Digital Logic for Protection & Control

35th Annual Hands-On Relay School

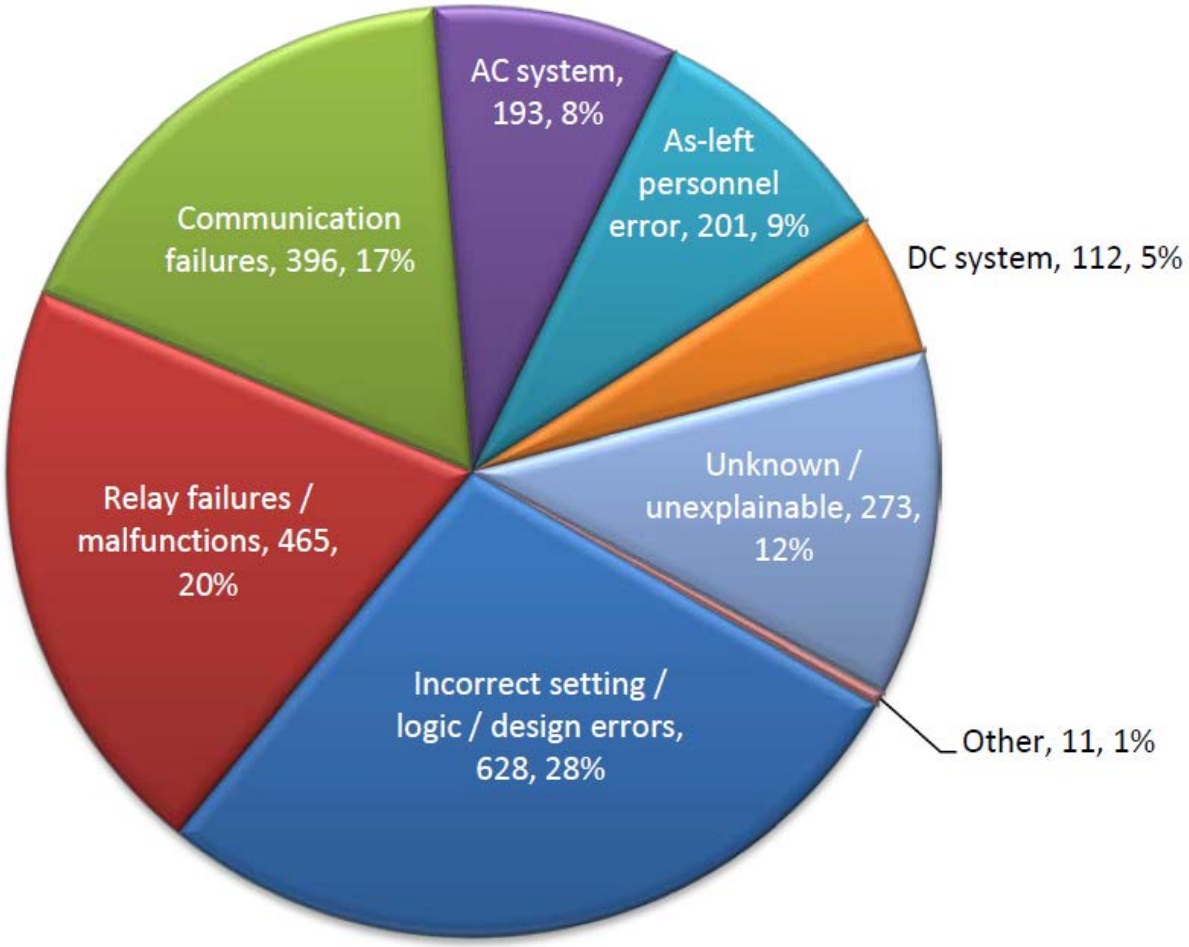
March-2018

Kevin Damron, PE

Sr. Protection Engineer – Avista Utilities
Adjunct Professor – Gonzaga University
kevin.damron@avistacorp.com

WASHINGTON STATE UNIVERSITY

Why?



NERC-wide Misoperations by Cause Code
(2011 – 2012)

Agenda

Fundamentals

- Logic Gates, Truth Tables, Logic Equations, and Schematics

Positive and Negative Logic

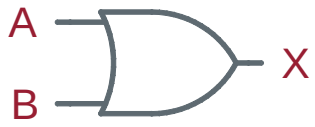
Logic Properties

More Logic

- Latches, Edge Triggers, Timers

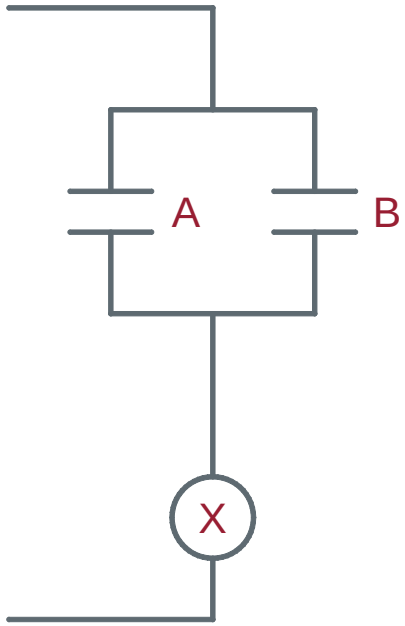
Examples

OR Gate



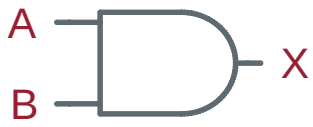
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

I1	I2	O1
false	false	false
false	true	true
true	false	true
true	true	true



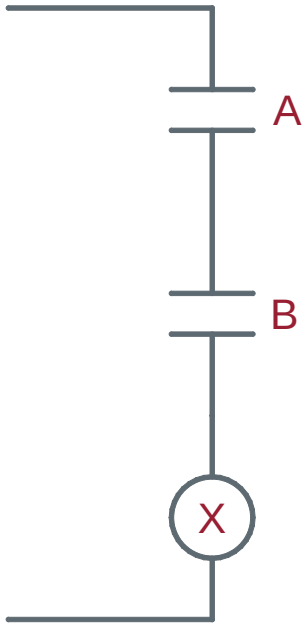
X = A + B = A OR B

AND Gate



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

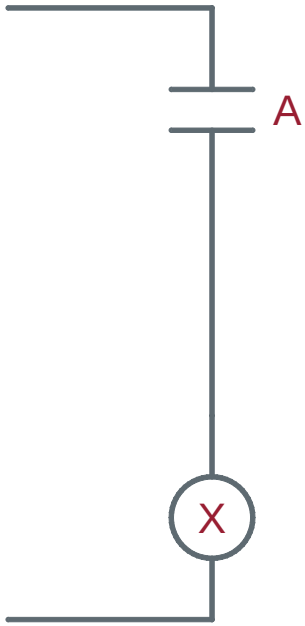
X = A * B = A AND B

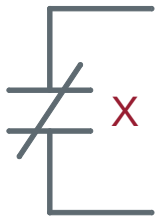


NOT Gate (Inverter)

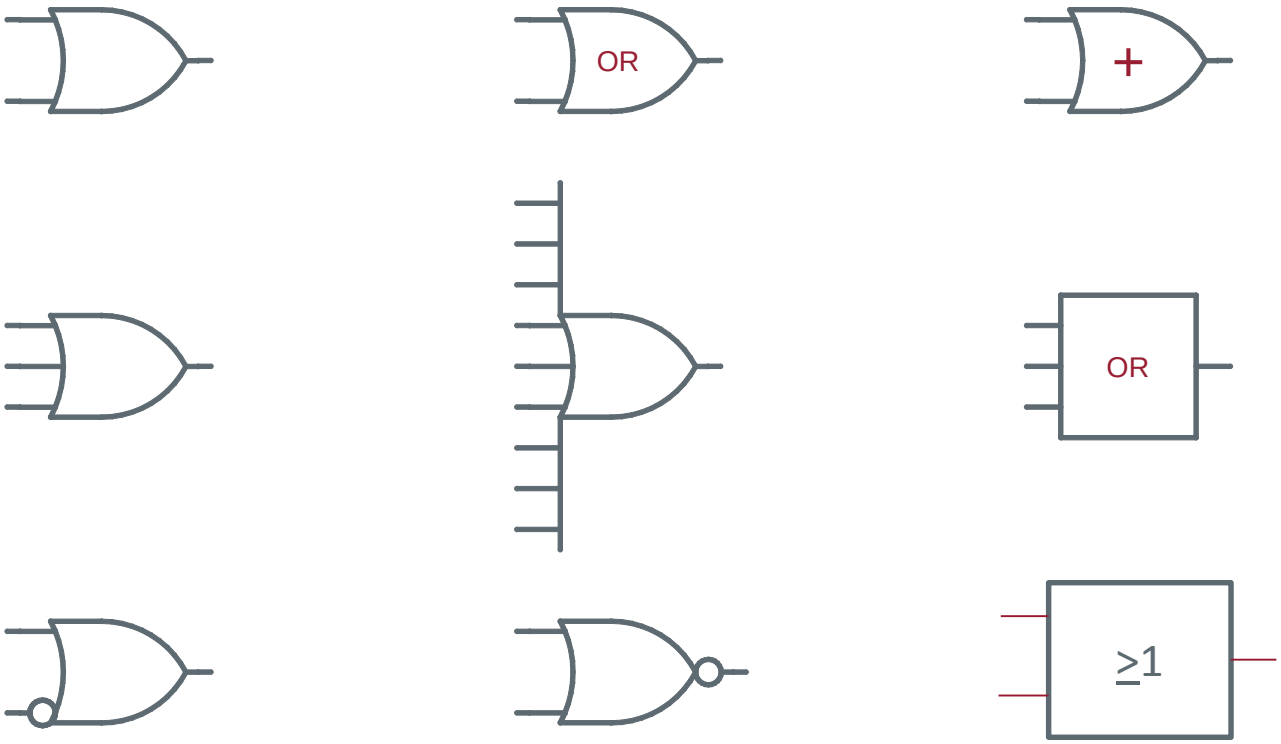
A	X
0	1
1	0

X = !A = NOT A

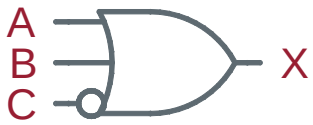




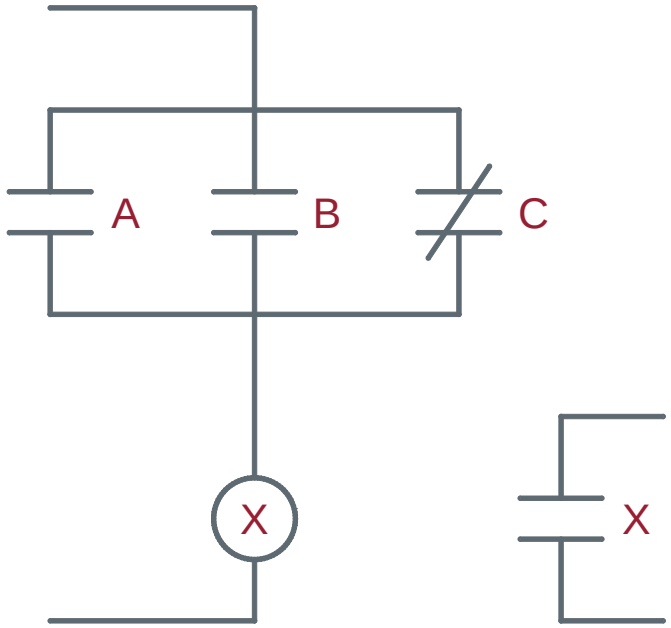
Gate Variations



Inverted Inputs



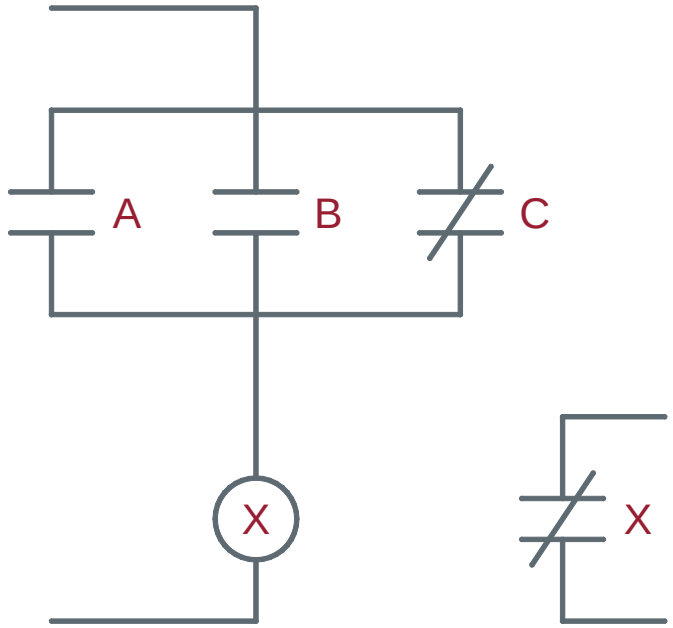
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



X = A + B + !C = A OR B OR NOT C

Inverted Outputs

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



X = ! (A + B + !C) = NOT (A OR B OR NOT C)

Positive and Negative Logic

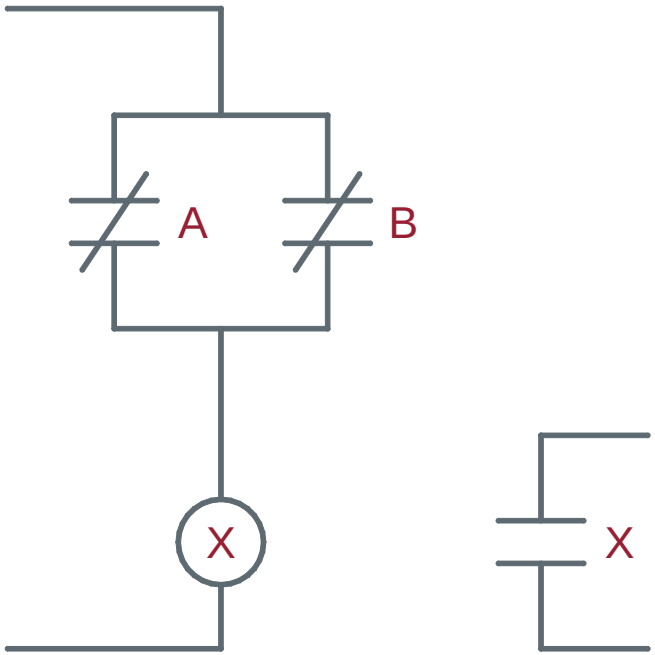
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

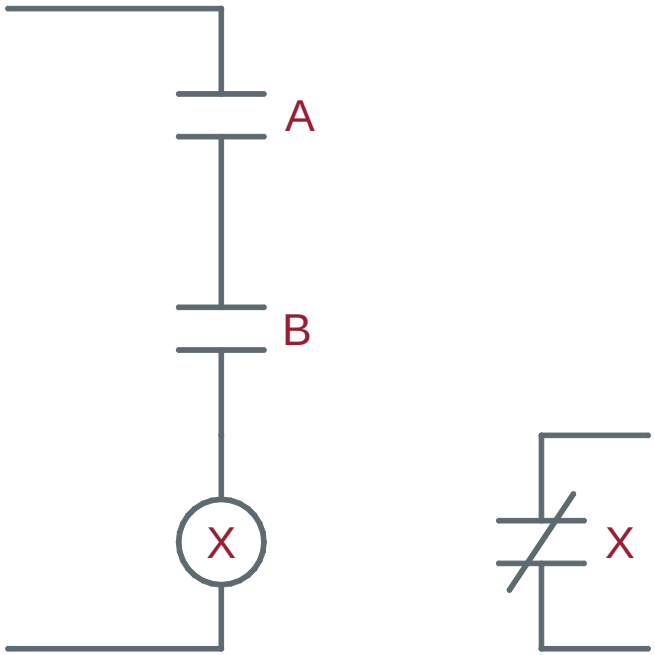
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

X = !A + !B = NOT A OR NOT B

X = !(A * B) = NOT (A AND B)

Positive and Negative Logic





Positive and Negative Logic

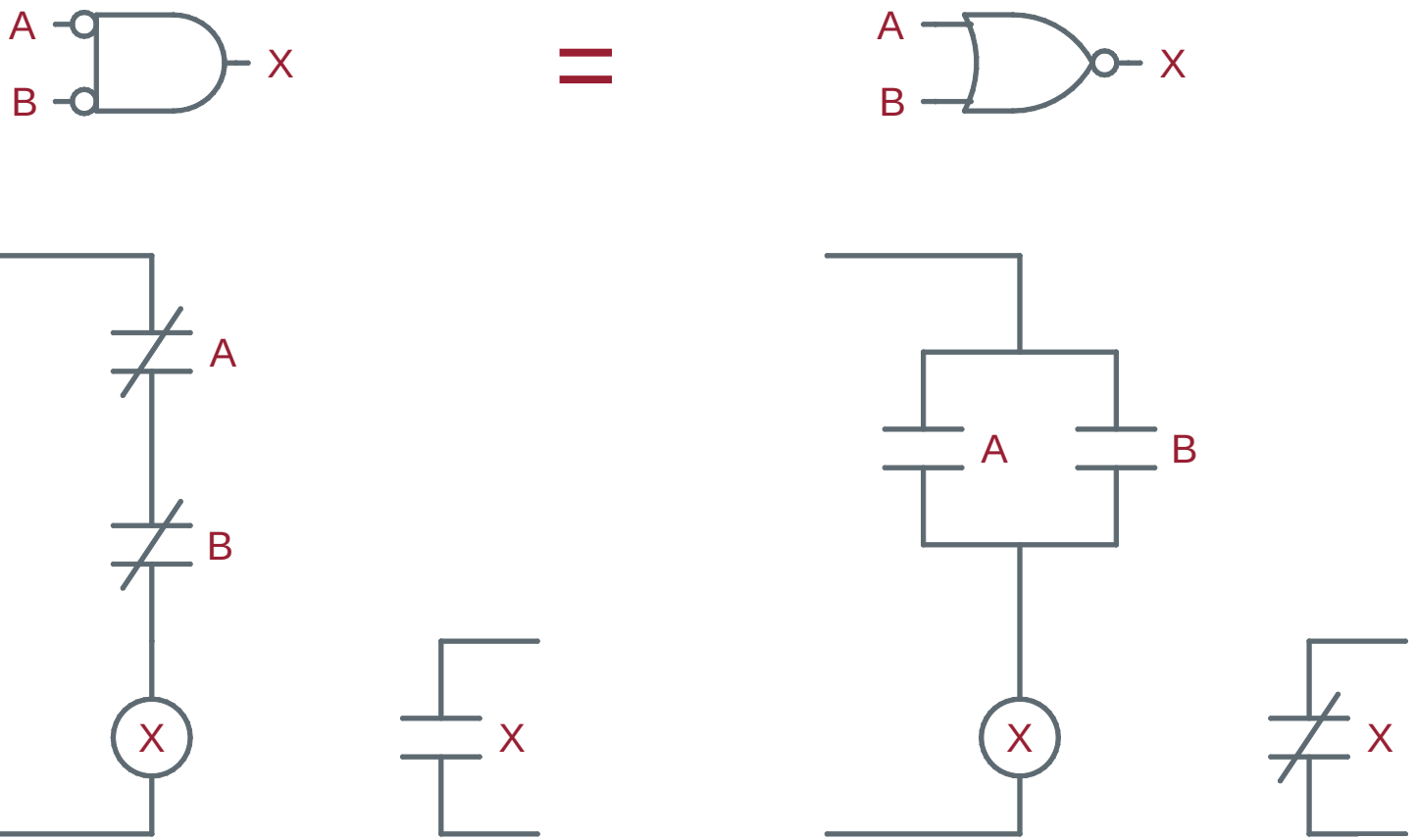
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

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

X = !A * !B = NOT A AND NOT B

X = !(A + B) = NOT (A OR B)

Positive and Negative Logic



Logic Gates

AND
OR
NOT

Most common for protection and control

NAND
NOR

Universal Gates
Very important for hardware design

XOR

XNOR

Logic Properties

Commutative Property:

- X = A + B = B + A
- X = A * B = B * A

Associative Property:

- X = (A + B) +C = A + (B + C)
- X = (A * B) * C = A * (B * C)

Distributive Property:

- X = A * (B + C) = A * B + A * C

Logic Precidence

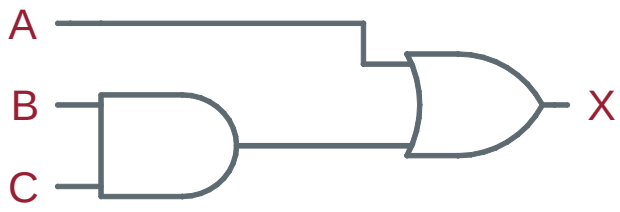
Math:

- My Dear Aunt Sally = Multiply -> Divide -> Add -> Subtract
  - 2 + 3 x 4 = ??
  - 2 + 3 x 4 = 2 + (3 x 4) = 14
  - 2 + 3 x 4 ≠ (2 + 3) x 4 = 20

Logic:

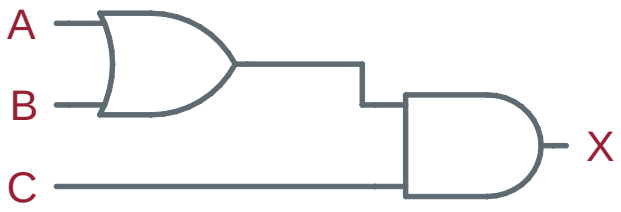
- NOT -> AND -> OR
  - X = A + B * C = A or B and C
  - X = A + B * C = A + (B * C)
  - X = A + B * C ≠ (A + B) * C

Combining Gates



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

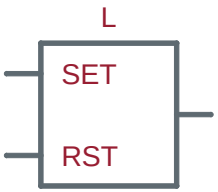
$X = A + (B * C) = A + B * C$

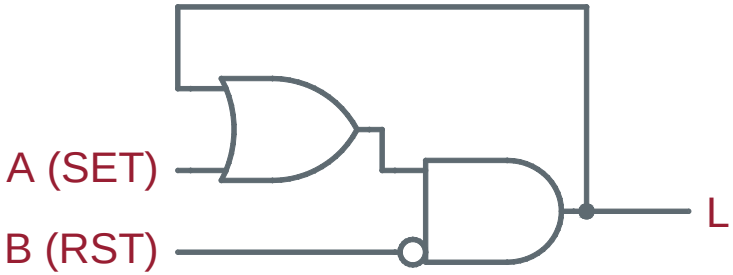


A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$X = (A + B) * C \neq A + B * C$

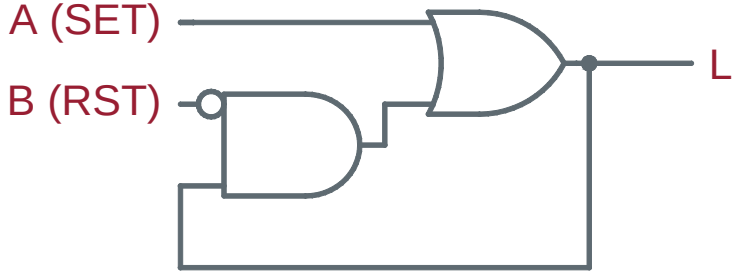
Latches





L = (A + L) * !B

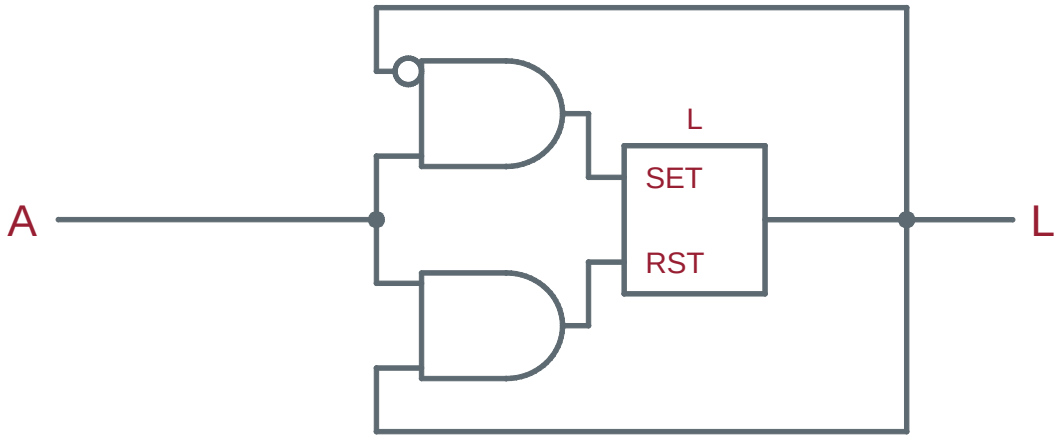
RESET DOMINANT LATCH



L = A + (L * !B)

SET DOMINANT LATCH

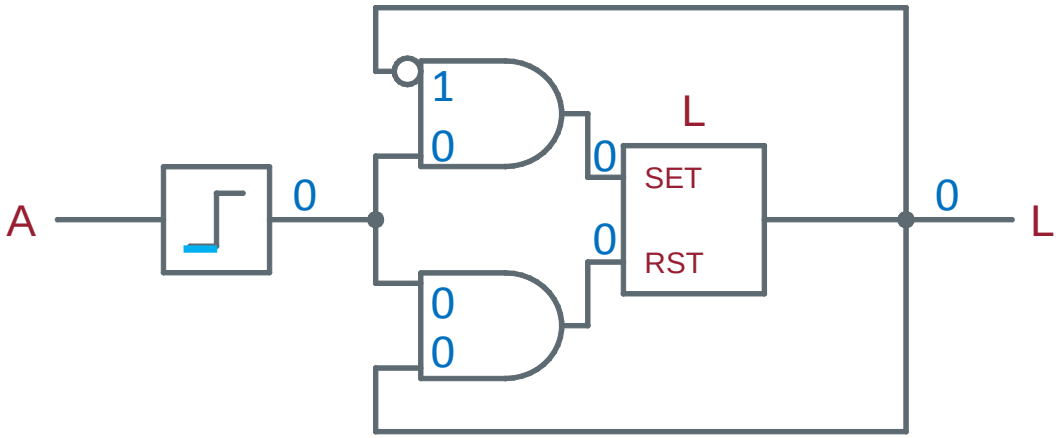
Latches



SETL = A * !L

RSTL = A * L

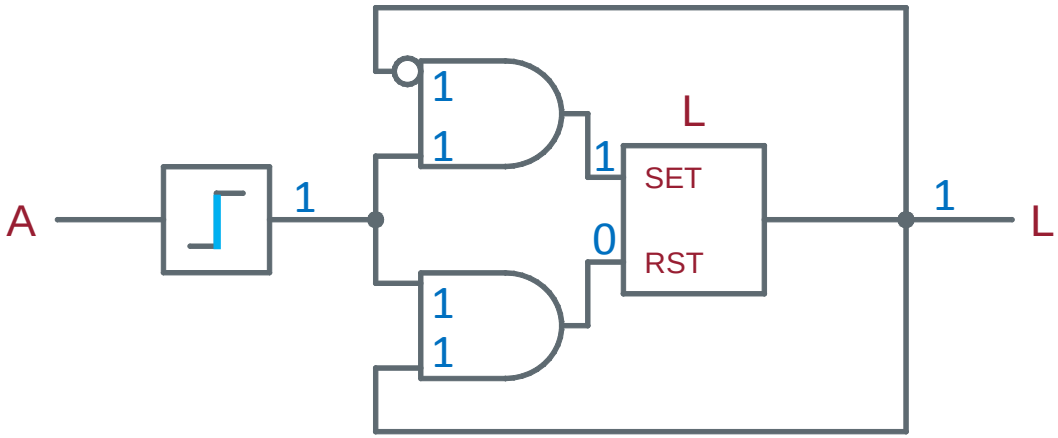
Edge Triggers



SETL = /A * !L

RSTL = /A * L

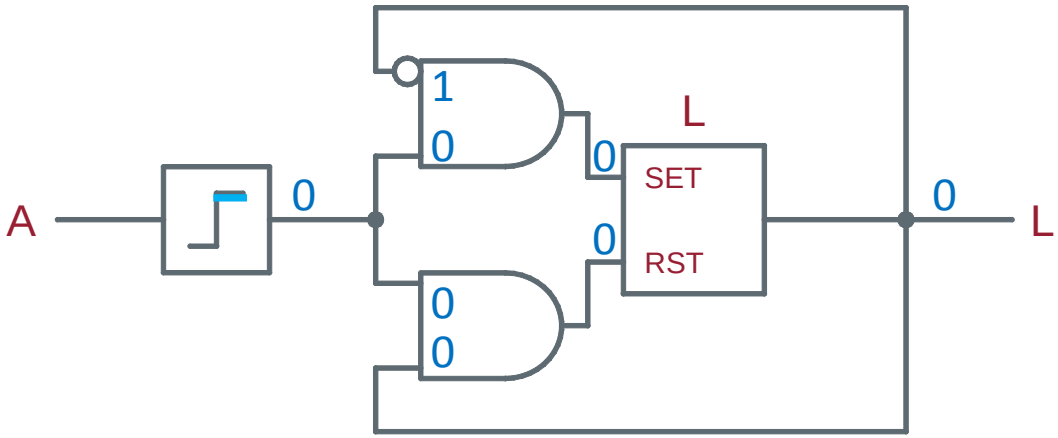
Edge Triggers



SETL = /A * !L

RSTL = /A * L

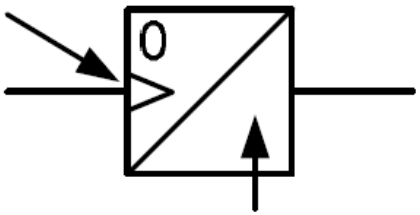
Edge Triggers



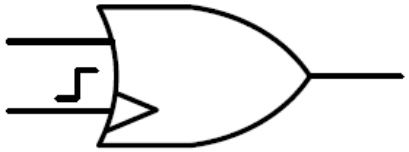
SETL = /A * !L

RSTL = /A * L

Edge Triggers

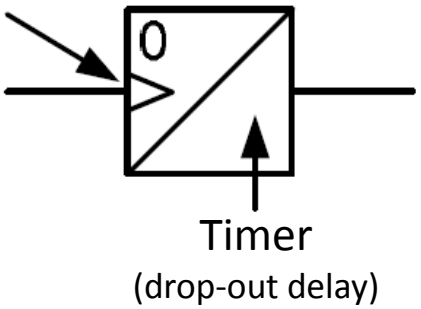


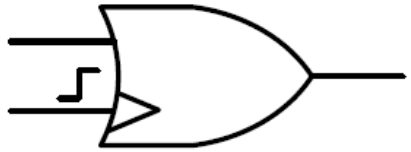
Timer
(drop-out delay)

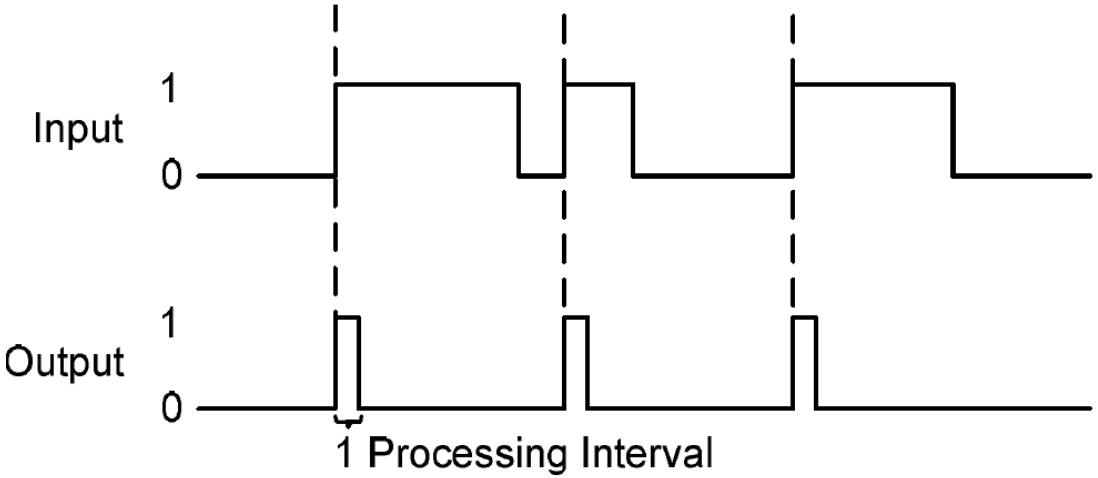


- Edge triggers can be shown a variety of ways in logic diagrams
- Rising-edge or falling-edge not always specified
  - Rising-edge = 0 → 1 transition
- Unless otherwise noted, a rising-edge trigger can be assumed

Edge Triggers

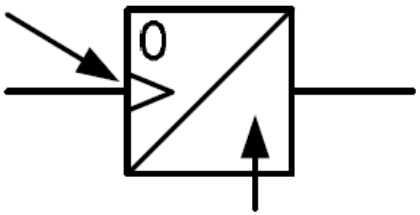


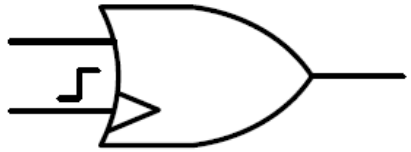


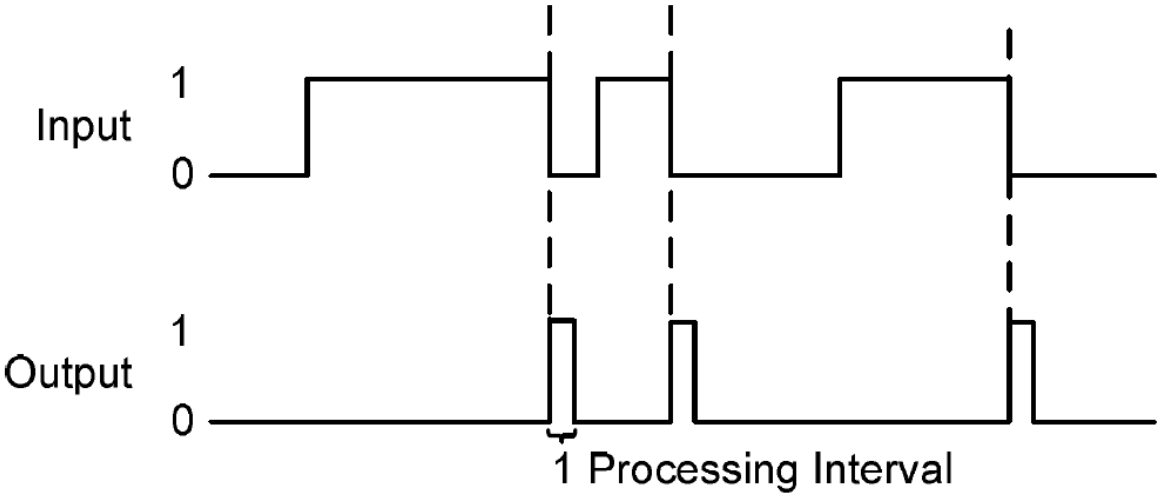


Rising-Edge Trigger Timing Diagram

Edge Triggers

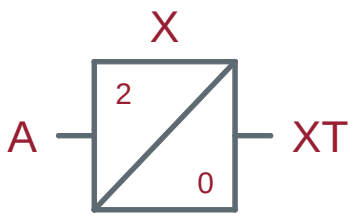




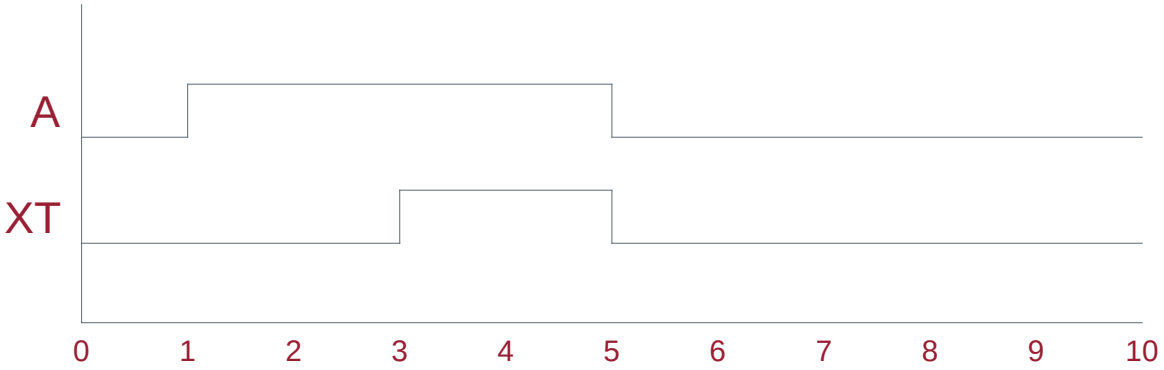


Falling-Edge Trigger Timing Diagram

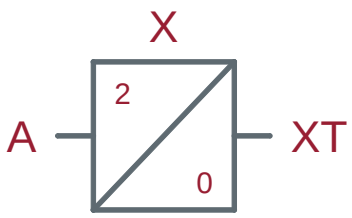
Timers



X = A
XPU = 2
XDO = 0

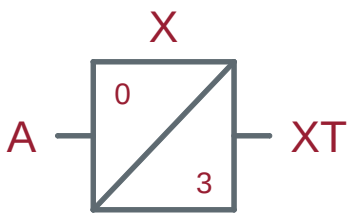


Timers



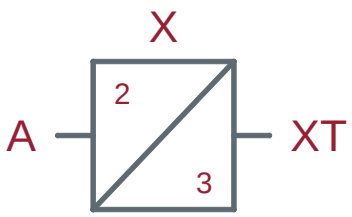
X = A
XPU = 2
XDO = 0

Timers

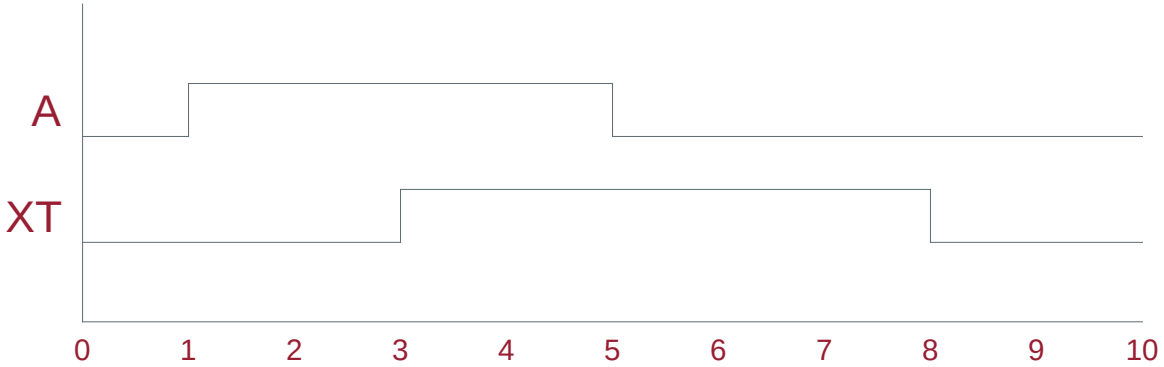


X = A
XPU = 0
XDO = 3

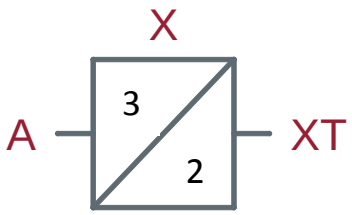
Timers

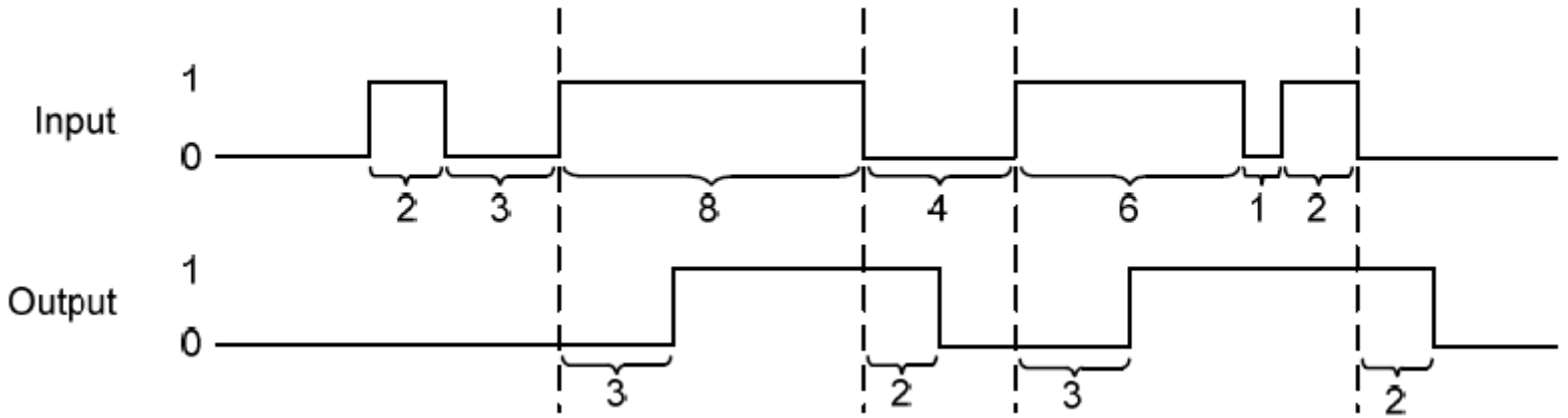


X = A
XPU = 2
XDO = 3

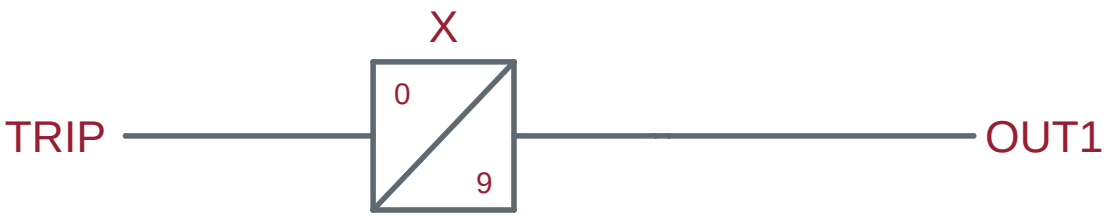


Timers

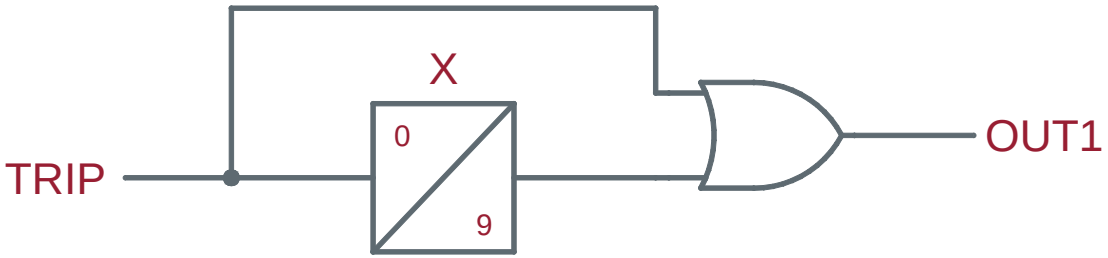




Timers

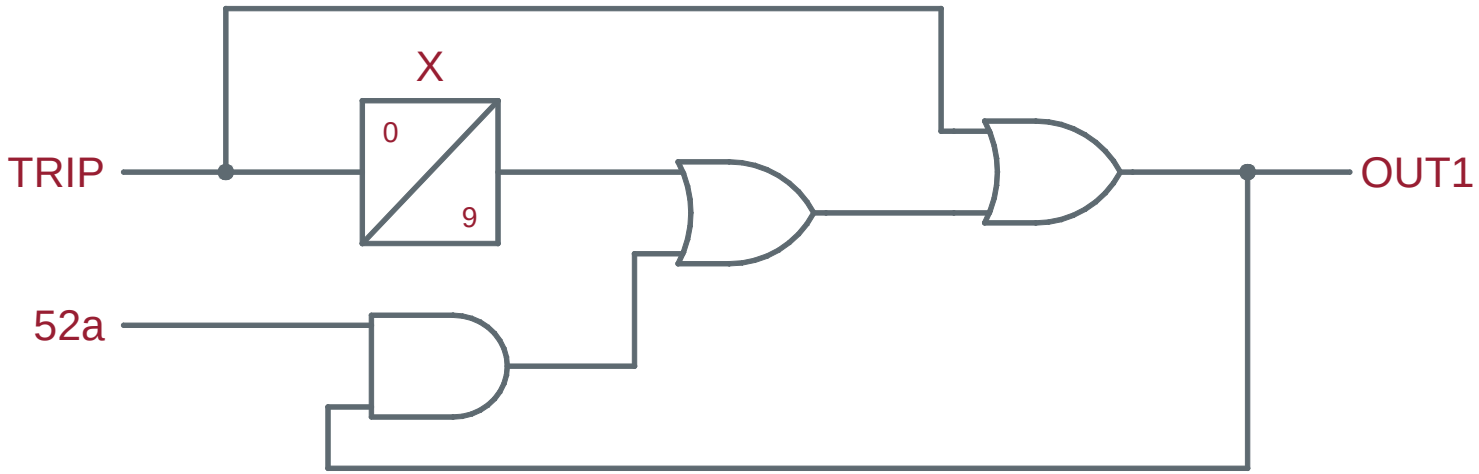


X = TRIP
OUT1 = XT
XPU = 0 cycles
XDO = 9 cycles



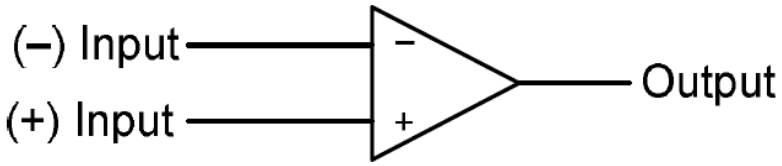
OUT1 = XT OR TRIP

Timers

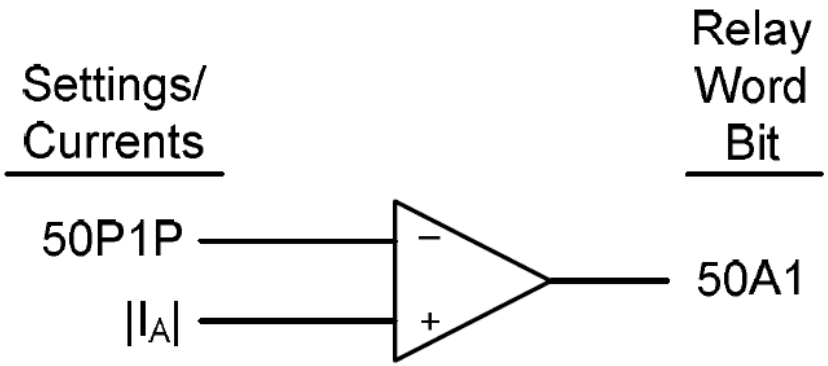


X = TRIP
OUT1 = TRIP + XT + 52a * OUT1
XPU = 0
XDO = 9

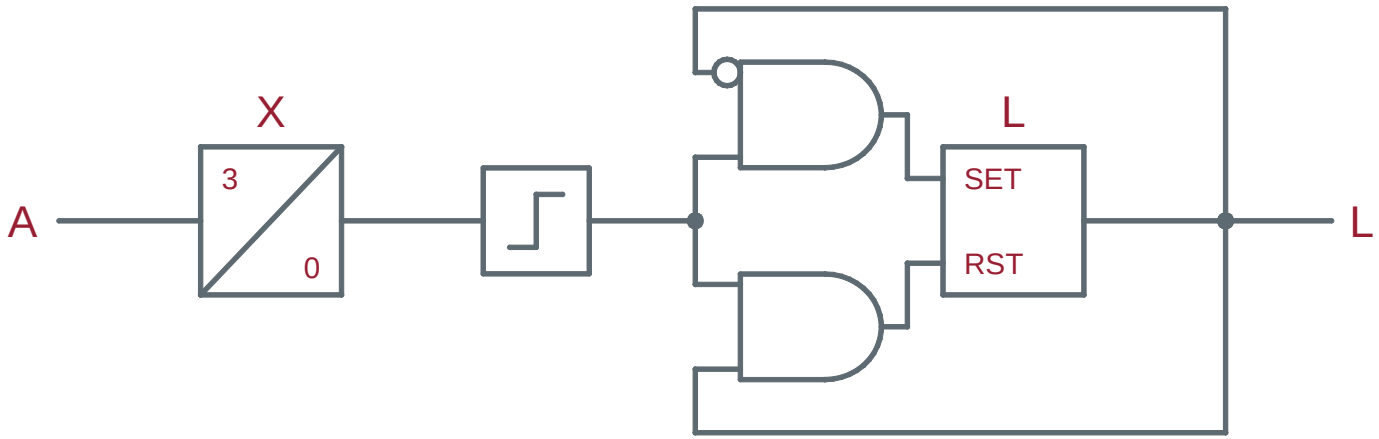
Comparator



Output = 1 if (+) Input > (-) Input

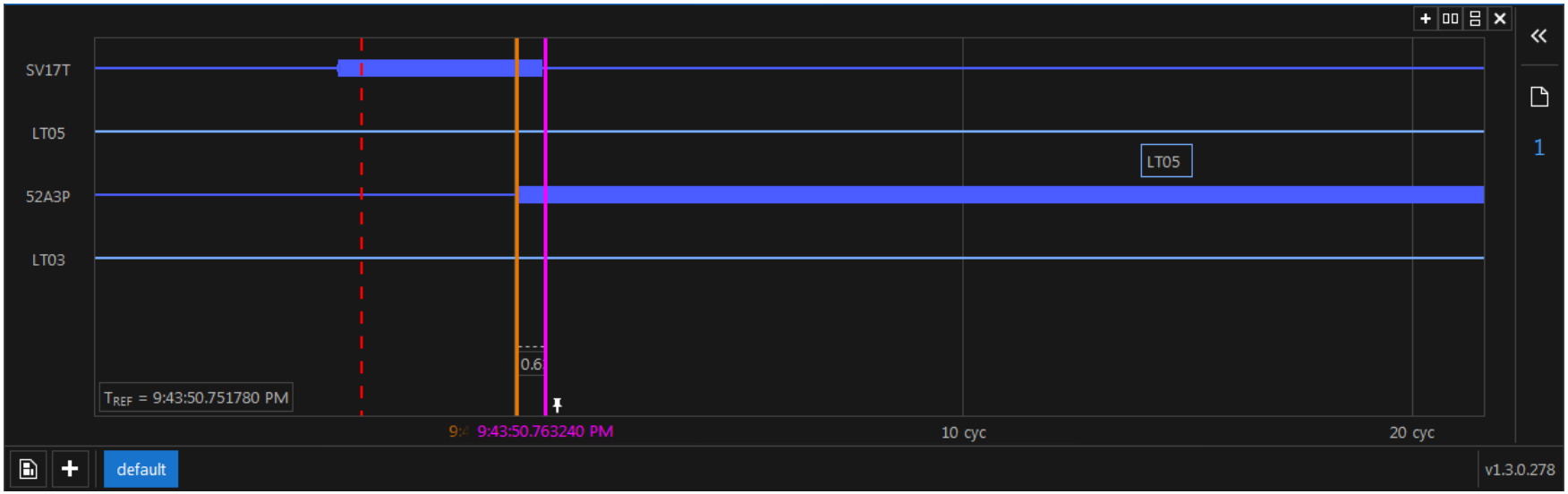


Complete Logic



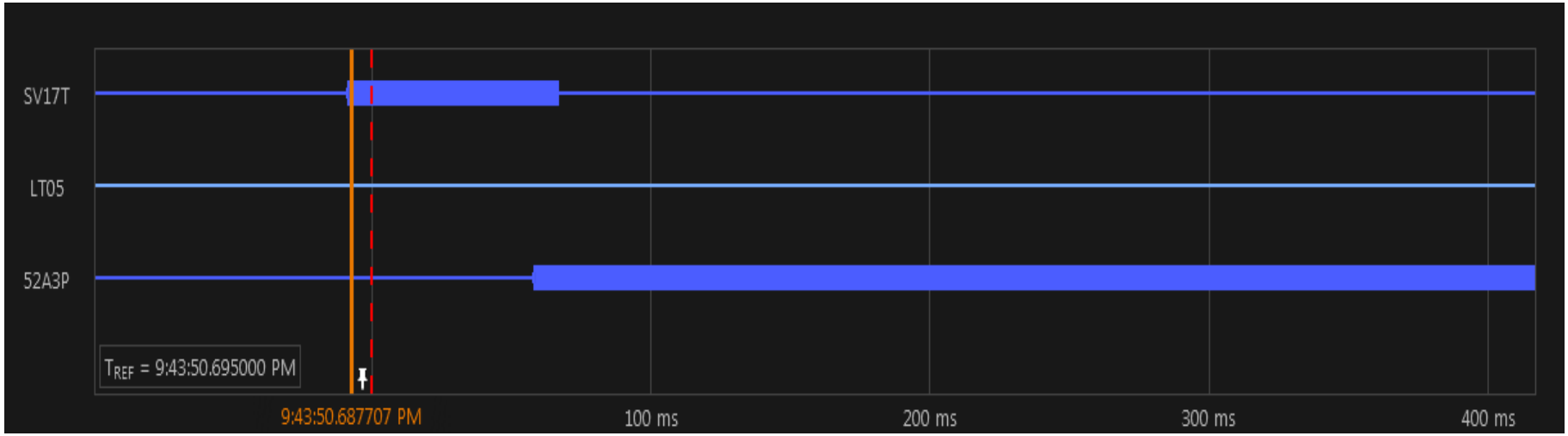
X=A
XPU = 3 seconds
XDO = 0 seconds
SETL = /XT * !L
RSTL = /XT * L

Example #1 – 43 Switch Enable on Close



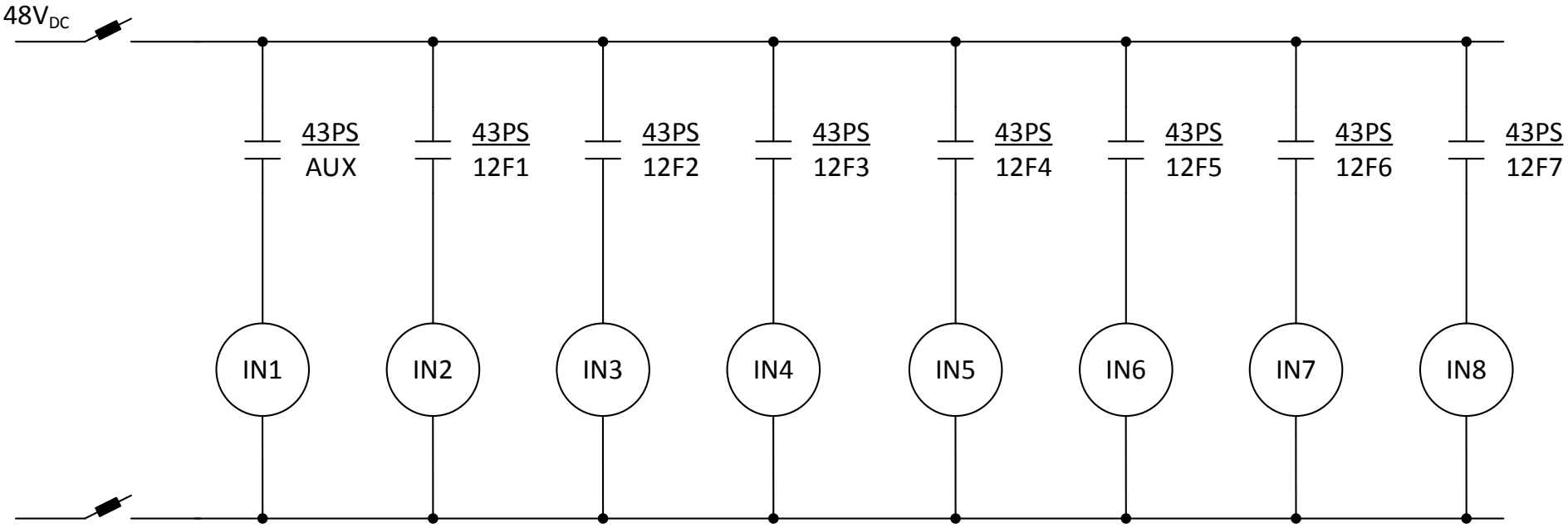
LT03 = F_TRIG SV17T AND NOT LT05 AND NOT 52A3P
= 1 * NOT (0) * NOT (1)
= 1 * 1 * 0
= 0

Example #1 – 43 Switch Enable on Close



LT03 = ~~F_TRIG~~ SV17T AND NOT LT05 AND NOT 52A3P
= 1 * NOT (0) * NOT (0)
= 1 * 1 * 1
= 1

Example #2


48V DC
43PS
AUX
43PS
12F1
43PS
12F2
43PS
12F3
43PS
12F4
43PS
12F5
43PS
12F6
43PS
12F7
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8

Example #2 cont.

IN1	IN2	IN3	POSITION	DECIMAL	BINARY
0	0	0	AUX	0	0
0	0	1	12F1	1	1
0	1	0	12F2	2	10
0	1	1	12F3	3	11
1	0	0	12F4	4	100
1	0	1	12F5	5	101
1	1	0	12F6	6	110
1	1	1	12F7	7	111

Example #2 cont.

Binary (Base 2)

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1024$

$2^{11} = 2048$

Slide Credit to Stephen Marx, BPA

Example #2 cont.

953 expressed in Decimal (Base 10)

$$953 = 9x10^2 + 5x10^1 + 3x10^0$$

$$9x100 + 5x10 + 3x1$$

$$900 + 50 + 3 = 953$$

Slide Credit to Stephen Marx, BPA

Example #2 cont.

Convert 1011 binary (base 2) to Decimal (base 10)

$$1011 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$$

$$1x8 + 0x4 + 1x2 + 1x1$$

$$8 + 0 + 2 + 1 = 11$$

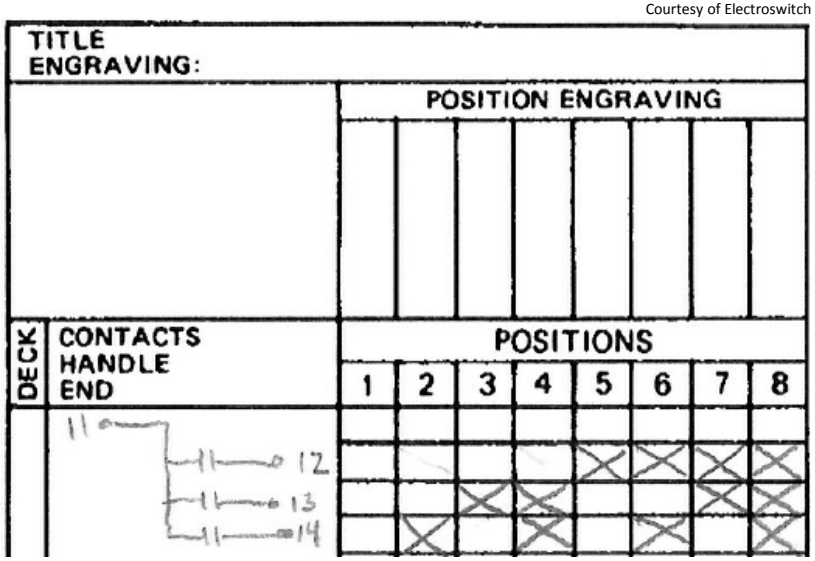
$$1011_2 = 11_{10}$$

Slide Credit to Stephen Marx, BPA

Example #2 cont.

X-CHART FOR INSTRUMENT & CONTROL SWITCH

Courtesy of Electroswitch



$2^0 \times 0 + 2^1 \times 0 + 2^2 \times 0 = 0$

$2^0 \times 1 + 2^1 \times 0 + 2^2 \times 0 = 1$

$2^0 \times 0 + 2^1 \times 1 + 2^2 \times 0 = 2$

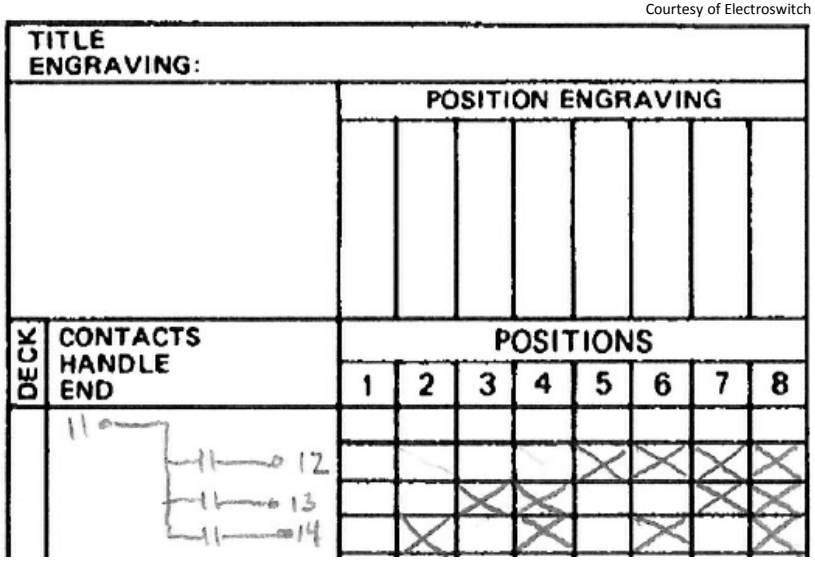
$2^0 \times 0 + 2^1 \times 0 + 2^2 \times 1 = 4$

$2^0 \times 1 + 2^1 \times 1 + 2^2 \times 1 = 7$

Example #2 cont.

X-CHART FOR INSTRUMENT & CONTROL SWITCH

Courtesy of Electroswitch



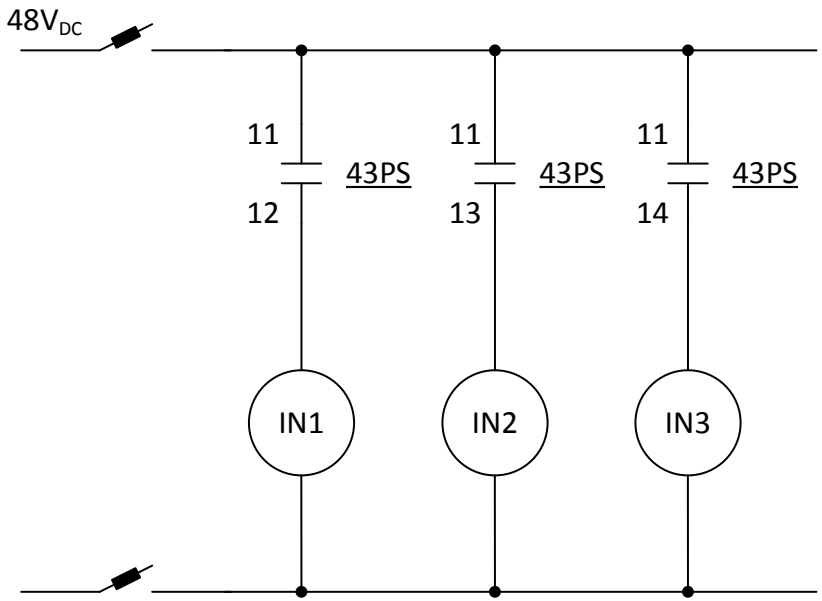
$2^0 \times 0 + 2^1 \times 0 + 2^2 \times 0 = 0$

$2^0 \times 1 + 2^1 \times 0 + 2^2 \times 0 = 1$

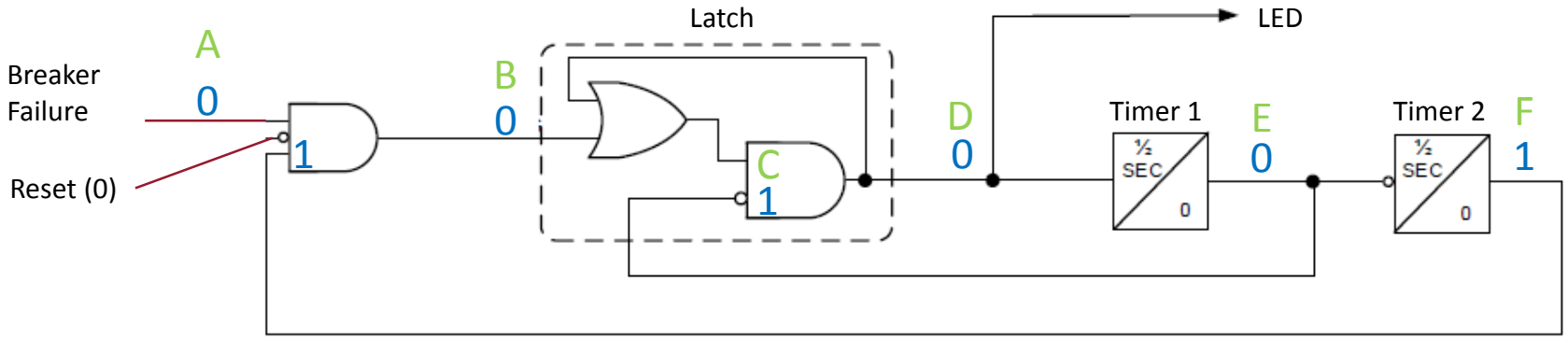
$2^0 \times 0 + 2^1 \times 1 + 2^2 \times 0 = 2$

$2^0 \times 0 + 2^1 \times 0 + 2^2 \times 1 = 4$

$2^0 \times 1 + 2^1 \times 1 + 2^2 \times 1 = 7$

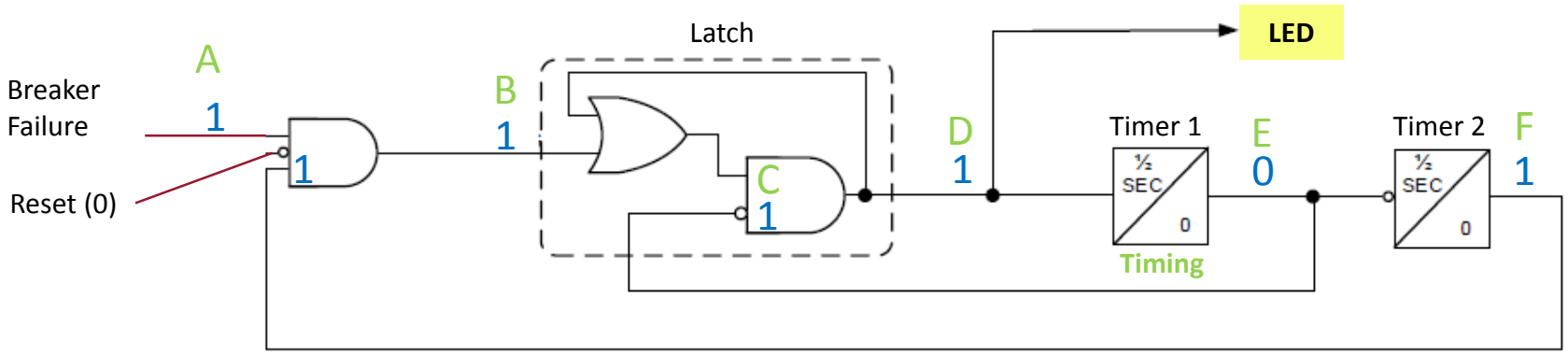


Example #3 – Breaker Failure “Blinking” LED



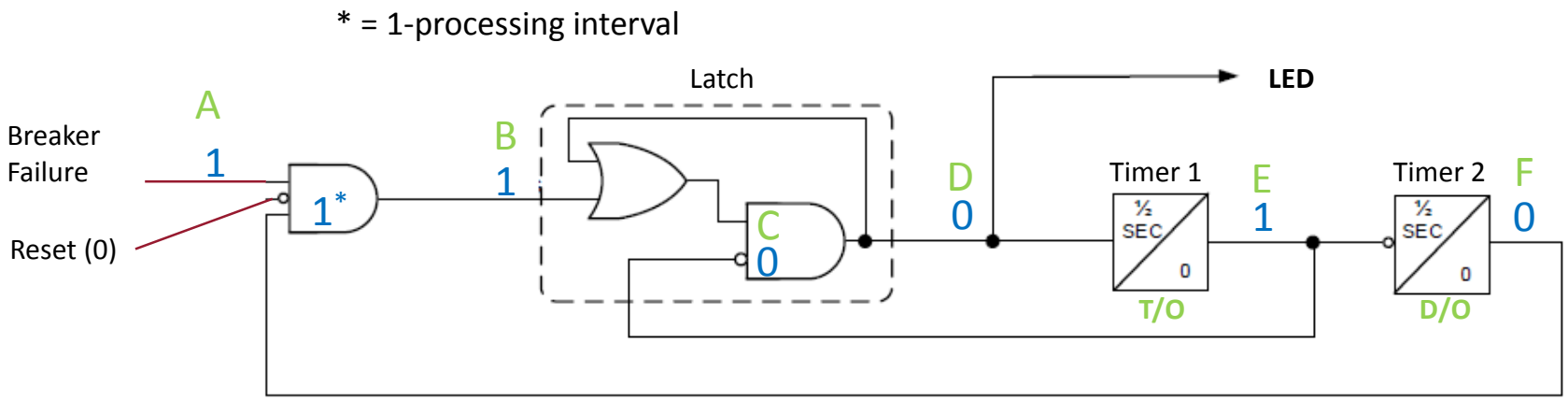
State	A	B	C	D	E	F	Timer 1	Timer 2
1	0	0	1	0	0	1	-	-
2	1	1	1	1	0	1	Timing	-
3	1	1	0	0	1	0	Time-Out	Drop-Out
4	1	0	0	0	0	0	Drop-Out	Timing
5	1	1	1	1	0	1	Timing	-

Example #3 – Breaker Failure “Blinking” LED



State	A	B	C	D	E	F	Timer 1	Timer 2
1	0	0	1	0	0	1	-	-
2	1	1	1	1	0	1	Timing	-
3	1	1	0	0	1	0	Time-Out	Drop-Out
4	1	0	0	0	0	0	Drop-Out	Timing
5	1	1	1	1	0	1	Timing	-

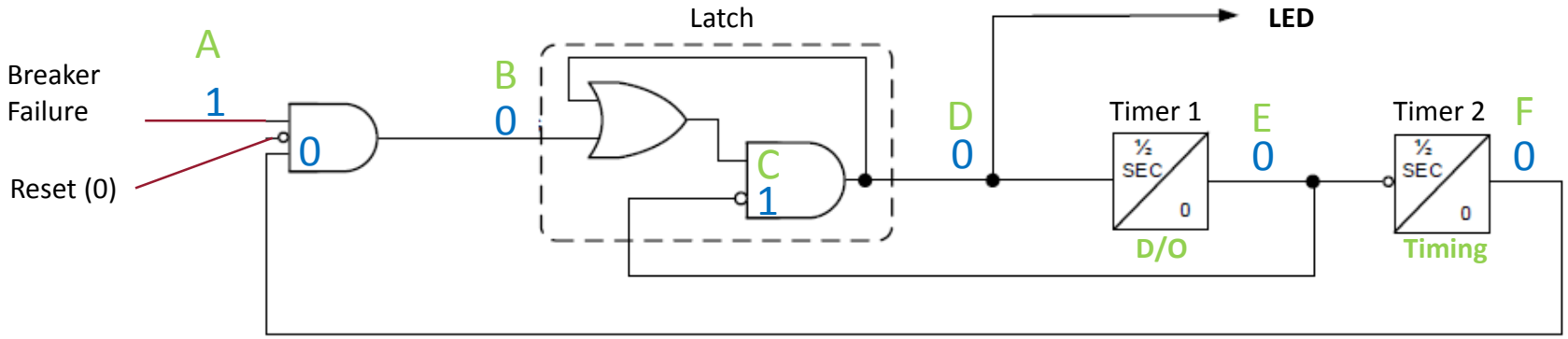
Example #3 – Breaker Failure "Blinking" LED



- After Timer #1 input asserted for ½ second

State	A	B	C	D	E	F	Timer 1	Timer 2
1	0	0	1	0	0	1	-	-
2	1	1	1	1	0	1	Timing	-
3	1	1	0	0	1	0	Time-Out	Drop-Out
4	1	0	0	0	0	0	Drop-Out	Timing
5	1	1	1	1	0	1	Timing	-

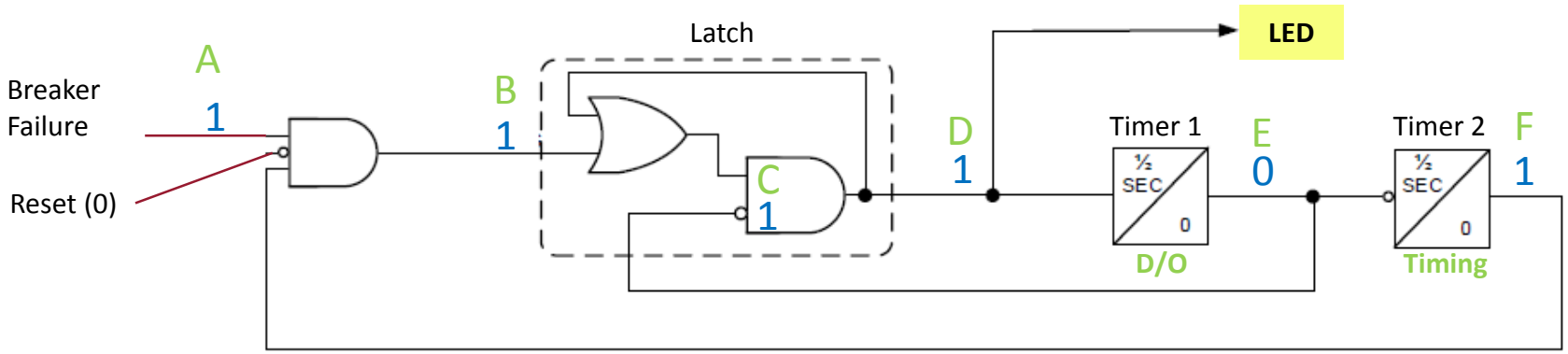
Example #3 – Breaker Failure “Blinking” LED



- After Timer #2 input asserted for ½ second

State	A	B	C	D	E	F	Timer 1	Timer 2
1	0	0	1	0	0	1	-	-
2	1	1	1	1	0	1	Timing	-
3	1	1	0	0	1	0	Time-Out	Drop-Out
4	1	0	0	0	0	0	Drop-Out	Timing
5	1	1	1	1	0	1	Timing	-

Example #3 – Breaker Failure “Blinking” LED

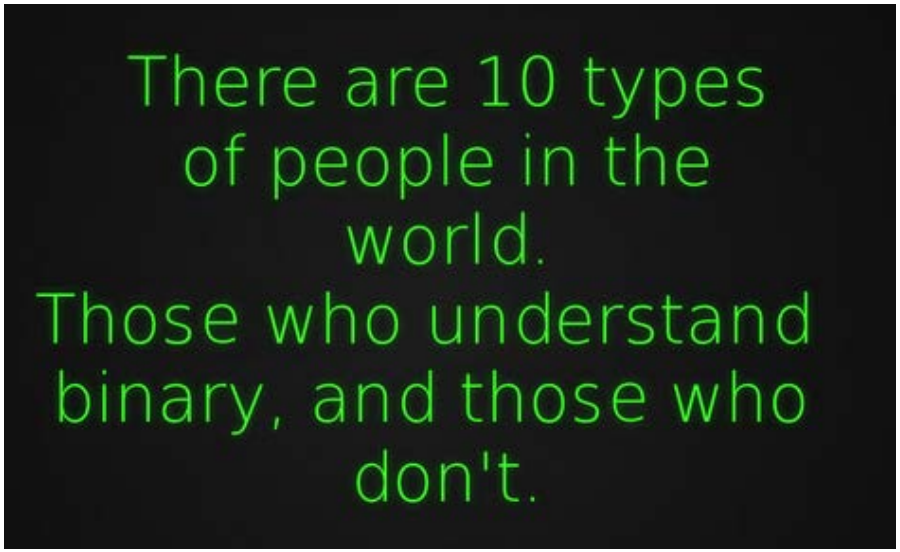


State	A	B	C	D	E	F	Timer 1	Timer 2
1	0	0	1	0	0	1	-	-
2	1	1	1	1	0	1	Timing	-
3	1	1	0	0	1	0	Time-Out	Drop-Out
4	1	0	0	0	0	0	Drop-Out	Timing
5	1	1	1	1	0	1	Timing	-

Recommendations on Testing

- Both positive and negative logic tests
- Understand state of return from power cycle/reset
- Always use the as-left relay settings to test output logic
    - All setting changes must be completed first.
- Use end-devices (breaker/disconnect switch staus) whenever possible.
- Question the design engineer!
- Ask for/compare against an Operating Description (if it exists)

Questions?



Kevin Damron, PE

Sr. Protection Engineer – Avista Utilities
Adjunct Professor – Gonzaga University
kevin.damron@avistacorp.com
509.495.4598

WASHINGTON STATE UNIVERSITY