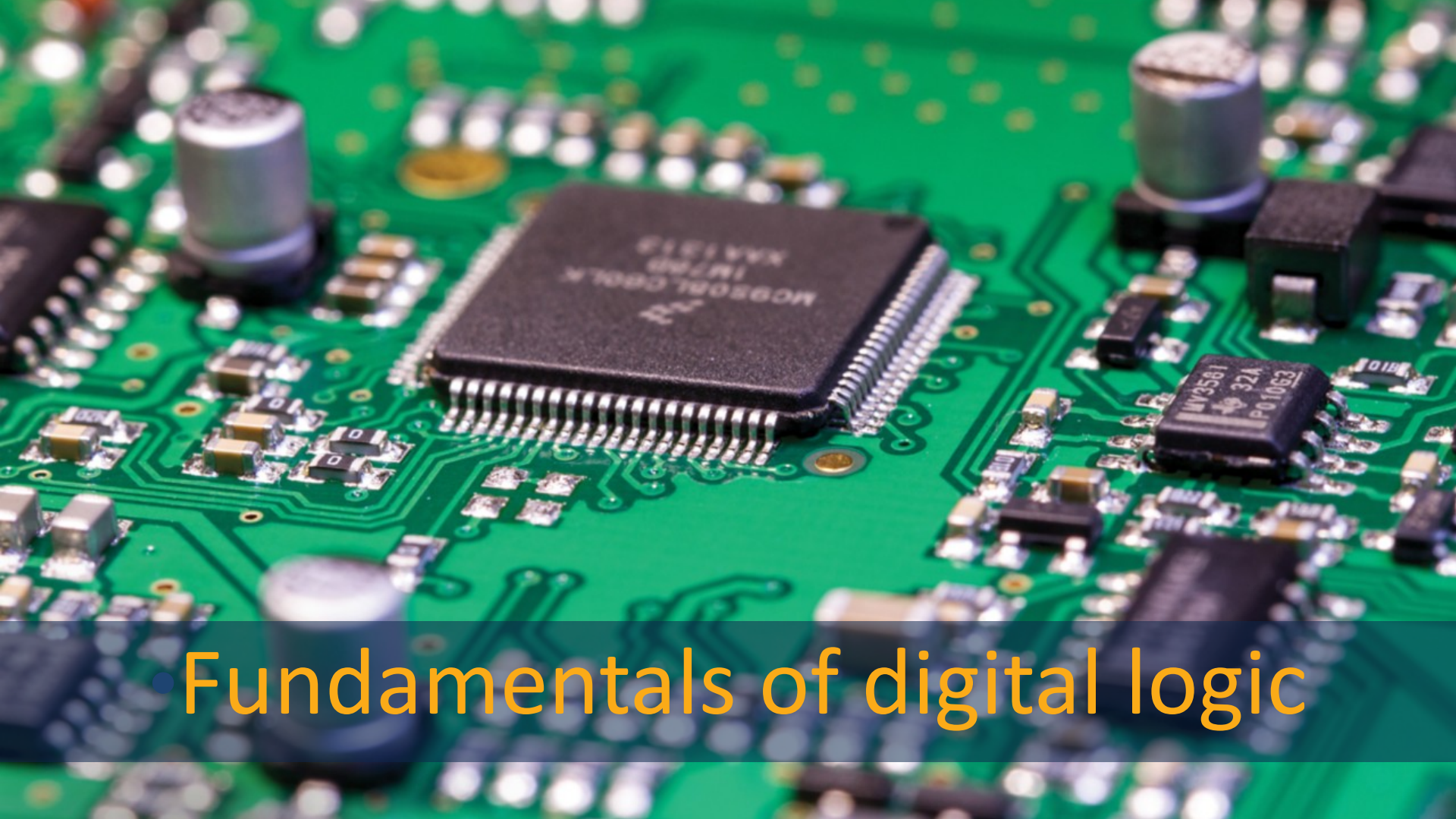
?

Hmm....

```
always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

Lifting the lid on FPGAs

oto courtesy Claudio Valdes



Fundamentals of digital logic

Bitwise NOT (~)

Bitwise NOT (~)

A	Out
0	1
1	0

Truth table

Bitwise NOT (~)

A	Out
0	1
1	0

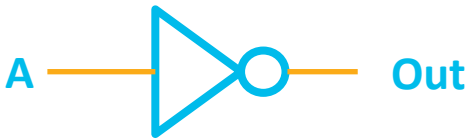
Circuit symbol

Bitwise NOT (~)

A	Out
0	1
1	0

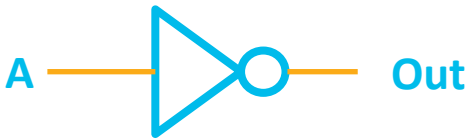
Bitwise AND (&)

A	B	Out
0	0	0
1	0	0
0	1	0
1	1	1



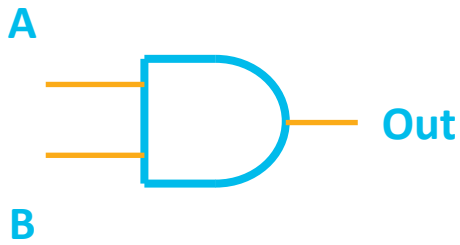
Bitwise NOT (~)

A	Out
0	1
1	0



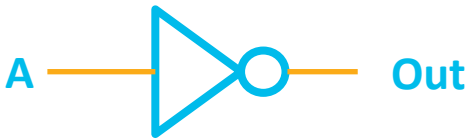
Bitwise AND (&)

A	B	Out
0	0	0
1	0	0
0	1	0
1	1	1



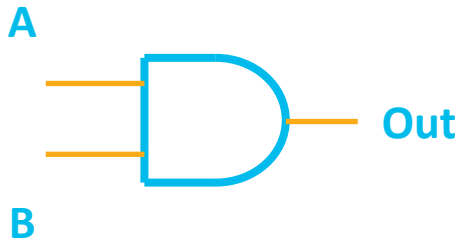
Bitwise NOT (~)

A	Out
0	1
1	0



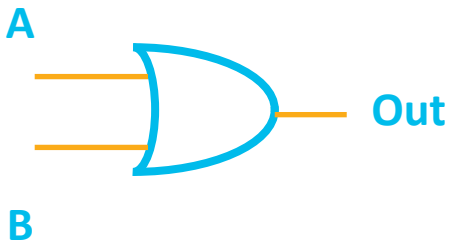
Bitwise AND (&)

A	B	Out
0	0	0
1	0	0
0	1	0
1	1	1



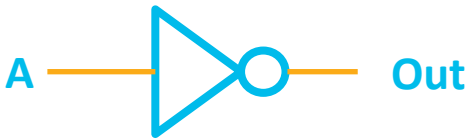
Bitwise OR (|)

A	B	Out
0	0	0
1	0	1
0	1	1
1	1	1



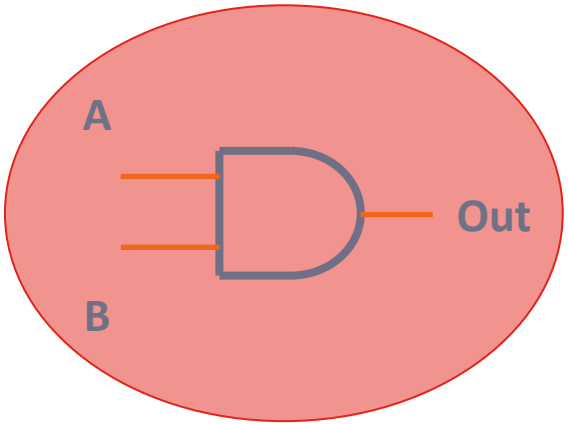
Bitwise NOT (~)

A	Out
0	1
1	0



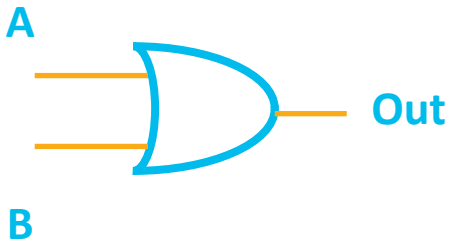
Bitwise AND (&)

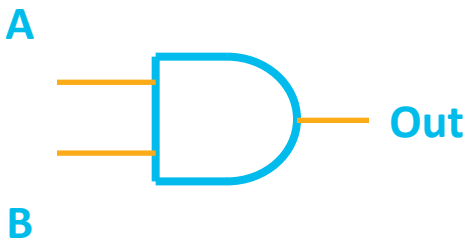
A	B	Out
0	0	0
1	0	0
0	1	0
1	1	1

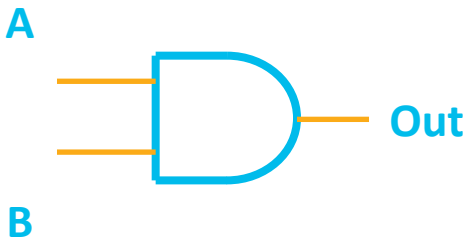


Bitwise OR (|)

A	B	Out
0	0	0
1	0	1
0	1	1
1	1	1

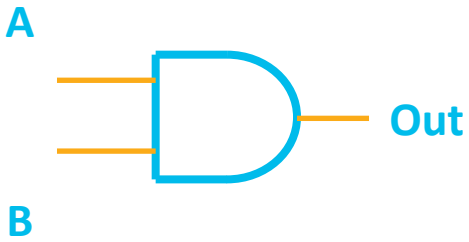






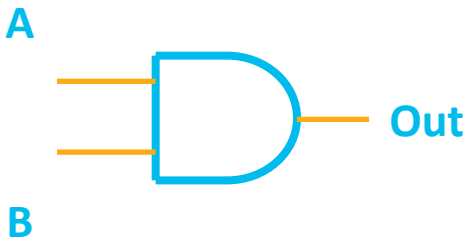
Truth Table

A	B	Out
0	0	0
1	0	0
0	1	0
1	1	1



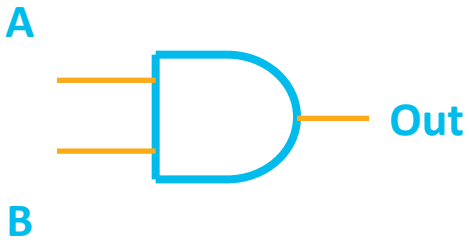
Truth Table

A	B	Out
0	0	0
1	0	0
0	1	0
1	1	1



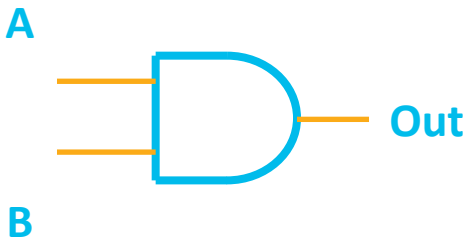
Truth Table

A	B	Out
0	0	0
1	0	0



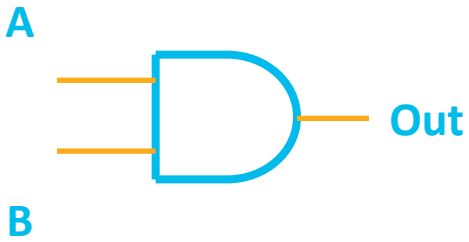
Truth Table

A	B	Out
0	0	0
1	0	0



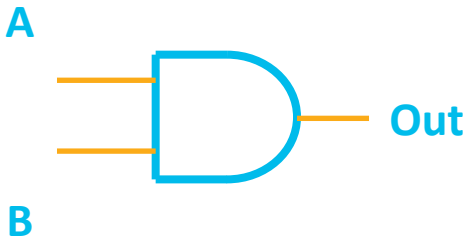
Truth Table

A	B	Out
0	0	0
1	0	0



Truth Table

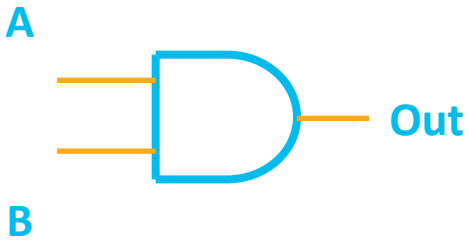
A	B	Out
x	0	0



Truth Table

A	B	Out
x	0	0

"x" could be 0 or 1

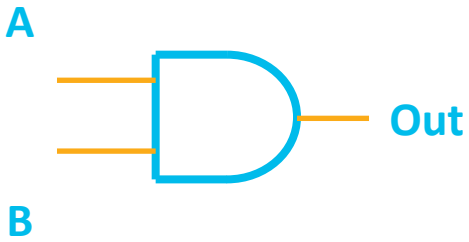


Truth Table

A	B	Out
x	0	0

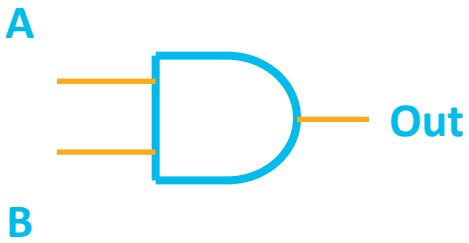
output is always 0

"x" could be 0 or 1



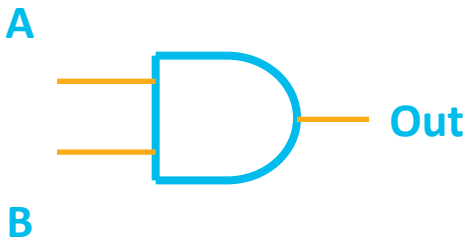
Truth Table

A	B	Out
0	x	0
0	1	0
1	1	1



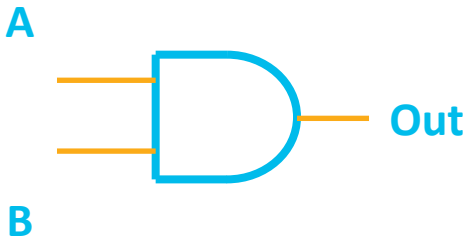
Truth Table

A	B	Out
0	x	0
0	1	0
1	1	1



Truth Table

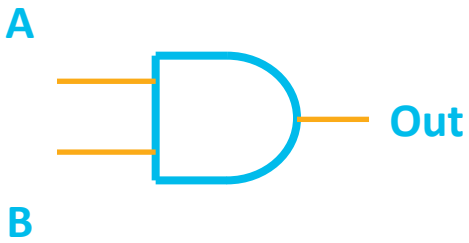
A	B	Out
0	x	0
0	1	0
1	1	1



Truth Table

A	B	Out
0	x	0
x	1	0
	1	1

"x" could be 0 or 1

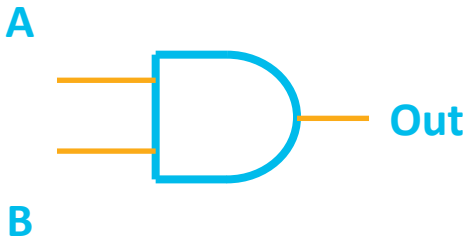


Truth Table

A	B	Out
0	x	0
x	1	x
	1	

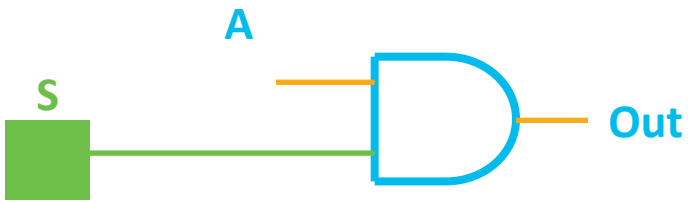
Output matches x

"x" could be 0 or 1



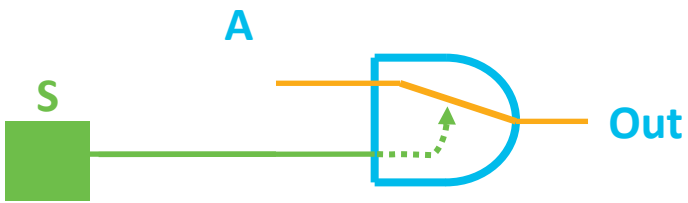
Truth Table

A	B	Out
x	0	0
x	1	x



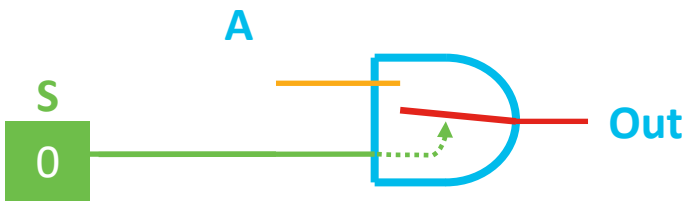
Truth Table

A	S	Out
x	0	0
x	1	x



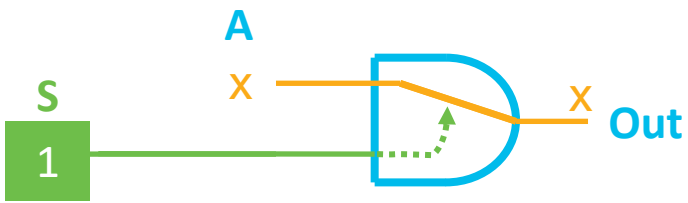
Truth Table

A	S	Out
x	0	0
x	1	x



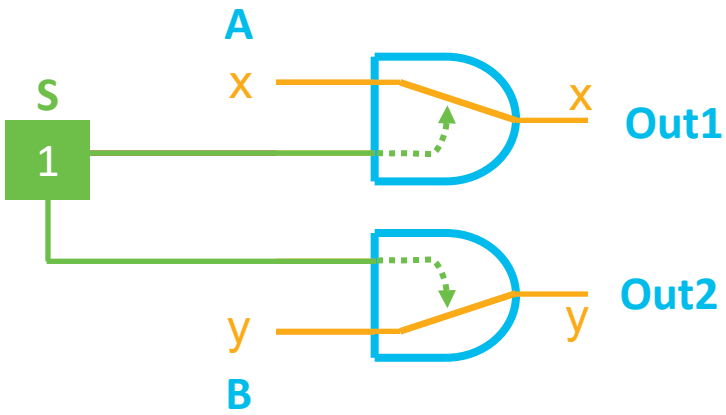
Truth Table

A	S	Out
x	0	0
x	1	x



Truth Table

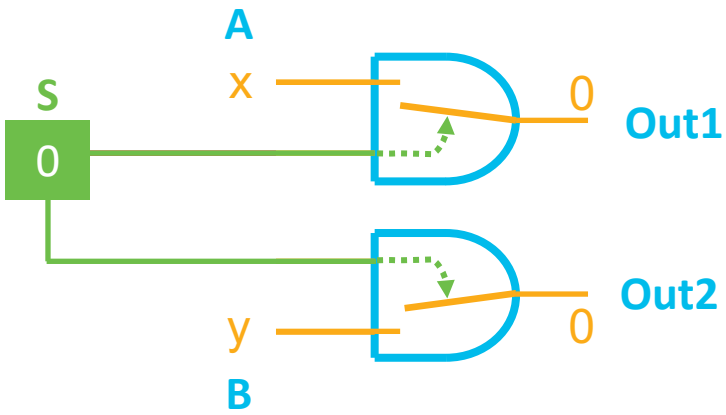
A	S	Out
x	0	0
x	1	x



Truth Tables

A	S	Out1
x	0	0
x	1	x

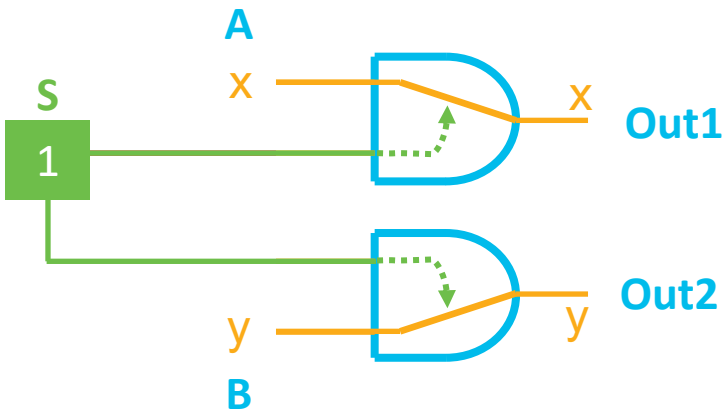
B	S	Out2
y	0	0
y	1	y



Truth Tables

A	S	Out1
x	0	0
x	1	x

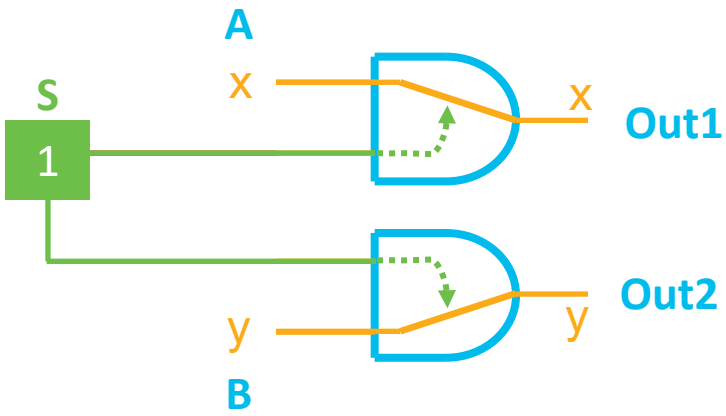
B	S	Out2
y	0	0
y	1	y



Truth Tables

A	S	Out1
x	0	0
x	1	x

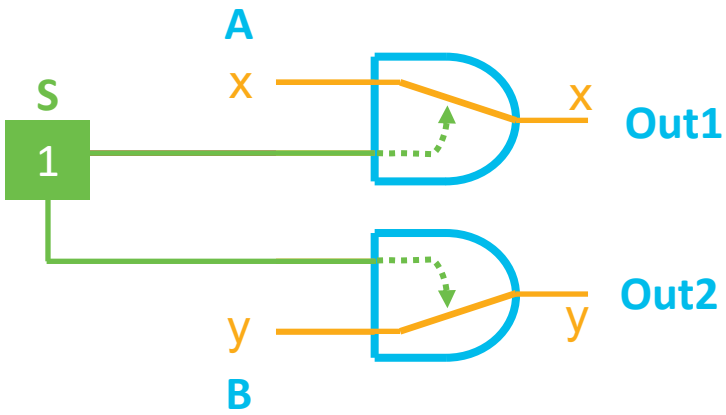
B	S	Out2
y	0	0
y	1	y



Truth Tables

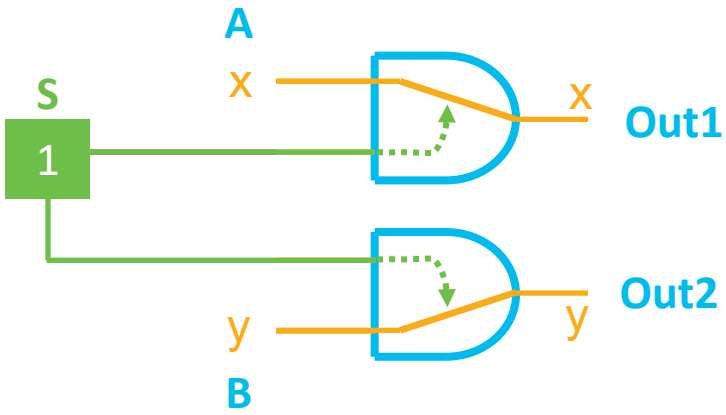
A	S	Out1
x	0	0
x	1	x

B	S	Out2
y	0	0
y	1	y



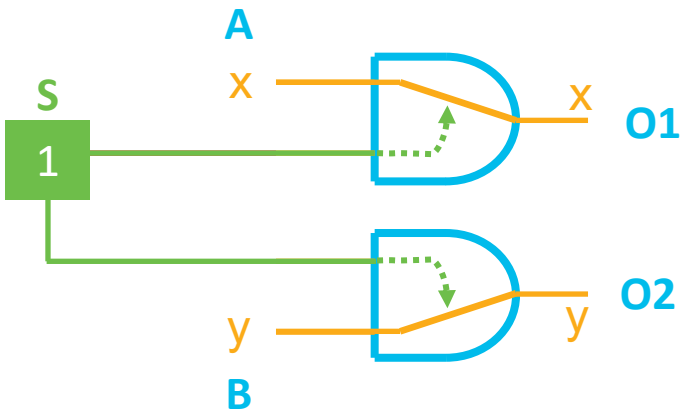
Truth Tables

S	A	Out1	B	Out2
0	x	0	y	0
1	x	x	y	y



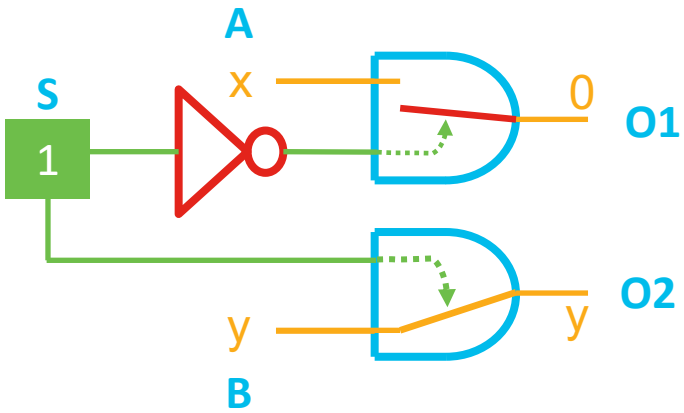
Truth Tables

S	A	B	Out1	Out2
0	x	y	0	0
1	x	y	x	y



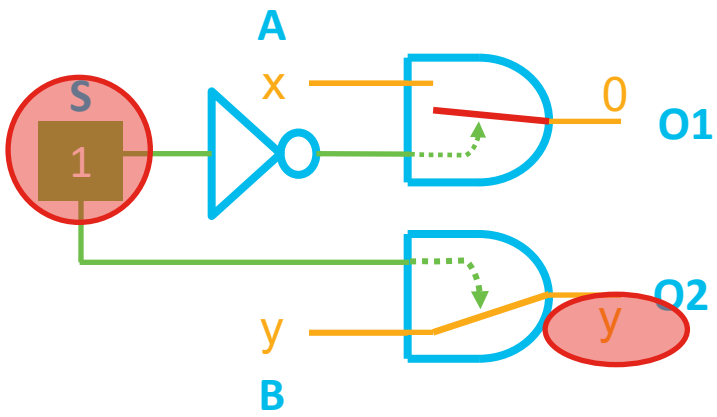
Truth Table

S	A	B	O1	O2
0	x	y	0	0
1	x	y	x	y



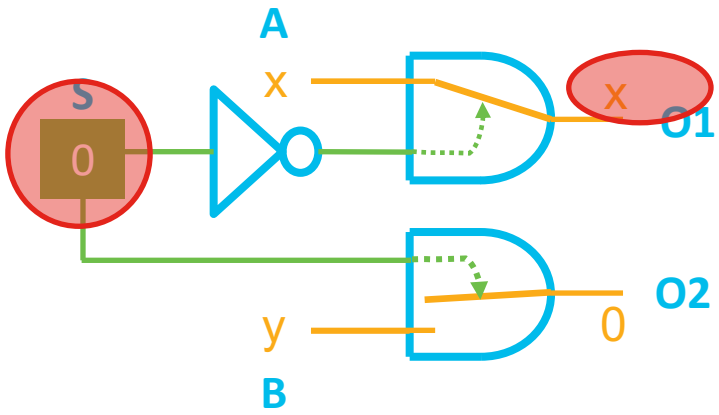
Truth Table

S	A	B	O1	O2
0	x	y	x	0
1	x	y	0	y



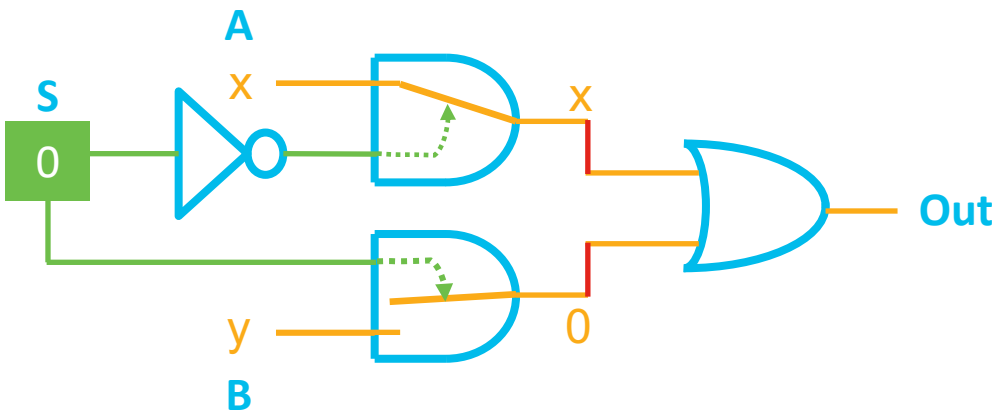
Truth Table

S	A	B	O1	O2
0	x	y	x	0
1	x	y	0	y



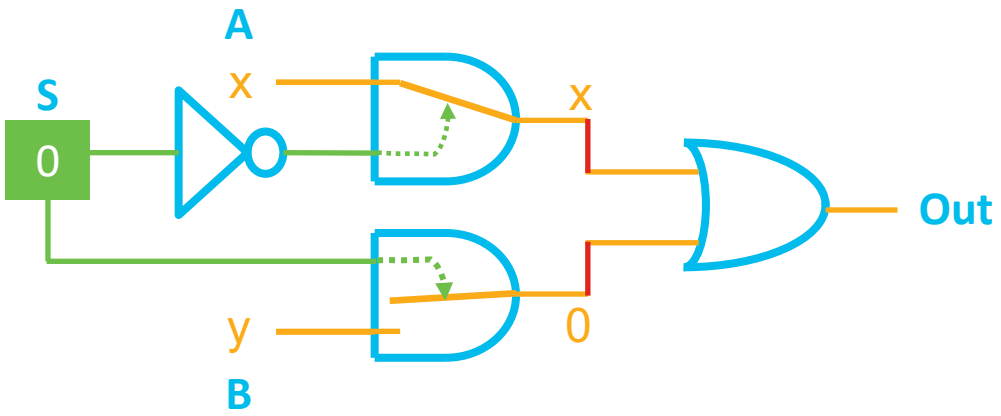
Truth Table

S	A	B	O1	O2
0	x	y	x	0
1	x	y	0	y



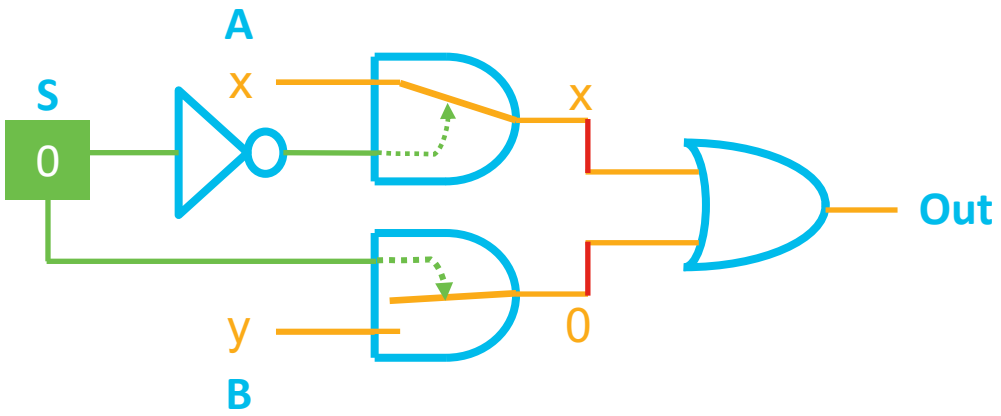
Truth Table

S	A	B	Out
0	x	y	x
1	x	y	y



Truth Table

S	A	B	Out
0	x	y	x
1	x	y	y

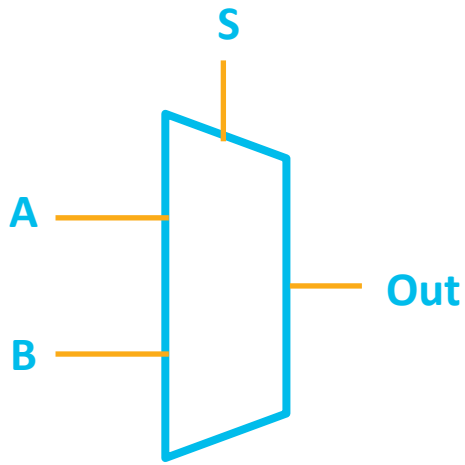


Truth Table

S	A	B	Out
0	x	y	x
1	x	y	y

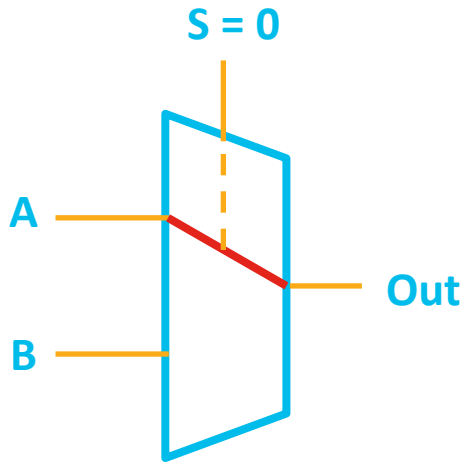

0
X


1
Y



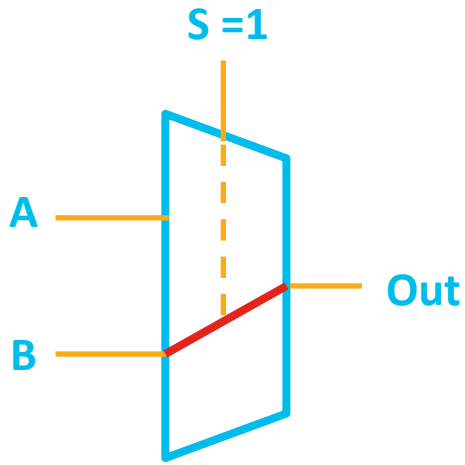
Truth Table

S	A	B	Out



Truth Table

S	A	B	Out
0	x	y	x



Truth Table

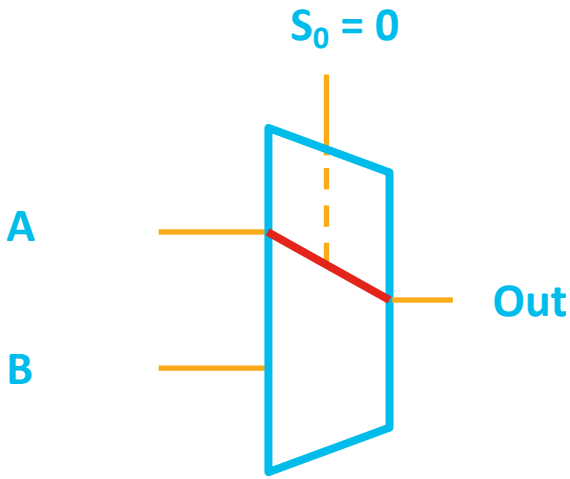
S	A	B	Out
0	x	y	x
1	x	y	y

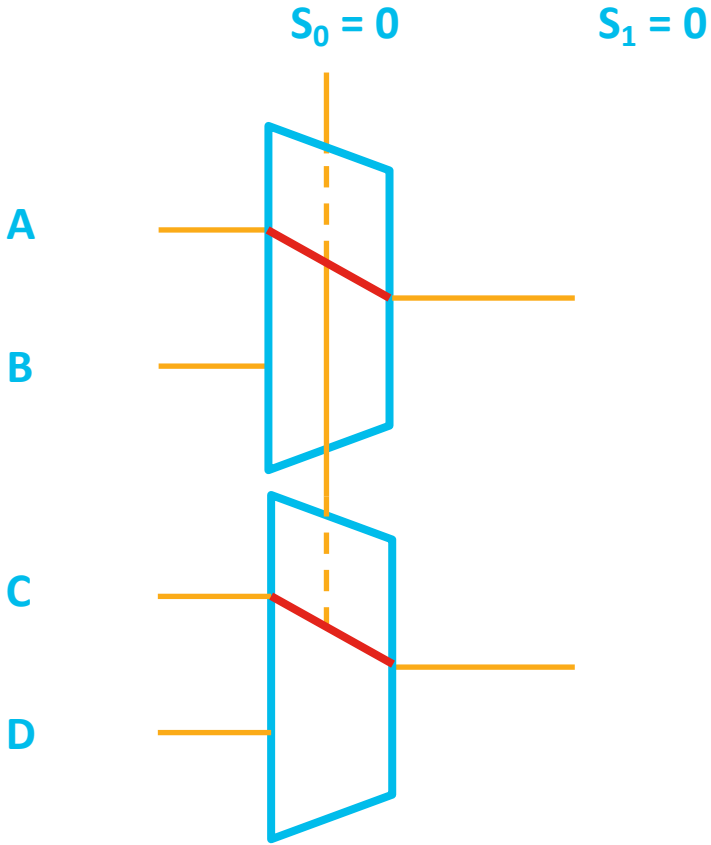
Multiplexer (MUX):

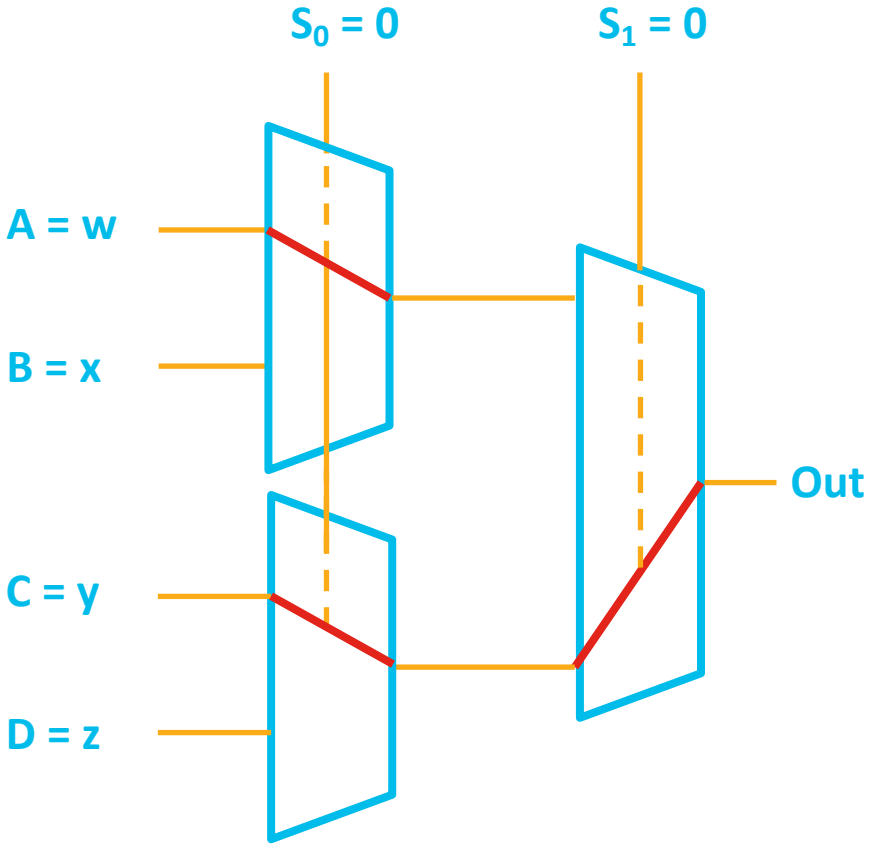
"Different inputs are selected(or routed) to a common output based on a selector"

S	A	B	Out
0	x	y	x
1	x	y	y

- Scaling up!

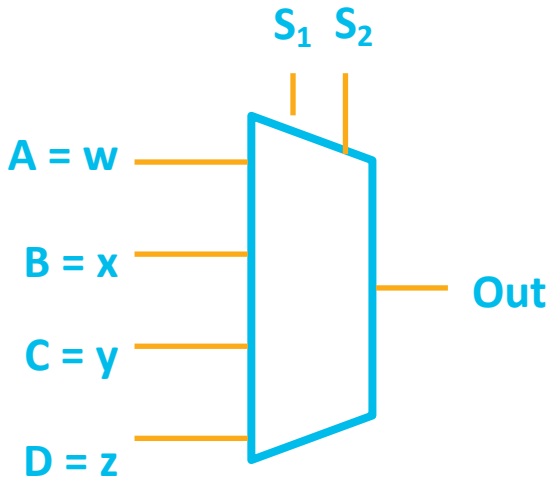


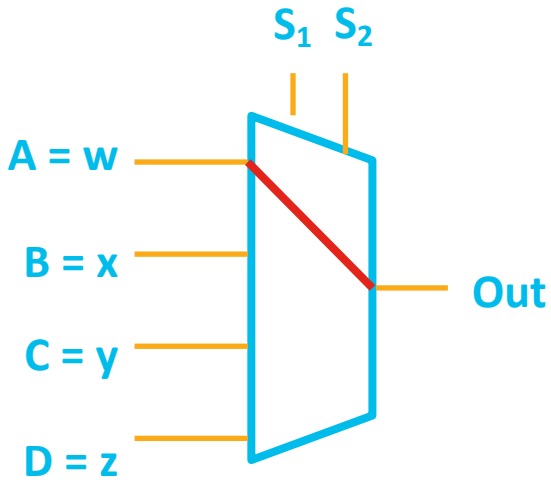




Truth Table

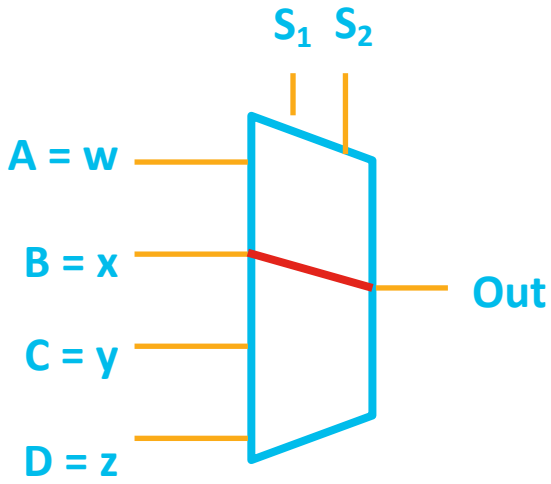
S_0	S_1	Out
0	0	w
1	0	x
1	1	z
0	1	y





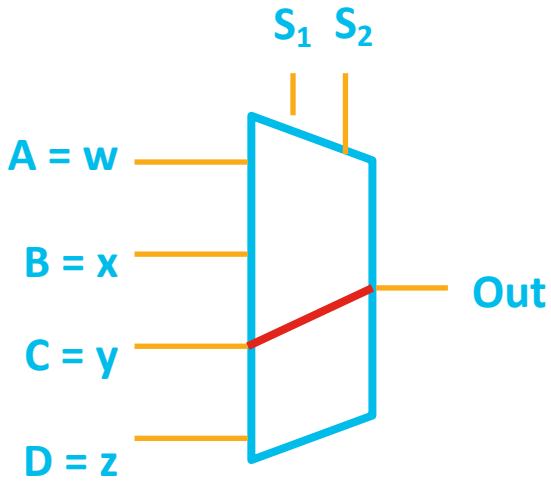
Truth Table

S_1	S_2	Out
0	0	w



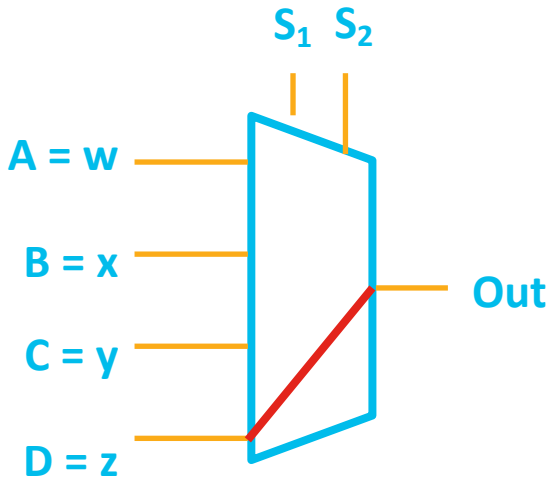
Truth Table

S_1	S_2	Out
0	0	w
1	0	x



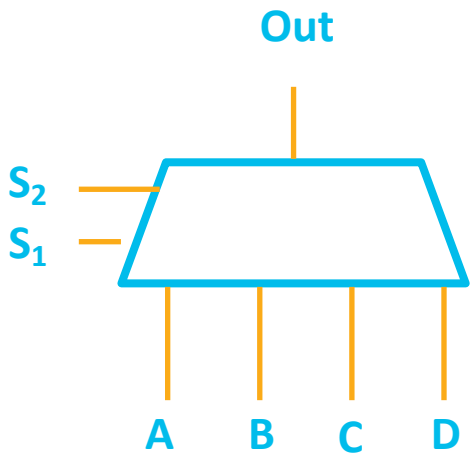
Truth Table

S_1	S_2	Out
0	0	w
1	0	x
0	1	y



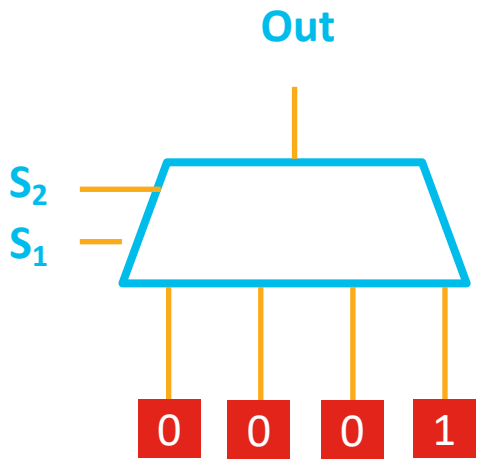
Truth Table

S_1	S_2	Out
0	0	w
1	0	x
0	1	y
1	1	z



Truth Table

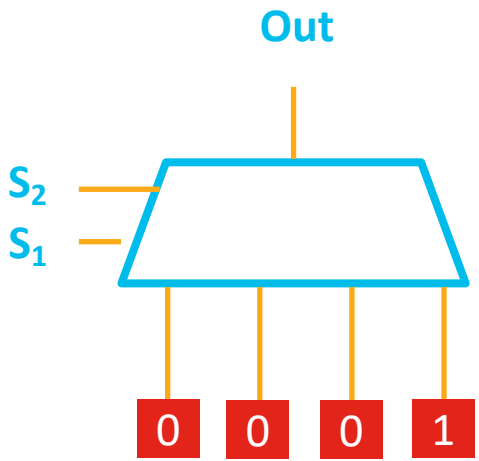
S_1	S_2	Out
0	0	w
1	0	x
0	1	y
1	1	z



1 bit of memory

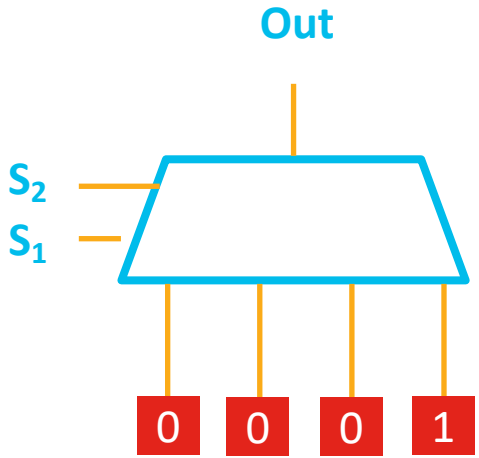
Truth Table

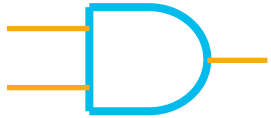
S_1	S_2	Out
0	0	w
1	0	x
0	1	y
1	1	z



Truth Table

S_1	S_2	Out
0	0	0
1	0	0
0	1	0
1	1	1

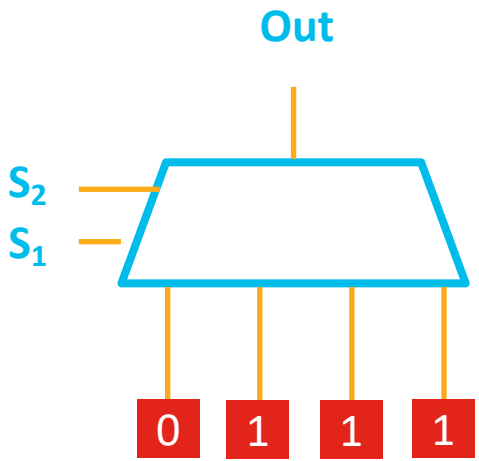




Truth Table

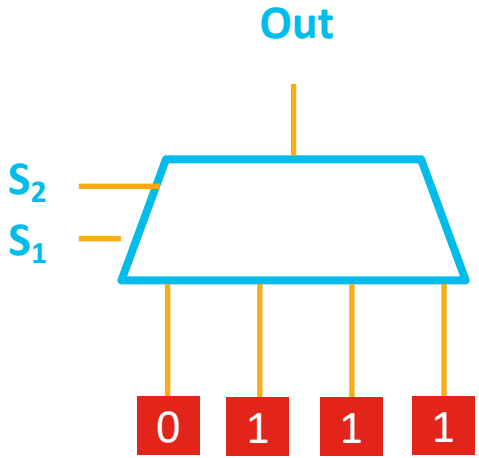
S_1	S_2	Out
0	0	0
1	0	0
0	1	0
1	1	1

AND Gate



Truth Table

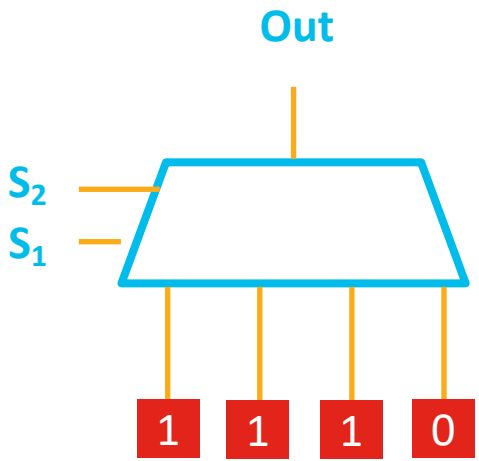
S_1	S_2	Out
0	0	0
1	0	1
0	1	1
1	1	1



Truth Table

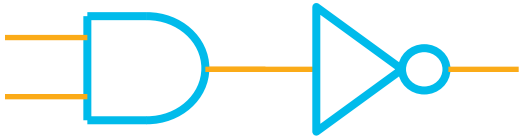
S_1	S_2	Out
0	0	0
1	0	1
0	1	1
1	1	1

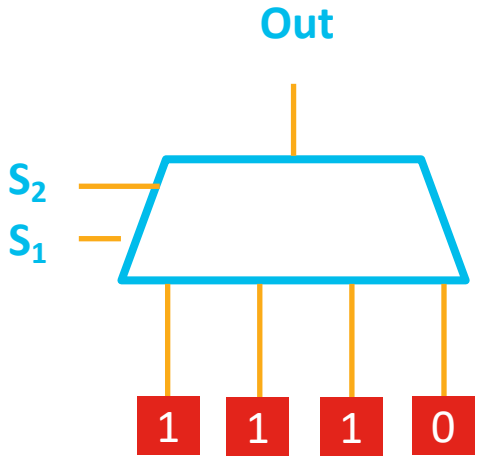
OR Gate



Truth Table

S_1	S_2	Out
0	0	1
1	0	1
0	1	1
1	1	0

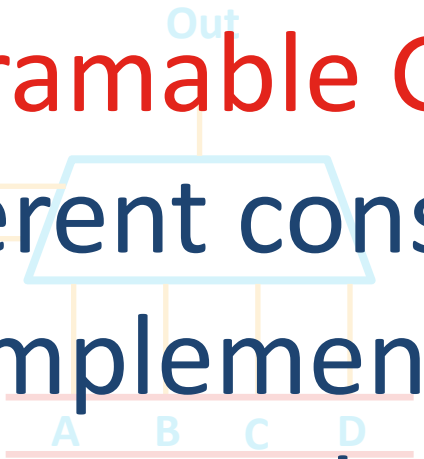




Truth Table

S_1	S_2	Out
0	0	1
1	0	1
0	1	1
1	1	0

NOT AND (NAND)

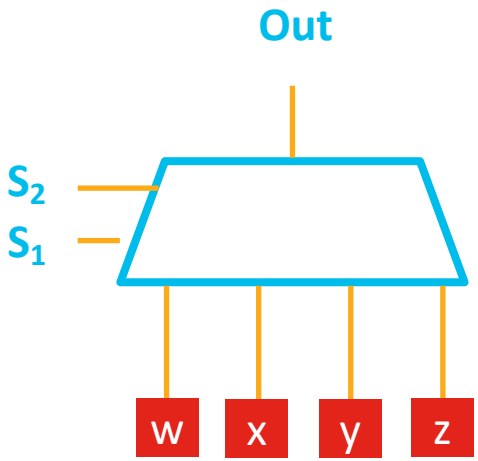


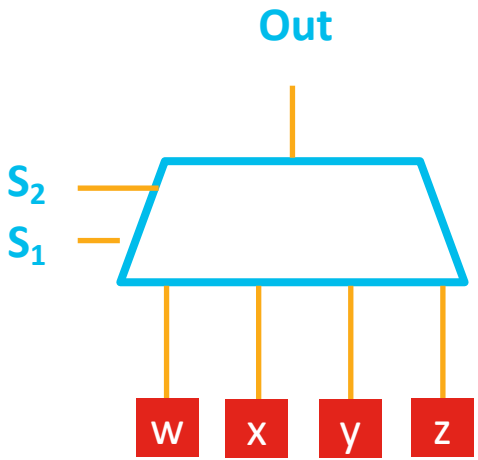
Truth Table

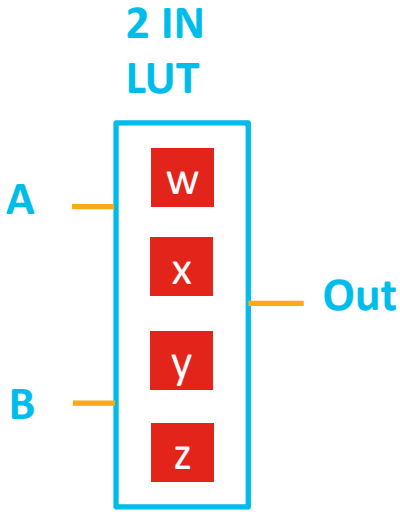
S_1	S_2	Out
0	0	w
1	0	x
0	1	y
1	1	z

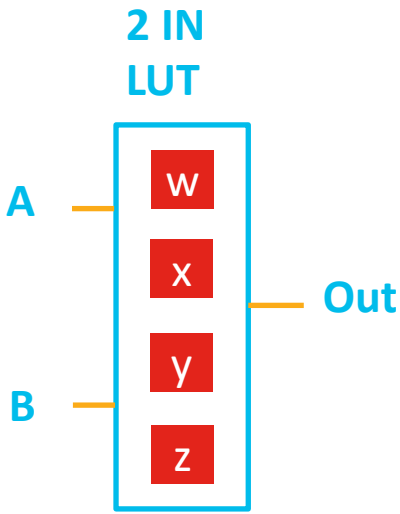
Programable Gate:

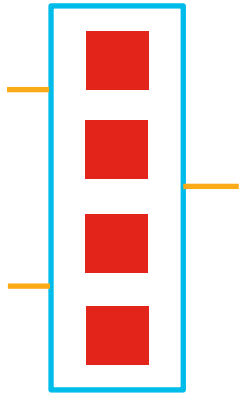
“different constants can implement any logical function with 2 inputs”

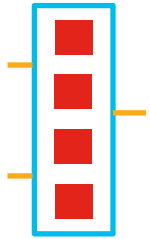


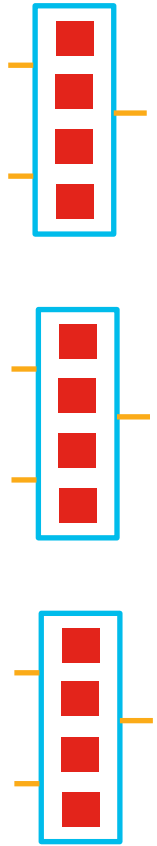


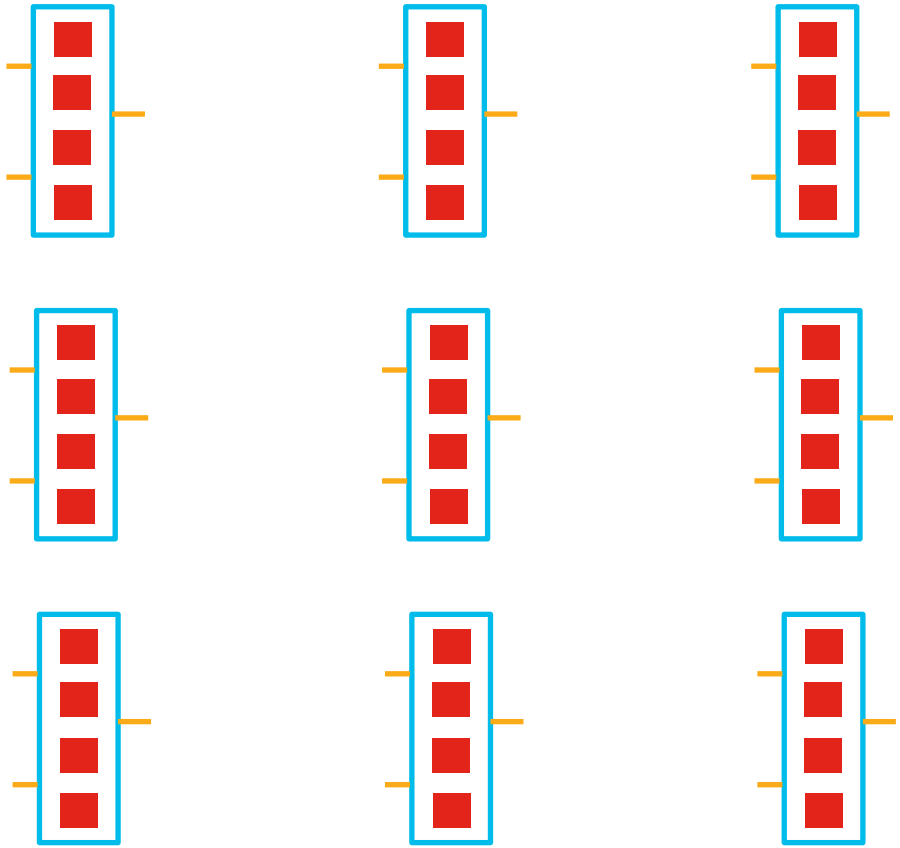


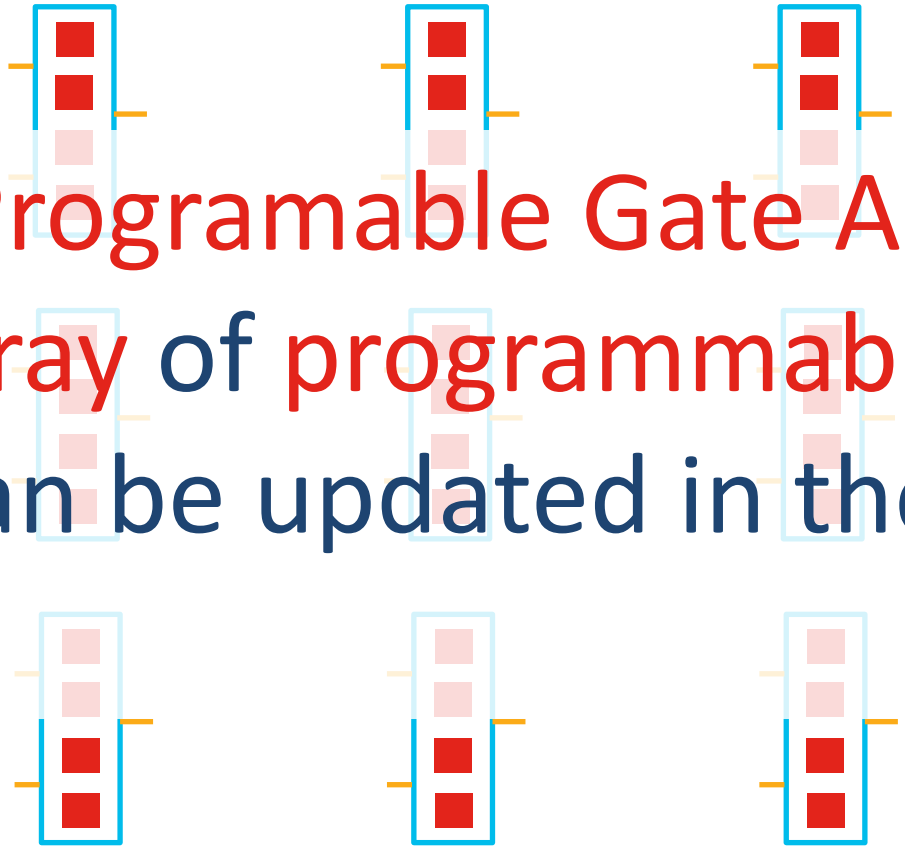






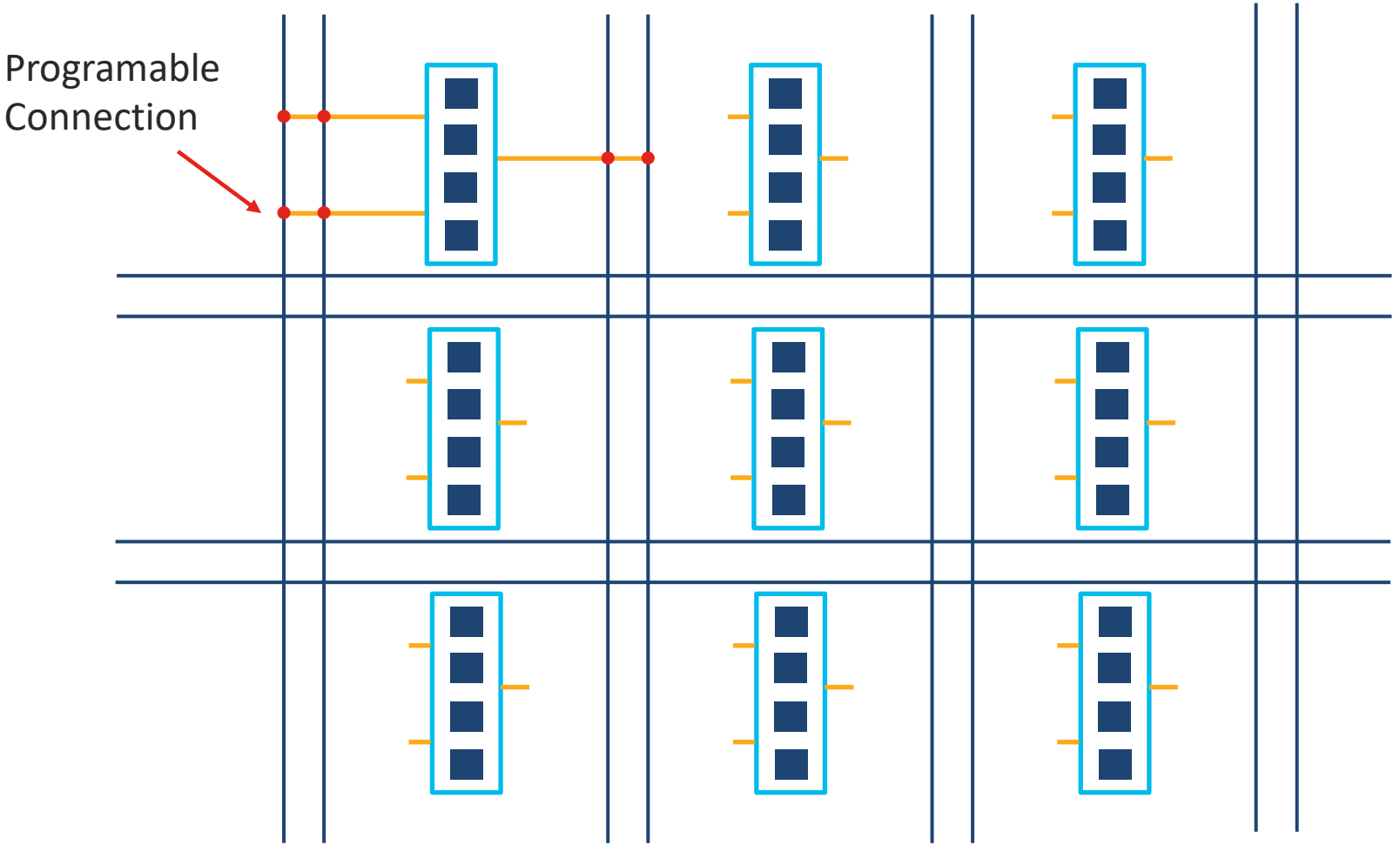


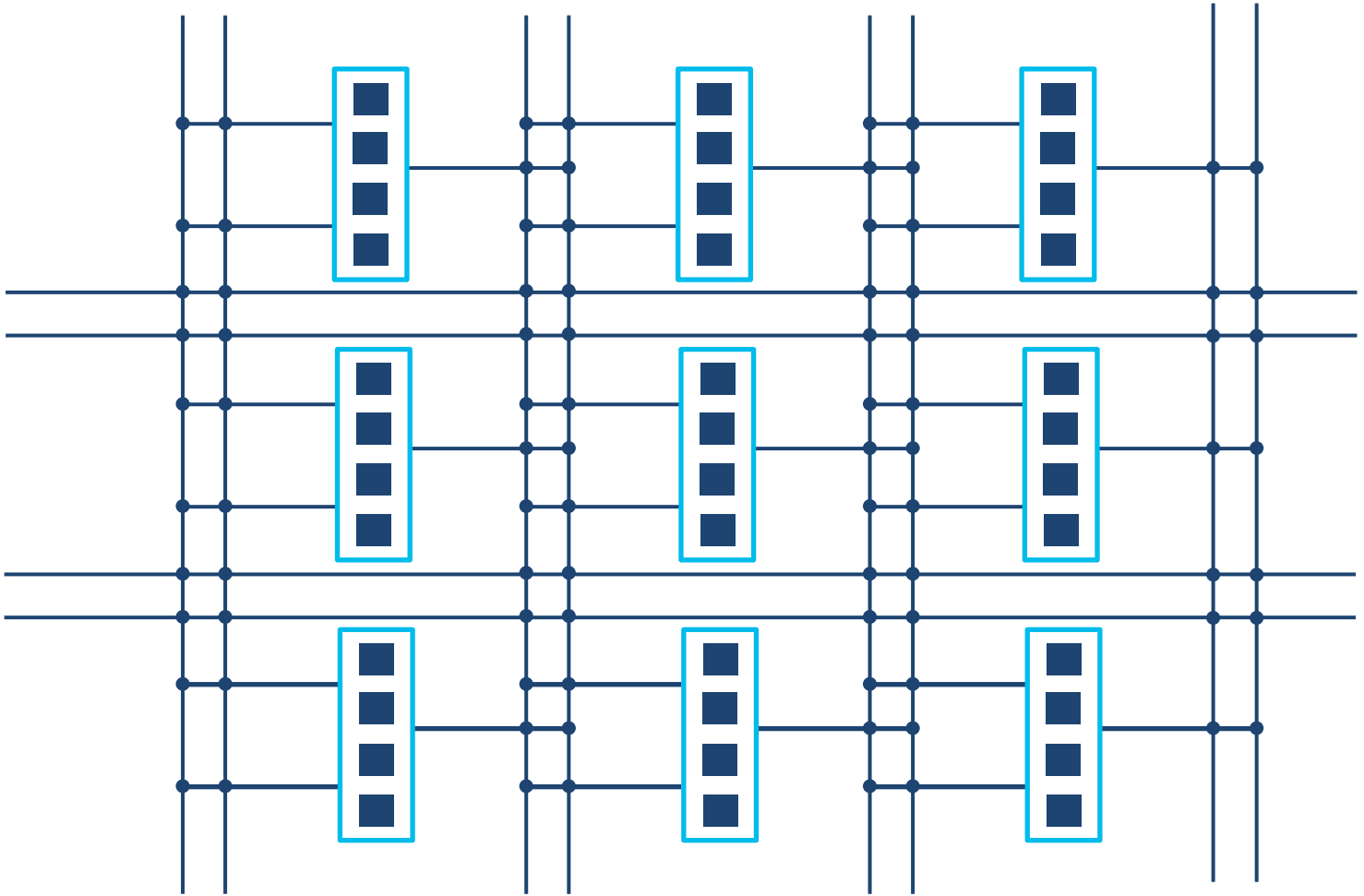


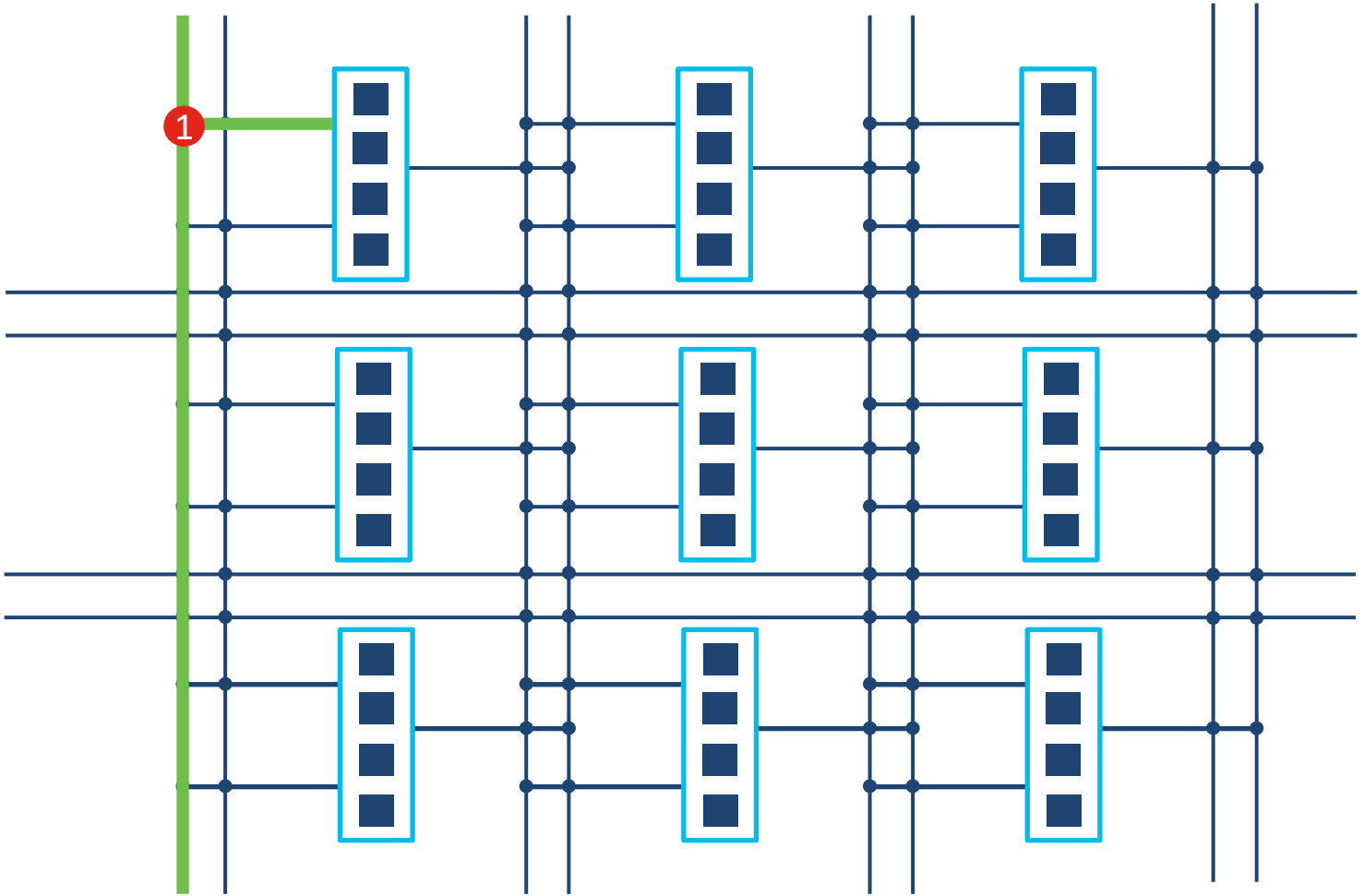


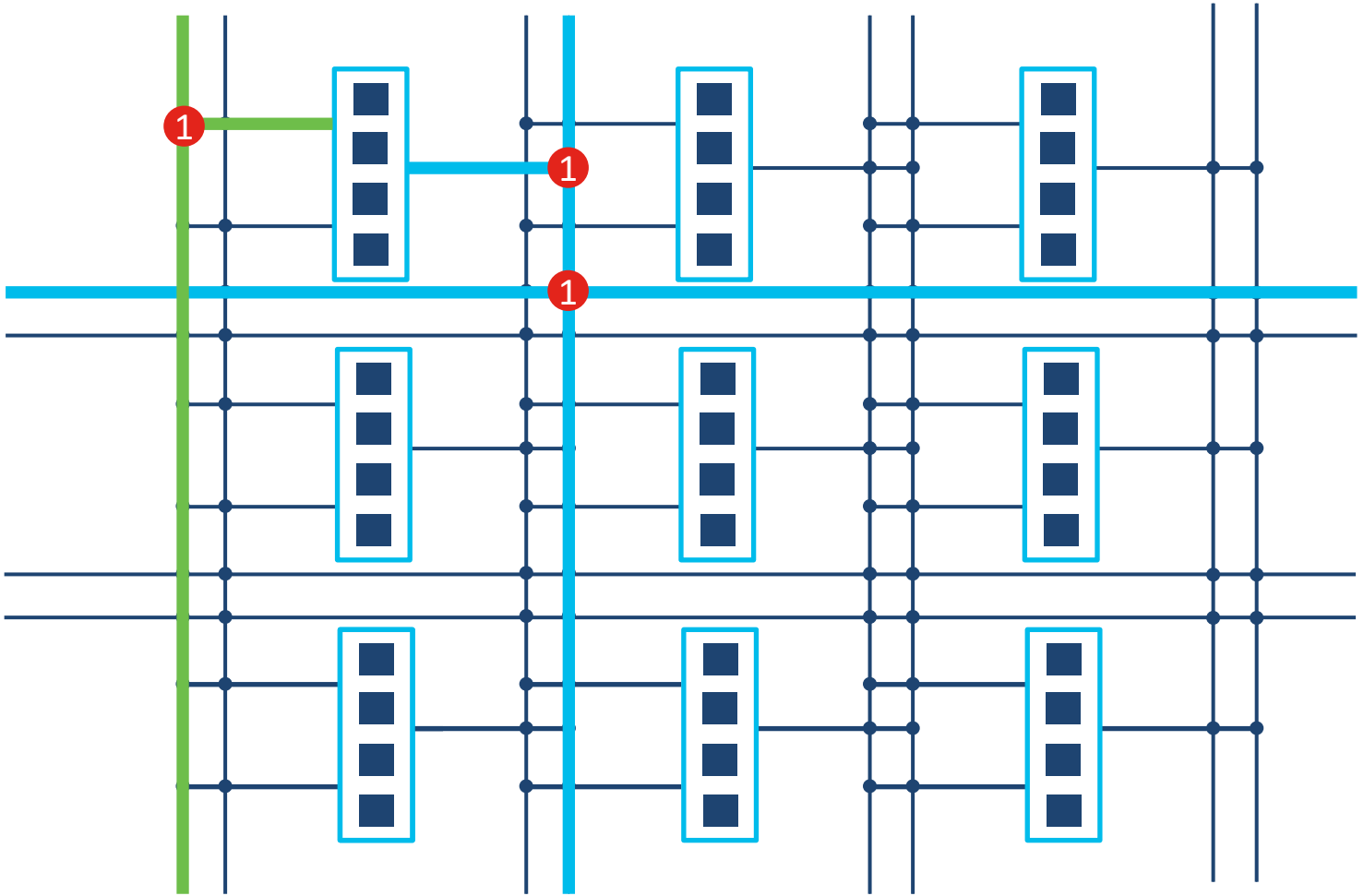
Field Programable Gate Array:

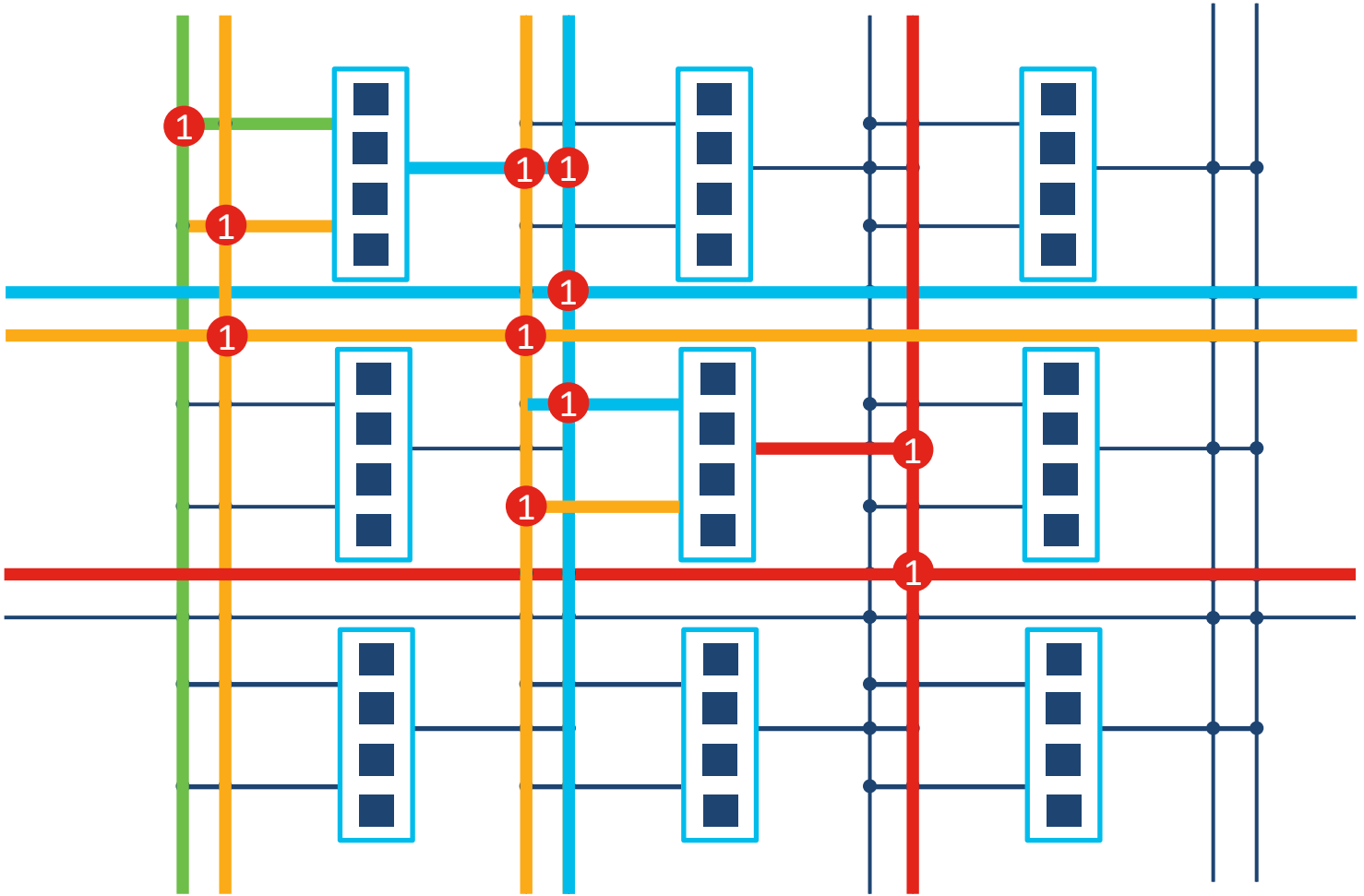
"An array of programmable gates that can be updated in the field"

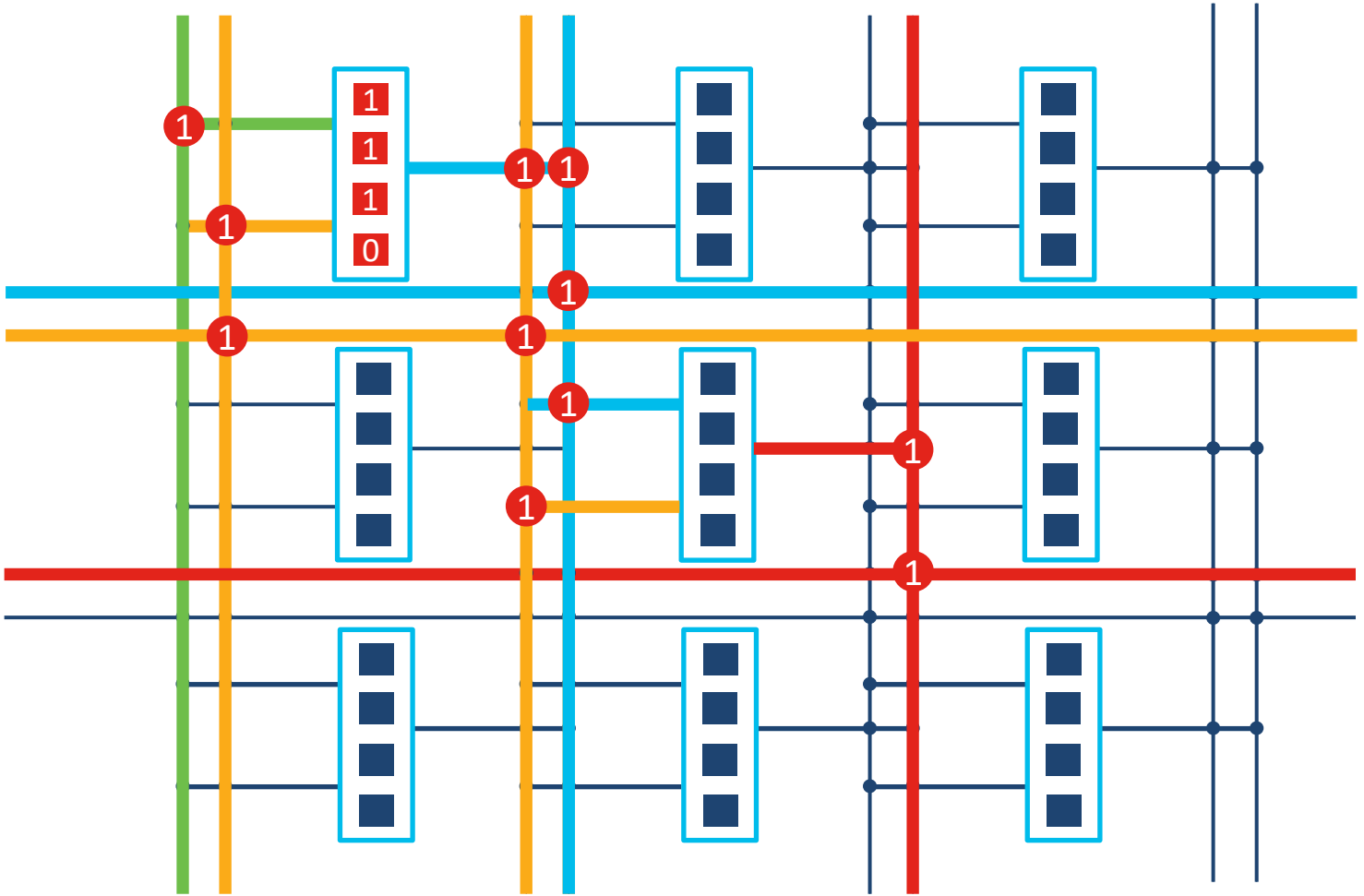


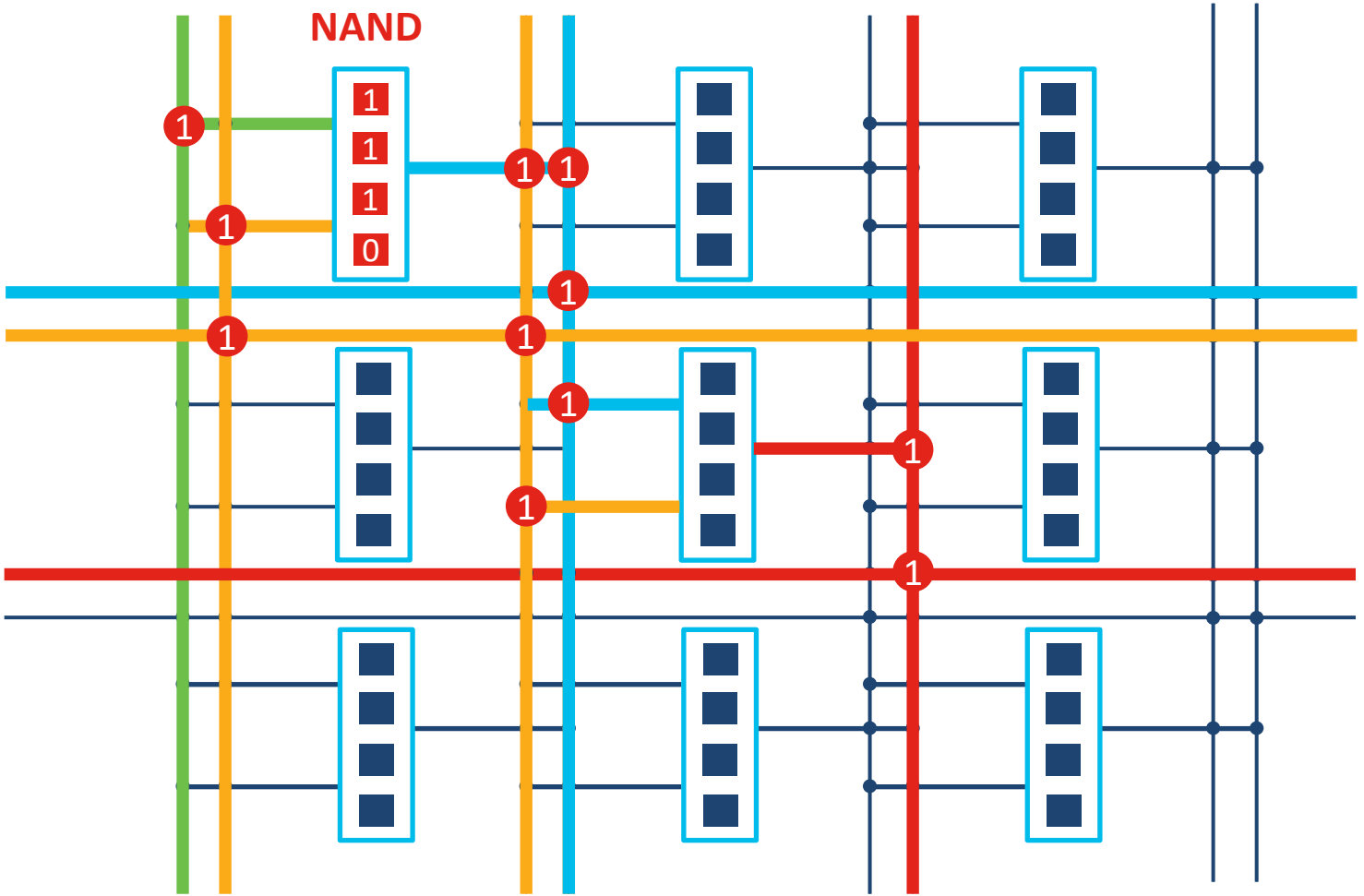


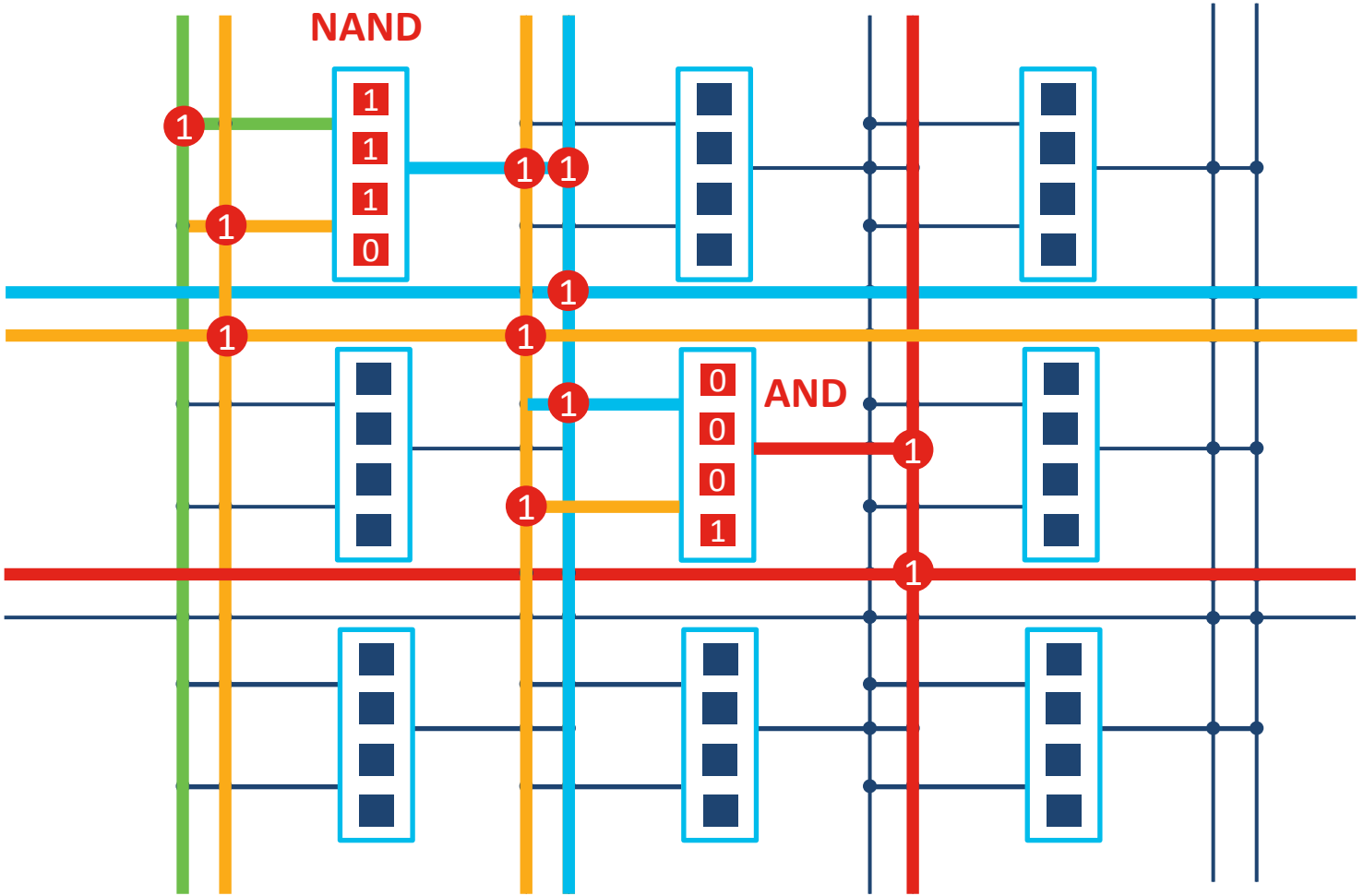


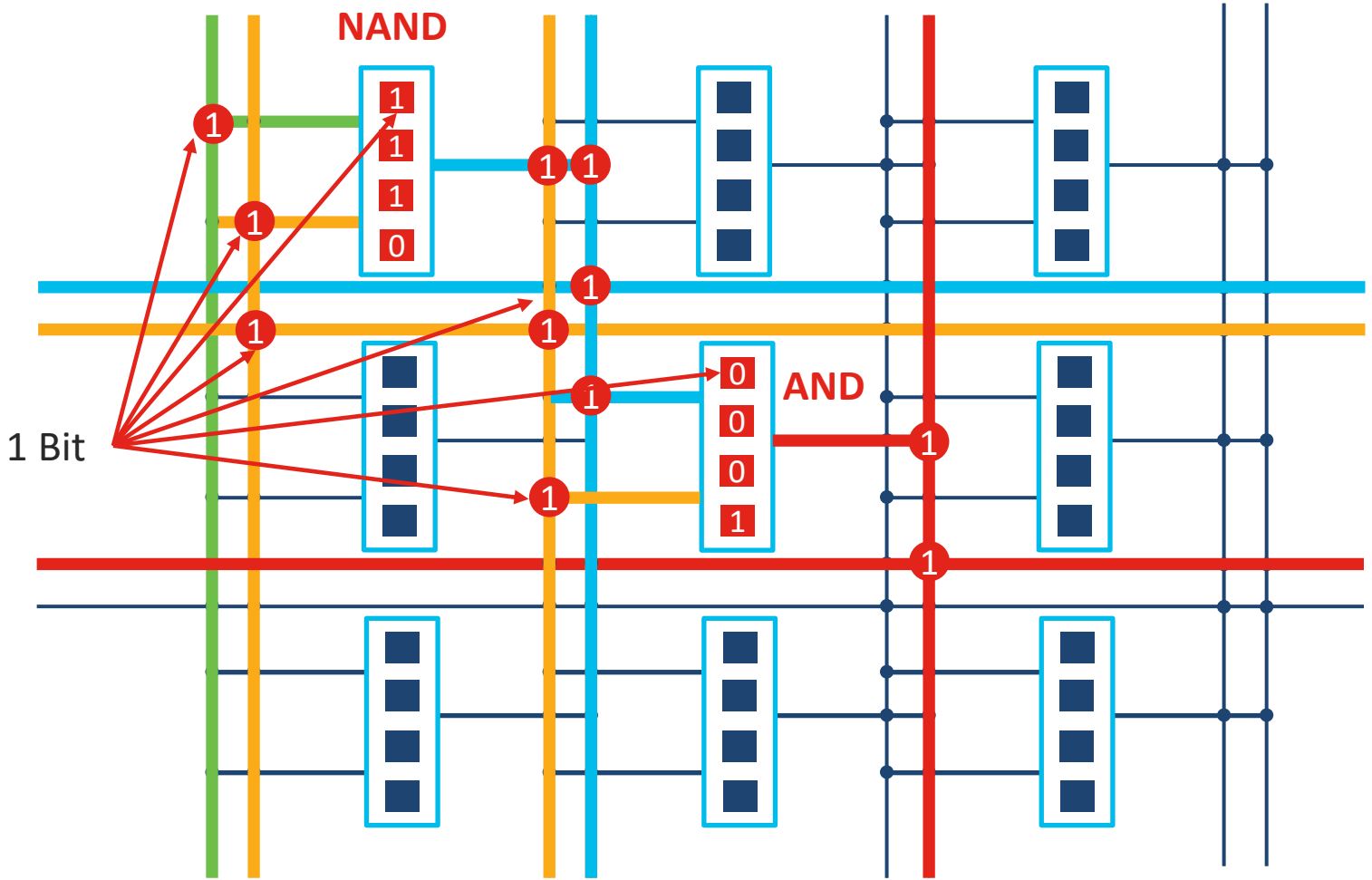


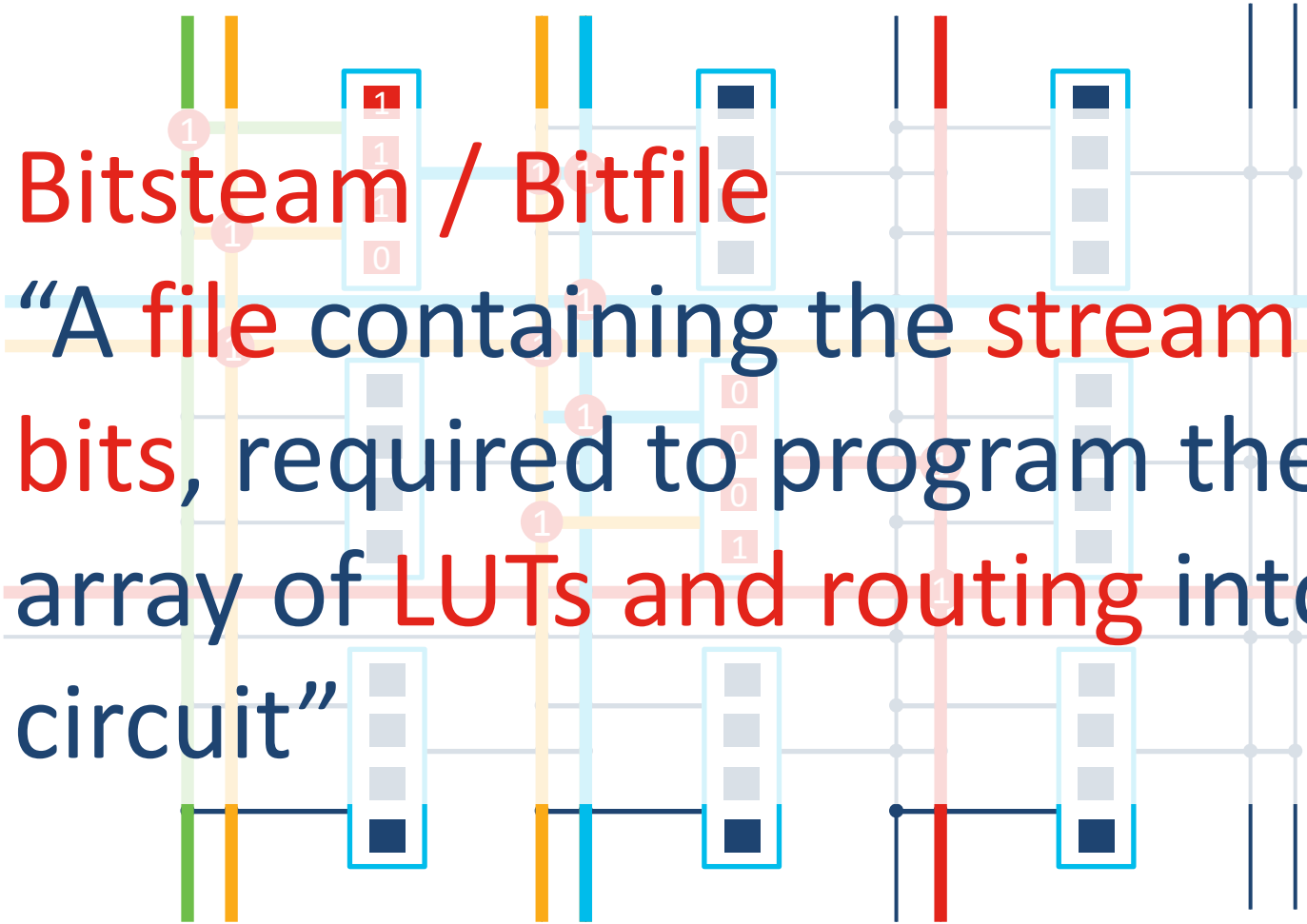






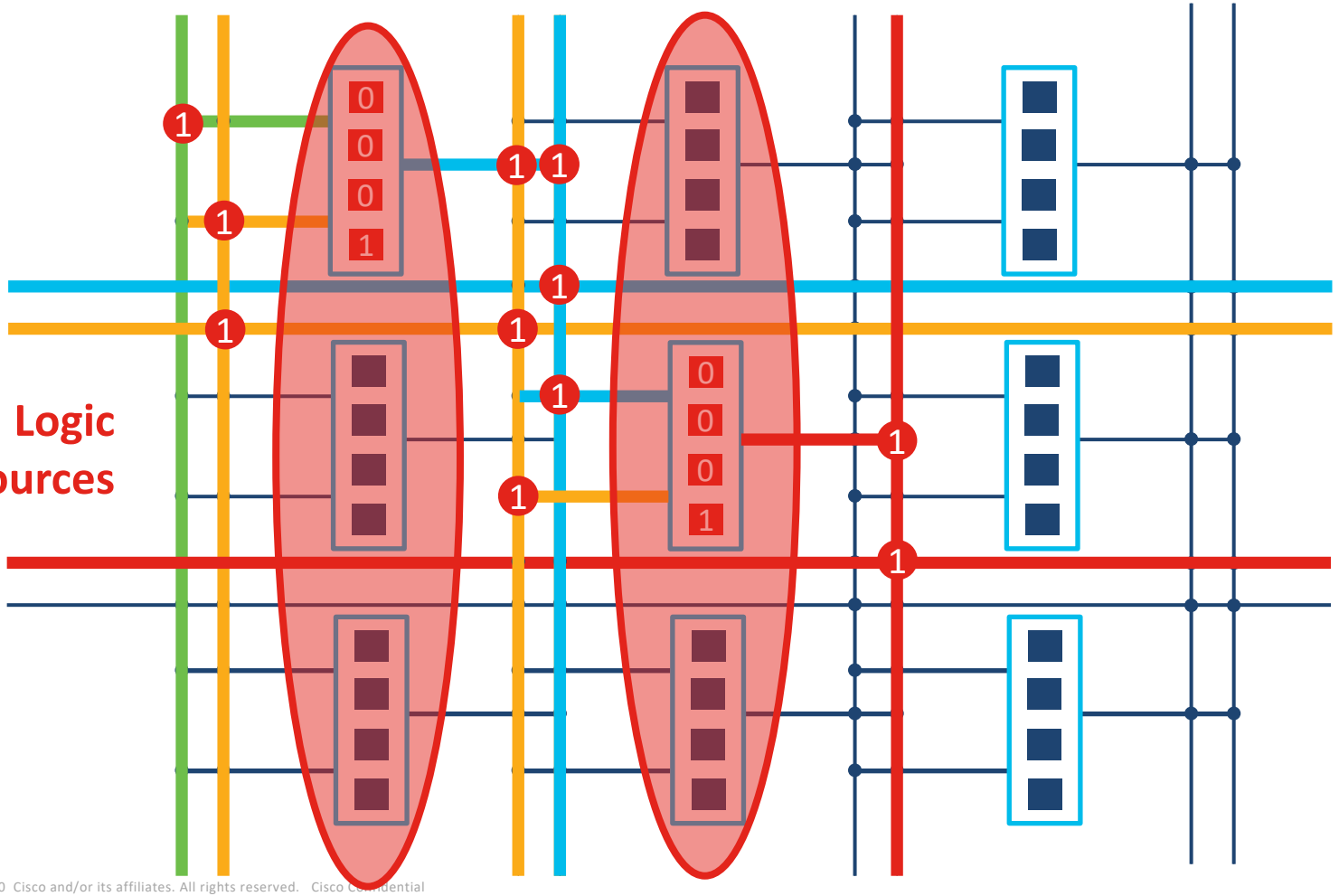


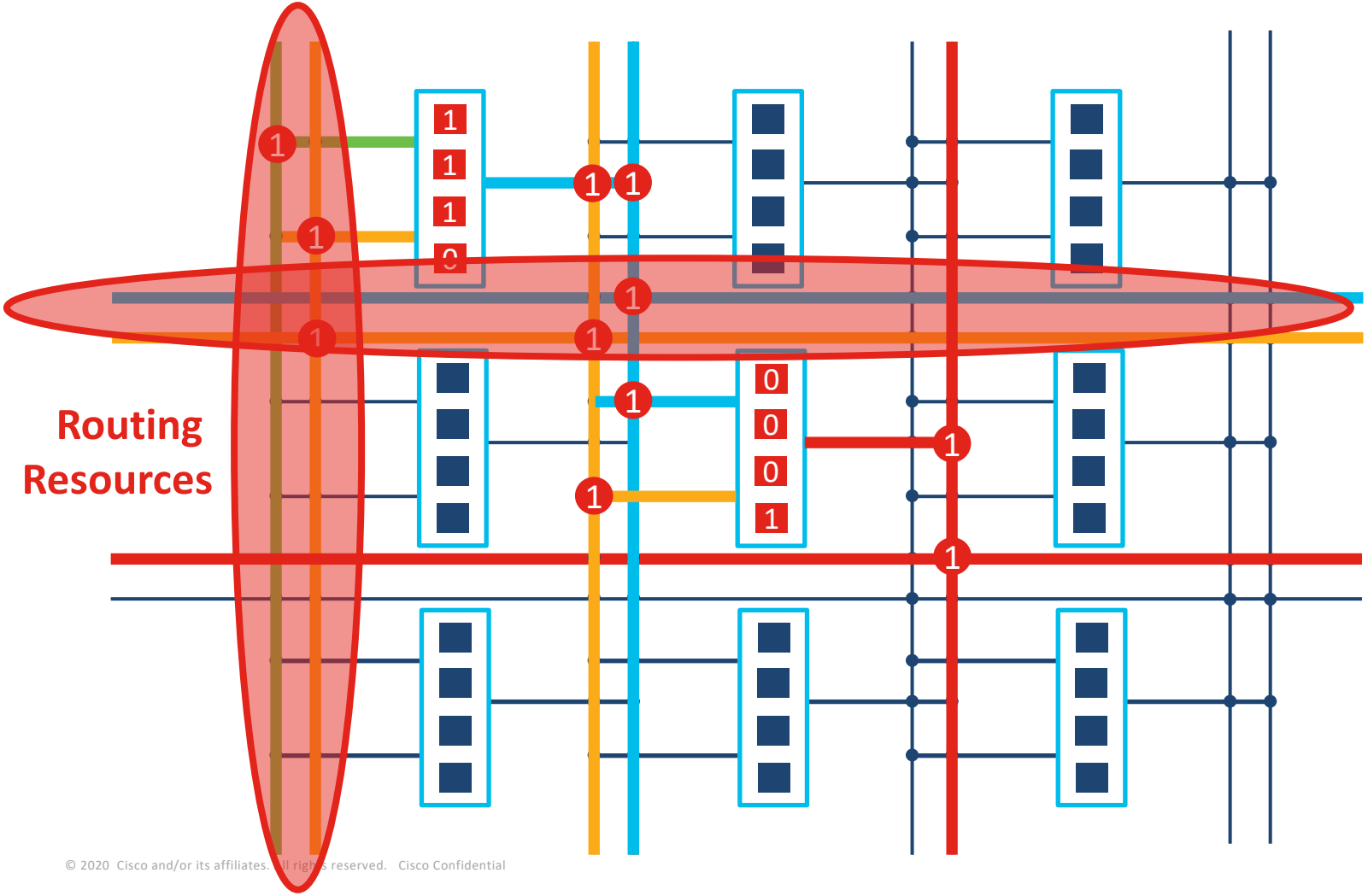


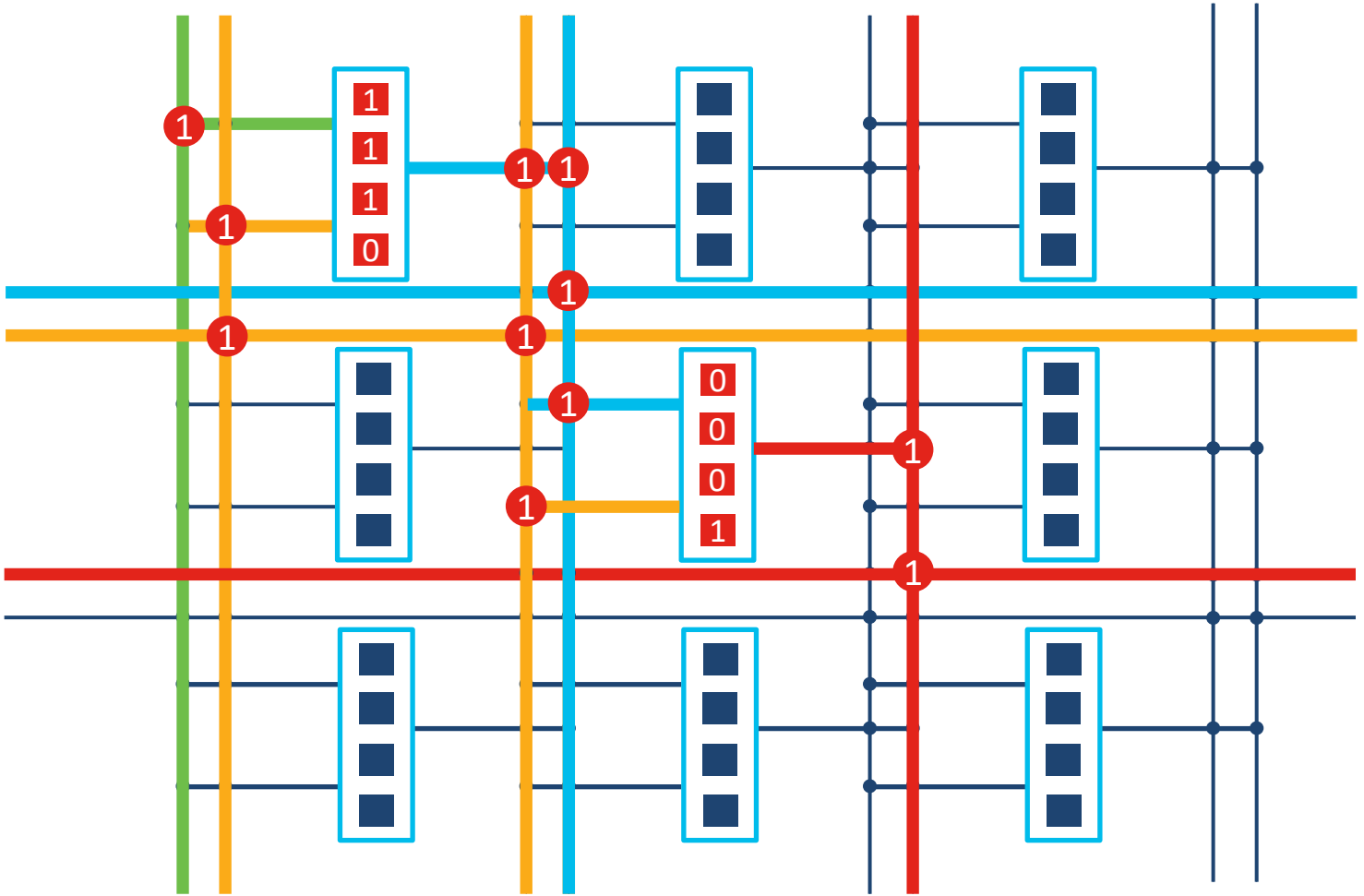


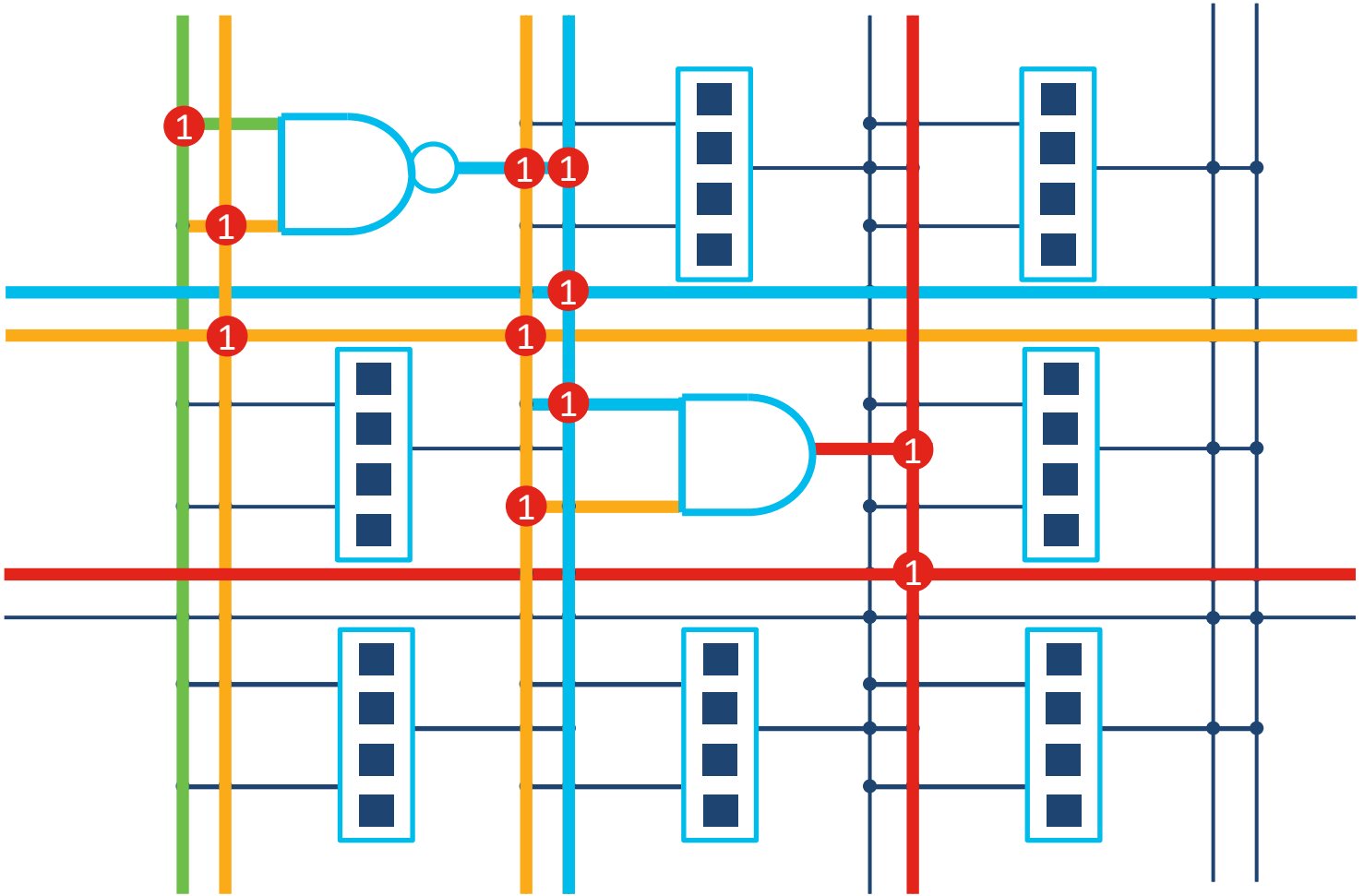
Bitsteam / Bitfile

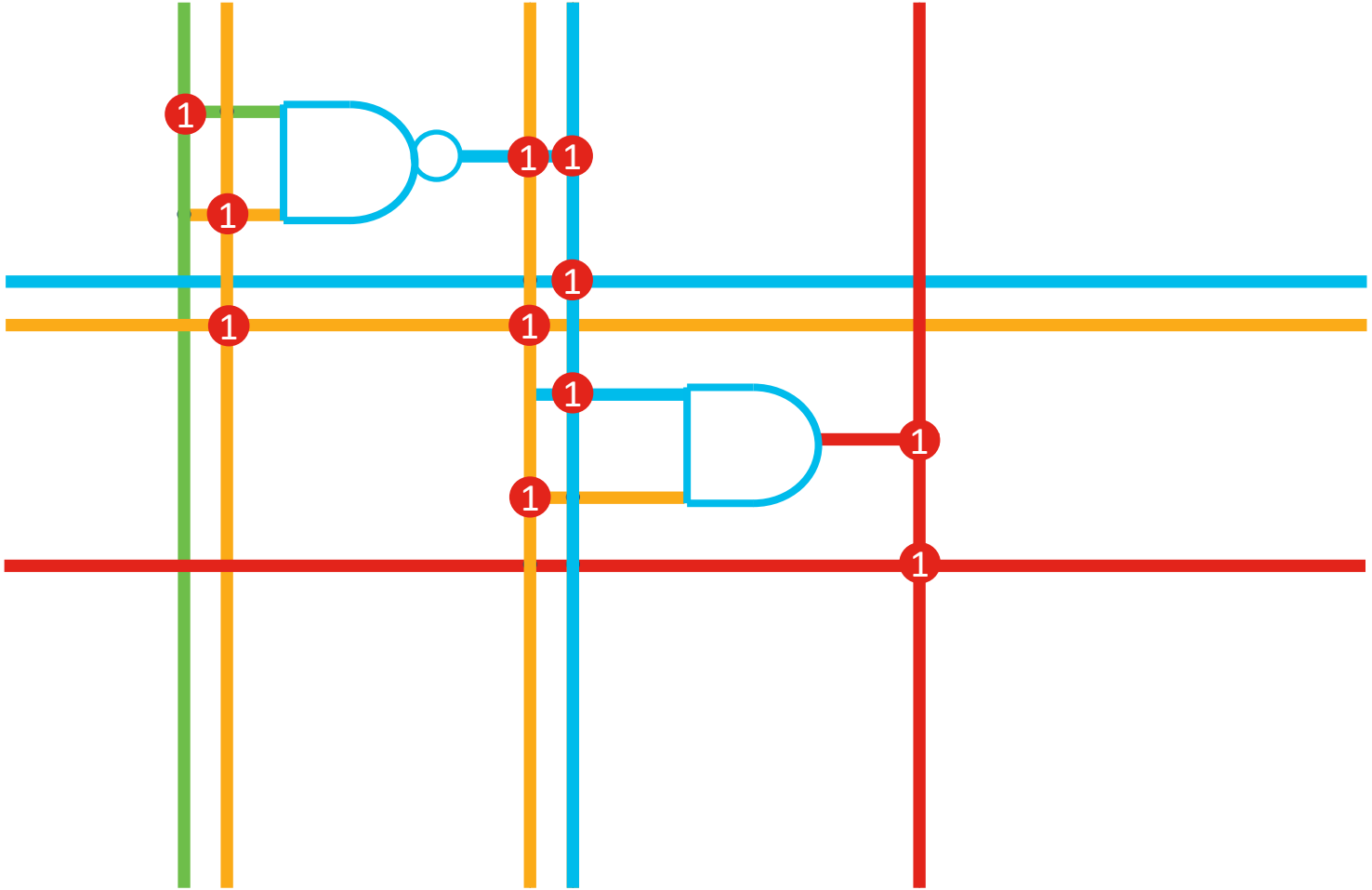
"A file containing the stream of bits, required to program the array of LUTs and routing into any circuit"

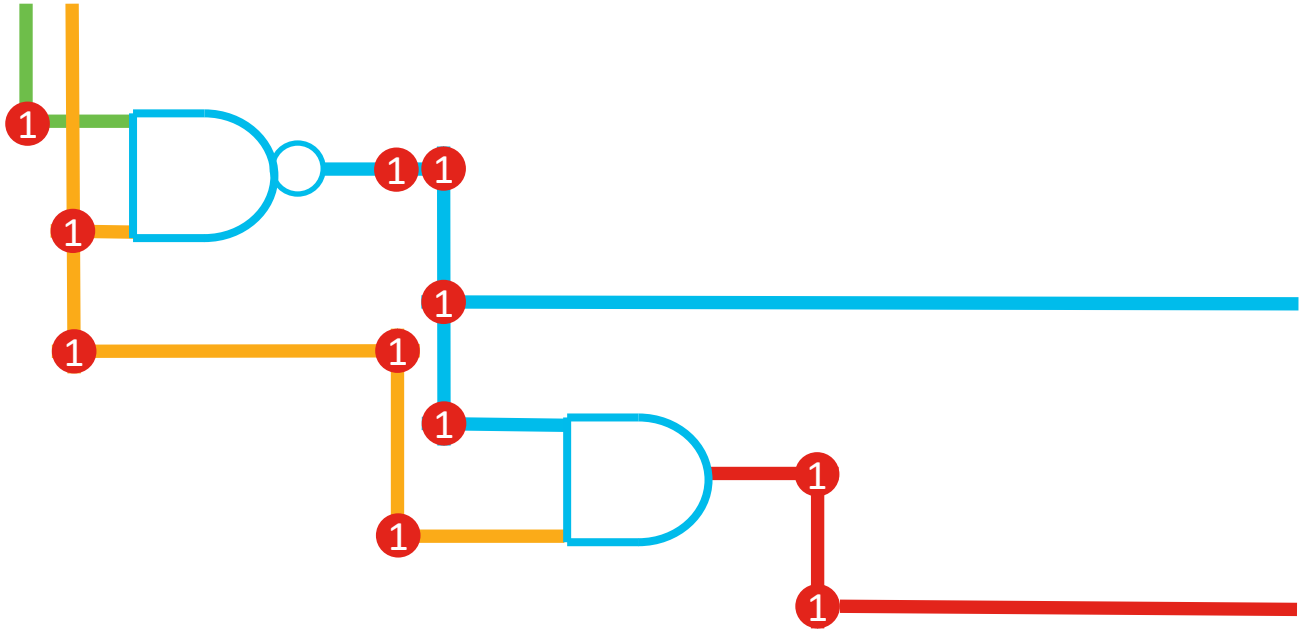


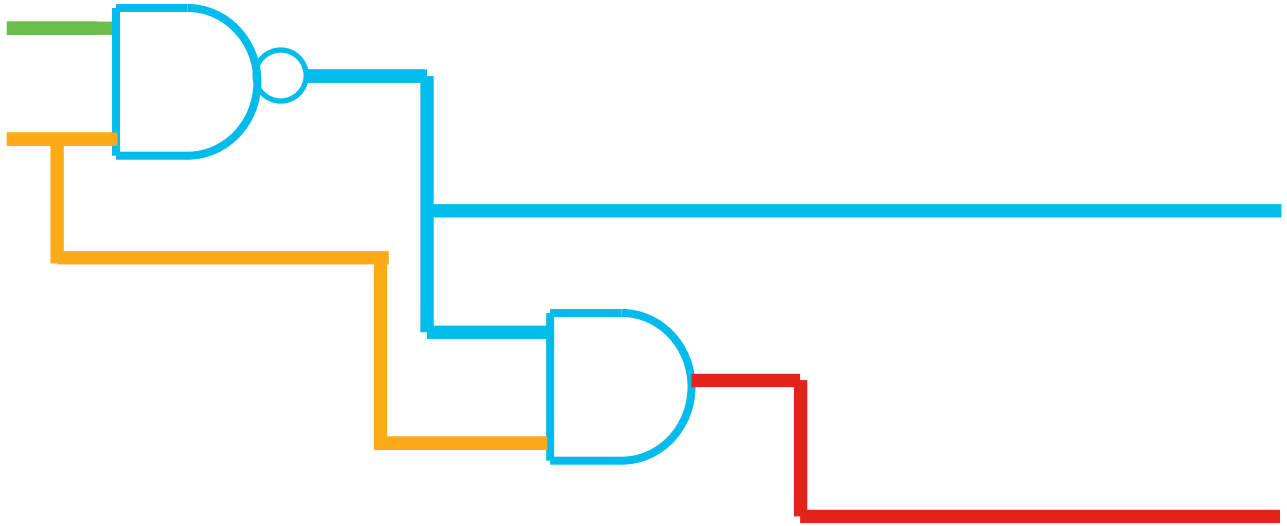


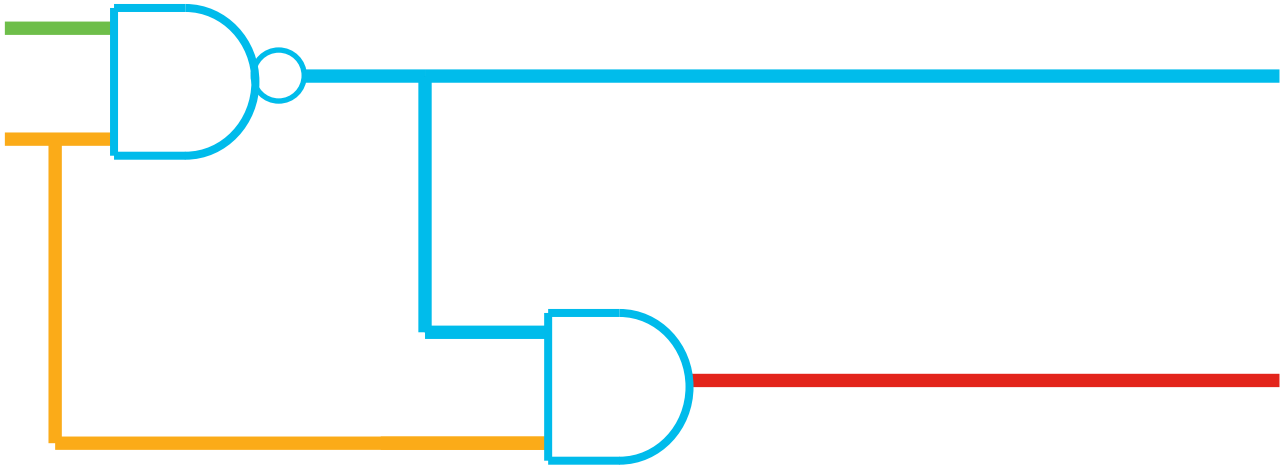


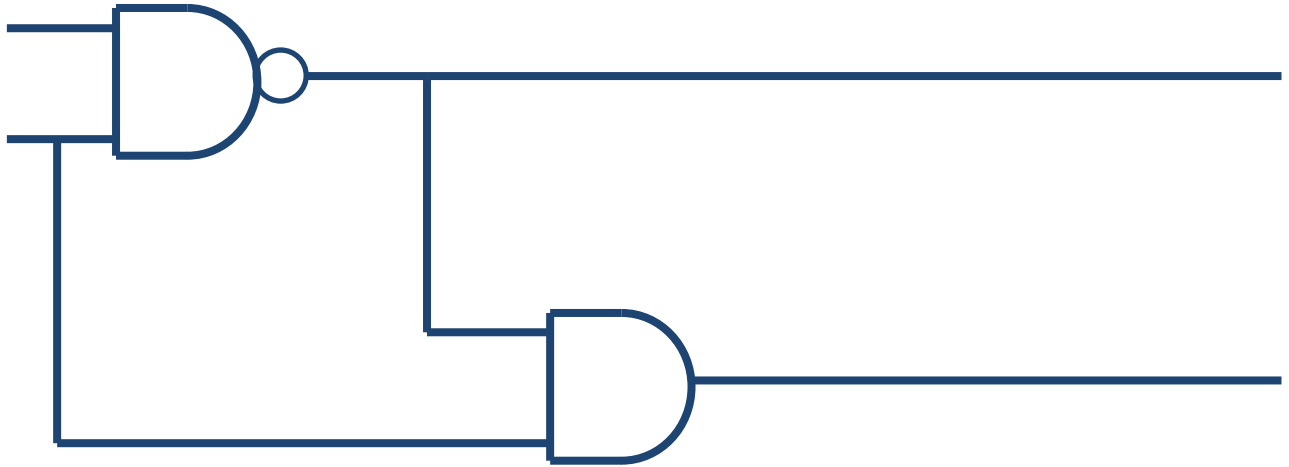








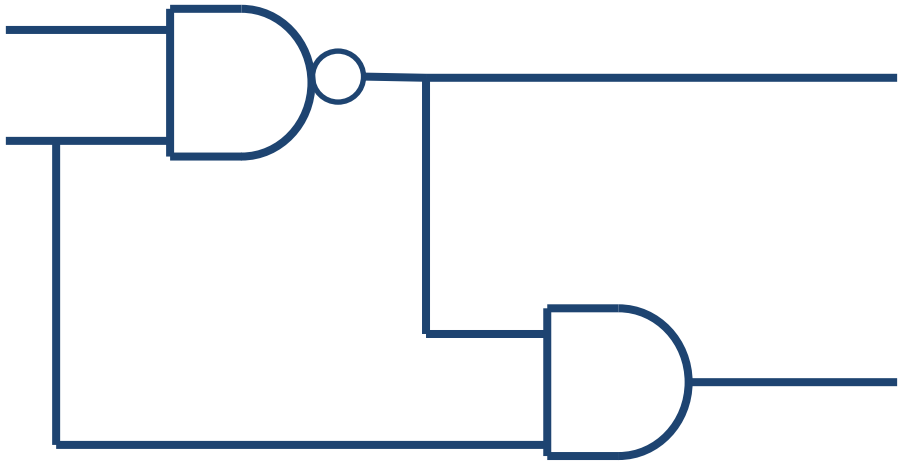


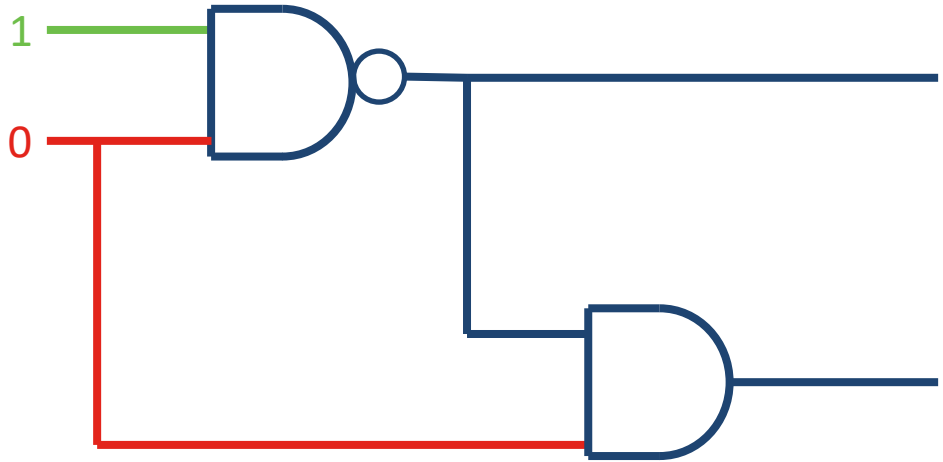


Combinational Circuit:

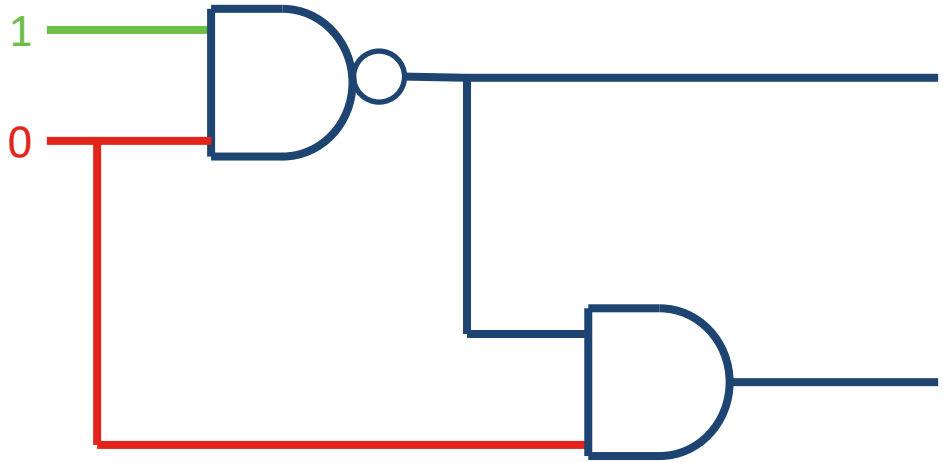
"Any circuit composed of only logical gates."


About Time!

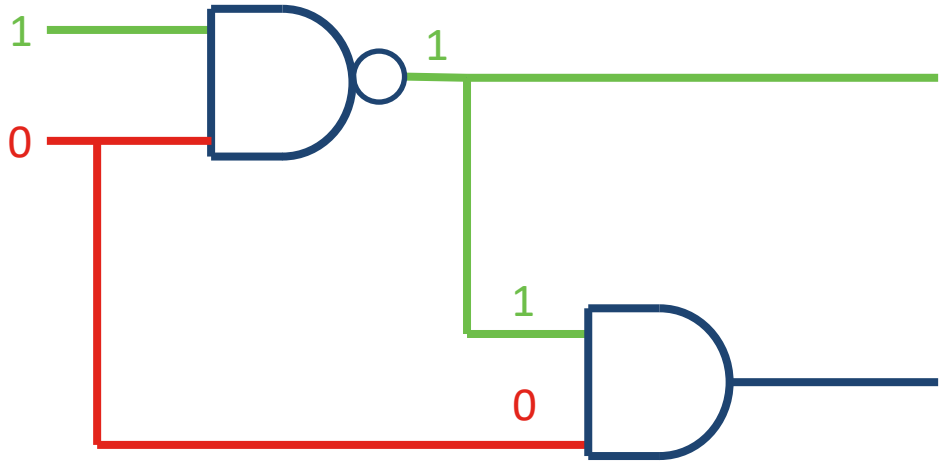




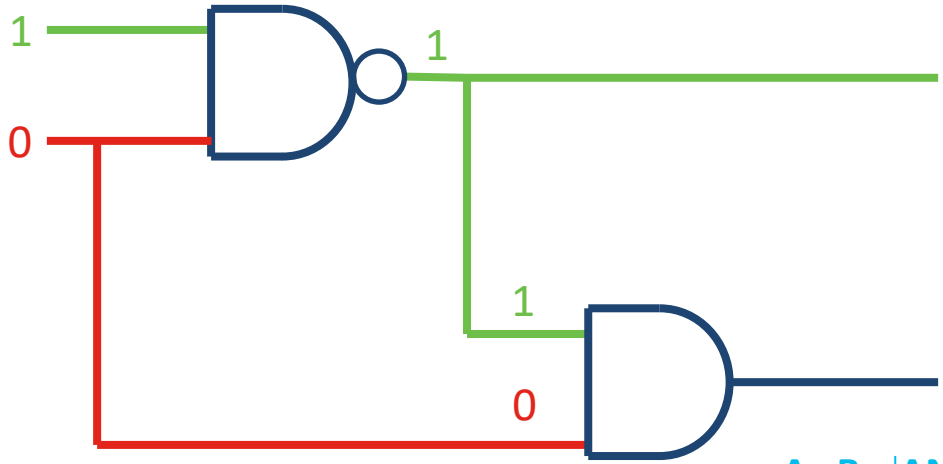
A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0



A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0

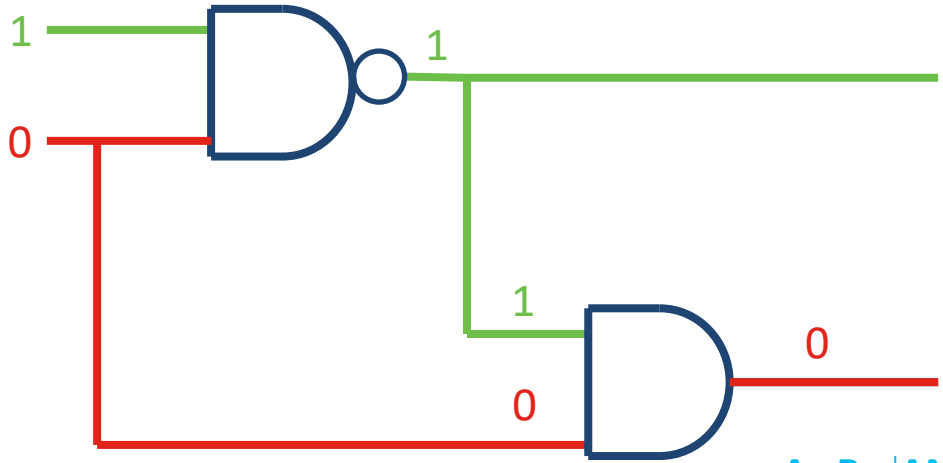


A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0



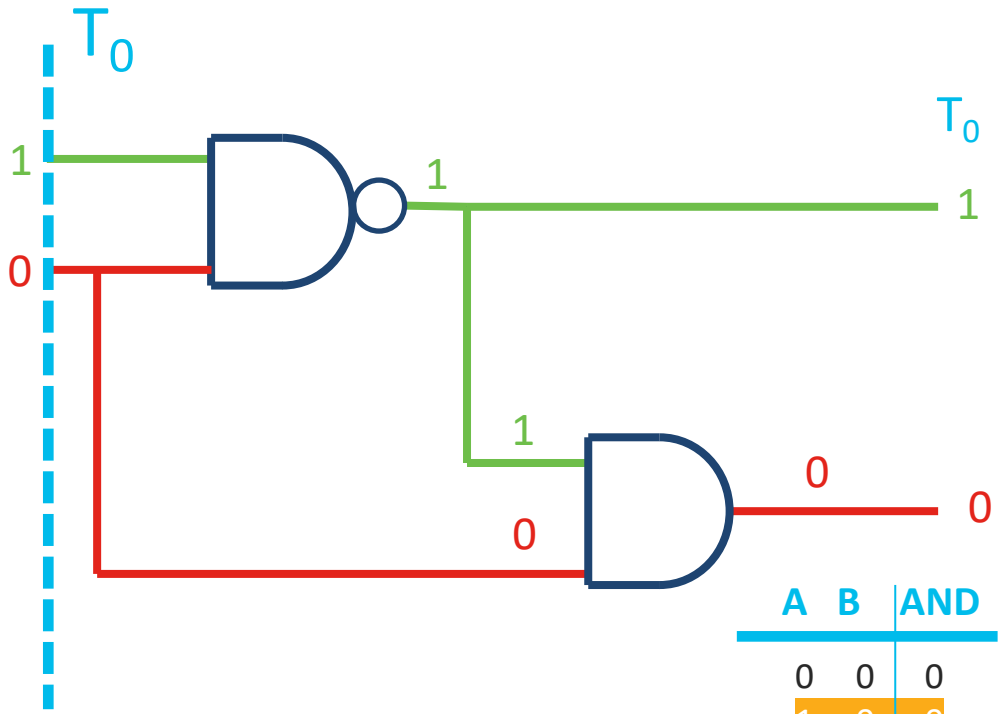
A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0



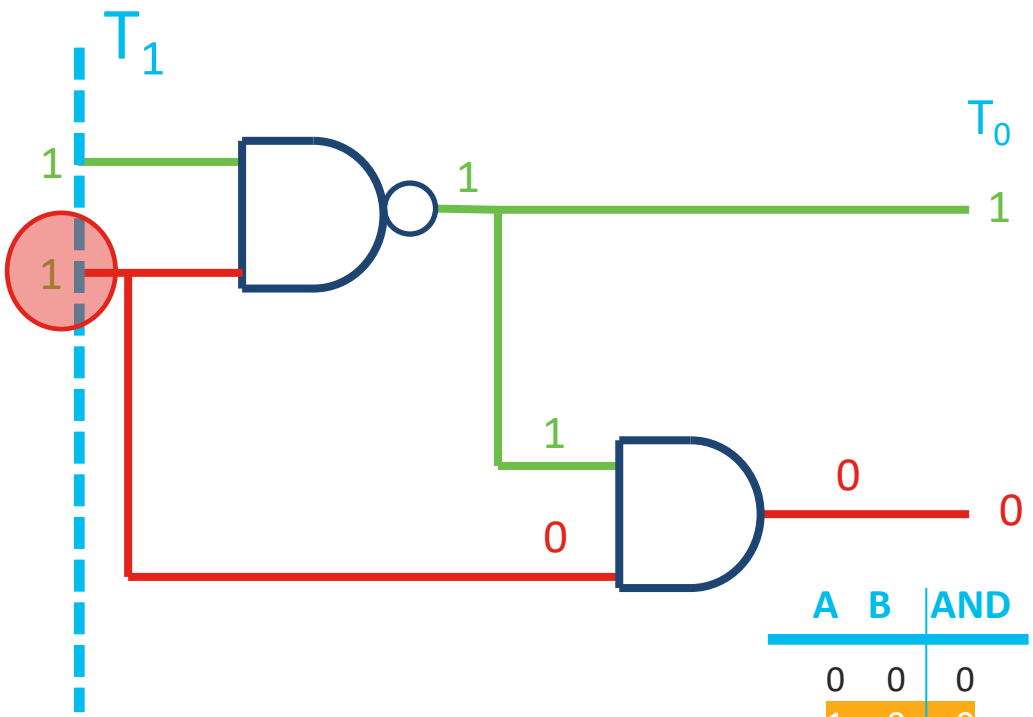
A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0

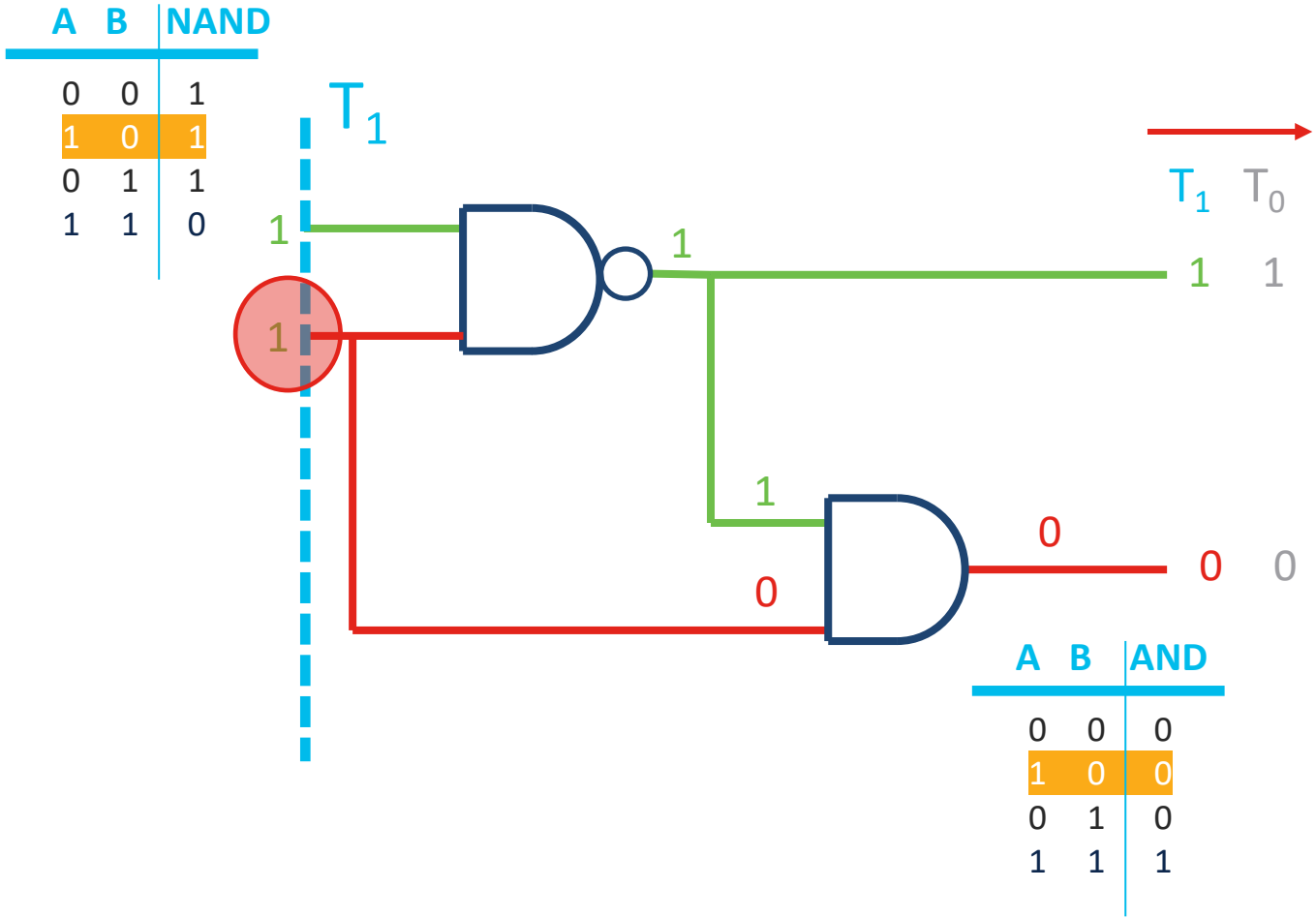


A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

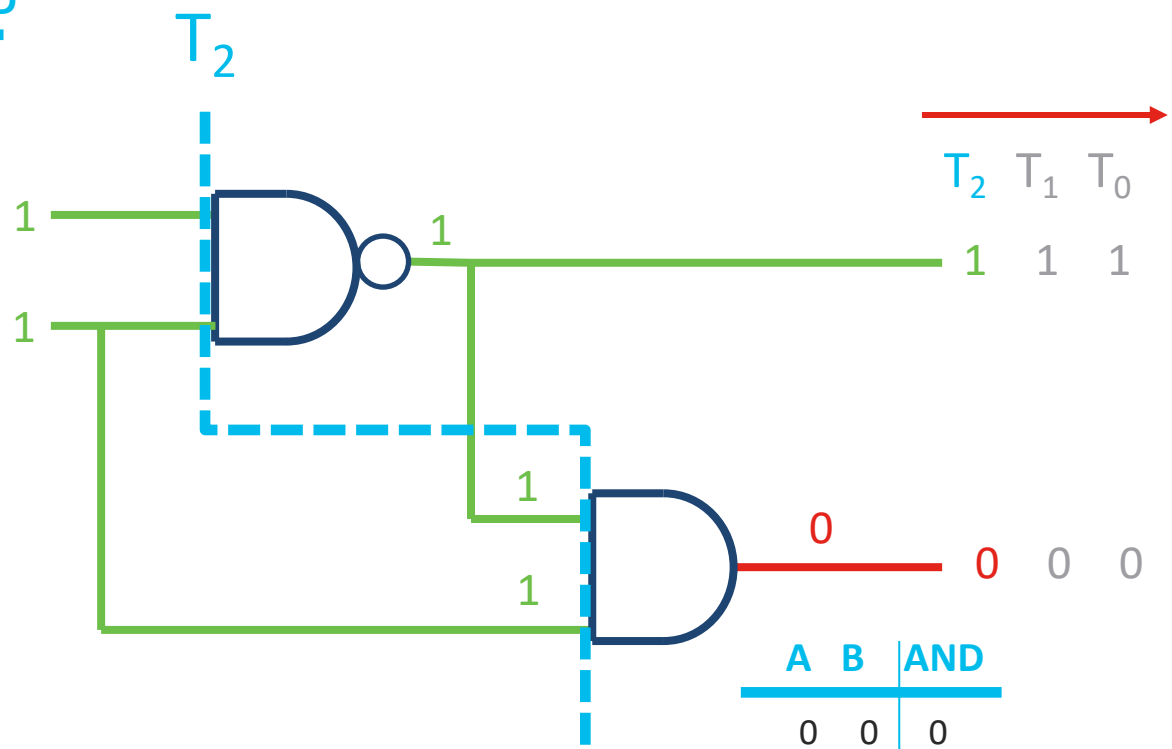
A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0



A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

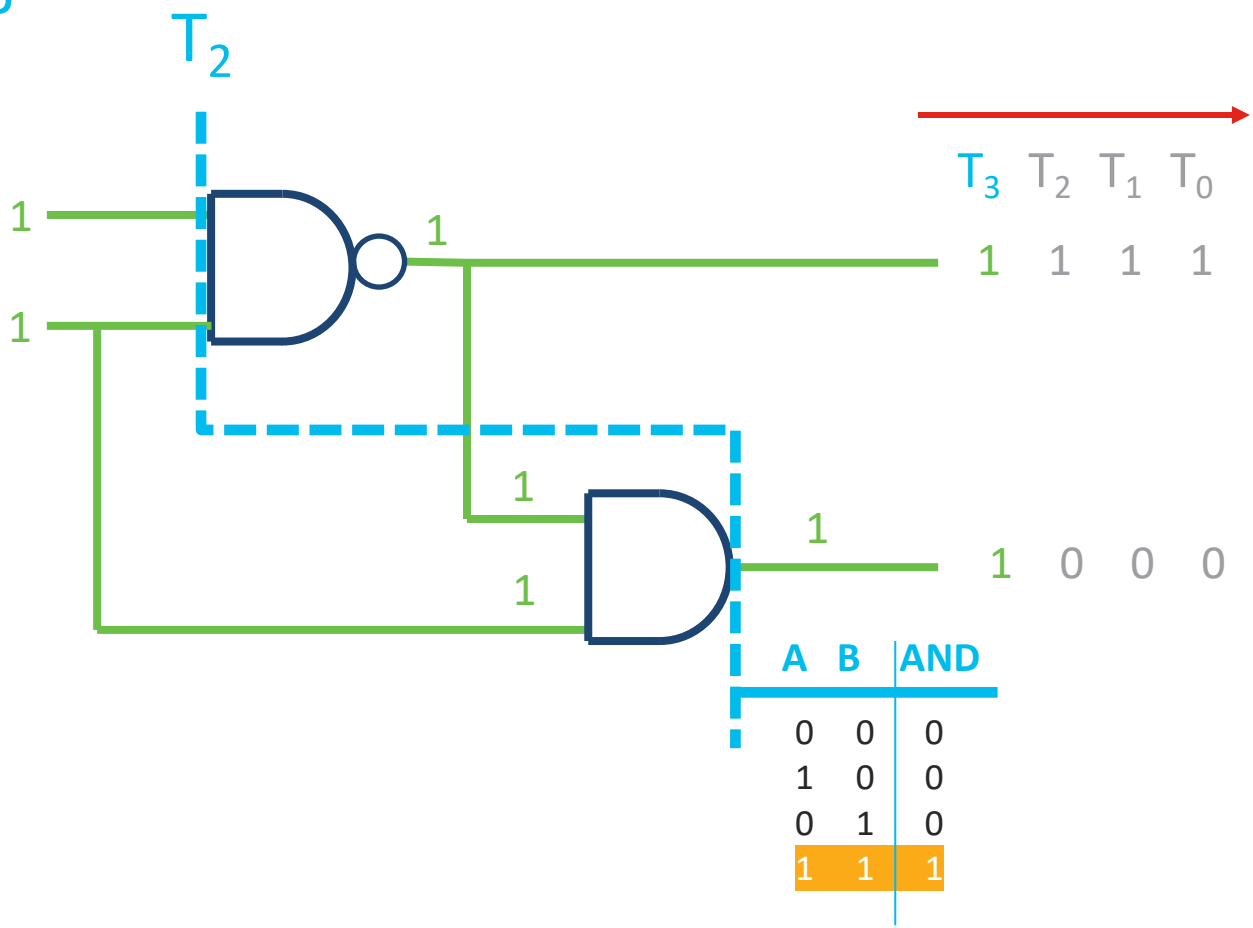


A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0



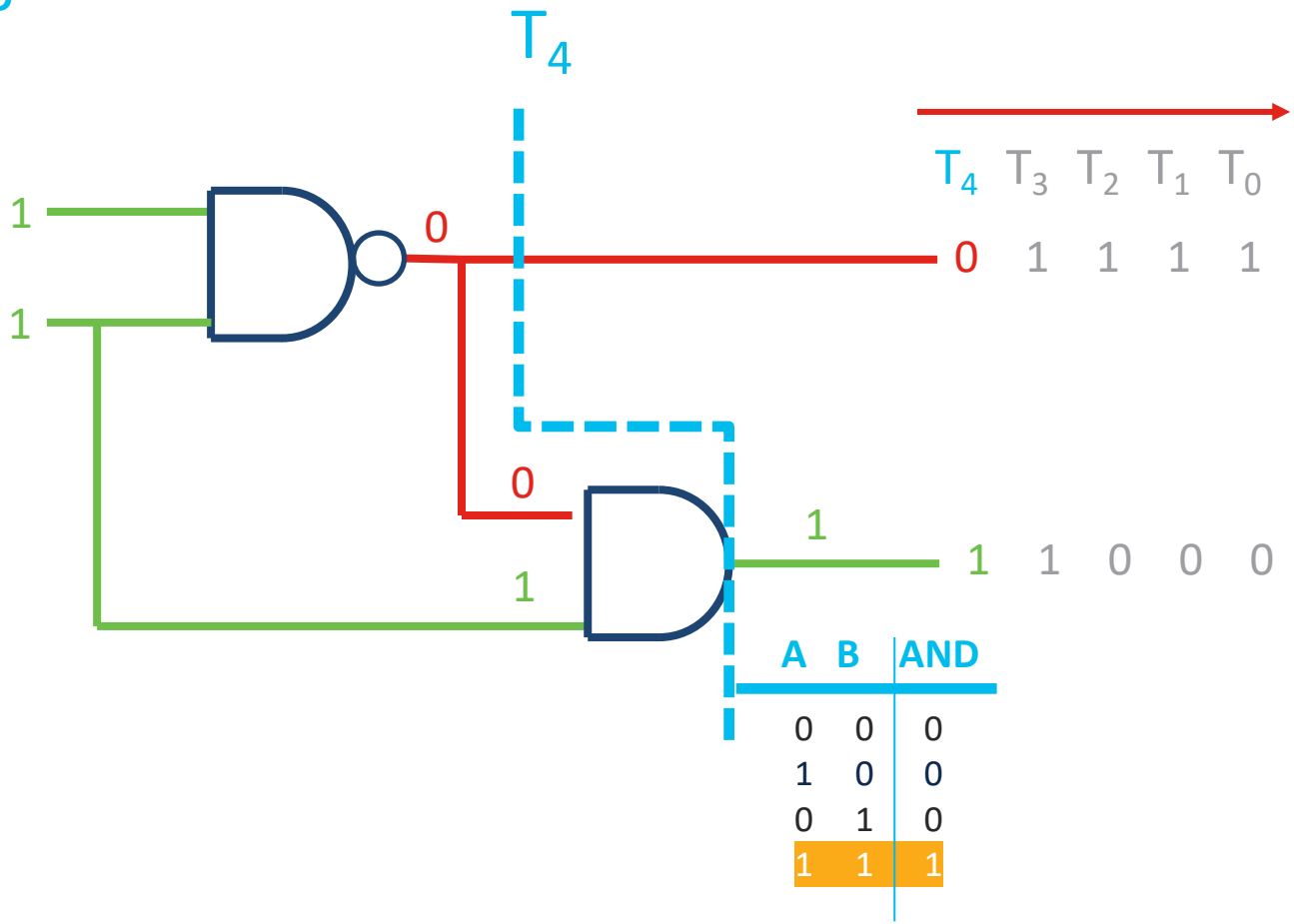
A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0

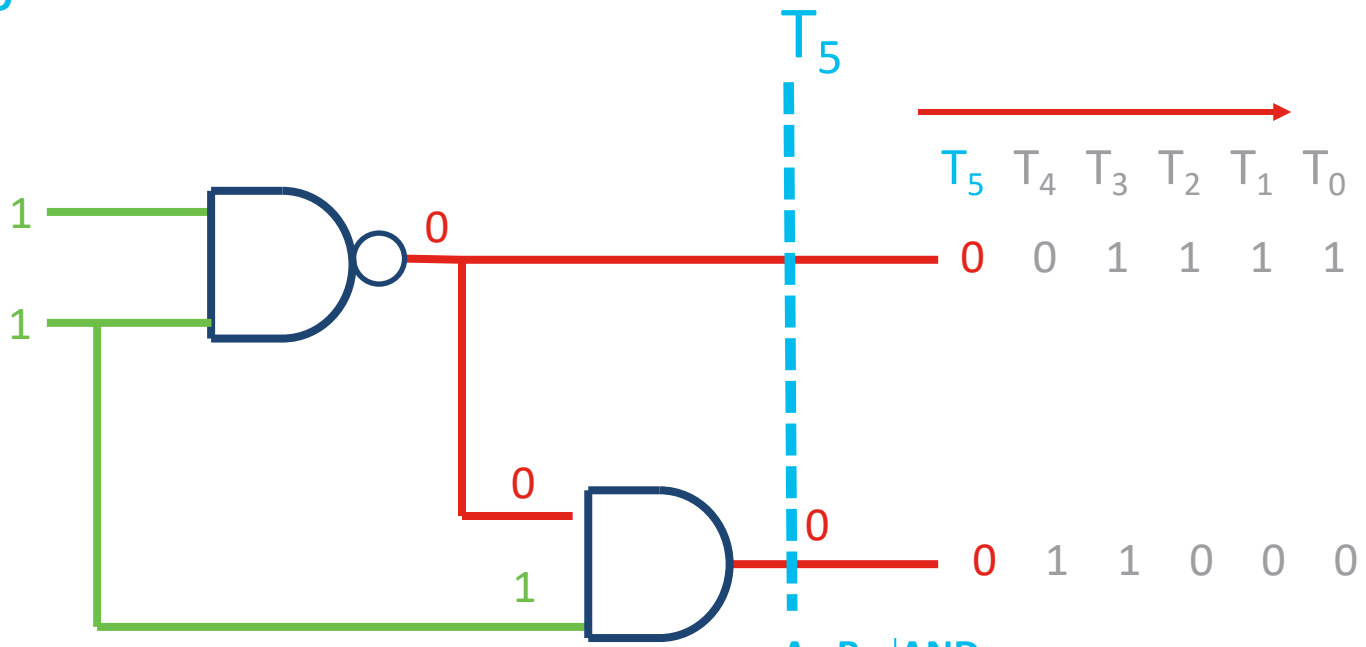


A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0

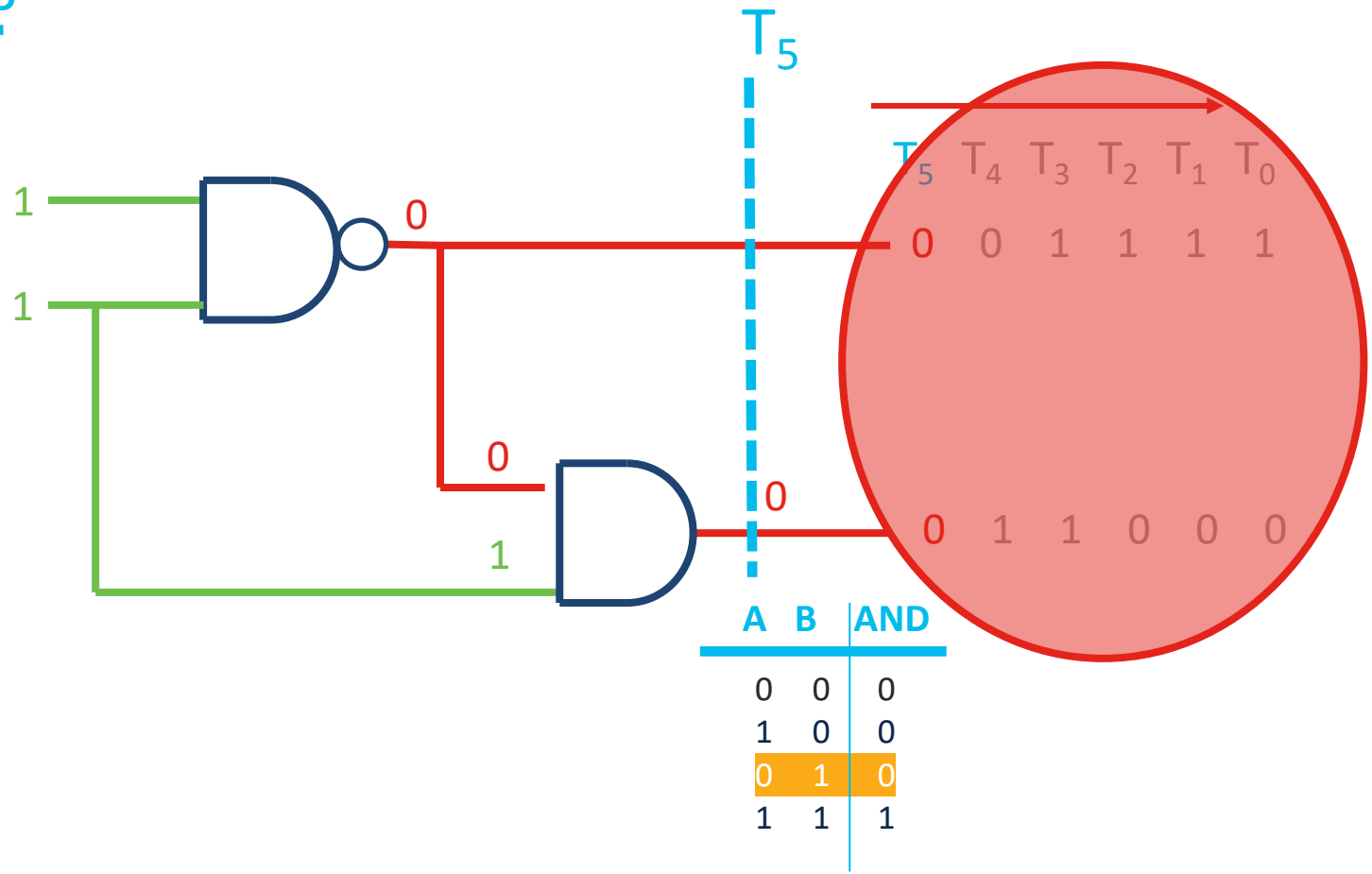


A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0



A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

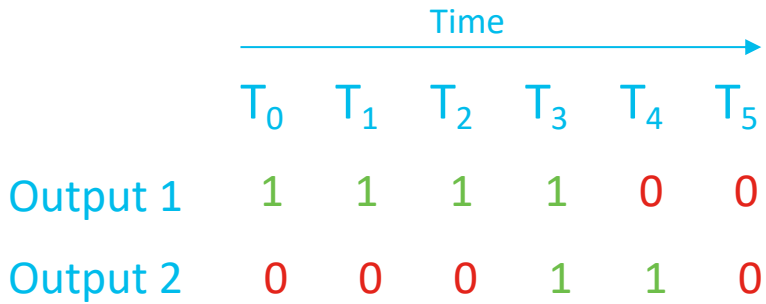
A	B	NAND
0	0	1
1	0	1
0	1	1
1	1	0

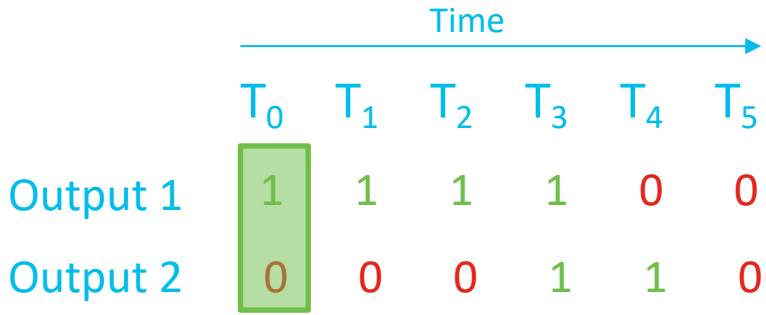


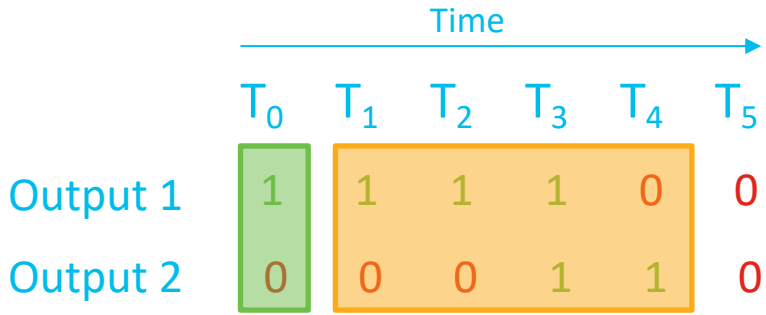
A	B	AND
0	0	0
1	0	0
0	1	0
1	1	1

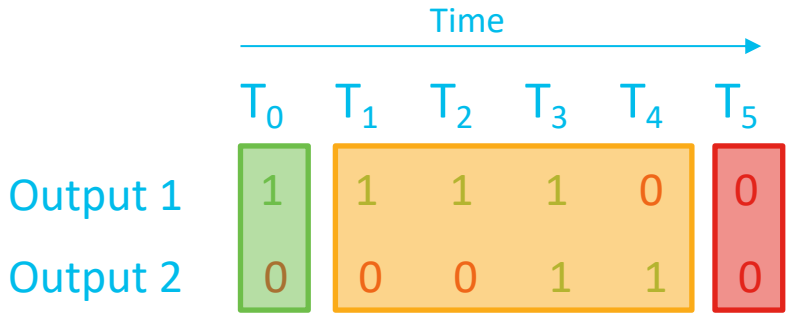
	T_5	T_4	T_3	T_2	T_1	T_0
Output 1	0	0	1	1	1	1
Output 2	0	1	1	0	0	0

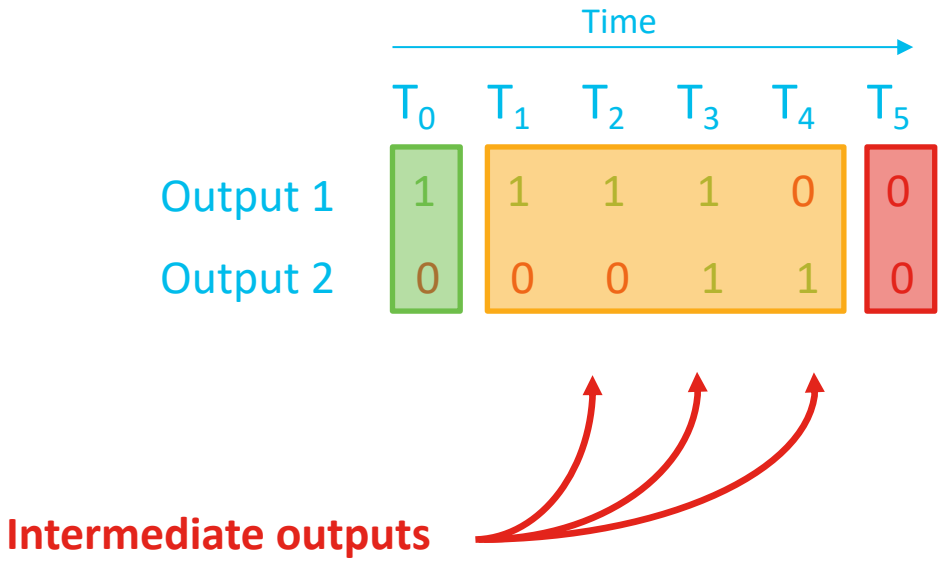
	T_5	T_4	T_3	T_2	T_1	T_0
Output 1	0	0	1	1	1	1
Output 2	0	1	1	0	0	0

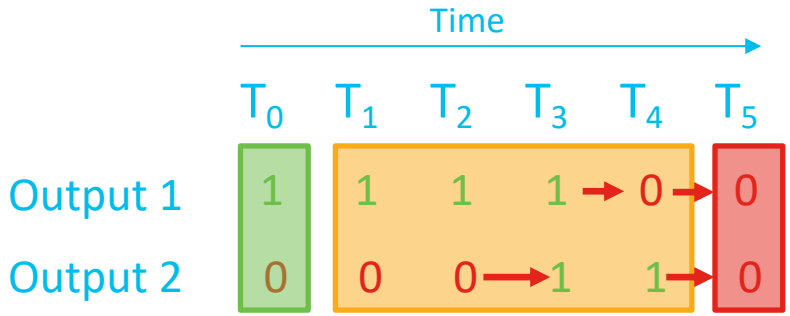


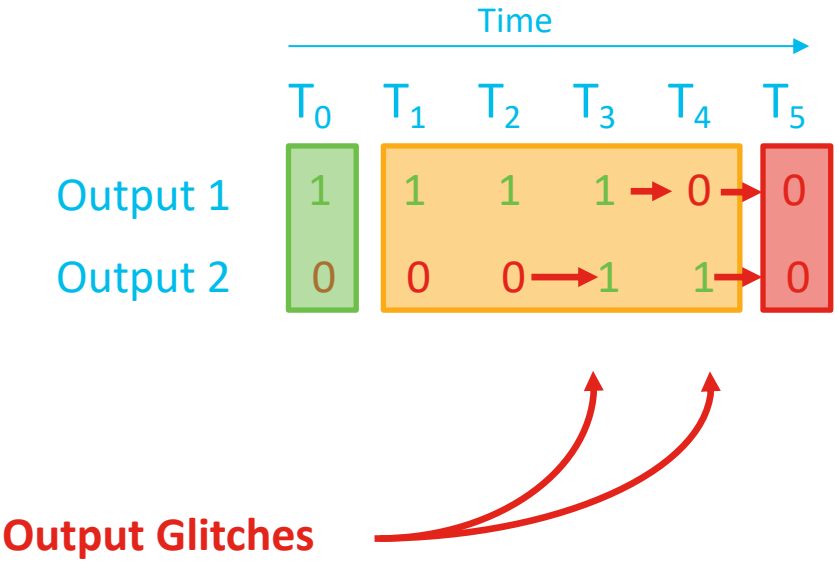






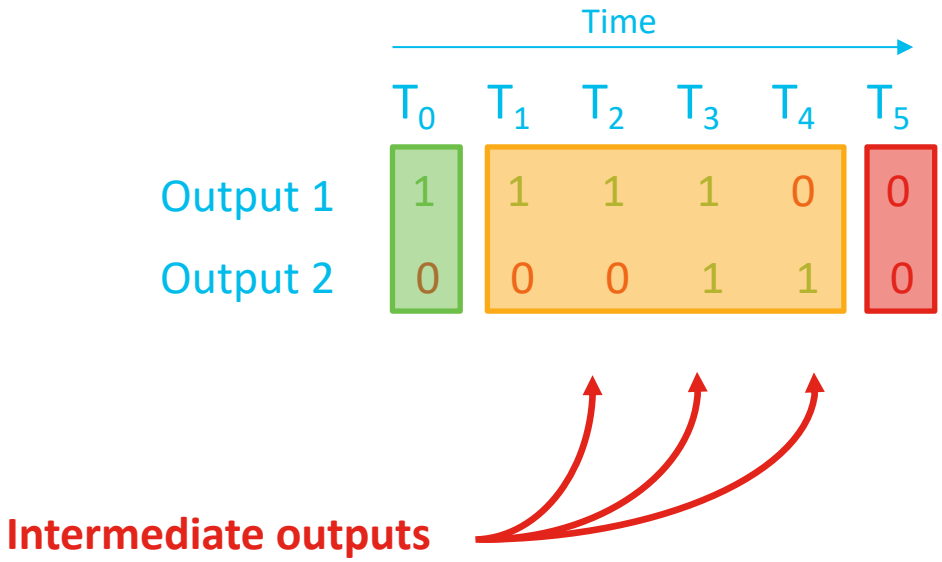


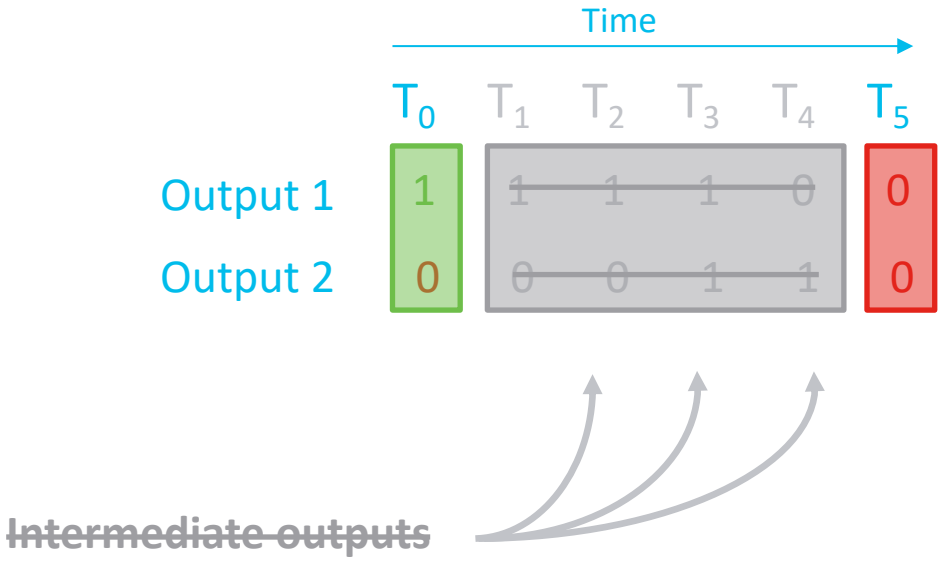


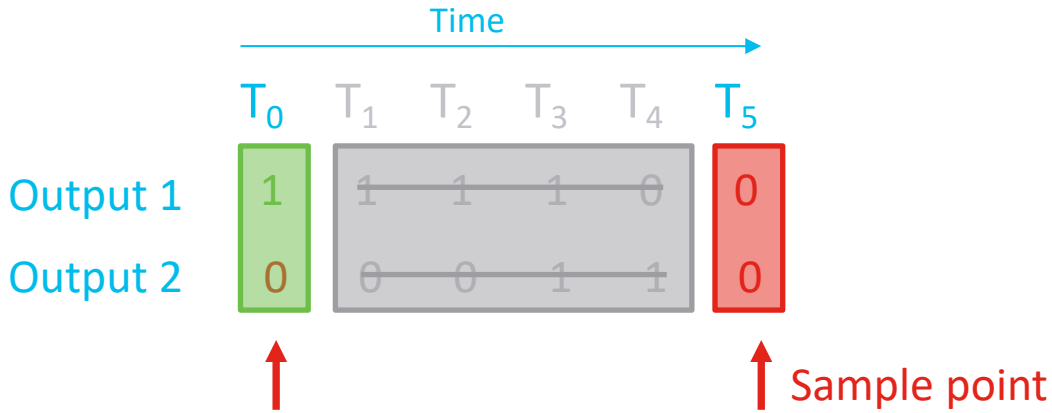


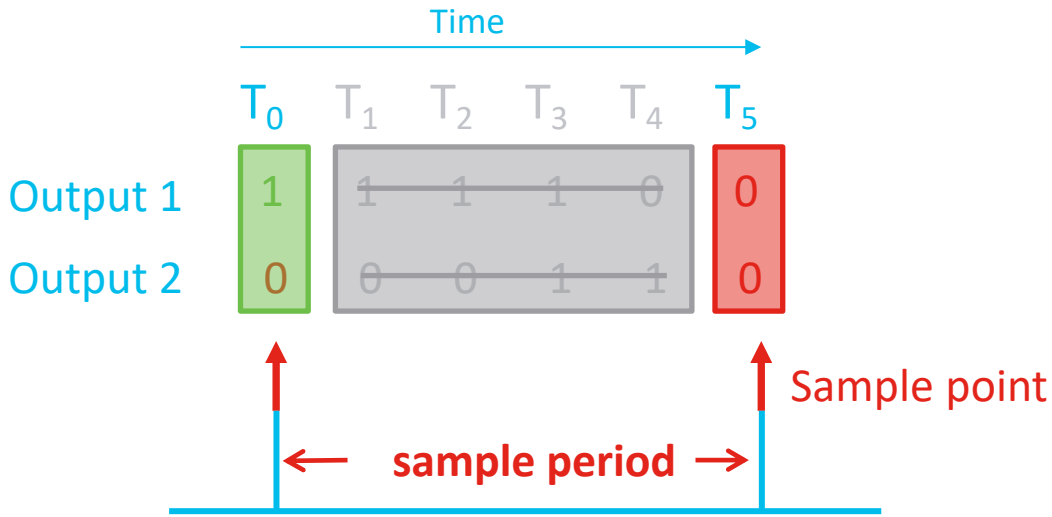
Glitching:

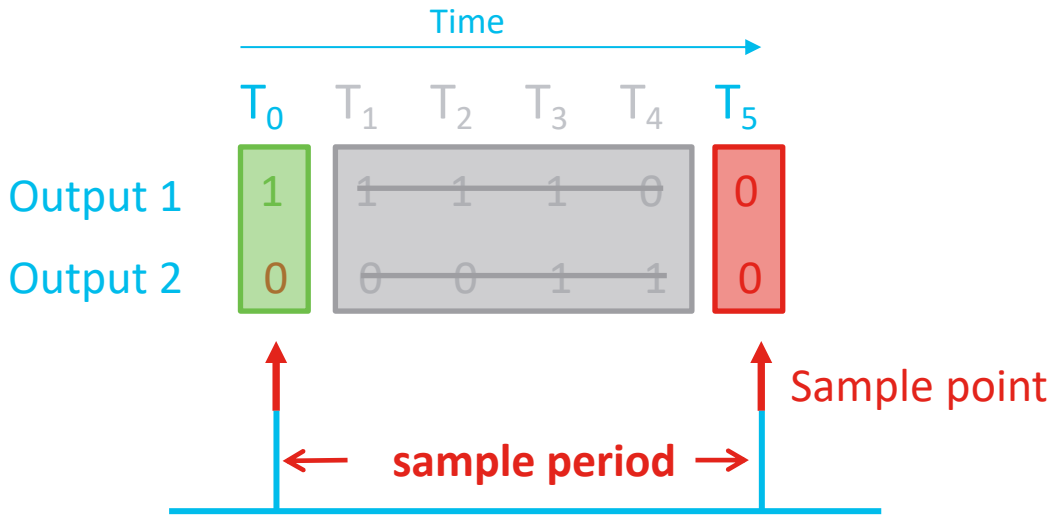
“Intermediate outputs while circuit to stabilizes to a final output over time”





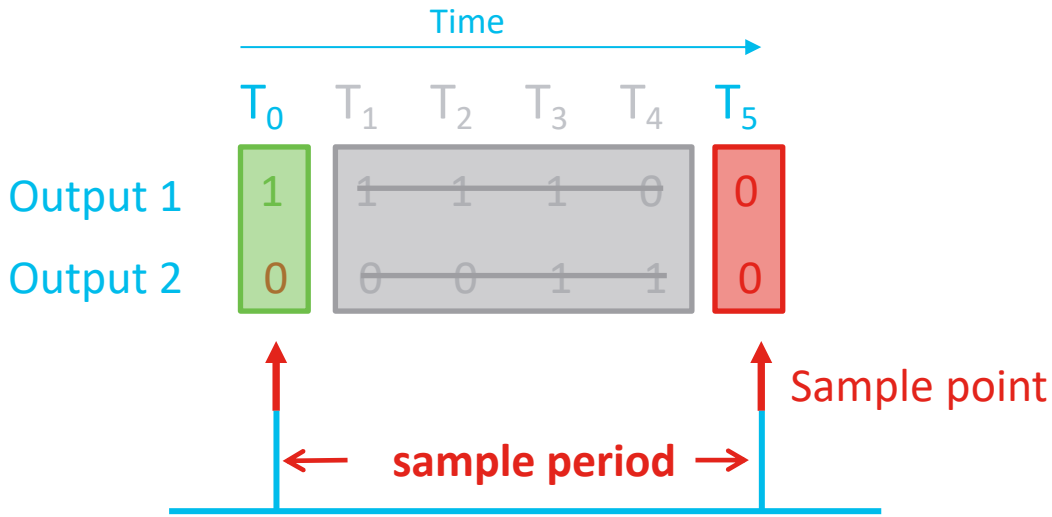






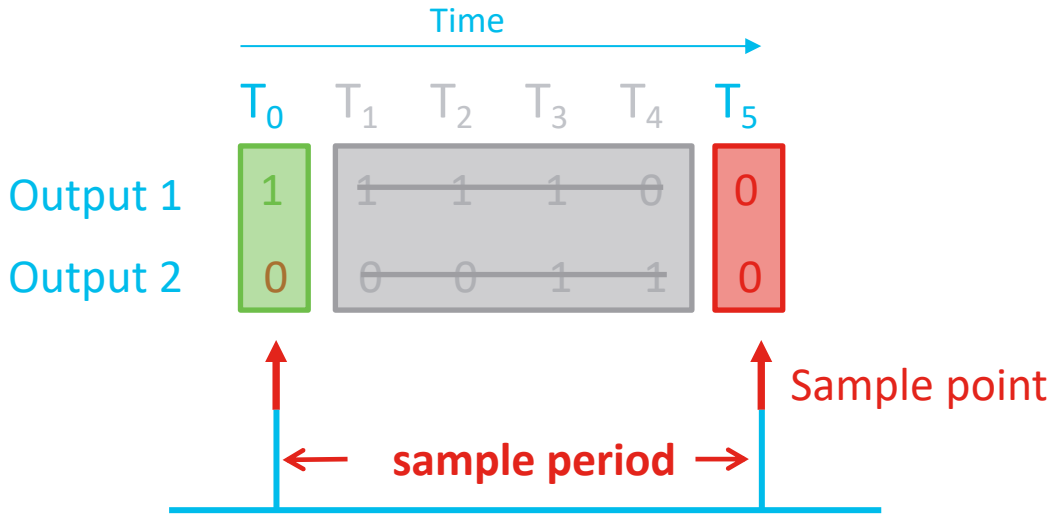
Sampled output 1 1

Sampled output 2 0



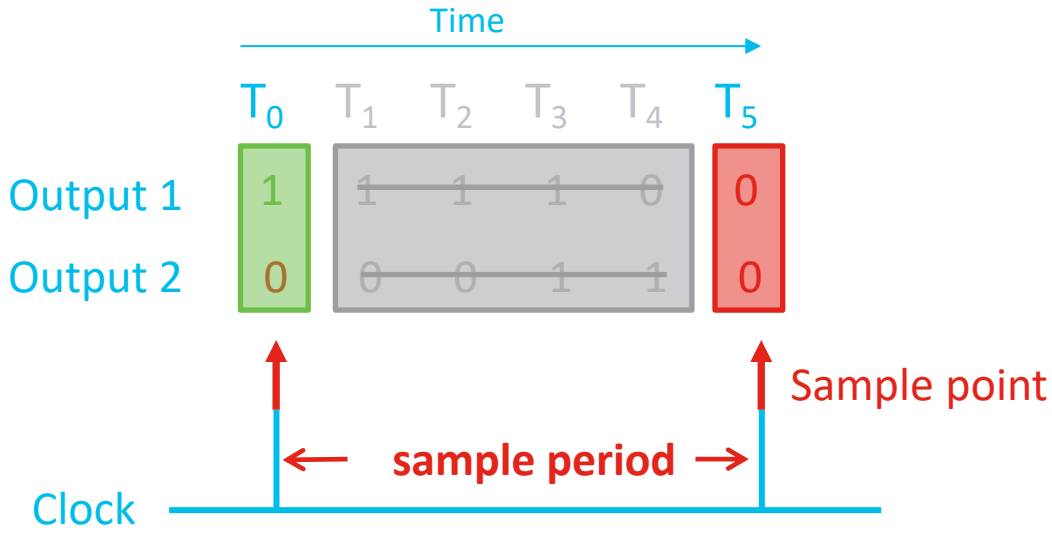
Sampled output 1 1

Sampled output 2 0



Sampled output 1 1 → 0

Sampled output 2 0 → 0

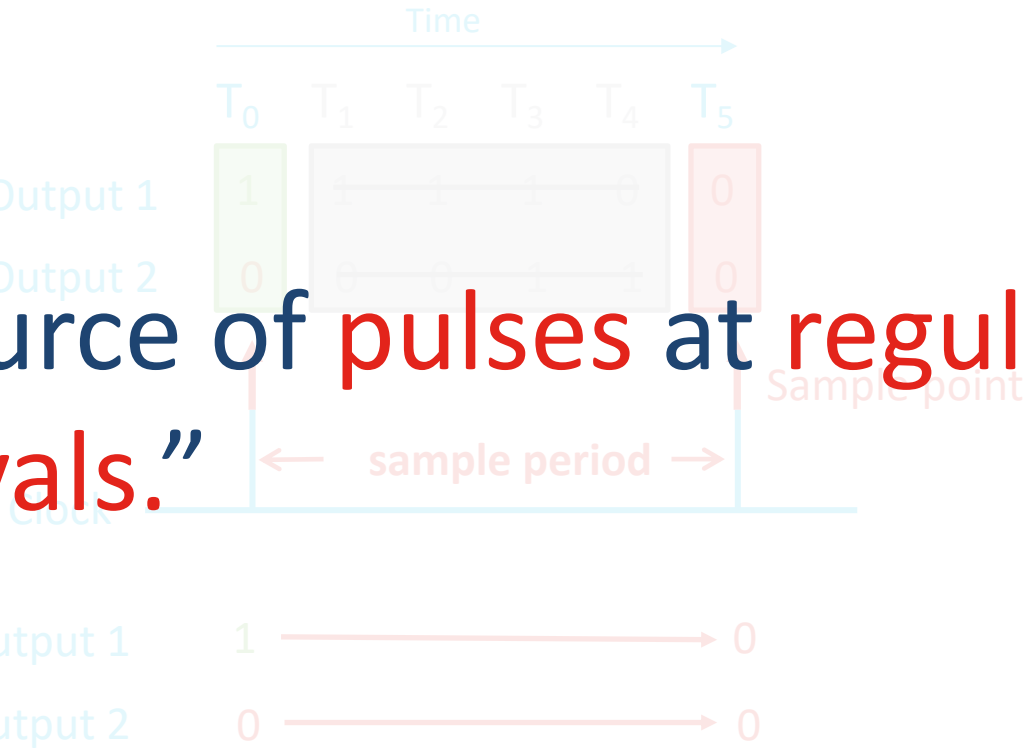


Sampled output 1 1 → 0

Sampled output 2 0 → 0

Clock

"A source of pulses at regular intervals."

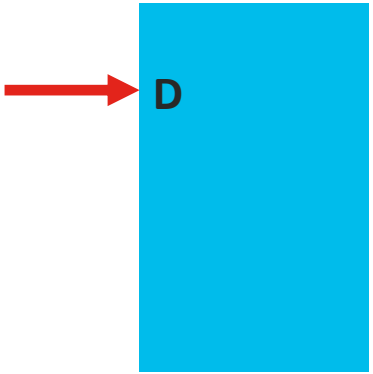


Clock

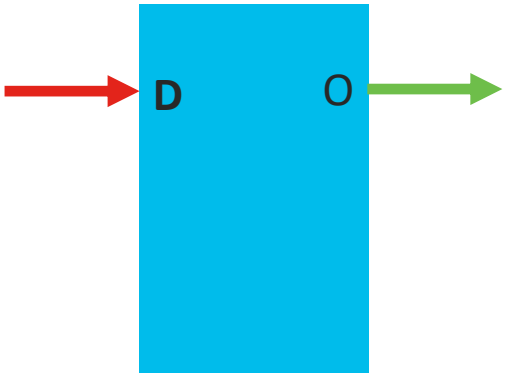
"A source of pulses at regular intervals. The clock speed is measured in Hz"


Get your flip-flops on

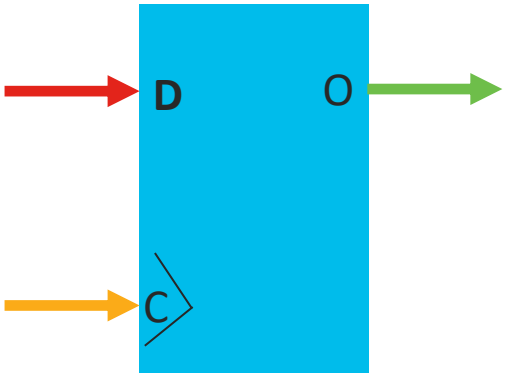
Flip-Flop Operation



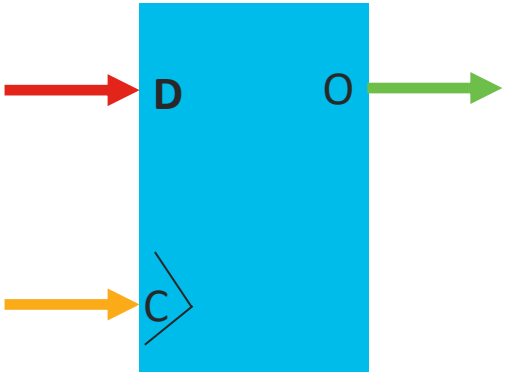
Flip-Flop Operation

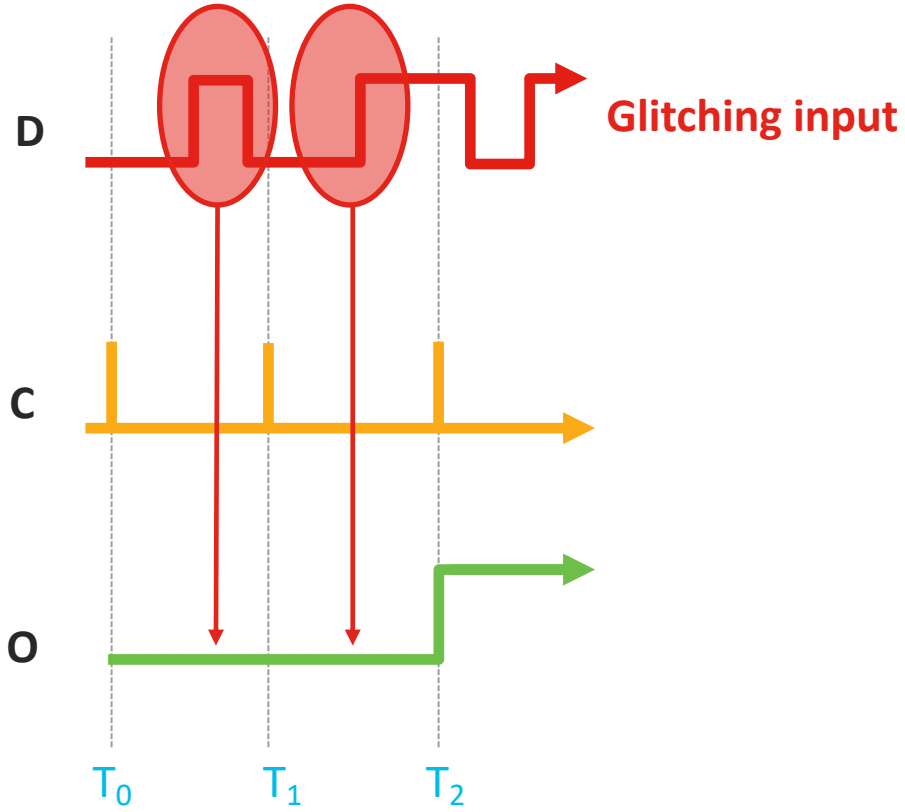


Flip-Flop Operation

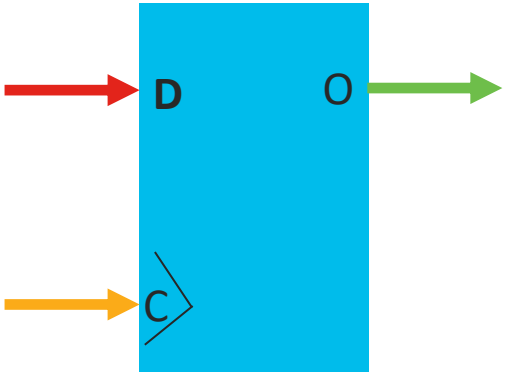


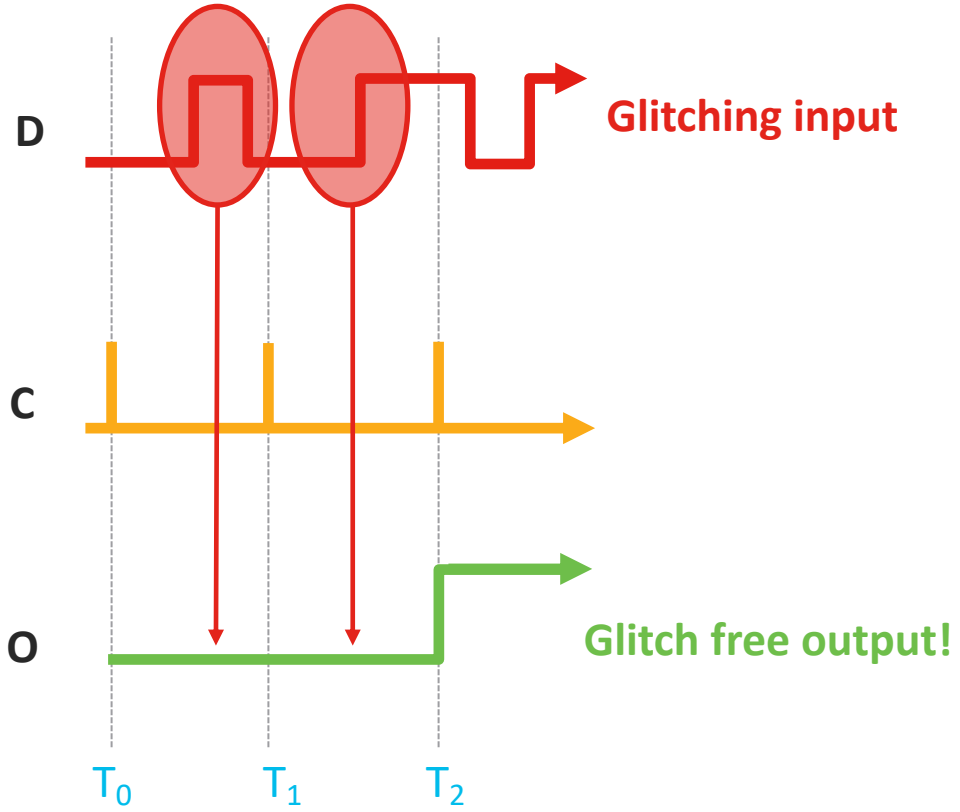
Flip-Flop Operation



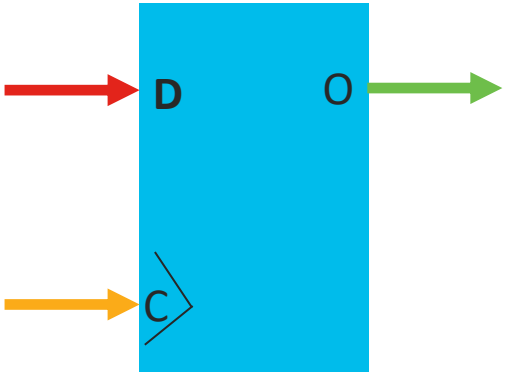


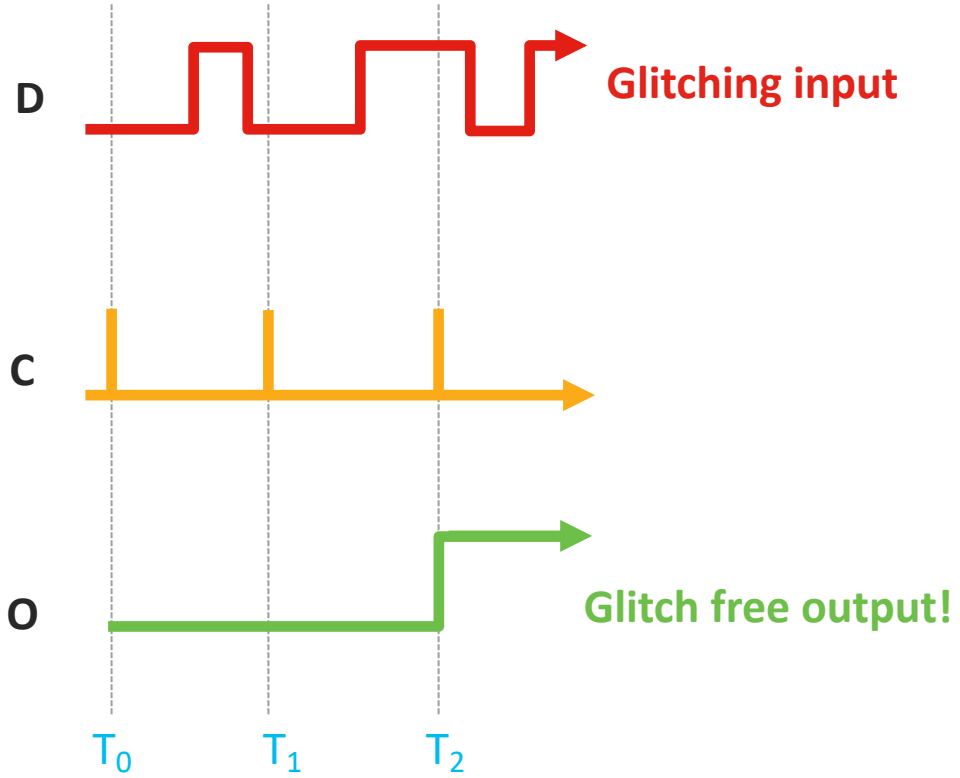
Flip-Flop Operation



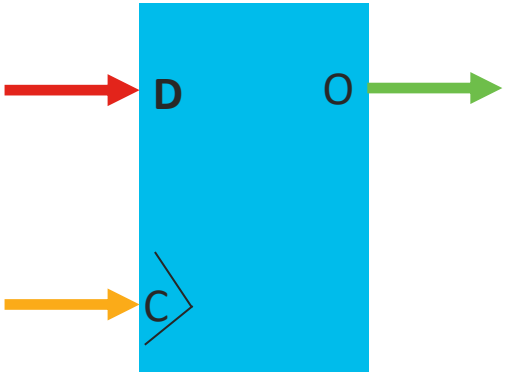


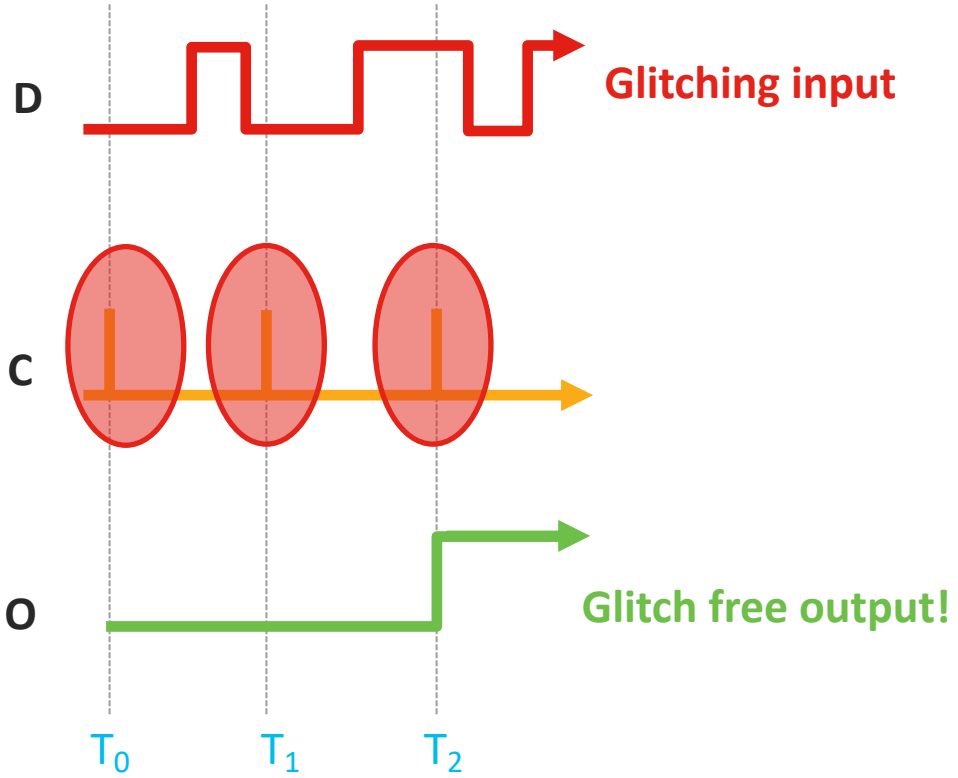
Flip-Flop Operation



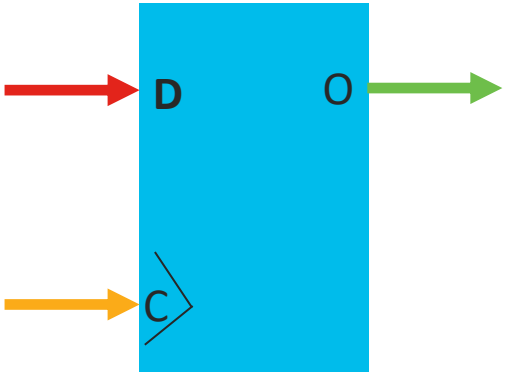


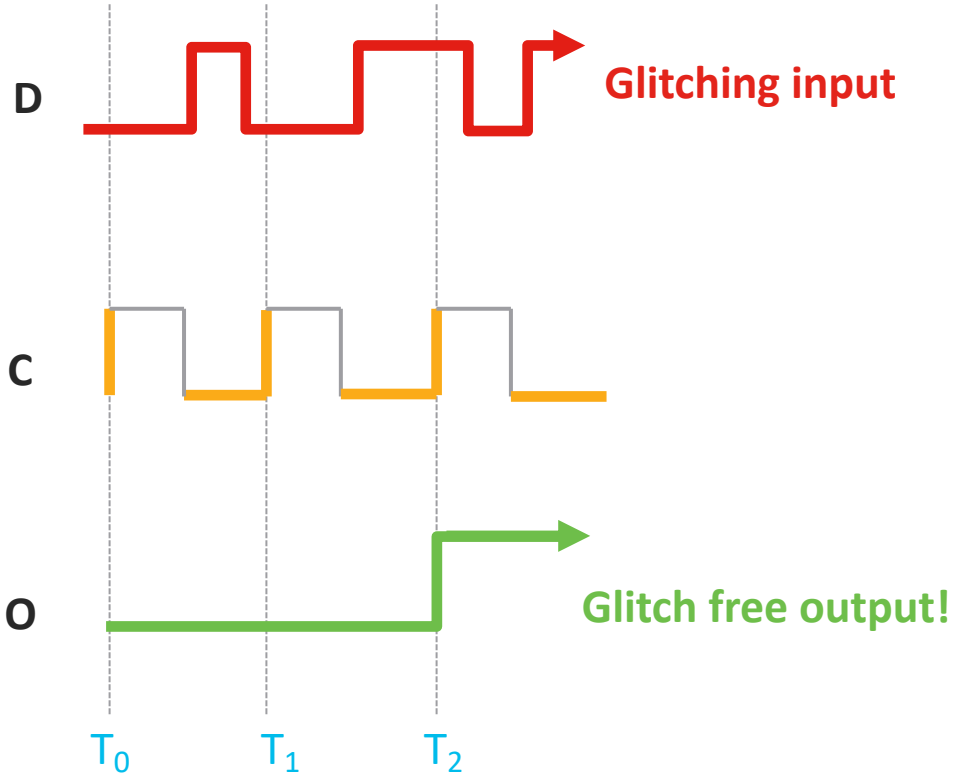
Flip-Flop Operation



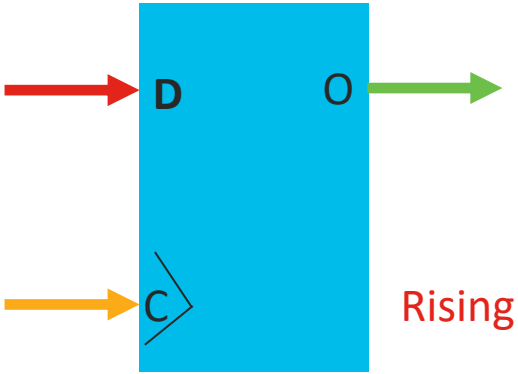


Flip-Flop Operation

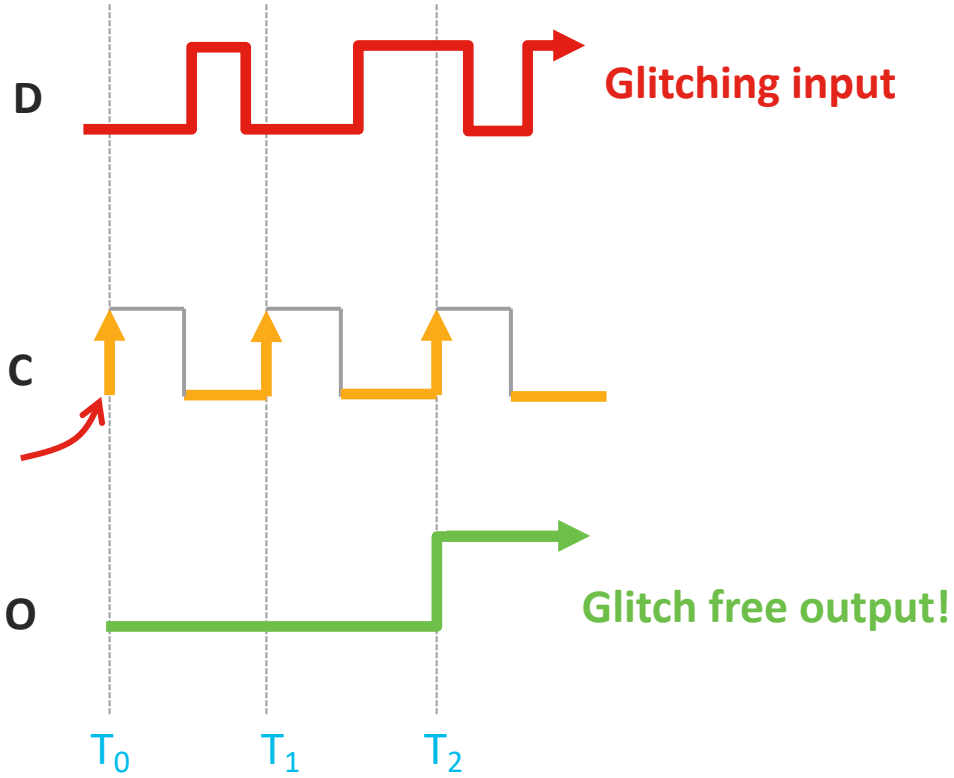




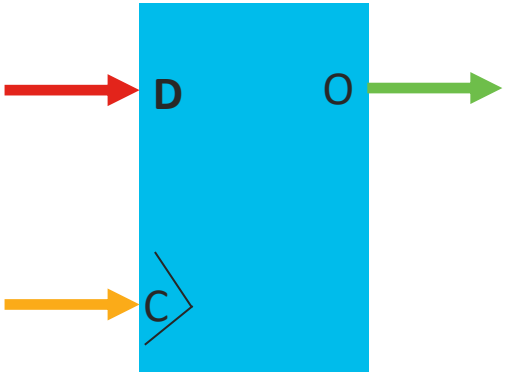
Flip-Flop Operation

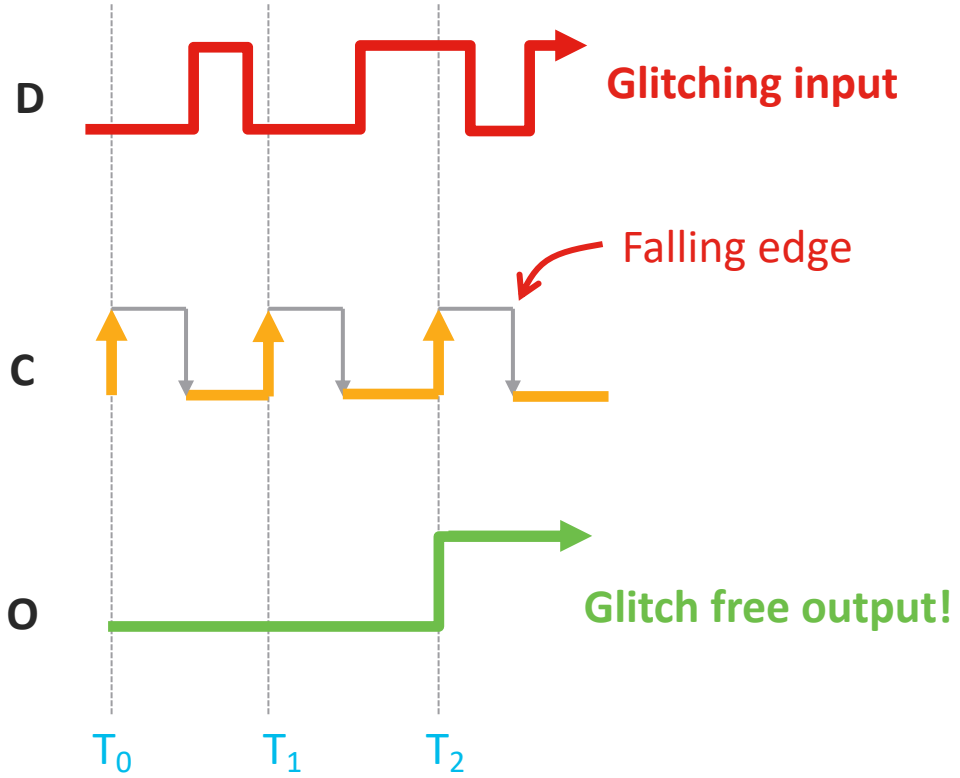


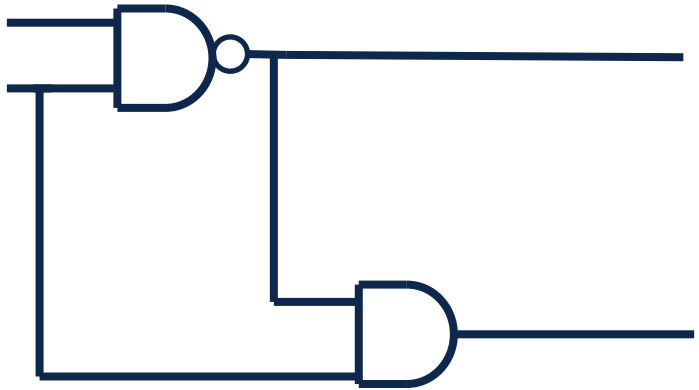
Rising edge

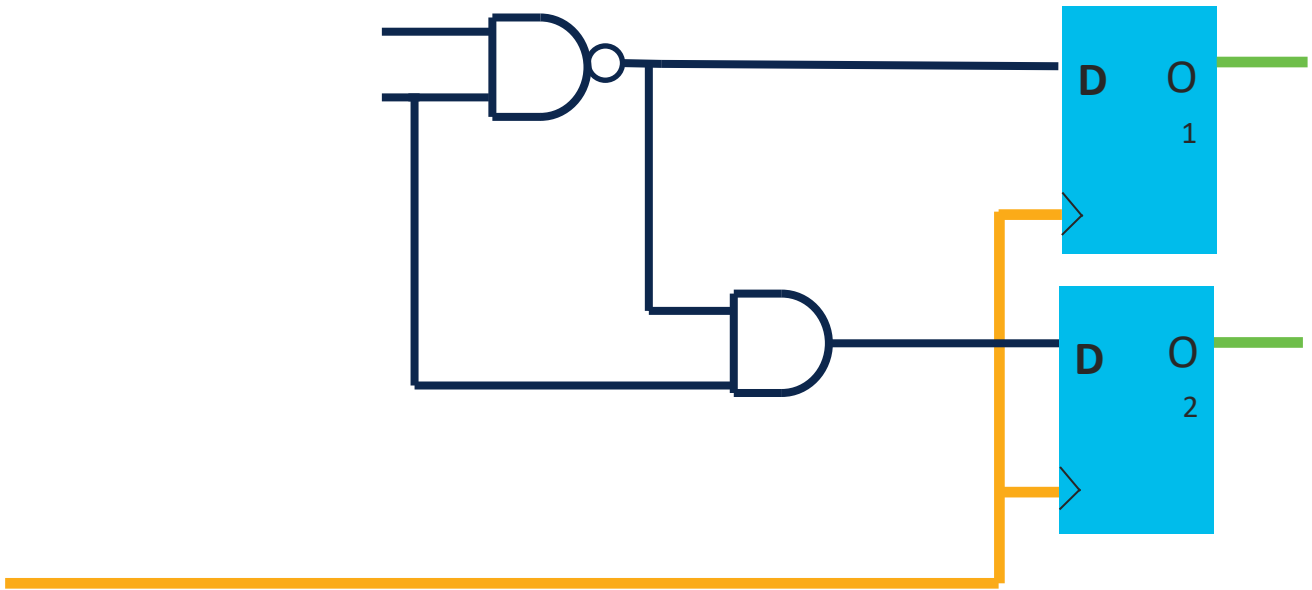


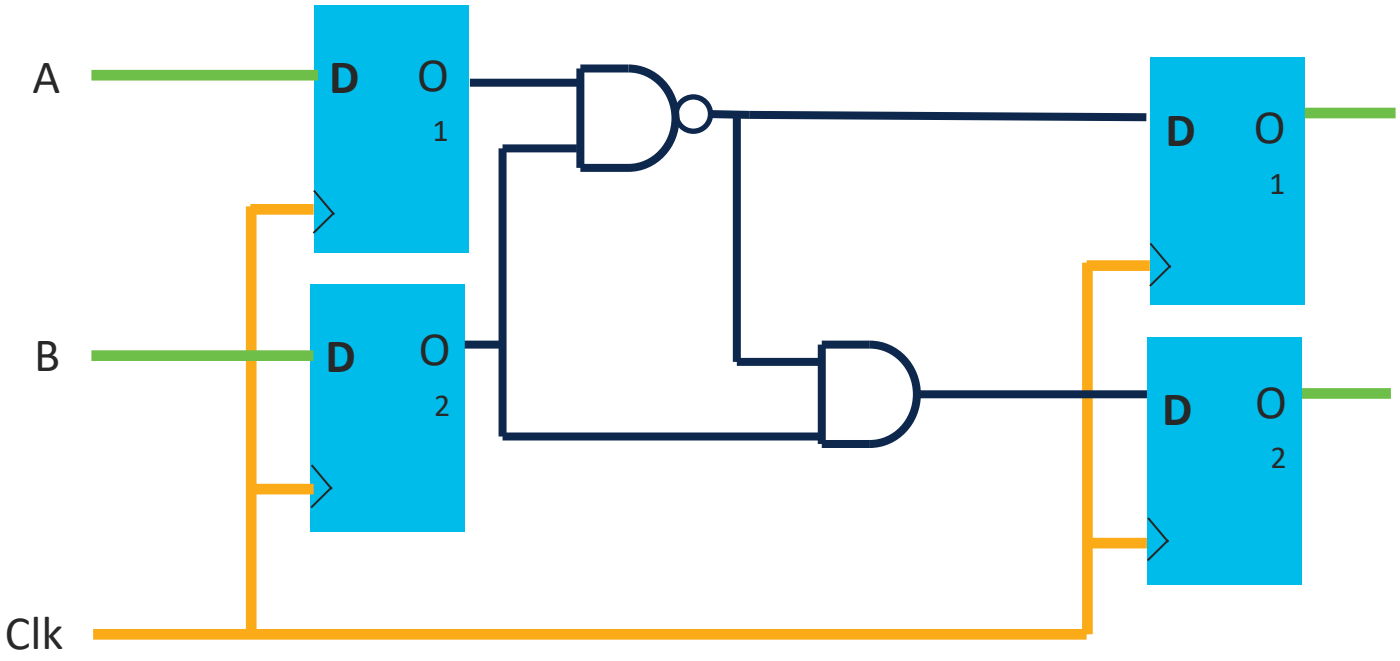
Flip-Flop Operation





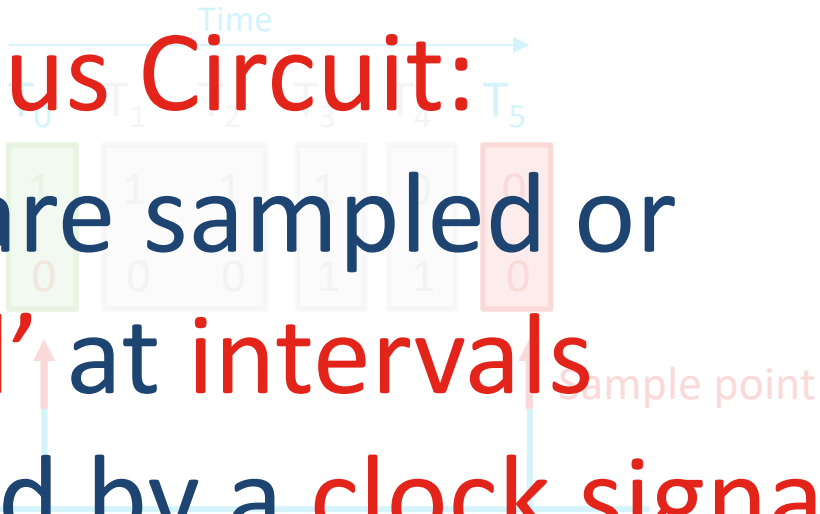



D
O
1
D
O
2


A
B
Clk
D
O
1
D
O
2
D
O
1
D
O
2

Synchronous Circuit:

"Outputs are sampled or 'registered' at intervals determined by a clock signal"



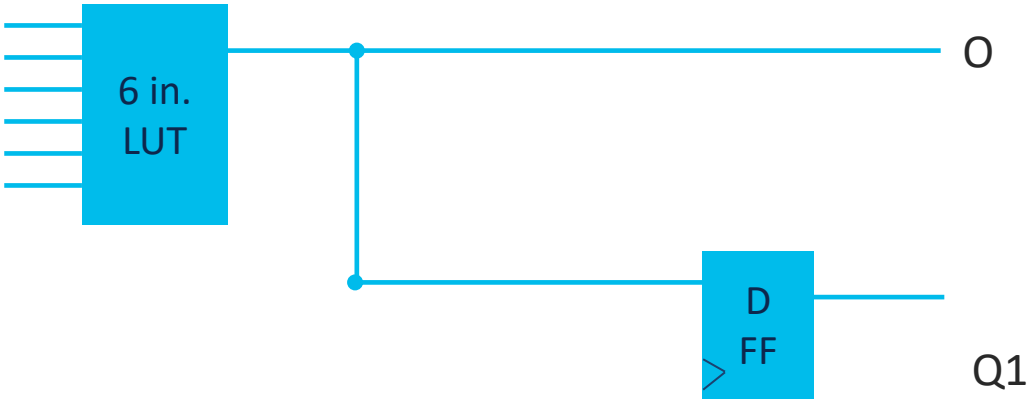
Sampled output 1 1 → 0

Sampled output 2 0 → 0

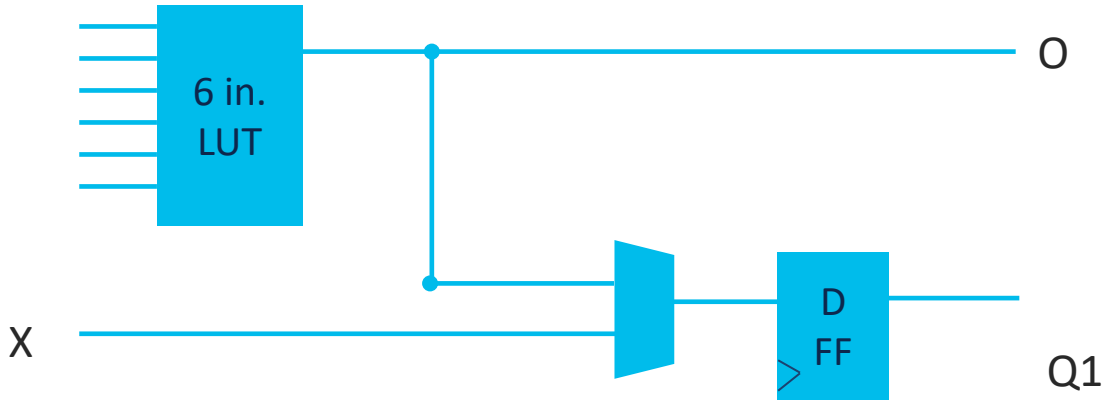
Putting all the pieces together

Xilinx Ultrascale+ FPGA Logic Slice

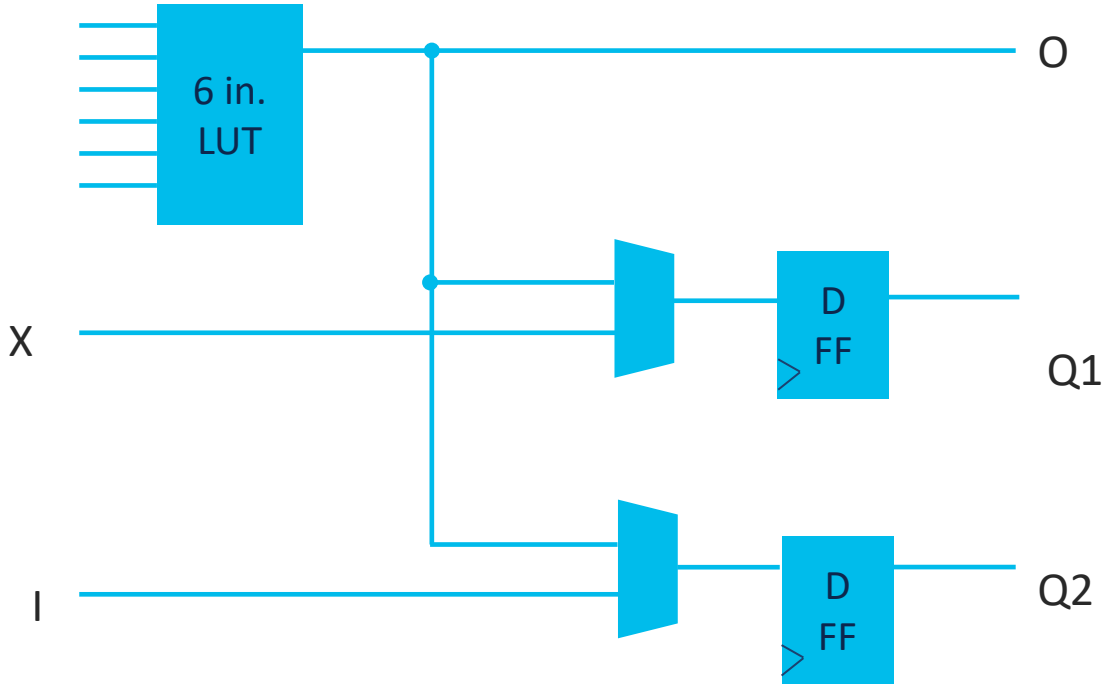
Xilinx Ultrascale+ FPGA Logic Slice



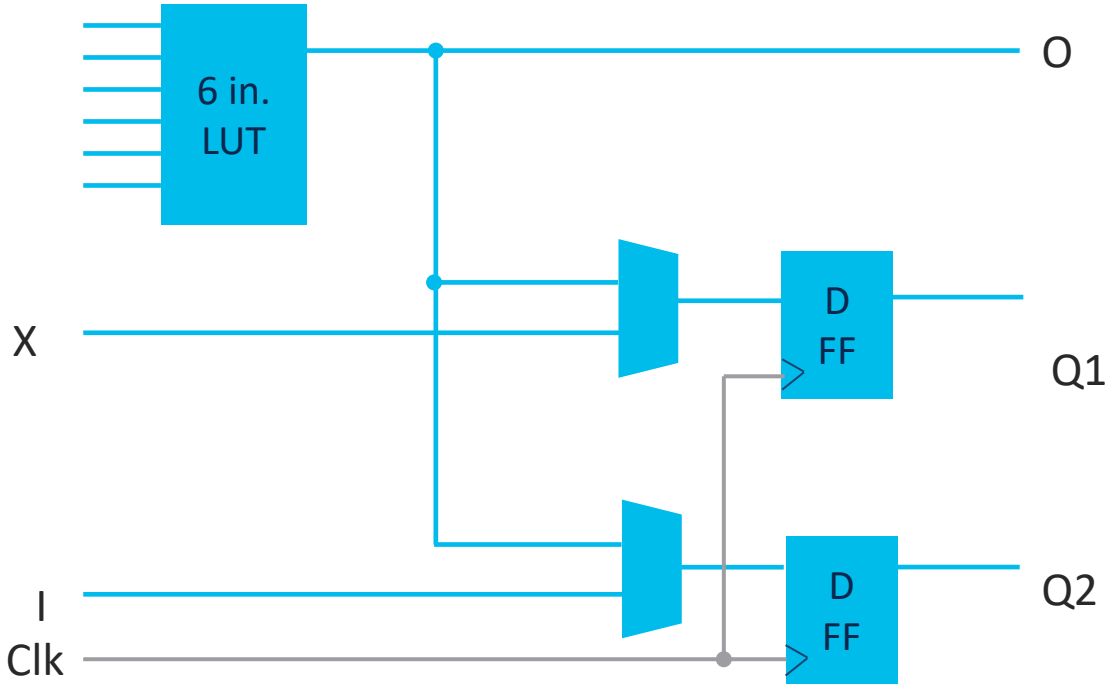
Xilinx Ultrascale+ FPGA Logic Slice



Xilinx Ultrascale+ FPGA Logic Slice

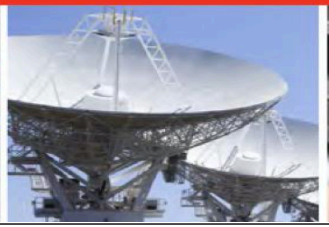


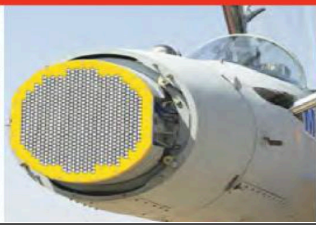
Xilinx Ultrascale+ FPGA Logic Slice



UltraScale+ FPGAs
Product Tables and Product Selection Guide



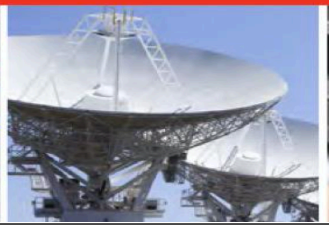


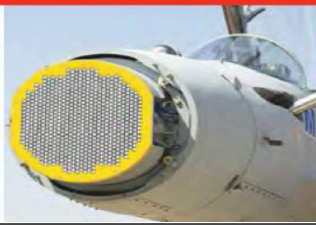


Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

UltraScale+ FPGAs
Product Tables and Product Selection Guide

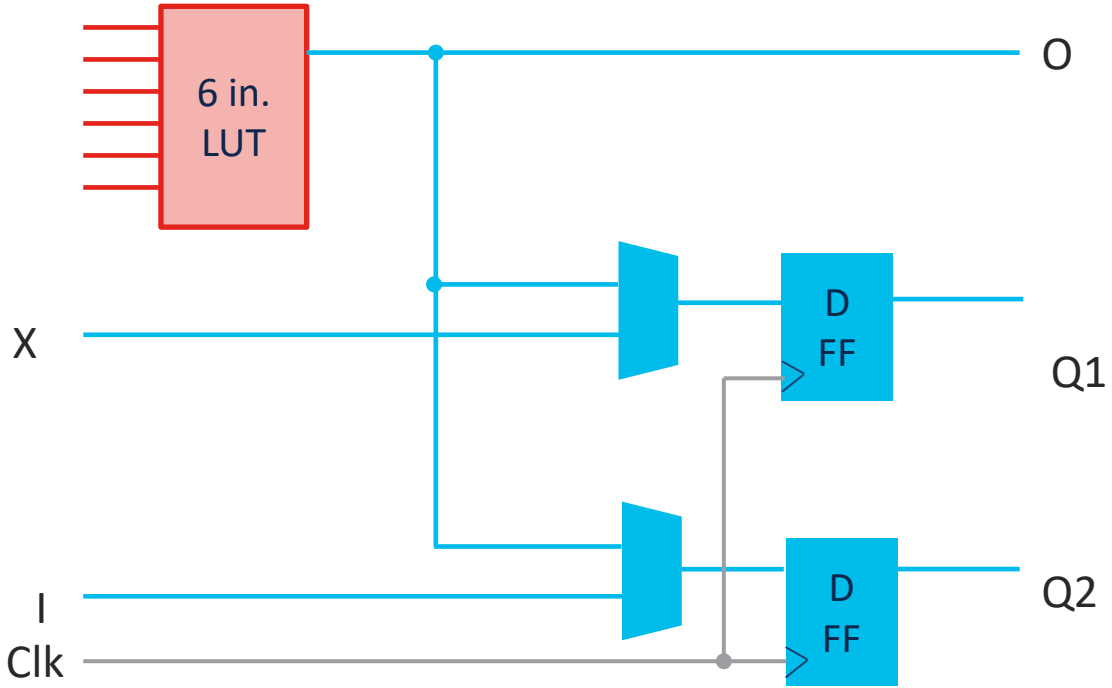






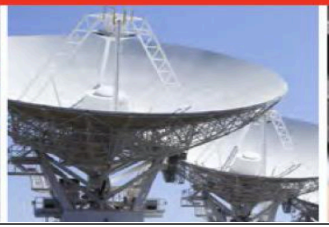
Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

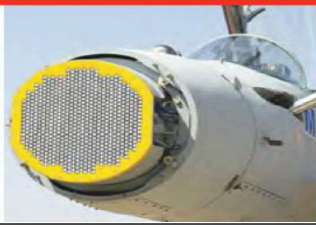
Xilinx Ultrascale+ FPGA Logic Slice



UltraScale+ FPGAs
Product Tables and Product Selection Guide



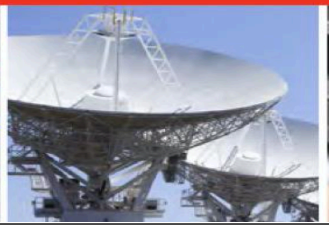


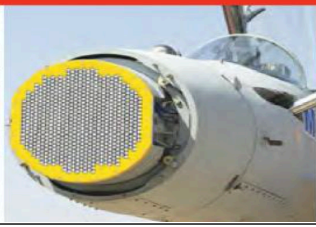


Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

UltraScale+ FPGAs
Product Tables and Product Selection Guide



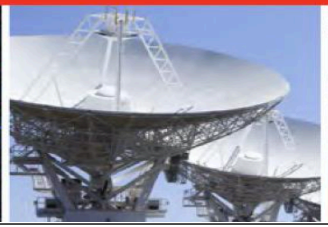




Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394,000	601,000	788,000	1,182,000	1,296,000	1,728,000	4,086,000
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

UltraScale+ FPGAs
Product Tables and Product Selection Guide





Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

>1M LUTs!!

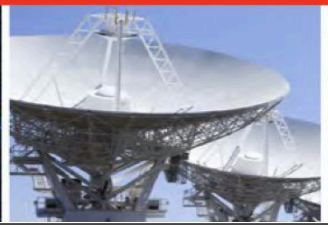
UltraScale+ FPGAs
Product Tables and Product Selection Guide

Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

SLC?

UltraScale+ FPGAs
Product Tables and Product Selection Guide



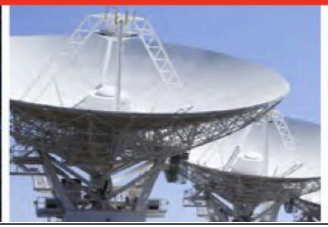


Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	18.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

6 input LUTs

UltraScale+ FPGAs
Product Tables and Product Selection Guide





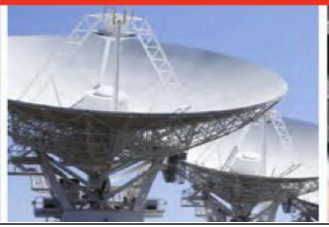
Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOPS)	7.1	10.8	14.2	21.3	28.7	38.3	18.1
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

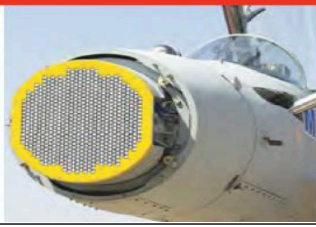
~4 input LUTs

6 input LUTs

UltraScale+ FPGAs
Product Tables and Product Selection Guide

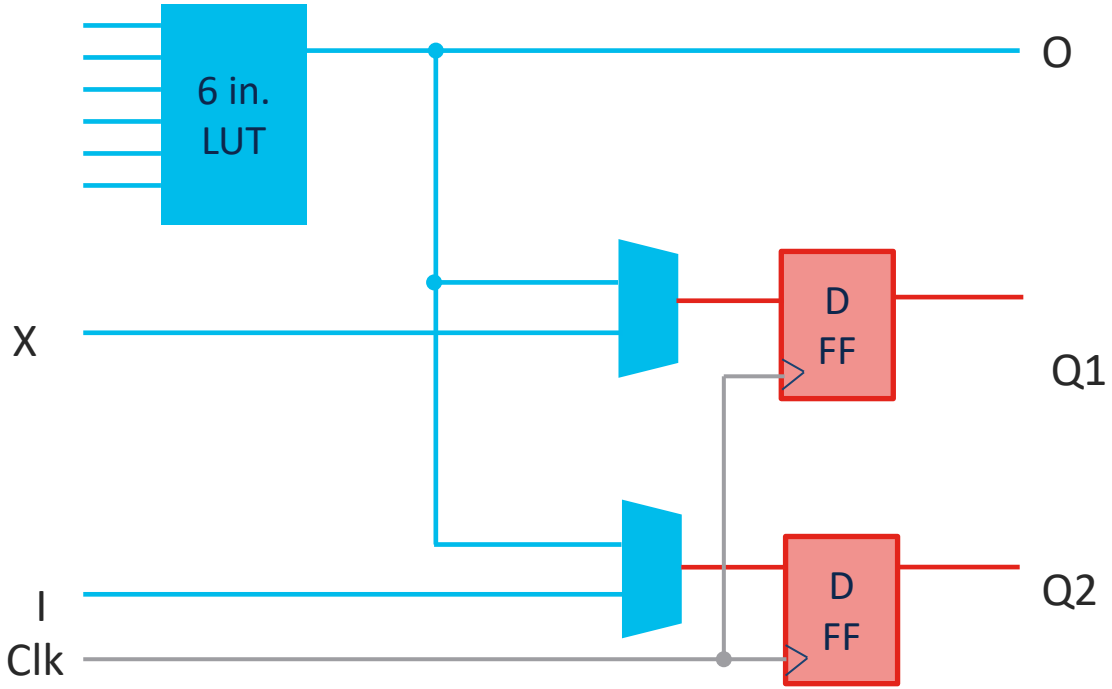






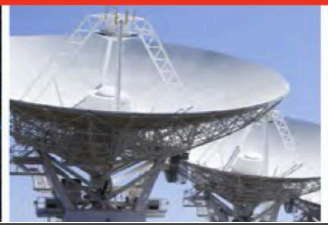
Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

Xilinx Ultrascale+ FPGA Logic Slice



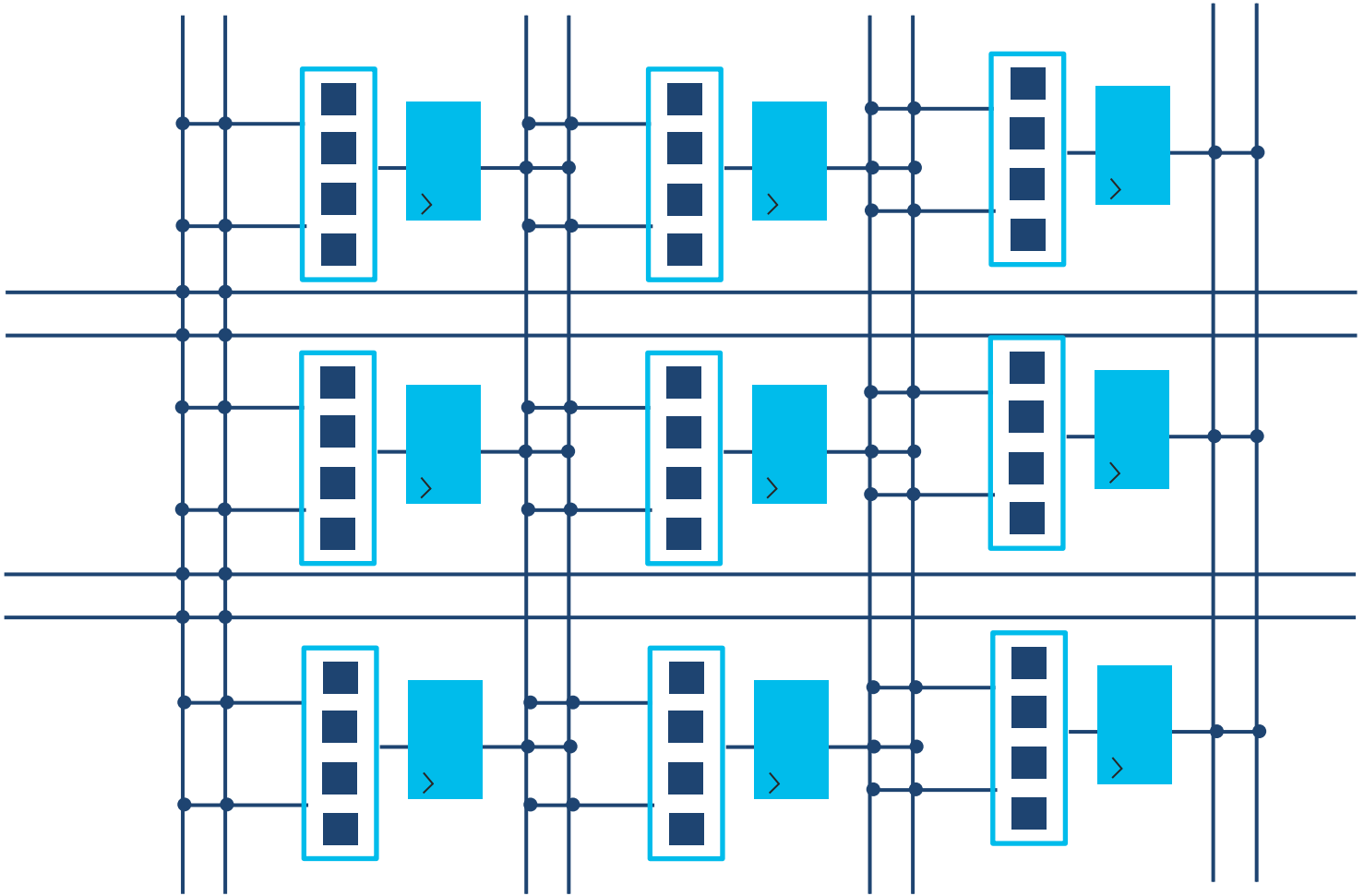
UltraScale+ FPGAs
Product Tables and Product Selection Guide

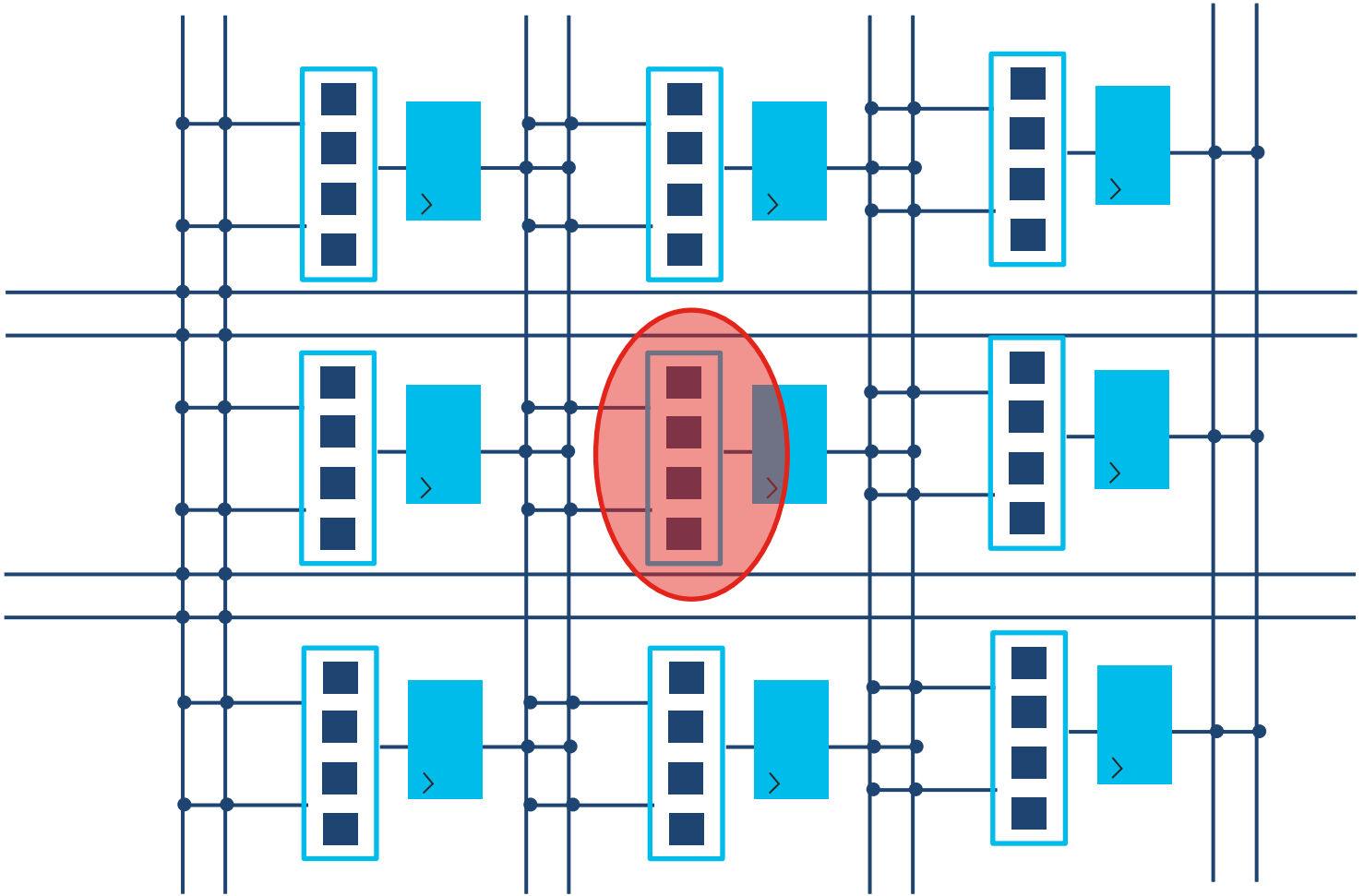


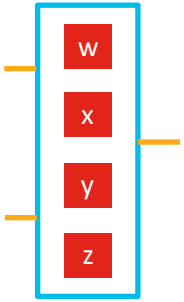


Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	[illegible]	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	[illegible]	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	[illegible]	–	–	8
150G Interlaken	3	4	6	[illegible]	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

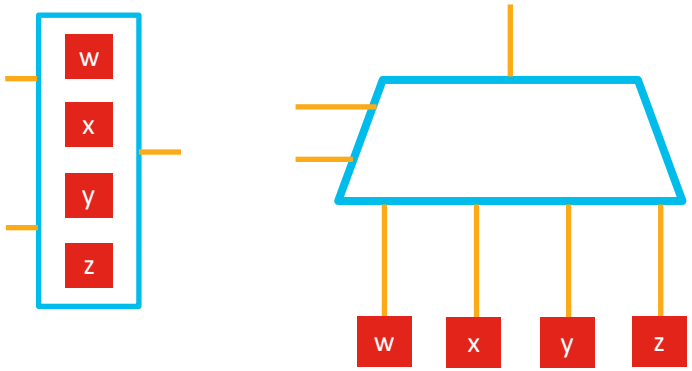

FPGA’S Naughty little secret





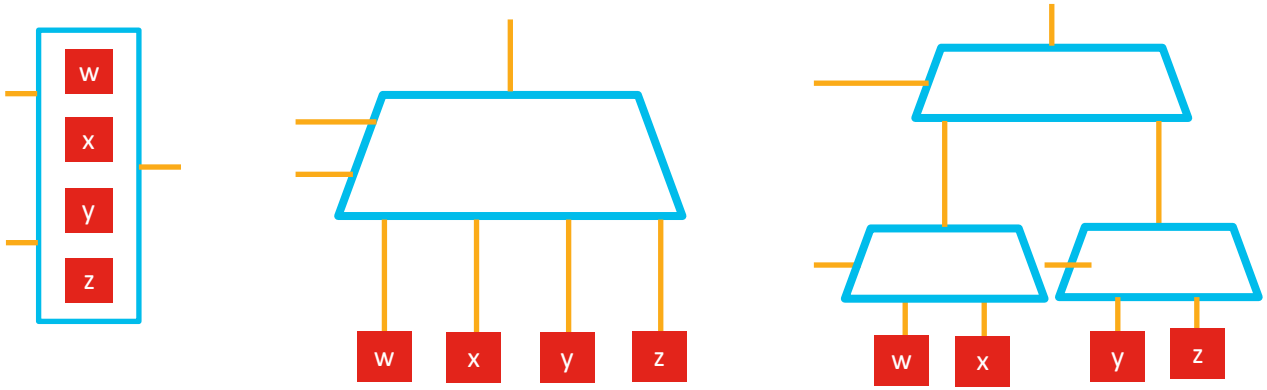


2 input LUT:



2 input LUT:

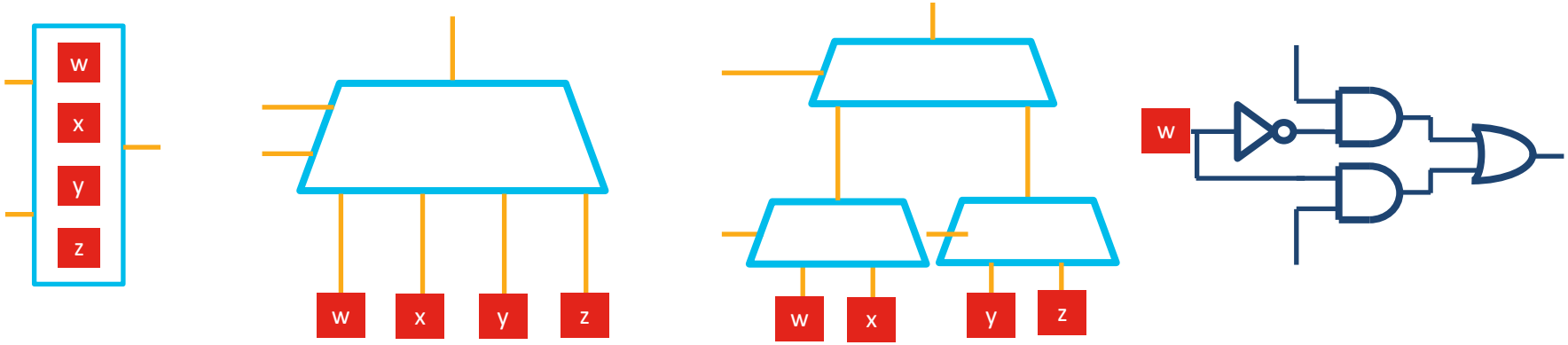
–1x 4-way mux + 4x 1bit memory



2 input LUT:

–1x 4-way mux + 4x 1bit memory

– 3x 2-way mux + 4x 1bit memory

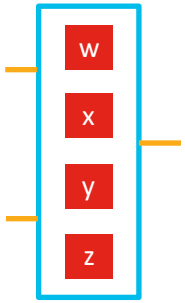


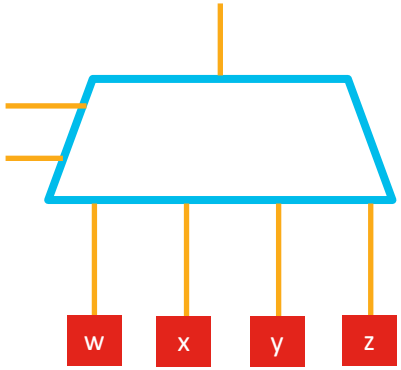
2 input LUT:

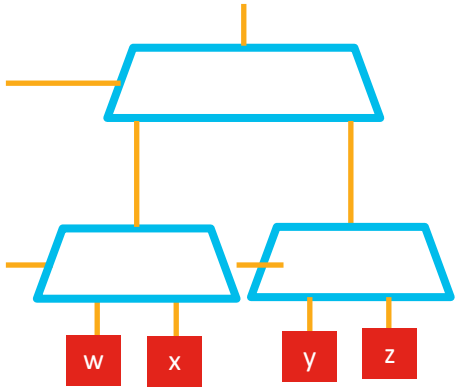
–1x 4-way mux + 4x 1bit memory

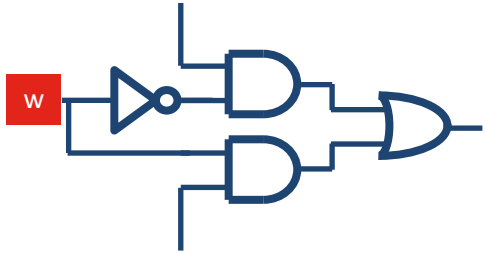
– 3x 2-way mux + 4x 1bit memory

- 3x 4 gates + 4x 1 bit memory









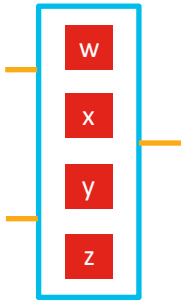
2 input LUT:

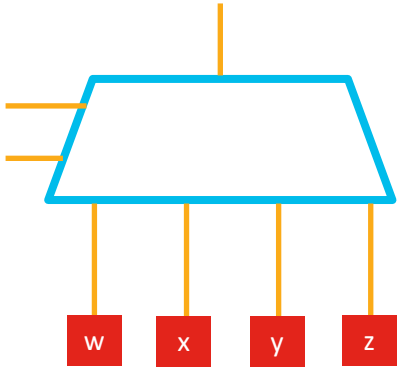
–1x 4-way mux + 4x 1bit memory

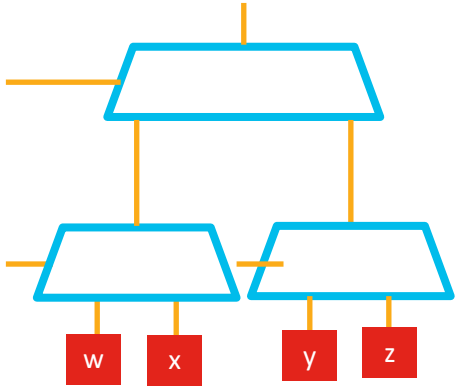
– 3x 2-way mux + 4x 1bit memory

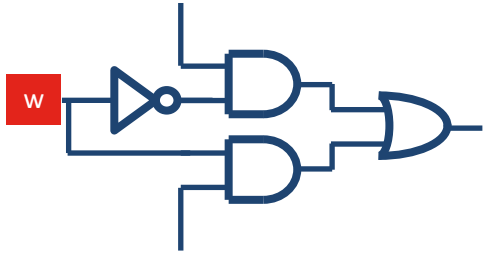
- 3x 4 gates + 4x 1 bit memory

- 3x (7) + 4x (4) transistors



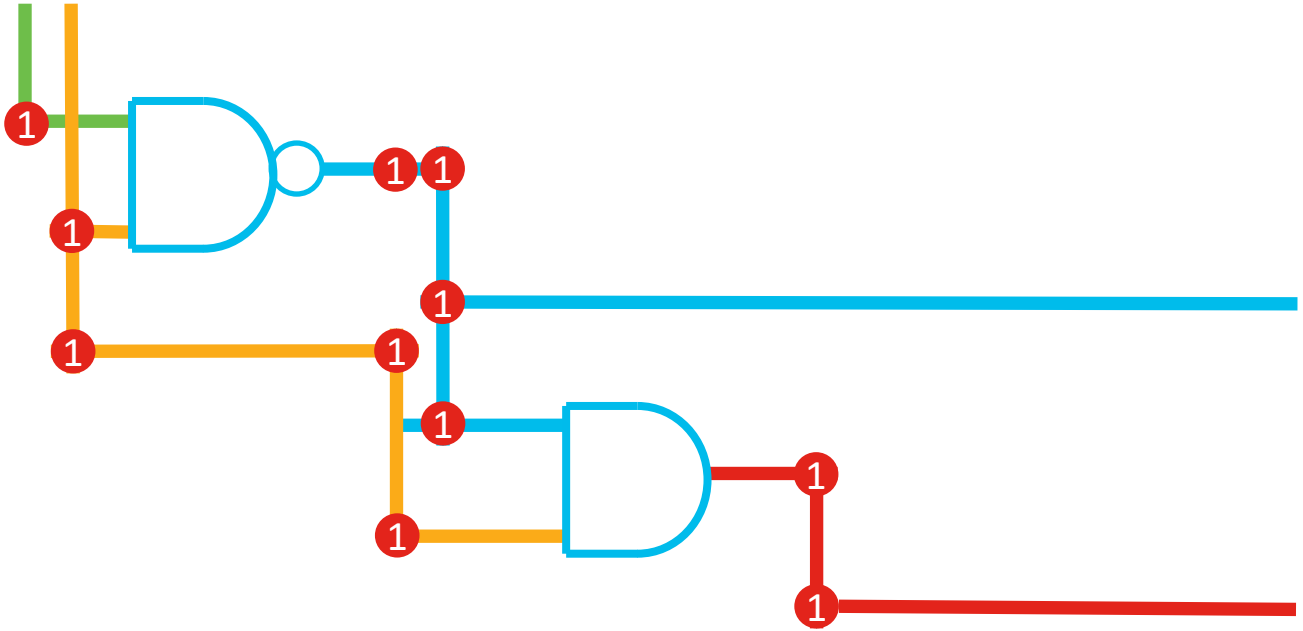




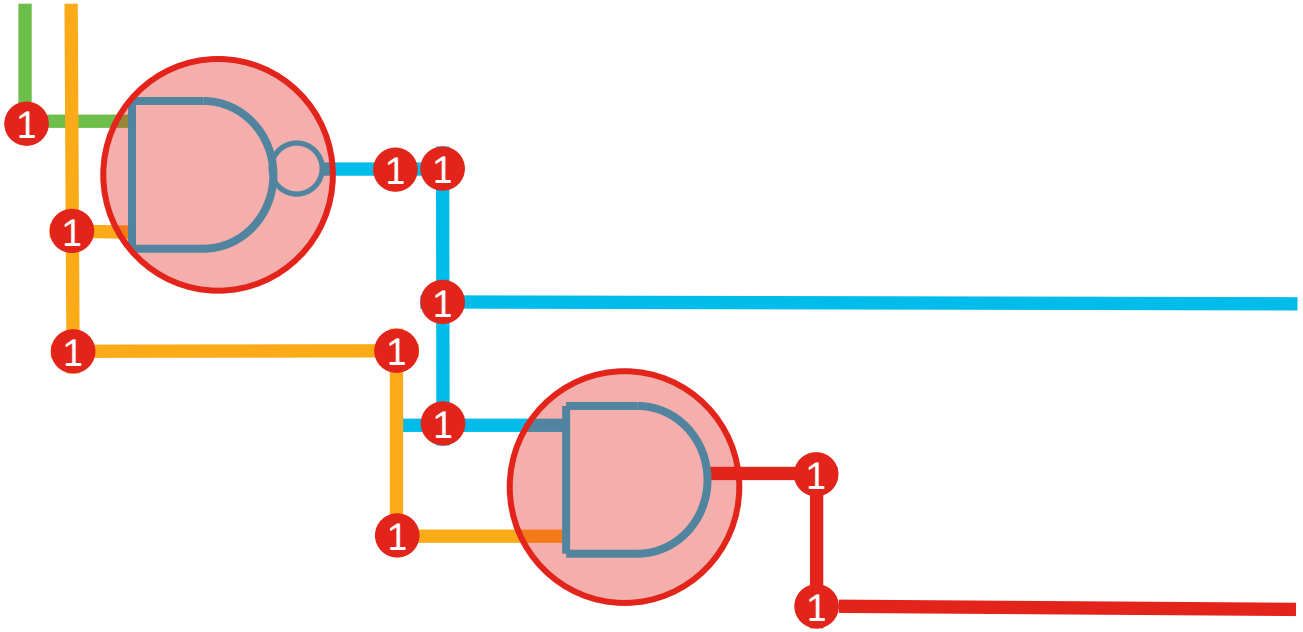


2 input LUT:

- 37 transistors!

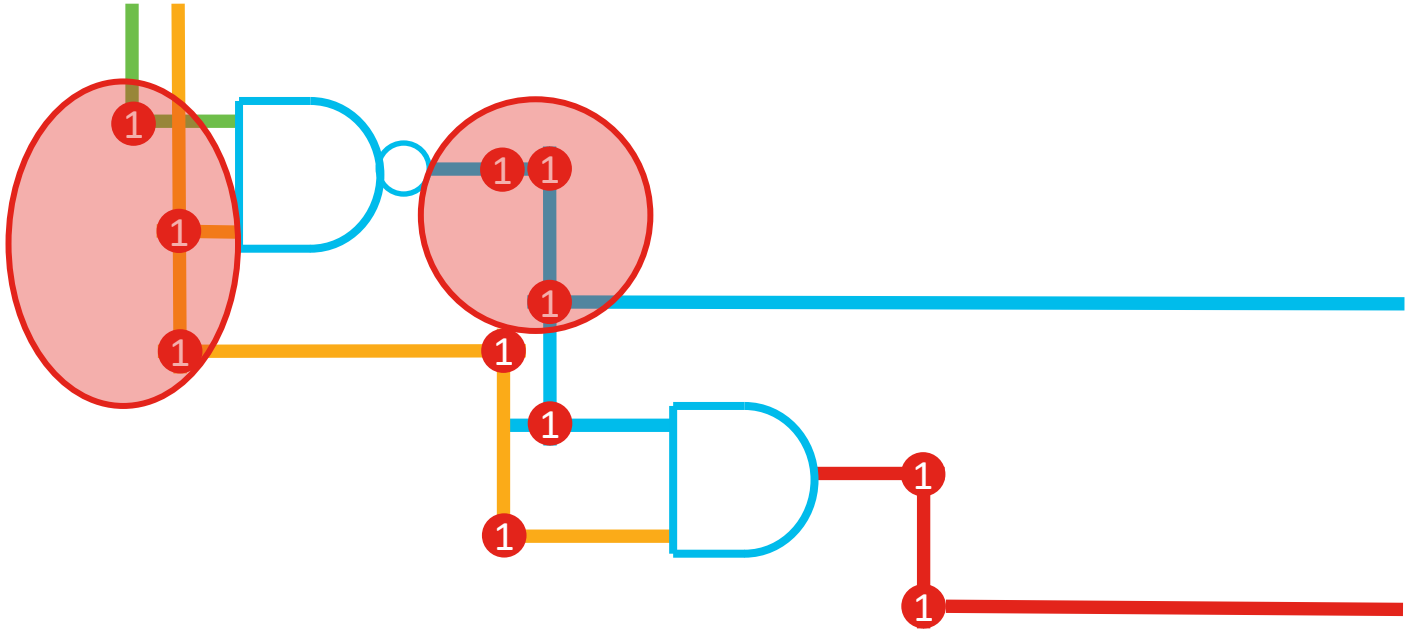


2 logic gate circuit



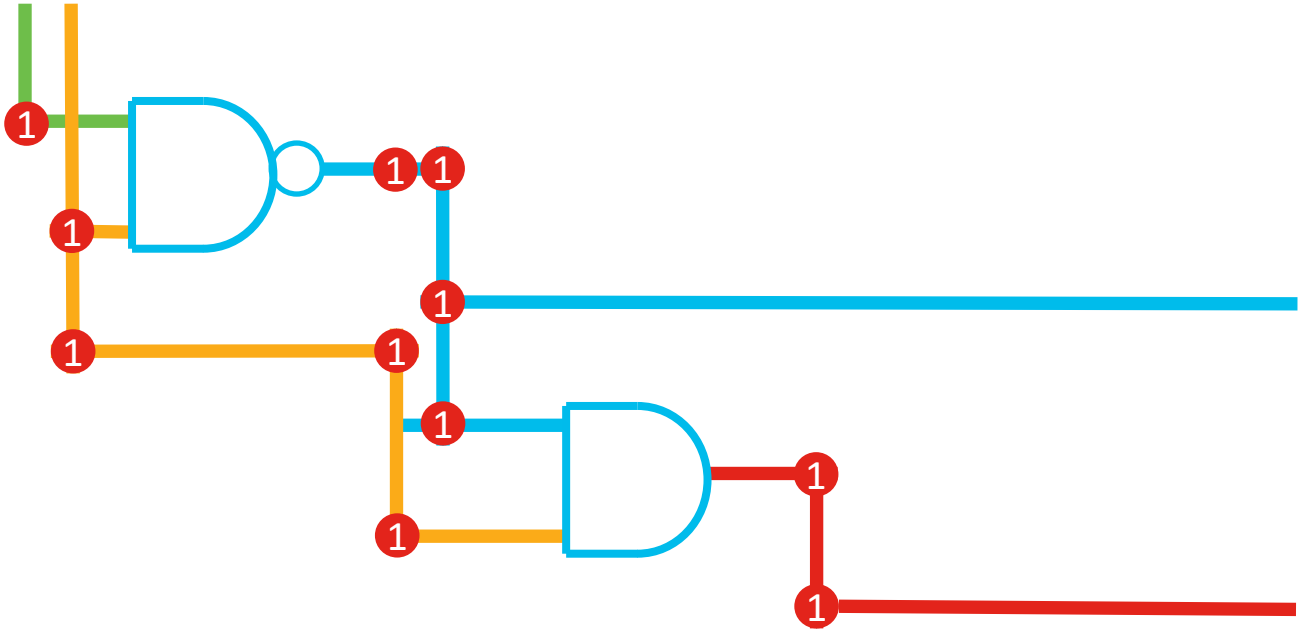
2 logic gate circuit

- 2x 2 input LUTs



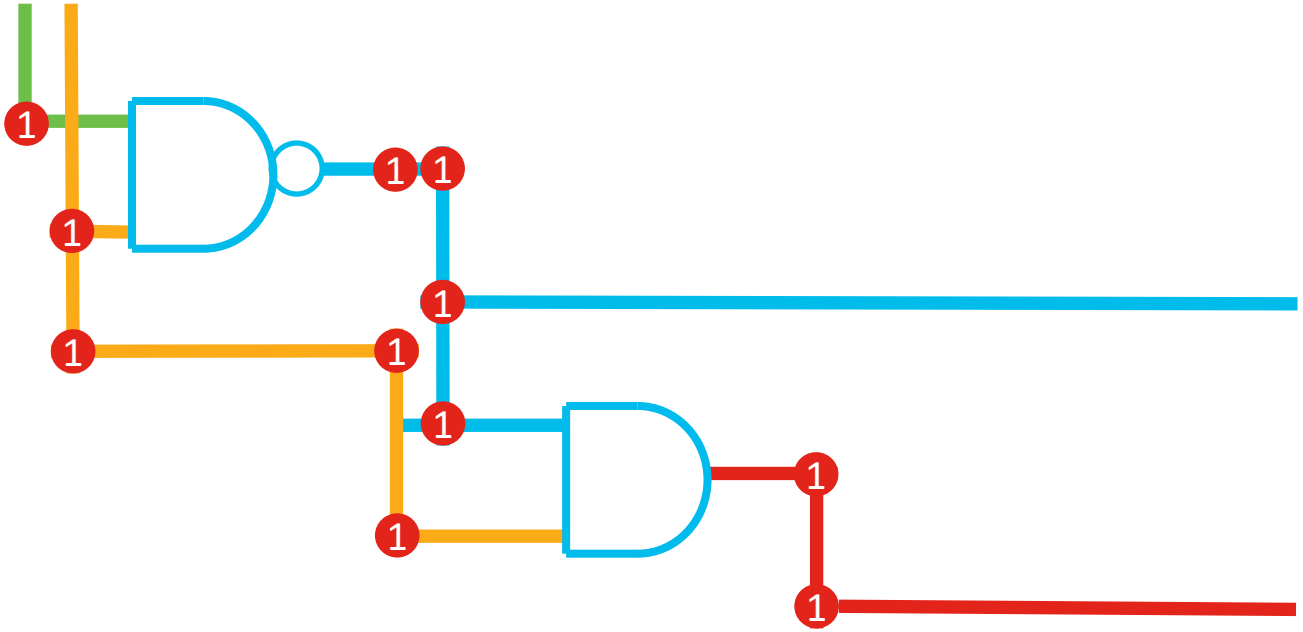
2 logic gate circuit

- 2x 2 input LUTs
- 11 programmable connections



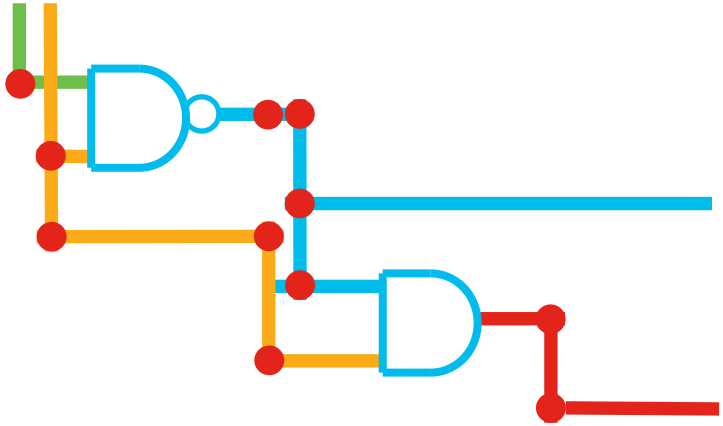
2 logic gate circuit

- 2x 2 input LUTs +11 connections
- 2x 37 + 11x4 transistors!!!



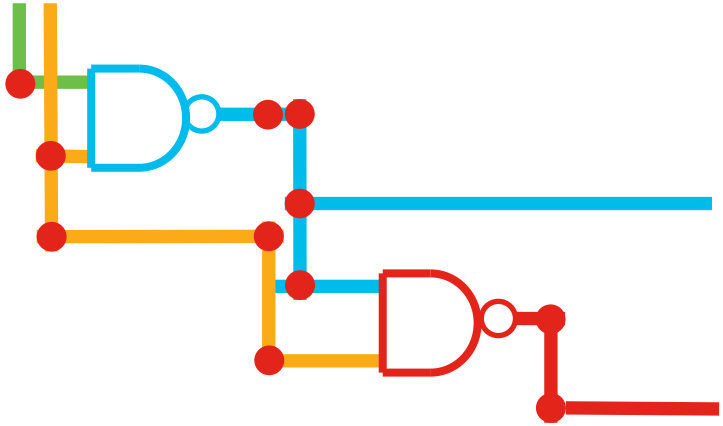
2 logic gate circuit

-118 transistors



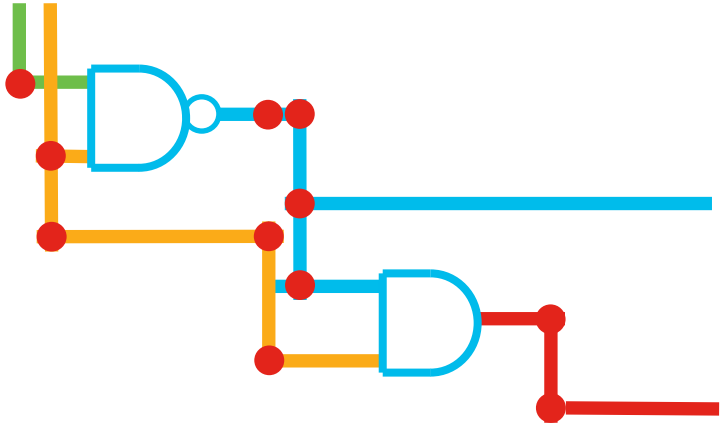
FPGA

- ~118 transistors



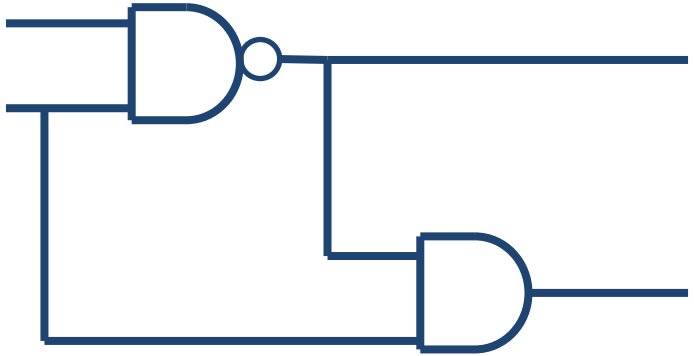
FPGA

- ~118 transistors
- Fully flexible!



FPGA

- ~118 transistors
- Fully flexible!



Direct “hard” (ASIC)

- ~4 transistors!!!
- fixed config

FPGAs:

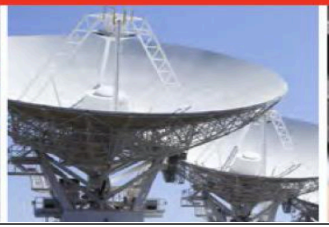
“Offer high flexibility in exchange for low efficiency”

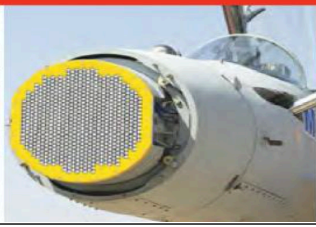
FPGA
~60 transistors!!!
Flexible

Direct “hard” (ASIC)
~4 transistors!!!
Fixed config

UltraScale+ FPGAs
Product Tables and Product Selection Guide



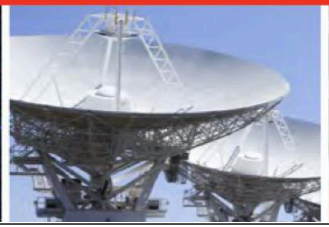




Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

UltraScale+ FPGAs
Product Tables and Product Selection Guide



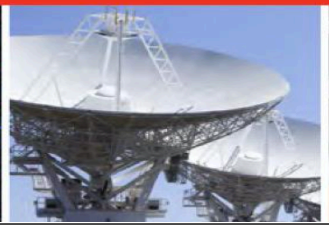


Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	34.6	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.3	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

Efficient “hard” function implementations

UltraScale+ FPGAs
Product Tables and Product Selection Guide



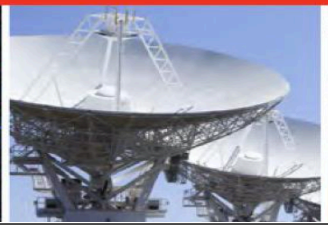


Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)							
Total Block RAM (Mb)							
UltraRAM (Mb)							
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

Different types/density of memory (RAM)

UltraScale+ FPGAs
Product Tables and Product Selection Guide



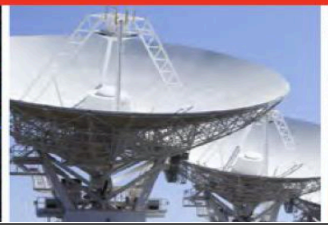


Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)							0
DSP Slices							40
Peak INT8 DSP (TOP/s)							4
PCIe® Gen3 x16							
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	0
Max. Single-Ended HP I/Os	520	832	832	832	624	832	1,976
Max. Single-Ended HD I/Os	0	0	0	0	0	0	96
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	80
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

Dedicated math functions

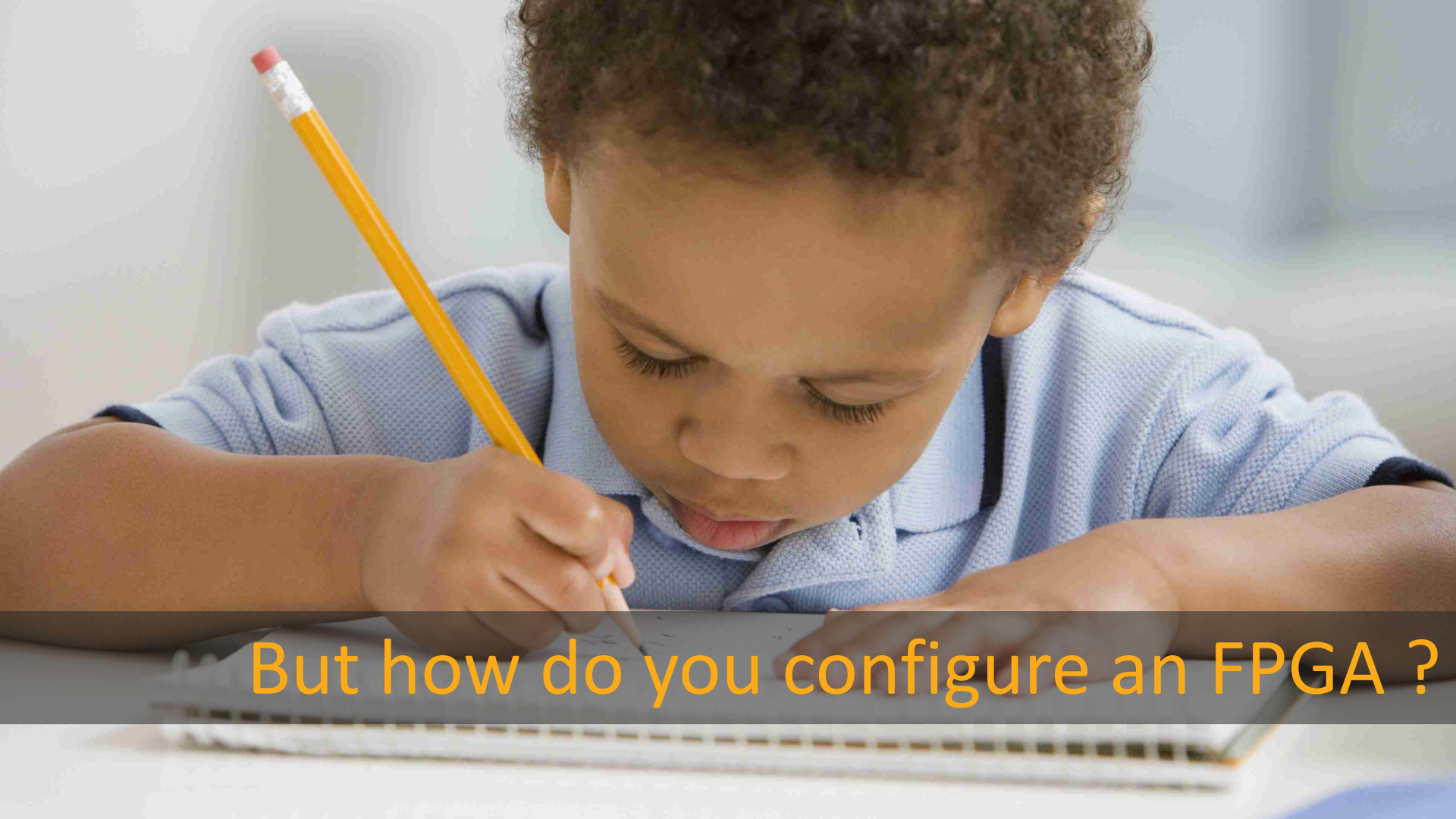
UltraScale+ FPGAs
Product Tables and Product Selection Guide





Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU19P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	8,938
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	8,172
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	4,086
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	58.4
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	75.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	90.0
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	3,840
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	10.4
PCIe® Gen3 x16	2	4	4	6	3	4	0
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	8
150G Interlaken	3	4	6	9	6	8	0
100G Ethernet w/ KR4 RS-FEC							
Max. Single-Ended HP I/Os							
Max. Single-Ended HD I/Os							
GTY 32.75Gb/s Transceivers							
GTM 58Gb/s PAM4 Transceivers	–	–	–	–	–	–	–
100G / 50G KP4 FEC	–	–	–	–	–	–	–
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–

External connectivity (PCIe, Ethernet, GPIO)


But how do you configure an FPGA ?

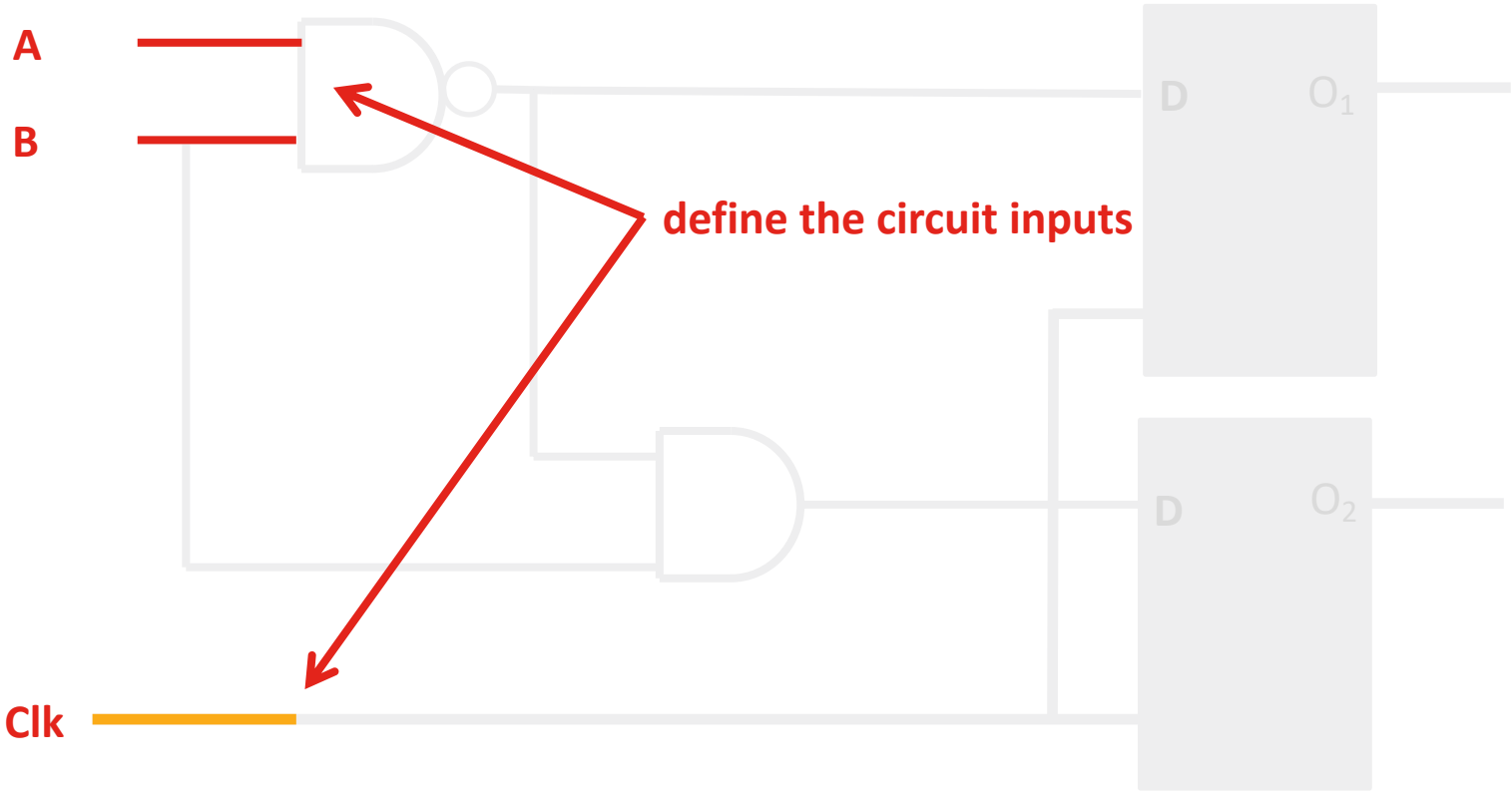
Verilog HDL Code Example

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

Verilog HDL Code Example

```
input A, B, clk;
```

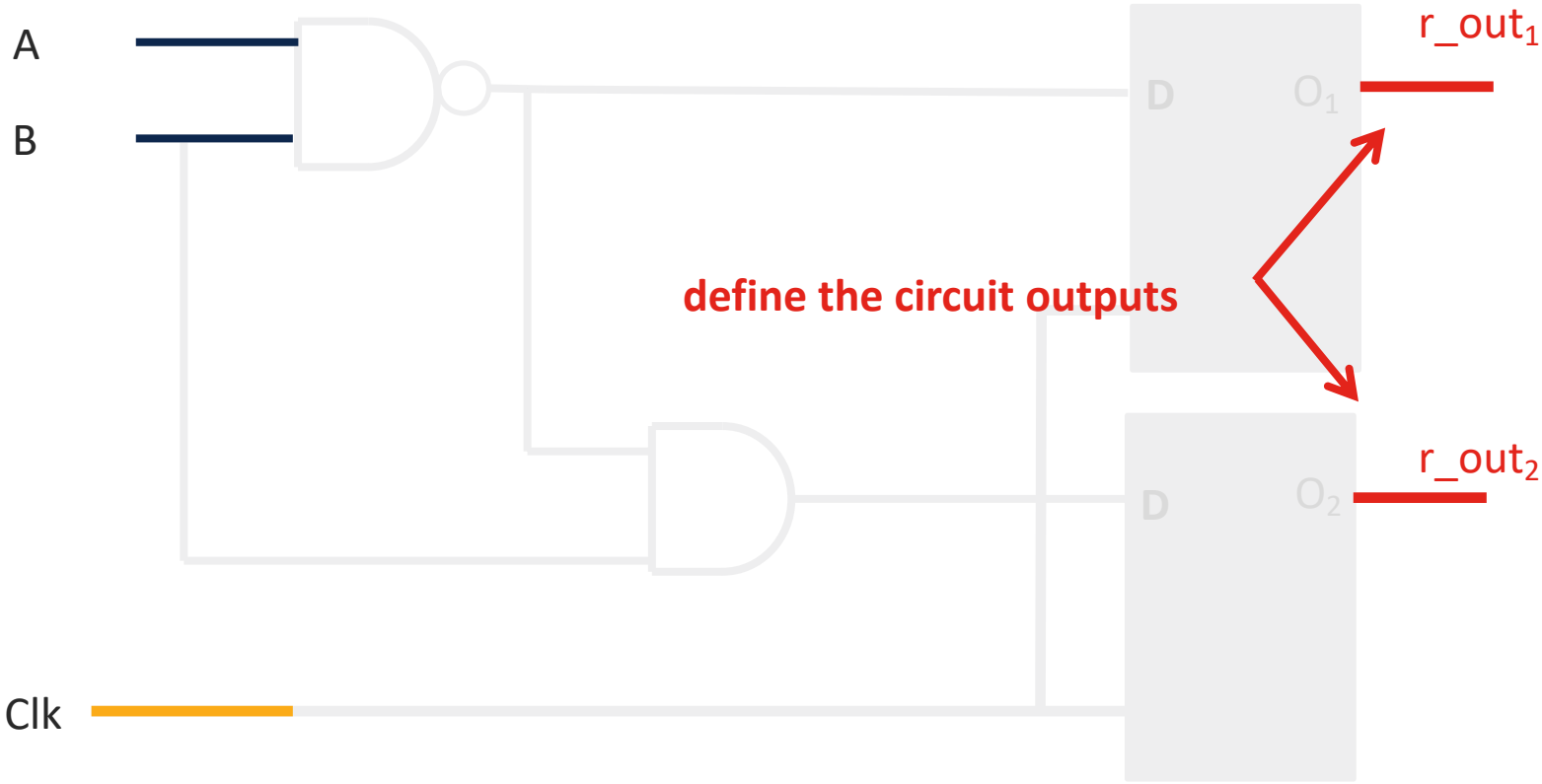
define the circuit inputs



Verilog HDL Code Example

```
input A, B, clk;
output r_out1, r_out2;
```

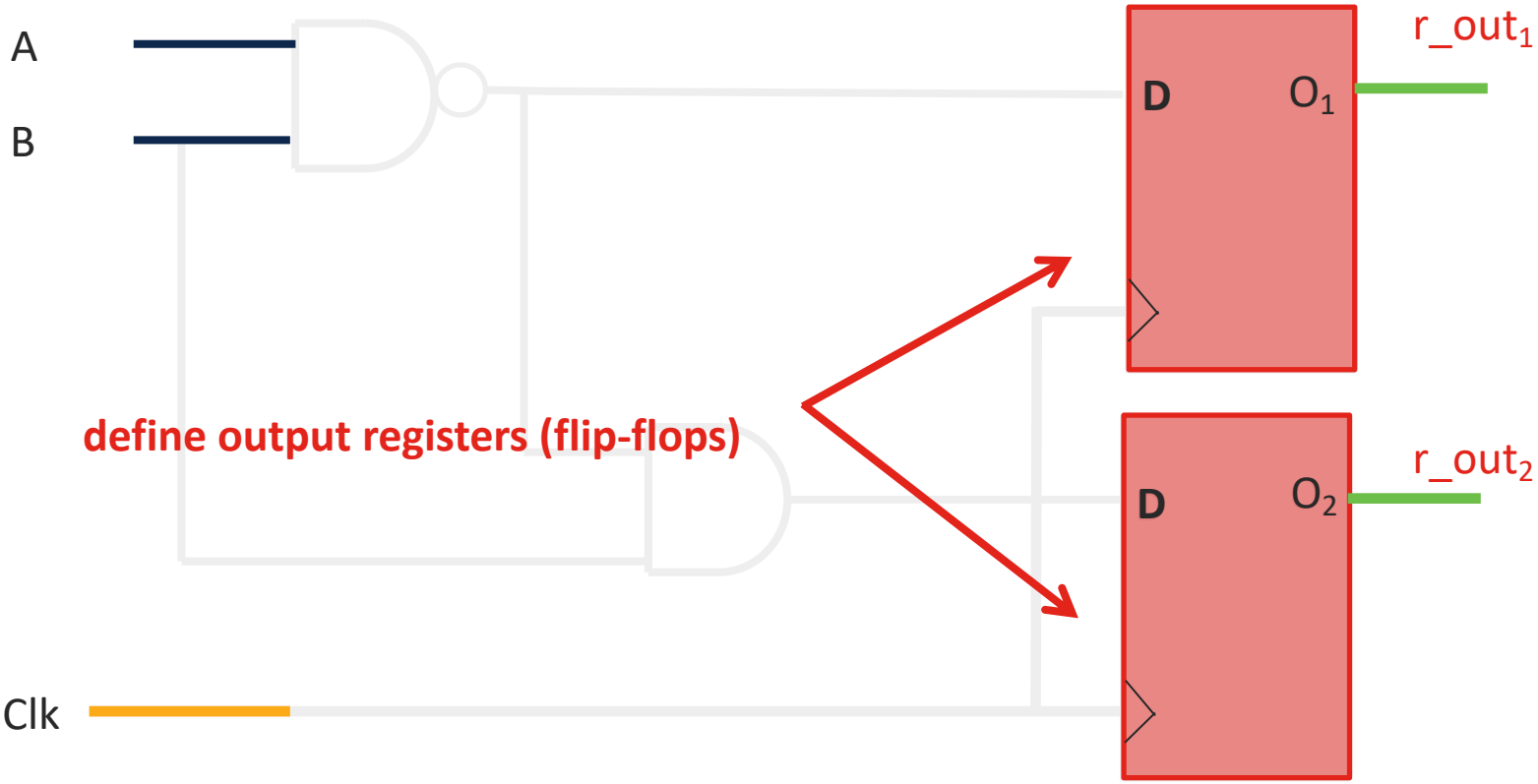
define the circuit outputs



Verilog HDL Code Example

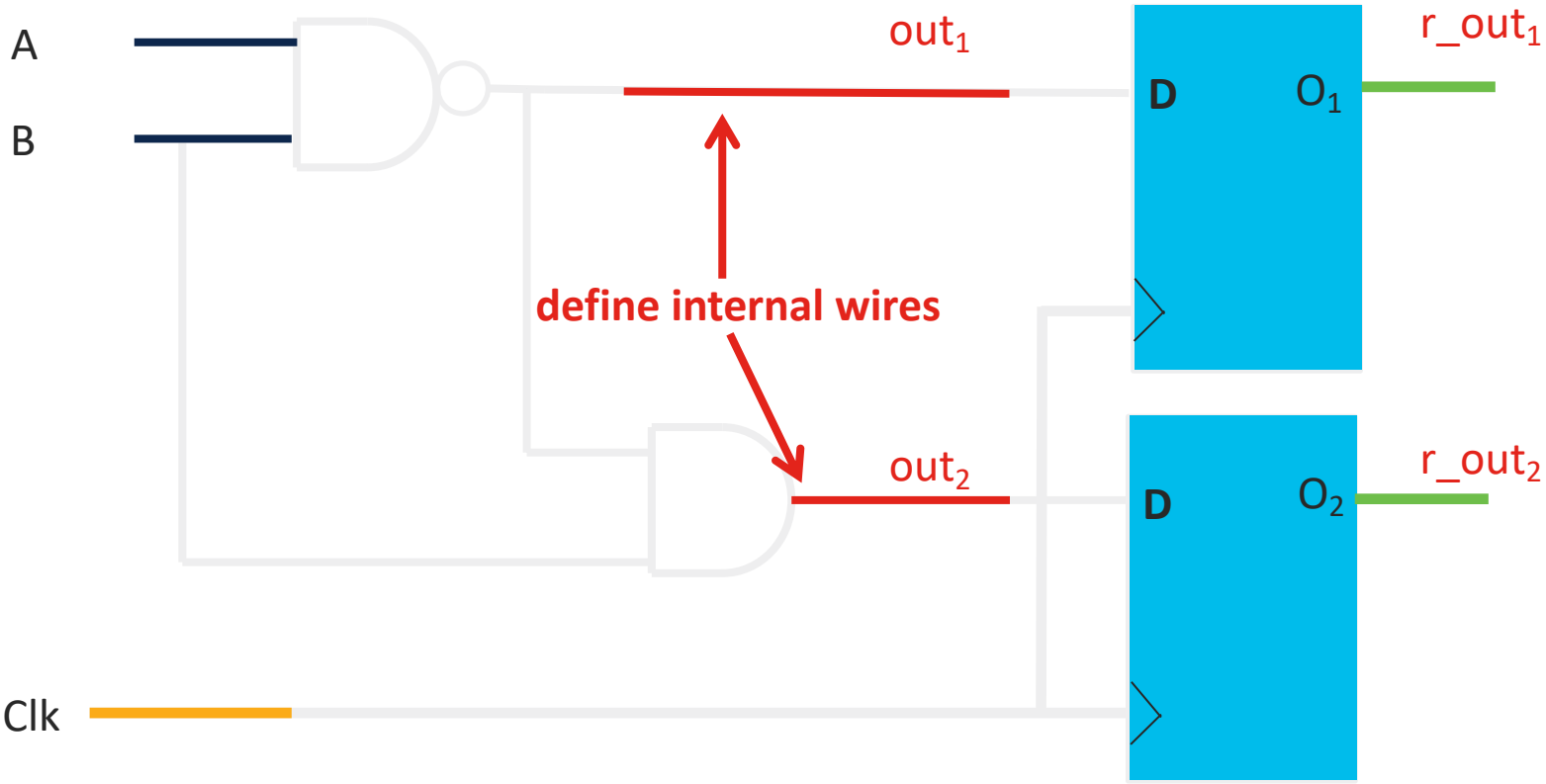
```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;
```

define output registers (flip-flops)



Verilog HDL Code Example

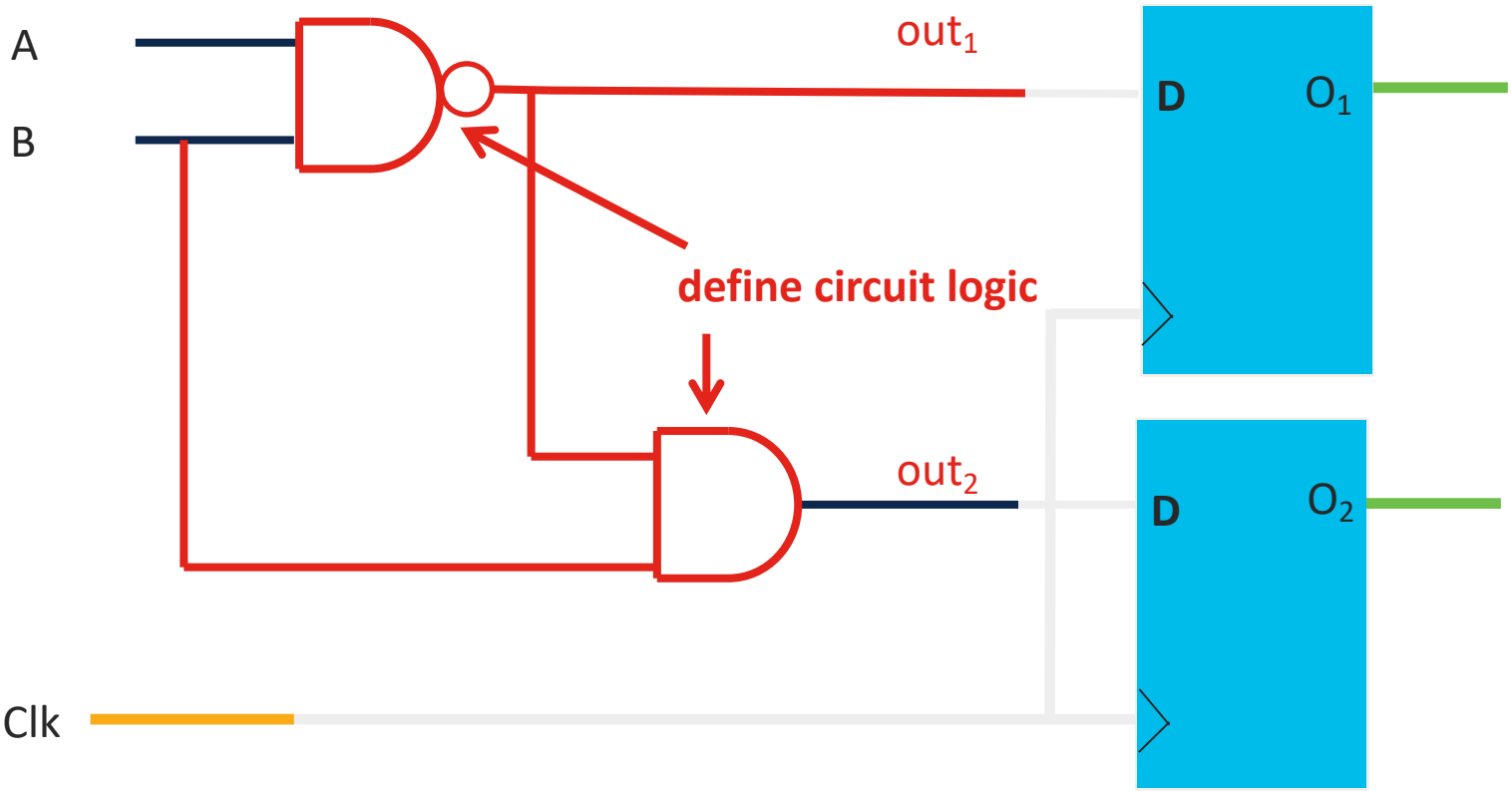
```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
```



Verilog HDL Code Example

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;
```

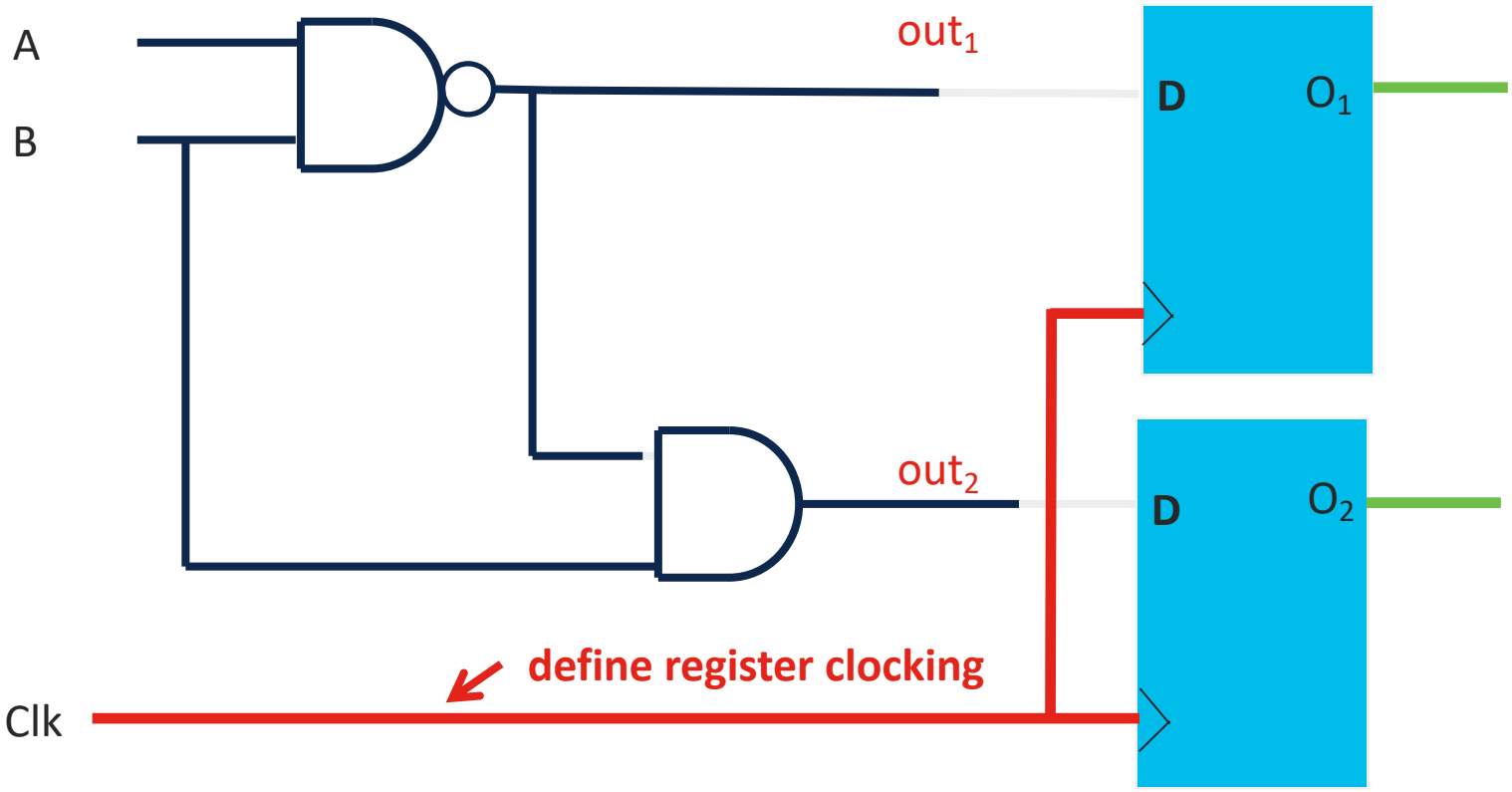
define circuit logic



Verilog HDL Code Example

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(          clk)
```

define register clocking

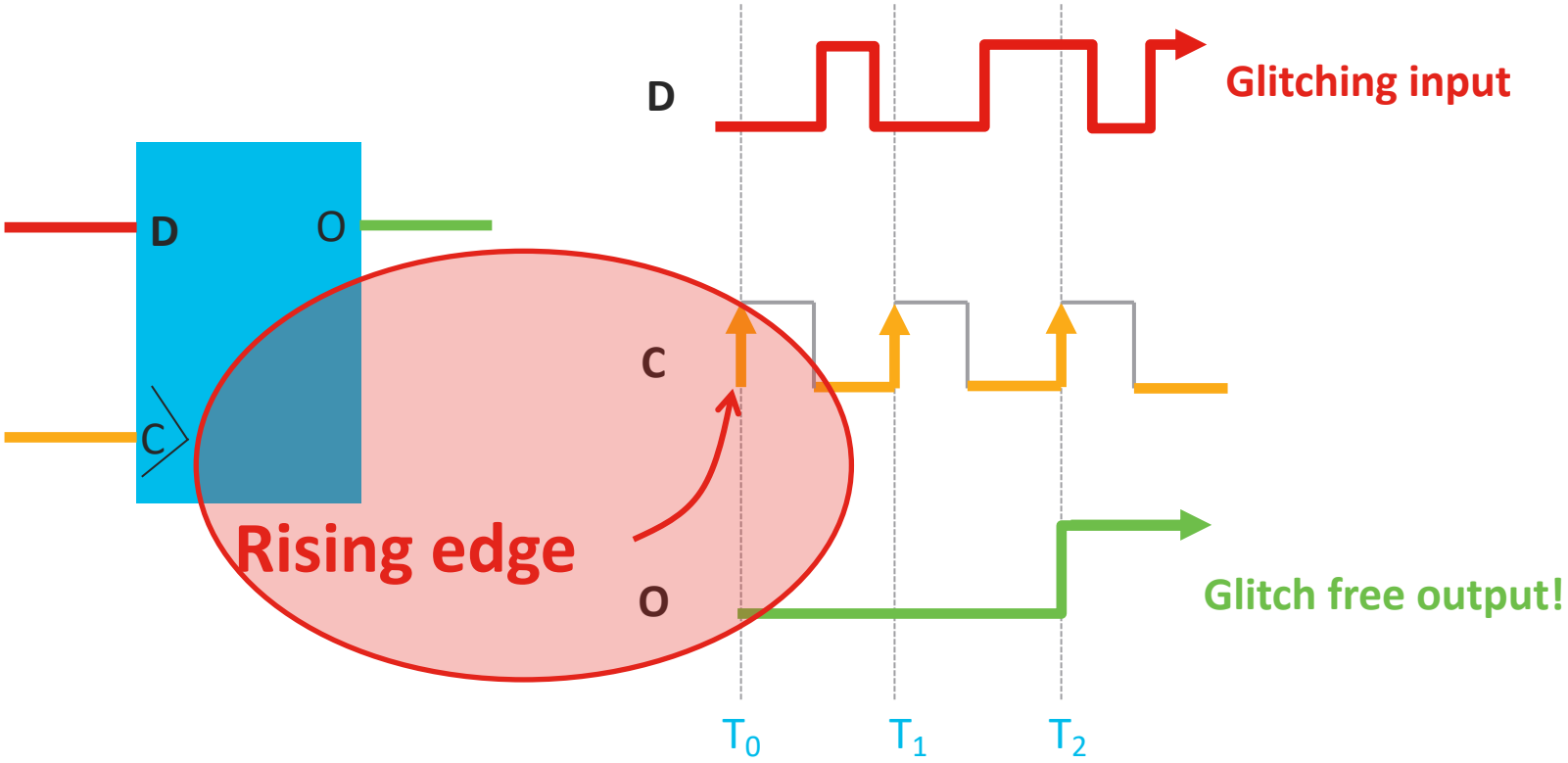


Verilog HDL Code Example

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
```

specify clock type

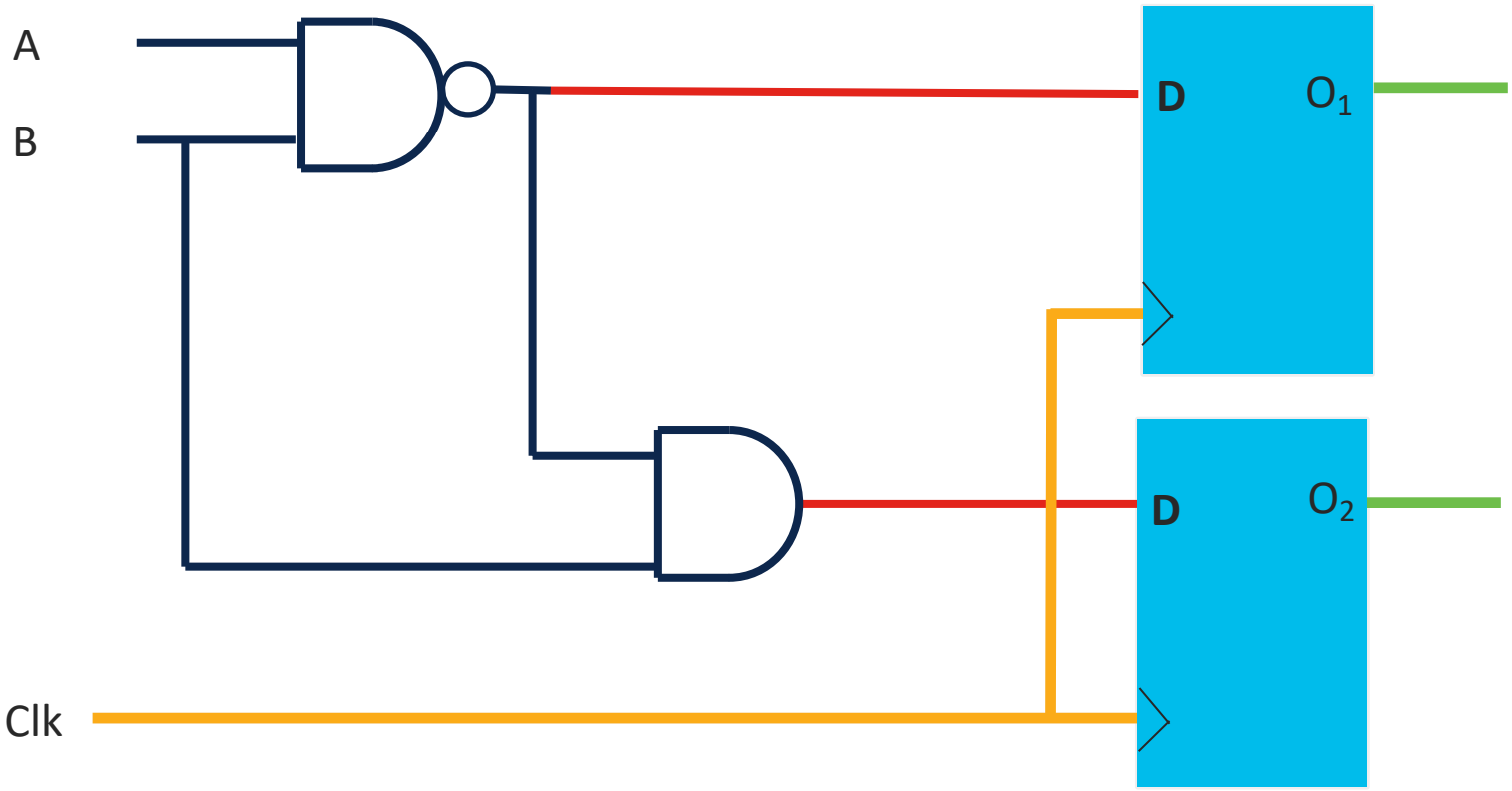
Flip-Flop Operation

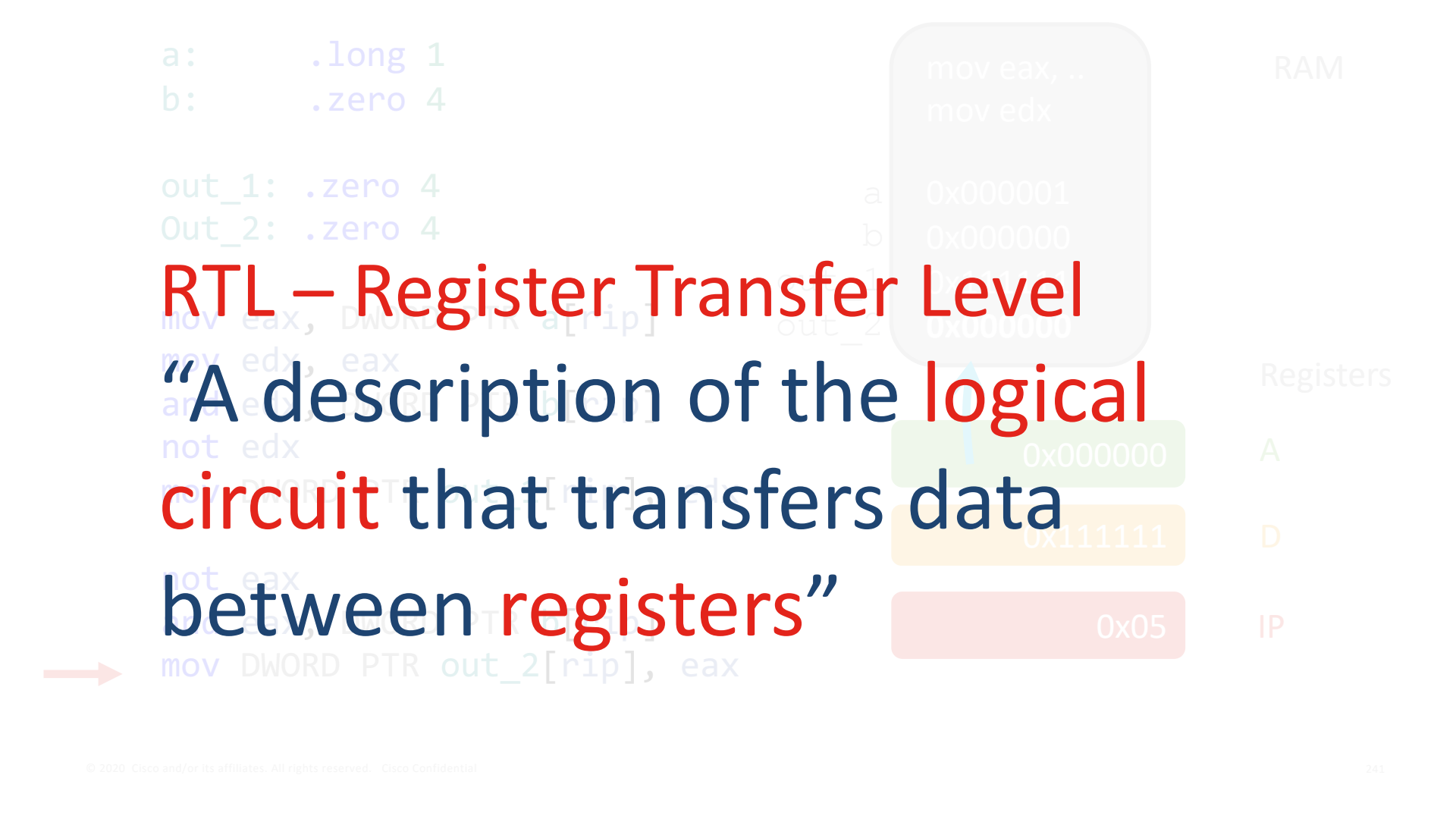


Verilog HDL Code Example

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
    r_out1 <= out1;
    r_out2 <= out2;
end
```

Connect combinational logic to sequential logic



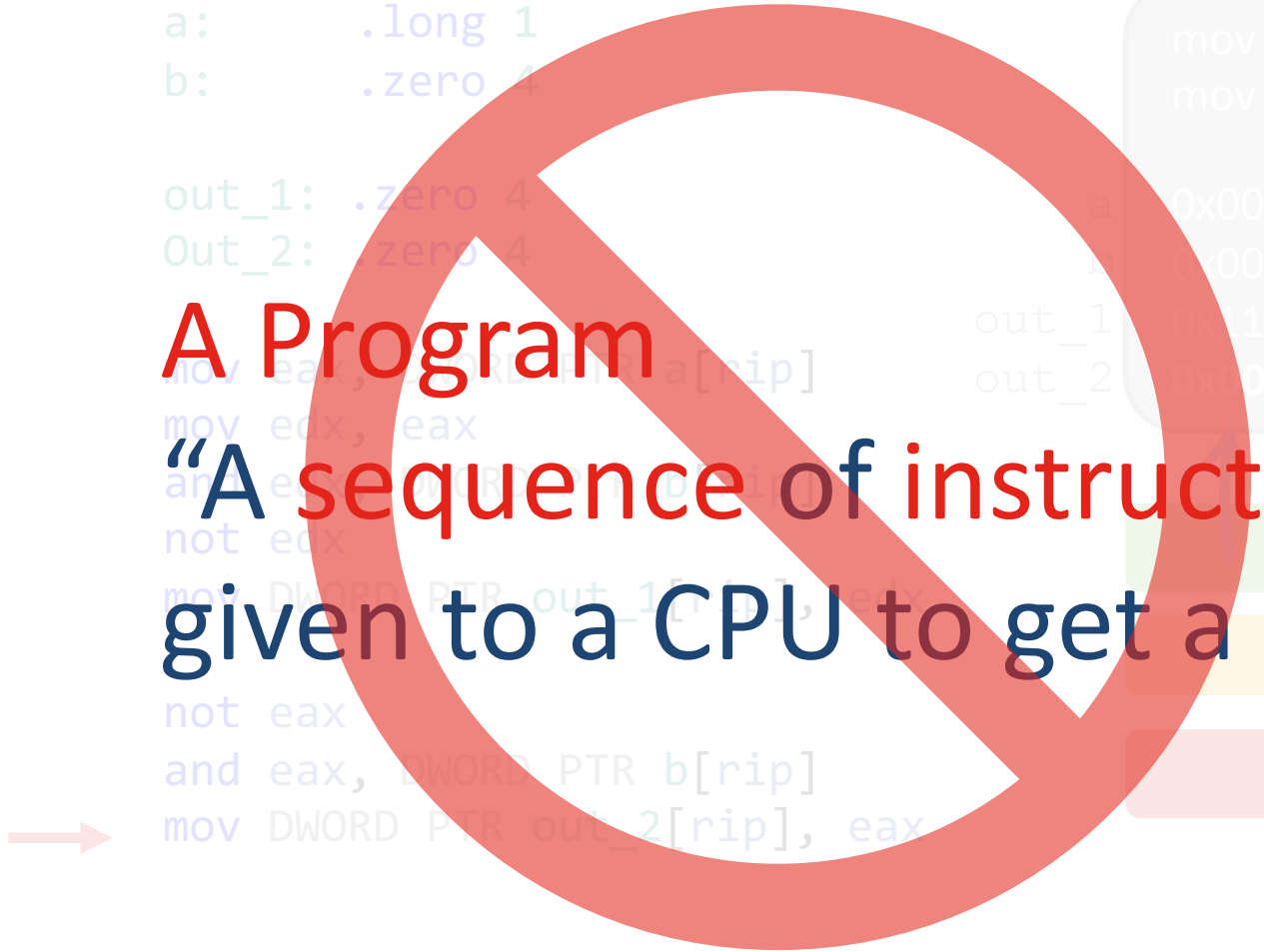


RTL – Register Transfer Level

"A description of the logical circuit that transfers data between registers"

Synthesis:

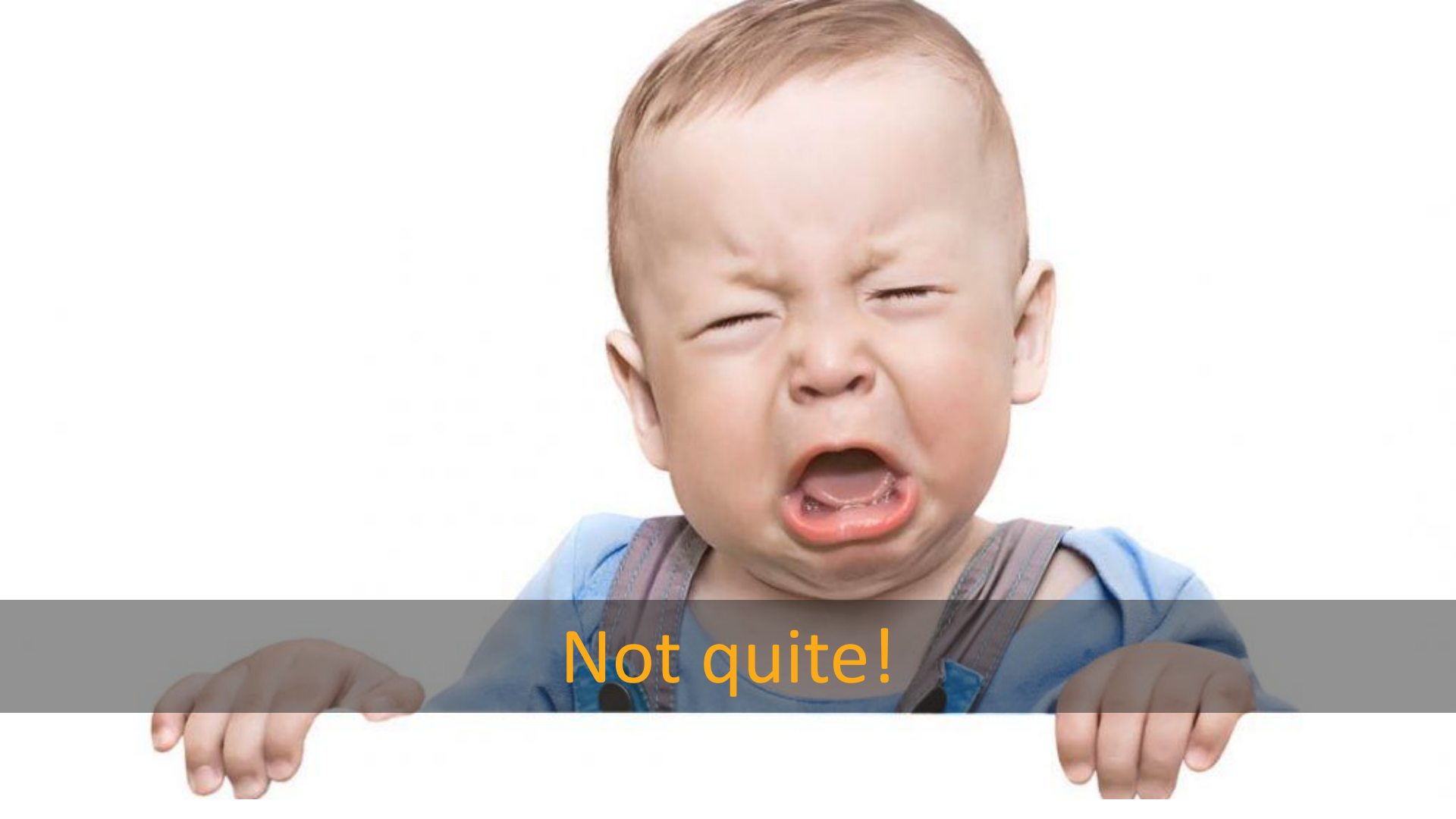
"Convert HDL code (Verilog) into circuit description (netlist)"

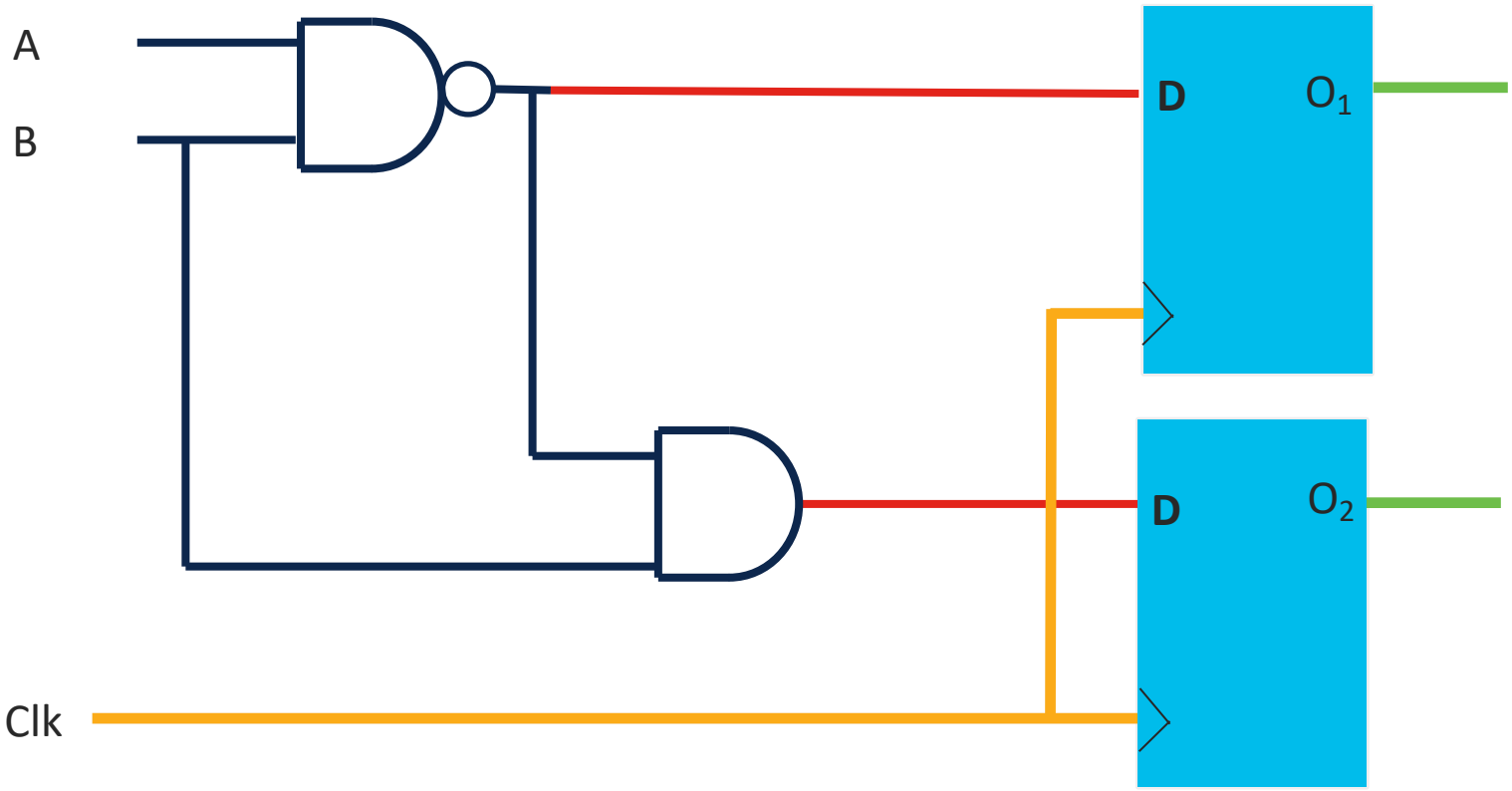


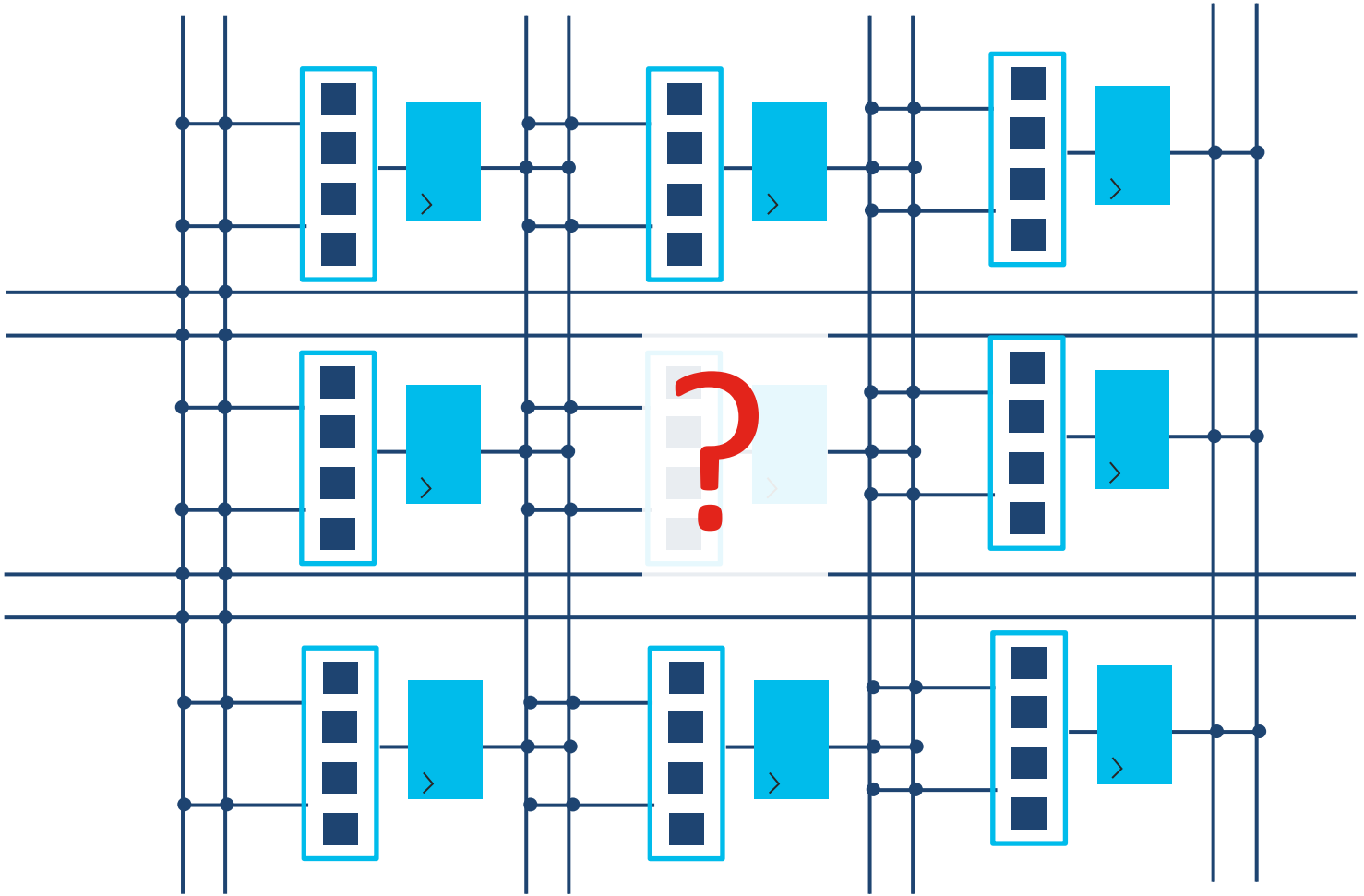
A Program

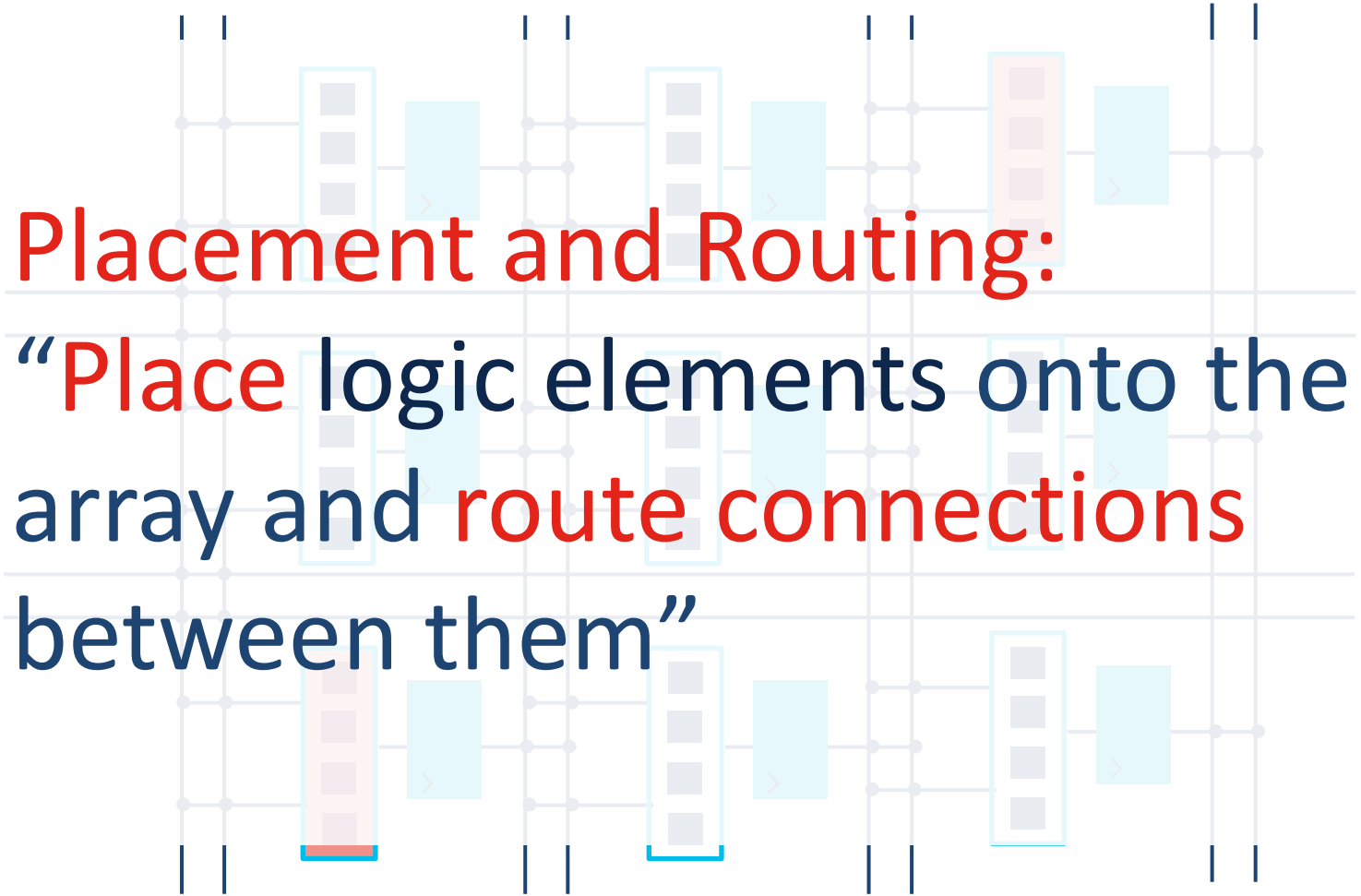
"A sequence of instructions given to a CPU to get a result"


Success!


Not quite!

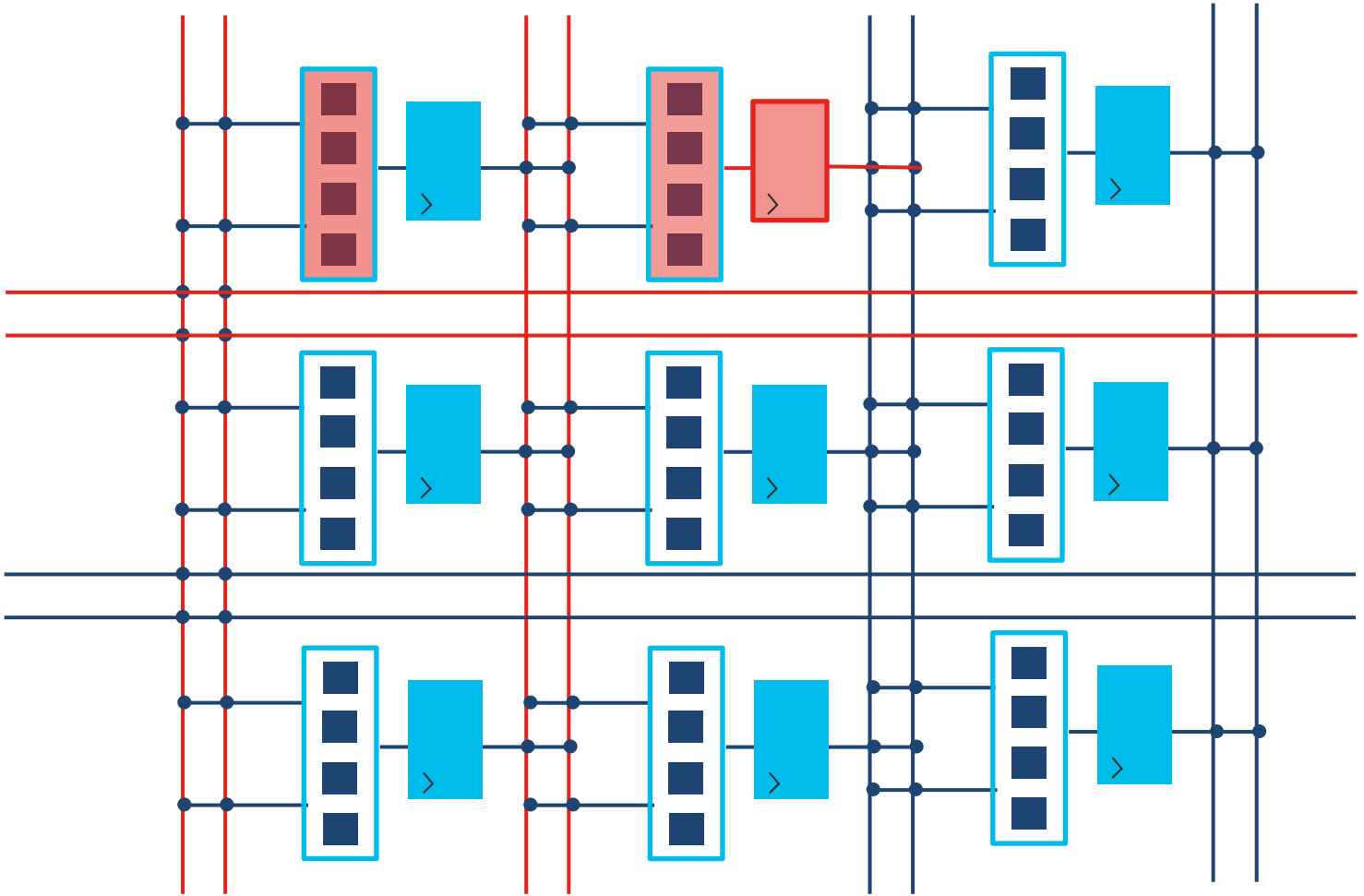


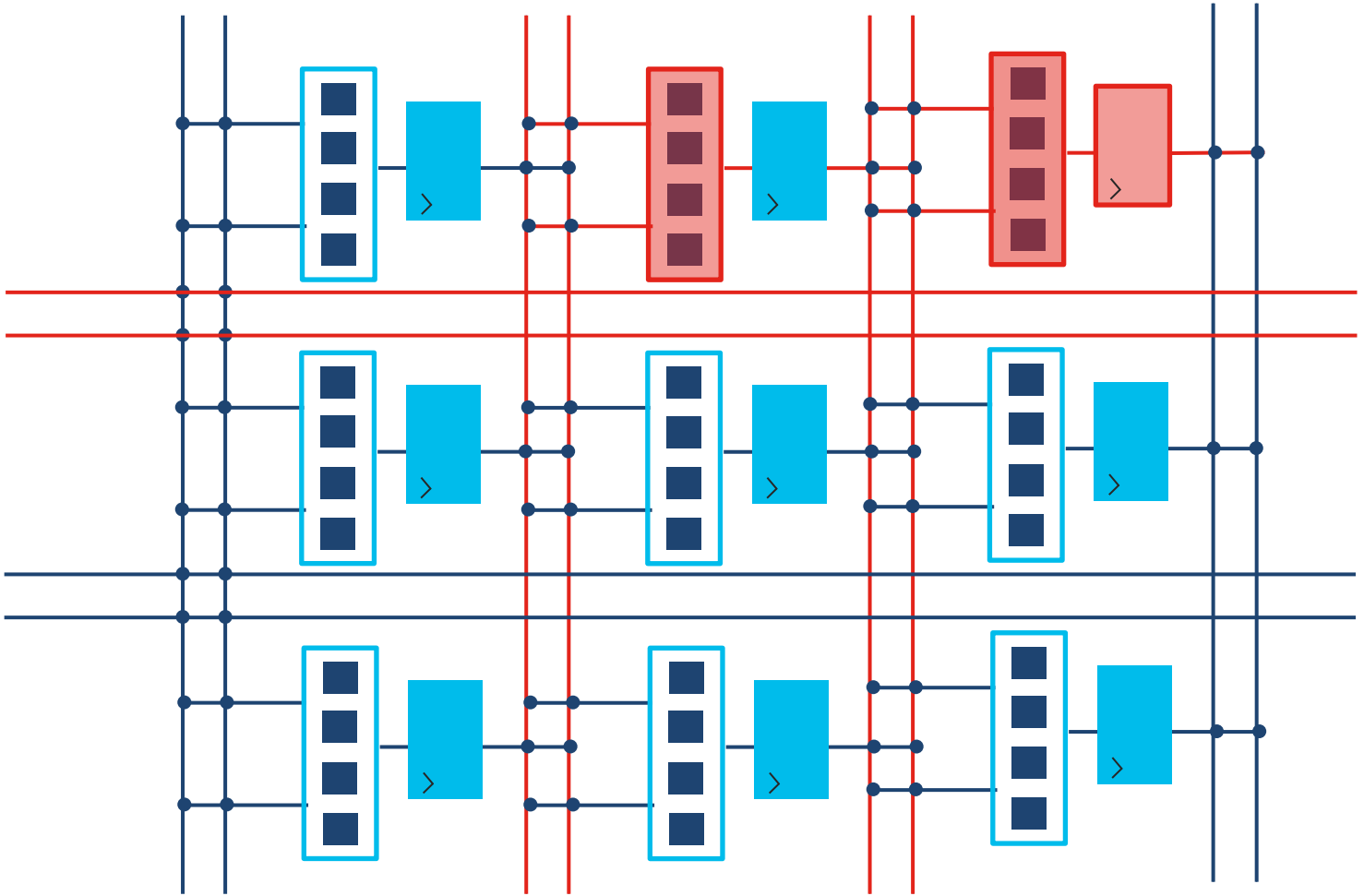


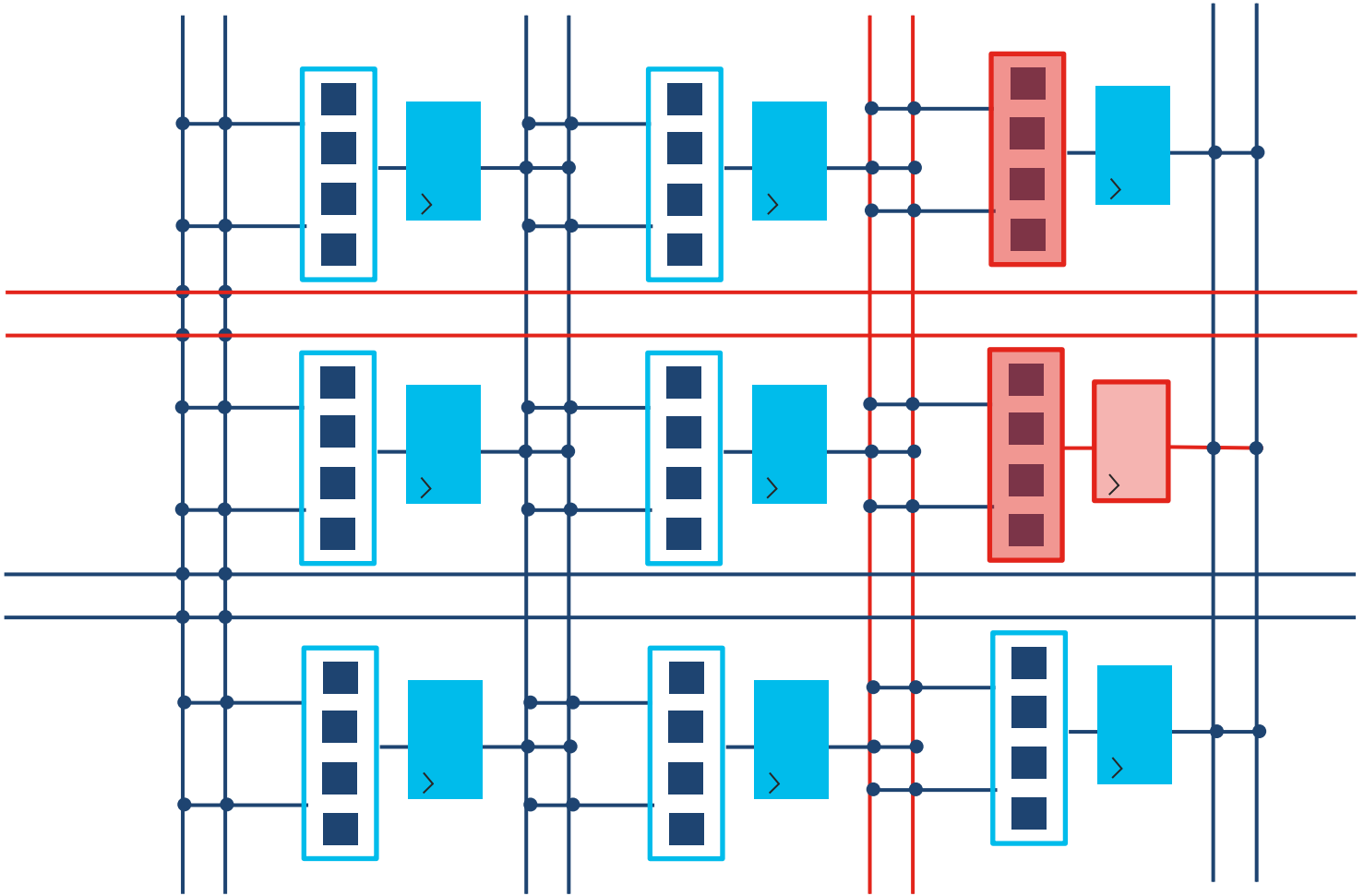


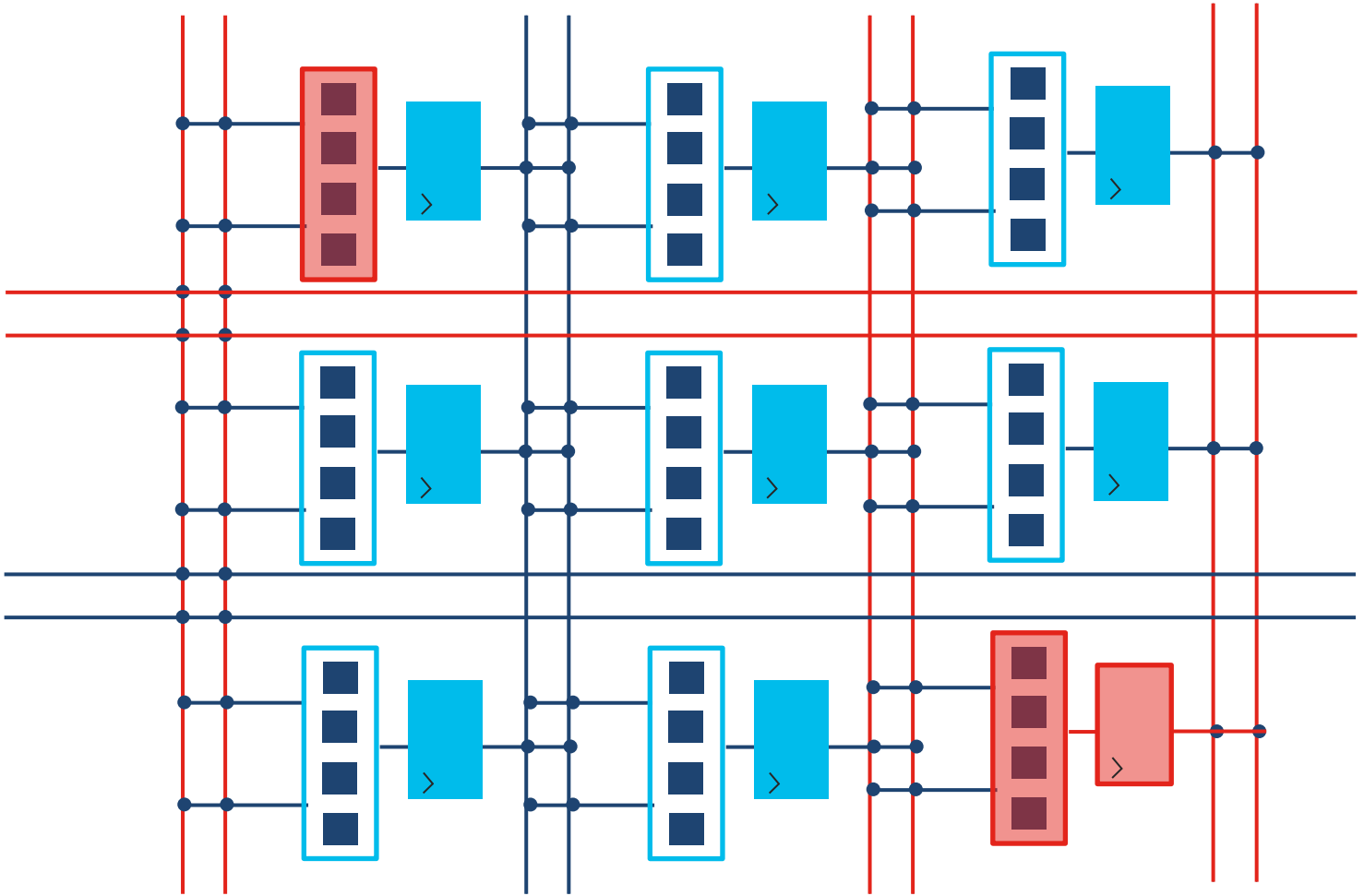
Placement and Routing:

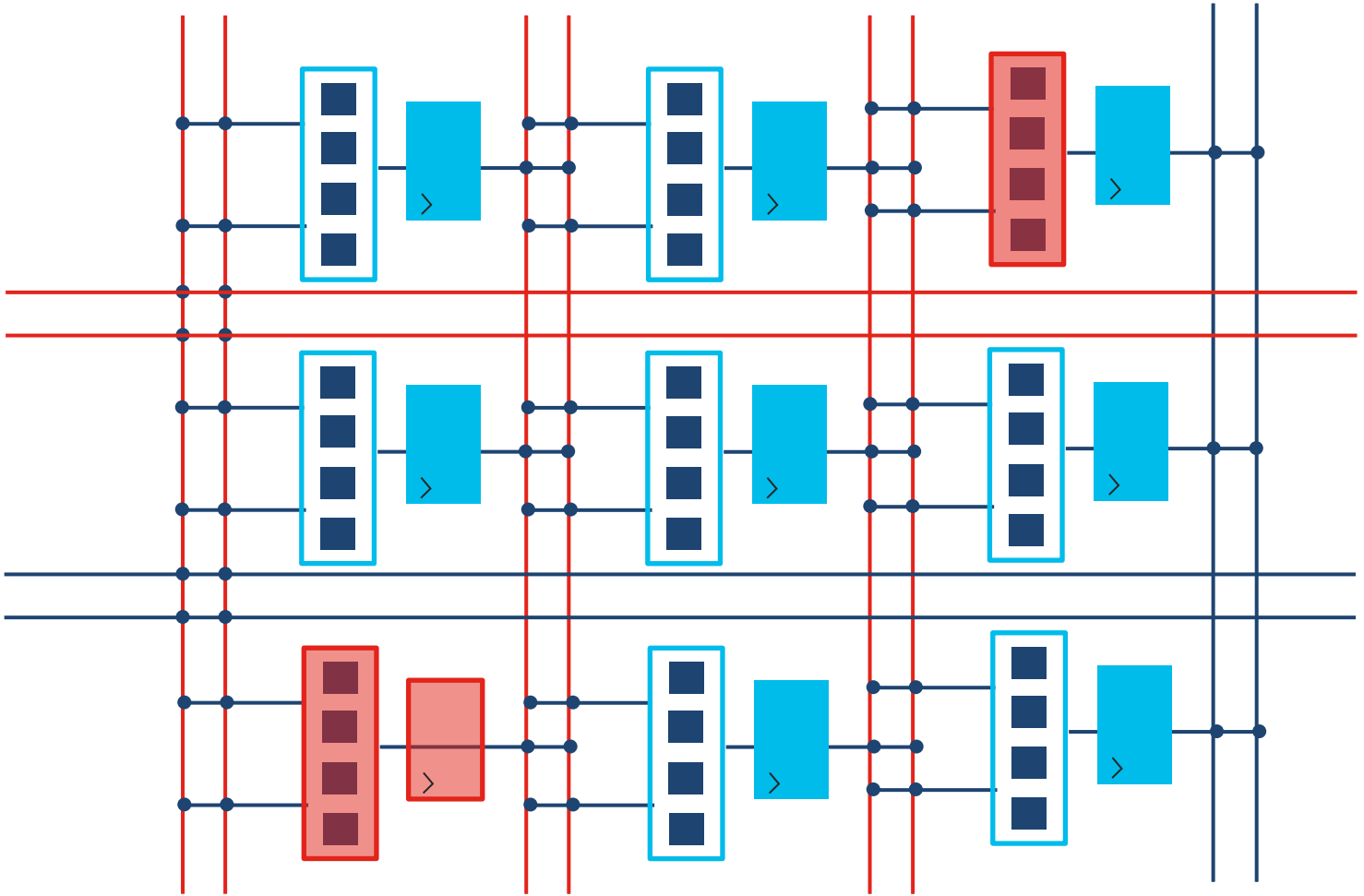
"Place logic elements onto the array and route connections between them"











Place and Route is NP-Hard

With just 9 LUTs, there are 36 different ways to lay out the same circuit

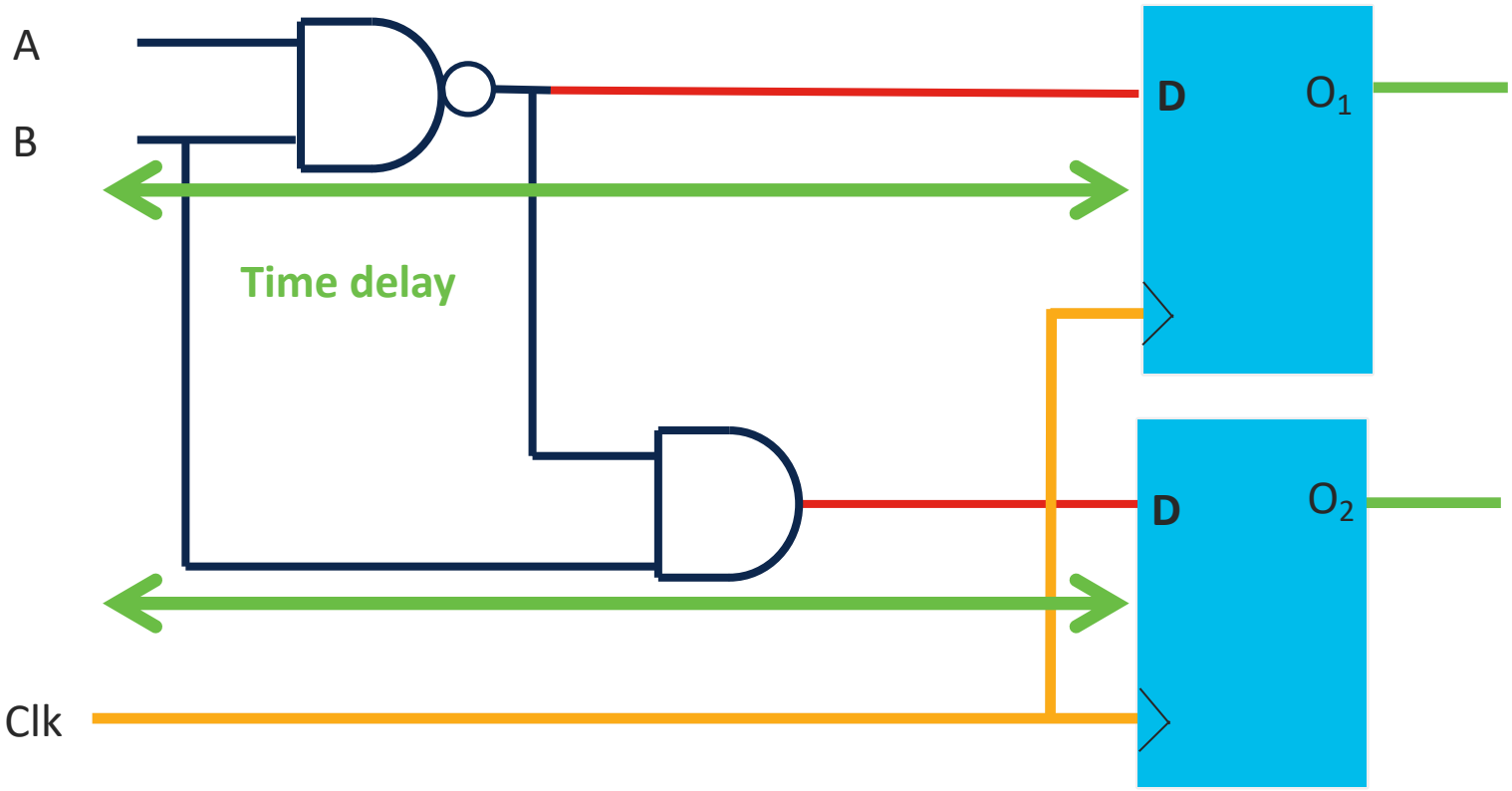
Place and Route is NP-Hard

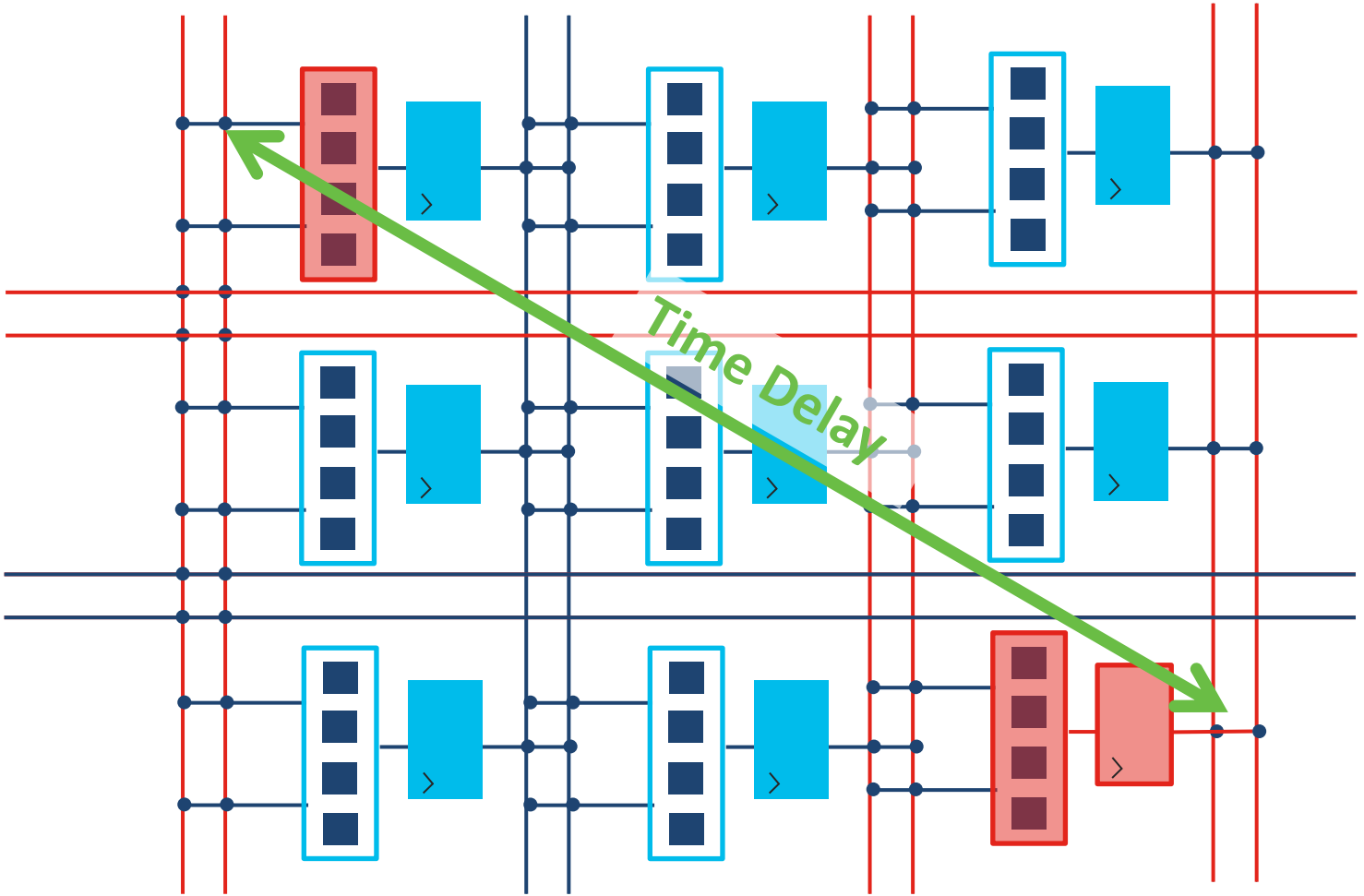
With just 9 LUTs, there are 36 different ways to lay out the same circuit – real FPGAs can have 2M+ LUTS!!

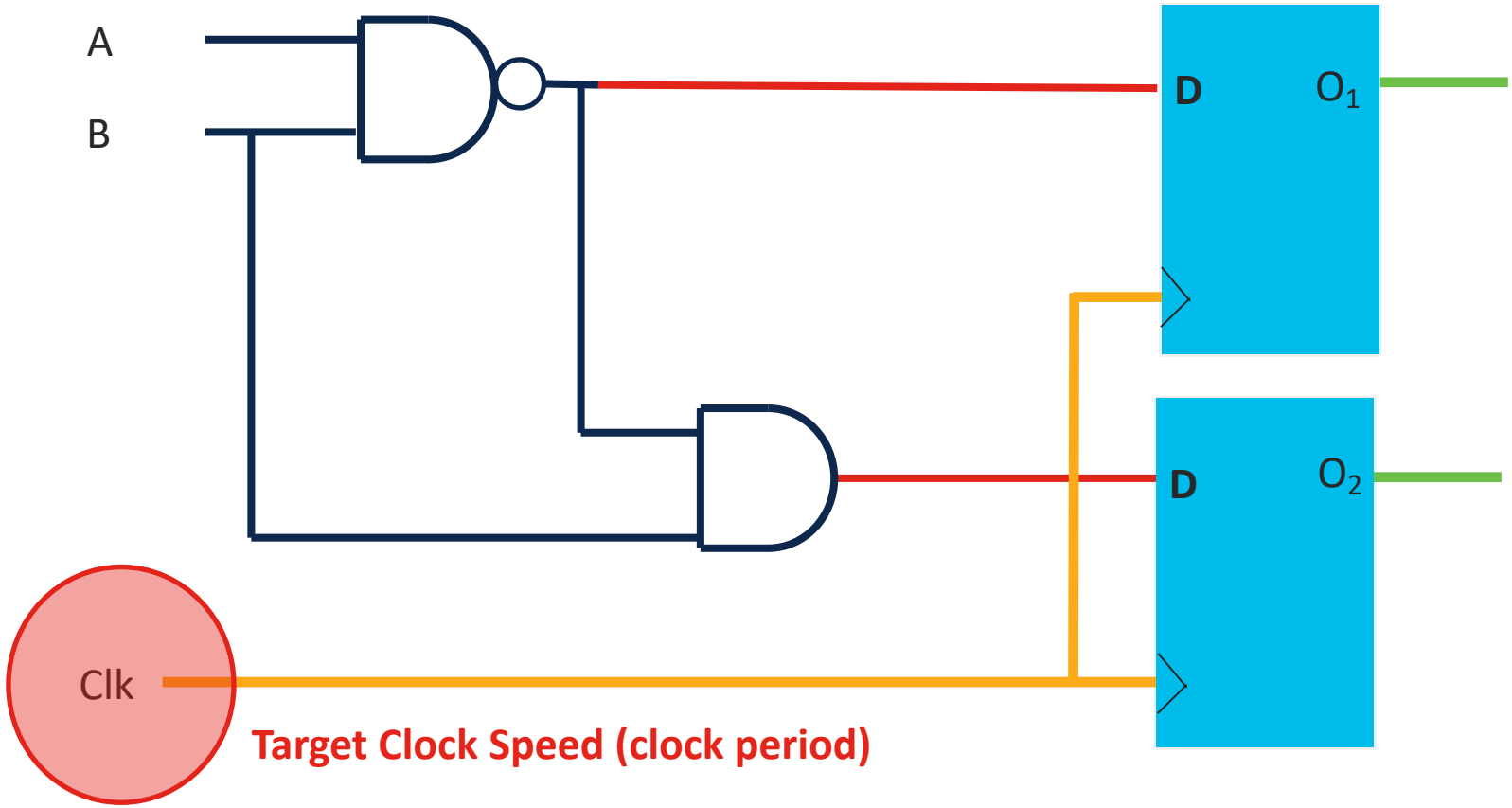

Placement uses random allocations and successive refinement to try to find good placement solutions


The same HDL code, can create totally different placements.


Successful placement does not guarantee a successful result







Timing Closure “meeting timing”

“The delay through logic and routing must not exceed the clock period.

If it does there is too much logic squeezed between flip-flops, or too long routing paths across the chip (or both)”


Success!

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

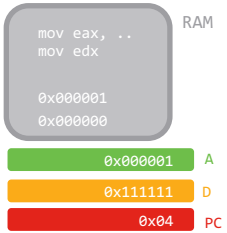
Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU



Summary

Software programming: Executing a sequence of predefined instructions

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU

RTL

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
         r_out1 <= out1;
         r_out2 <= out2;
end
```

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU

RTL

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
        r_out1 <= out1;
        r_out2 <= out2;
end
```

Synth.

Netlist

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU

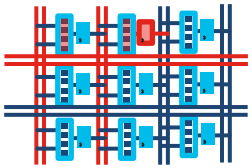
RTL

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
        r_out1 <= out1;
        r_out2 <= out2;
end
```

Synth.

Netlist

Place & Route



Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU

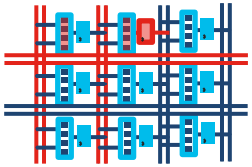
RTL

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
        r_out1 <= out1;
        r_out2 <= out2;
end
```

Synth.

Netlist

Place & Route



Timing Closure

Bitstream

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU

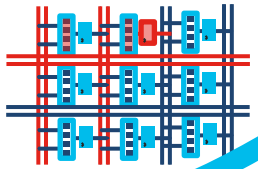
RTL

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
        r_out1 <= out1;
        r_out2 <= out2;
end
```

Synth.

Netlist

Place & Route



Timing Closure

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU

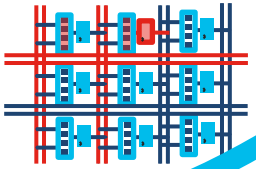
RTL

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
        r_out1 <= out1;
        r_out2 <= ou
end
```

Synth.

Netlist

Place & Route



Timing Closure

Summary

Program

```
int a = 1;
int b = 0;
int out_1;
int out_2;
out_1 = ~(a & b);
out_2 = out_1 & b;
```

Compiler

Machine Code

```
a: .long 1
b: .zero 4
out_1: .zero 4
out_2: .zero 4
mov eax, DWORD PTR a[rip]
mov edx, eax
and edx, DWORD PTR b[rip]
not edx
mov DWORD PTR out_1[rip],
```

Execute

CPU/GPU

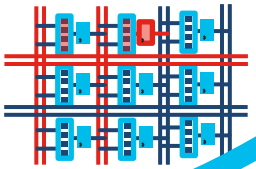
RTL

```
input A, B, clk;
output r_out1, r_out2;
reg r_out1, r_out2;

wire out1, out2;
assign out1 = ~(A & B);
assign out2 = out1 & B;

always @(posedge clk)
begin
        r_out1 <= out1;
        r_out2 <= ou
end
```

Synth.

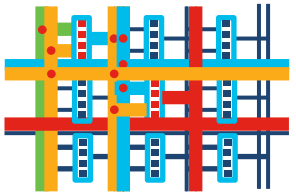
Netlist

Place & Route



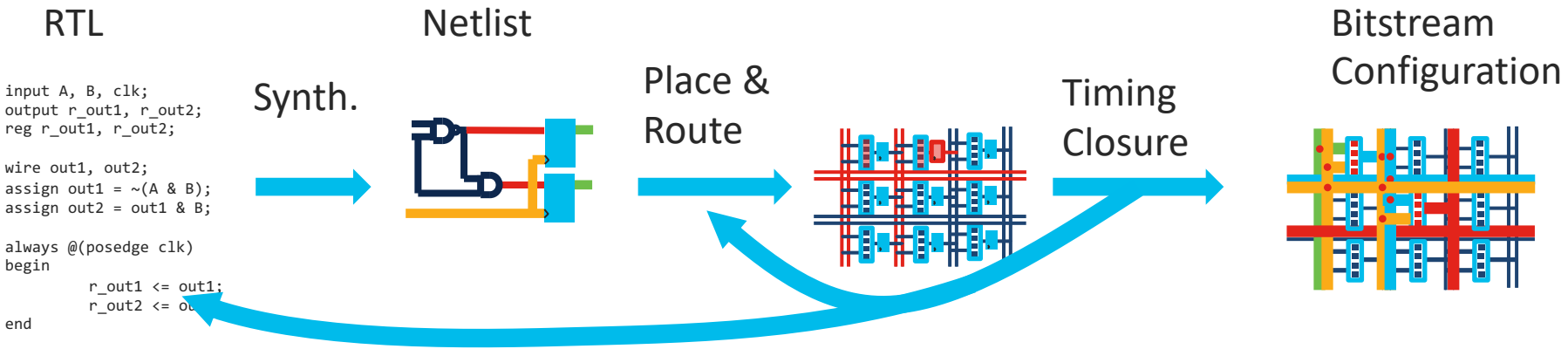
Timing Closure

Bitstream Configuration



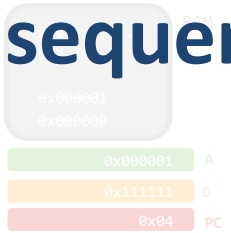
Summary

Software programming: Executing a sequence of predefined instructions



Summary

Software programming: Executing a sequence of predefined instructions



FPGA "programming": Specifying the digital logic circuit to be implemented


One more thing…

Firmware Development Kit (FDK)

Integrated FPGA development environment optimized for low latency applications on Nexus SmartNIC and Nexus 3550-F swithces.

Firmware Development Kit (FDK)

1. **FDK-PE Performance Edition**

   Optimized for Ultra Low latency applications (e.g. HFT)

Firmware Development Kit (FDK)

1. **FDK-PE Performance Edition**

   Optimized for Ultra Low latency applications (e.g. HFT)

2. **FDK-EV Evaluation Edition**

   Freely available ULL optimized (2hr operation limit)

Firmware Development Kit (FDK)

1. **FDK-PE Performance Edition**

   Optimized for Ultra Low latency applications (e.g. HFT)

2. **FDK-EV Evaluation Edition**

   Freely available ULL optimized (2hr operation limit)

3. **FDK-FE – Free Edition**

   Optimized for network acceleration. Freely available.

Firmware Development Kit (FDK)

1. **FDK-PE Performance Edition**

   Optimized for Ultra Low latency applications (e.g. HFT)

2. **FDK-EV Evaluation Edition**

   Freely available ULL optimized (2hr operation limit)

3. **FDK-FE – Free Edition** 

   Optimized for network acceleration. Freely available.

Tick the box or speak to our sales team to find out more!


CISCO