Efficient **M**eta**s**tability-**C**ontaining Gray Code 2-sort

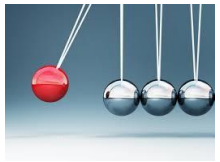
Moti Medina

Joint work with Christoph Lenzen

Max Planck Institute For Informatics, Saarbrucken, Germany

ASYNC 2016

In The Beginning there was **M**eta**s**tability



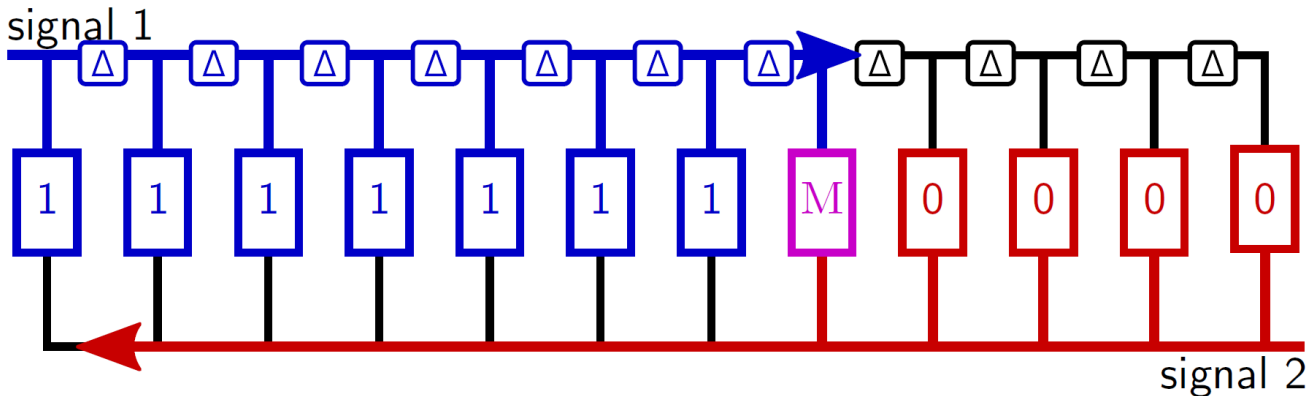
- Physics
  - Analog world (voltage) to 0-1 → non 0-1 equilibria = **M**eta**s**tability (Ms).
  - [Marino 81] Ms cannot be avoided, detected, or resolved.
  - ↑ clock rate → ↑ Pr. Metastability.
  - ↑ clock rate → ↑ #cc until metastability resolves.
  - ↑ #transistors → ↑ Pr. Metastability.
  - **Common solution: Wait!**

- Crossing clock domains
  - Signal arrives out of FF critical segment → Ms.

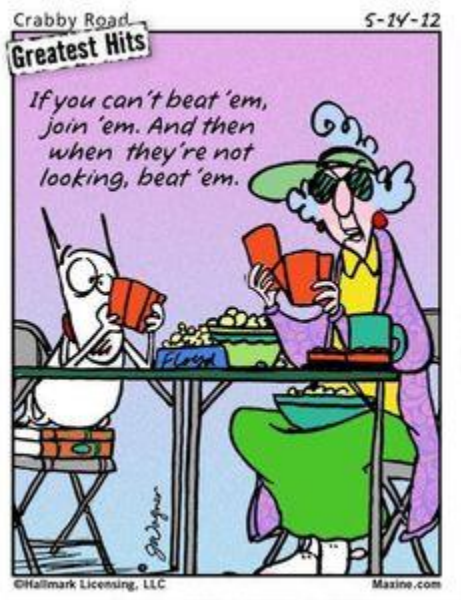
- **We want to eliminate this reason of Ms.**

Idea: Clock Synchronization
1st shot: a naïve “solution”

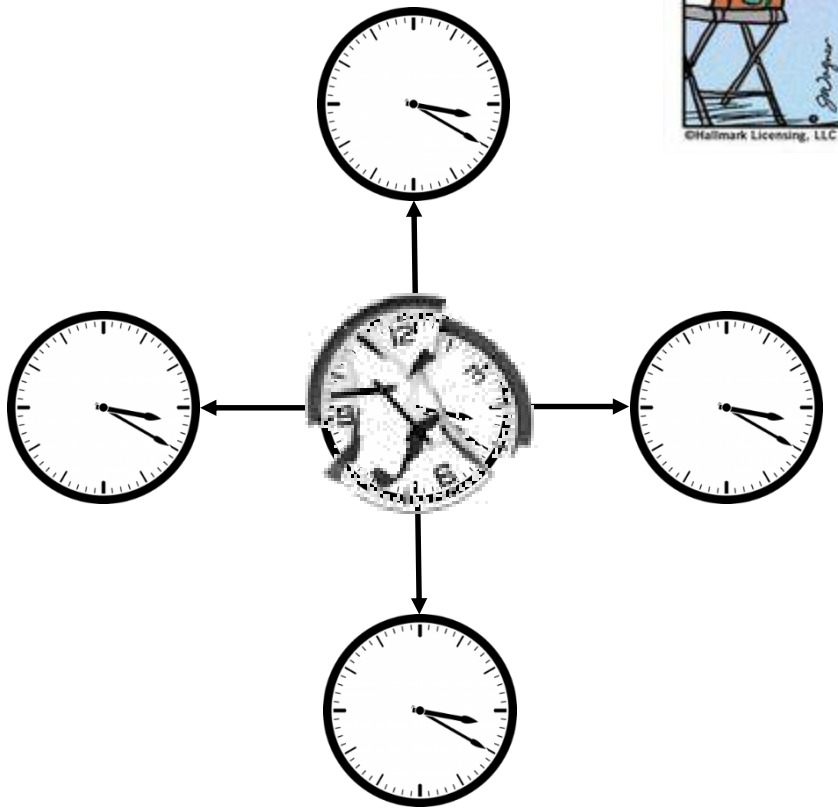
- 2 clocks
- Master-Slave approach.
- Slave clock measures Master’s clock.
  - Analog to Digital conversion = a TDC.
- Slave clock: Unary to Bin trans and update slave’s clock.
- Problem: Marino → Metastability.
- Conversion → Arbitrary clock update.
- **Problem: Clock synch. fails.**



2nd shot & **M**eta**s**tability-**C**ontaining



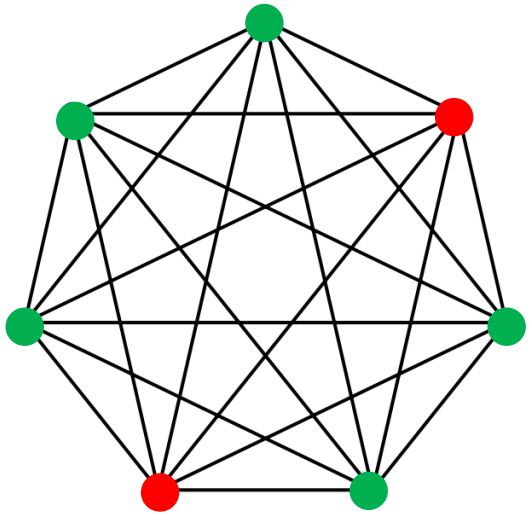
- "If you can't beat them, join them" → Contain metastability
- Unary string → analog correction of the oscillator.
  - We didn't lose any precision due to Ms = **metastable containing (Ms-C)**
- Master-slave approach→ nice property: bounded phase shift → crossing clock domains doesn't incur metastability.
- **Problem: the master clock fails → lost availability of a clock, i.e., not a fault tolerant system.**



Our Goal

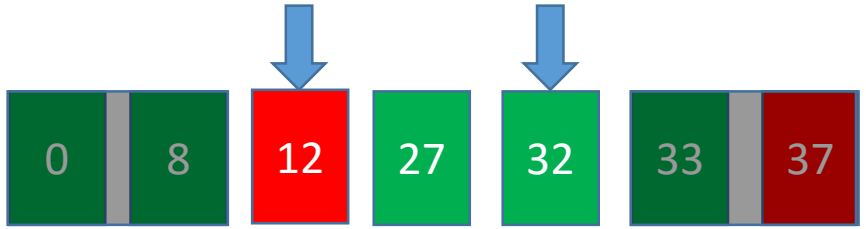
- Can we design a system which is
  - fault-tolerant, and
  - No loss of precision due to metastability?
    - i.e., Containing metastability

Yes to Fault-Tolerance!
The Lynch-Welch [88] Algorithm



- #number of faulty clocks: f
- #number of clocks: $n = 3f + 1$
- Example: $f = 2, n = 7$

- Algorithm's idea (animation):

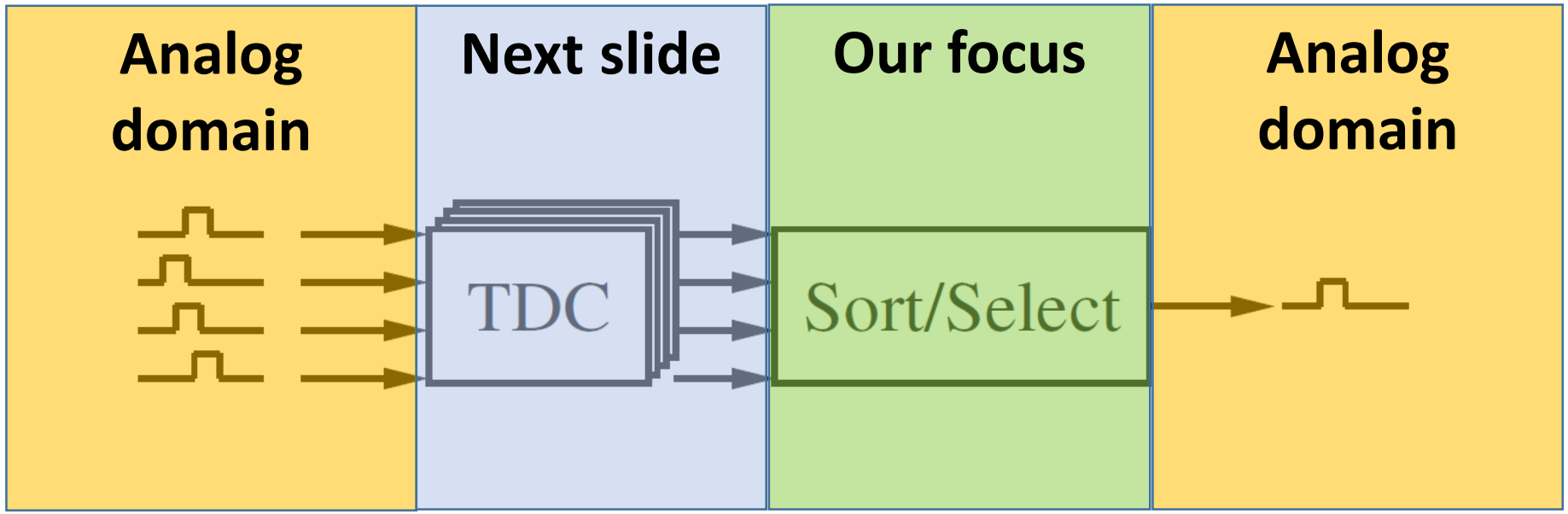


Implementation:

- In each clock (repeat forever)
  - Measure phase diff. from all the other clocks using TDCs.
  - **Sort** these measurements.
  - Adjust self clock to $\frac{diff[f+1]+diff[n-f]}{2}$.

The System

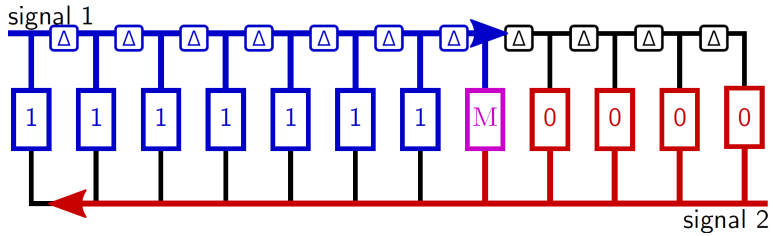
- Each clock has a $n-1$ TDCs.
- Each clock has a sorting network.



Our Time to Digital Converter (TDC) [Fuegger, Lenzen, Polzer 2016]

- (Ideal) TDC Specification:
  - Input: analog signals $a(t_1), b(t_2), t_2 > t_1$
  - Output: Gray code representation of $\lceil t_2 - t_1 \rceil$

- Corollary [Marino]: There is no ideal TDC!
- **[FLP2016] : TDC s.t. output contains 1 MS bit,**
  - **i.e., in unary $\mathbf{1}^*\mathbf{0}^* \cup \mathbf{1}^*\boldsymbol{M}\mathbf{0}^*$,**
  - **Mapping to Gray-code in the next slide.**



- **<u>Our question revisited</u>: Can we implement the LW algorithm? We cannot wait…**

Gray code & Valid (input/output) strings

- 3-bit Reflected Binary Gray Code

#	g[1]	g[2]	g[3]
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

U

- Valid (input/output) strings

g[1]	g[2]	g[3]
0	0	M
0	M	1
0	1	M
M	1	0
1	1	M
1	M	1
1	0	M

- Valid Unary to Valid Gray code is 1-2-1 & one bit is MS in both.
- $g[1] = M \rightarrow g[2:3]$ is the "biggest" number in two bits Gray code

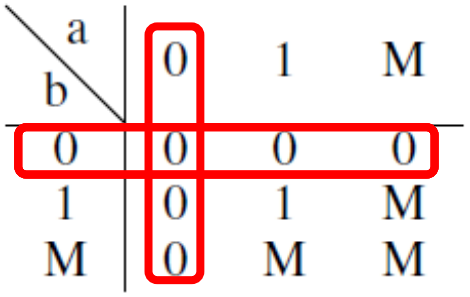
Our Question

- We need to sort!
- [LW88] does not solve the Ms-C issue.
- Sorting in a Ms-C manner + unary strings = exp. size circuits
  - Trivial min/max
  - Who is bigger? 100 or M00 ? 1M0 or 100?

- **Challenge: Find Ms-C Polynomial size circuits.**

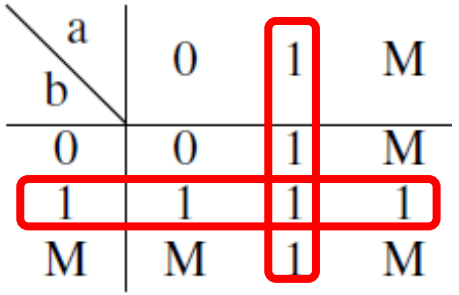
Metastability (Ms) model

- Worst case metastability gate propagation.
- Logical masking:
  - AND



a \ b	0	1	M
0	0	0	0
1	0	1	M
M	0	M	M

  - OR



a \ b	0	1	M
0	0	1	M
1	1	1	1
M	M	1	M

<u>Cost and Dealy of Combinational circ.:</u>

- Standard definitions.
- Cost = sum of basic gates' costs
- Delay = heaviest path from an input to an output.

$+ = OR$
$\cdot = AND$

Ms-C $2 - sort$ - Specification

- For u, v unary strings
  - $\max\limits_{un}\{u, v\} = u + v$
  - $\min\limits_{un}\{u, v\} = u \cdot v$
  - → **Ms-C**
- For u, v valid Gray code inputs
  - Let $\hat{u} \leftarrow un(u), \hat{v} \leftarrow un(v)$
  - $\max_{rg}\{u, v\} = rg\left(\max\limits_{un}\{\hat{u}, \hat{v}\}\right)$
  - $\min_{rg}\{u, v\} = rg\left(\min\limits_{un}\{\hat{u}, \hat{v}\}\right)$
  - → **Ms-C**

A MS-C Gray code $2 - sort(B)$ combinational circuit is defined as follows:

- **Input:** B bit valid gray code strings g, h
- **Output:** B bit valid gray code strings g', h'
- **Functionality:**
  - $g' \triangleq \max_{rg}\{g, h\}$
  - $h' \triangleq \min_{rg}\{g, h\}$

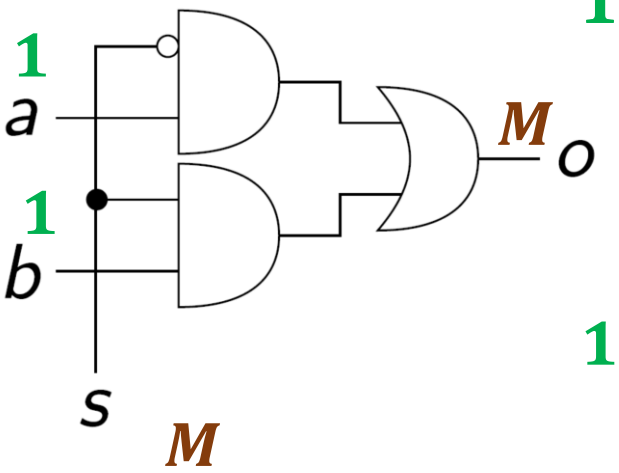
Our main result

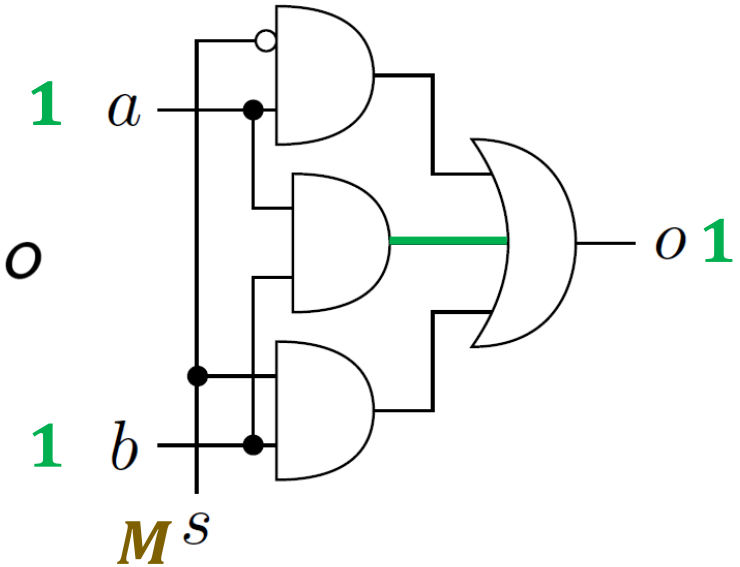
We propose an Ms-C circuit of

- Polynomial cost $O(B^2)$, and
- Linear depth $O(B)$
- that computes the $2 - sort(B)$ of two B-bit Gray code valid inputs.

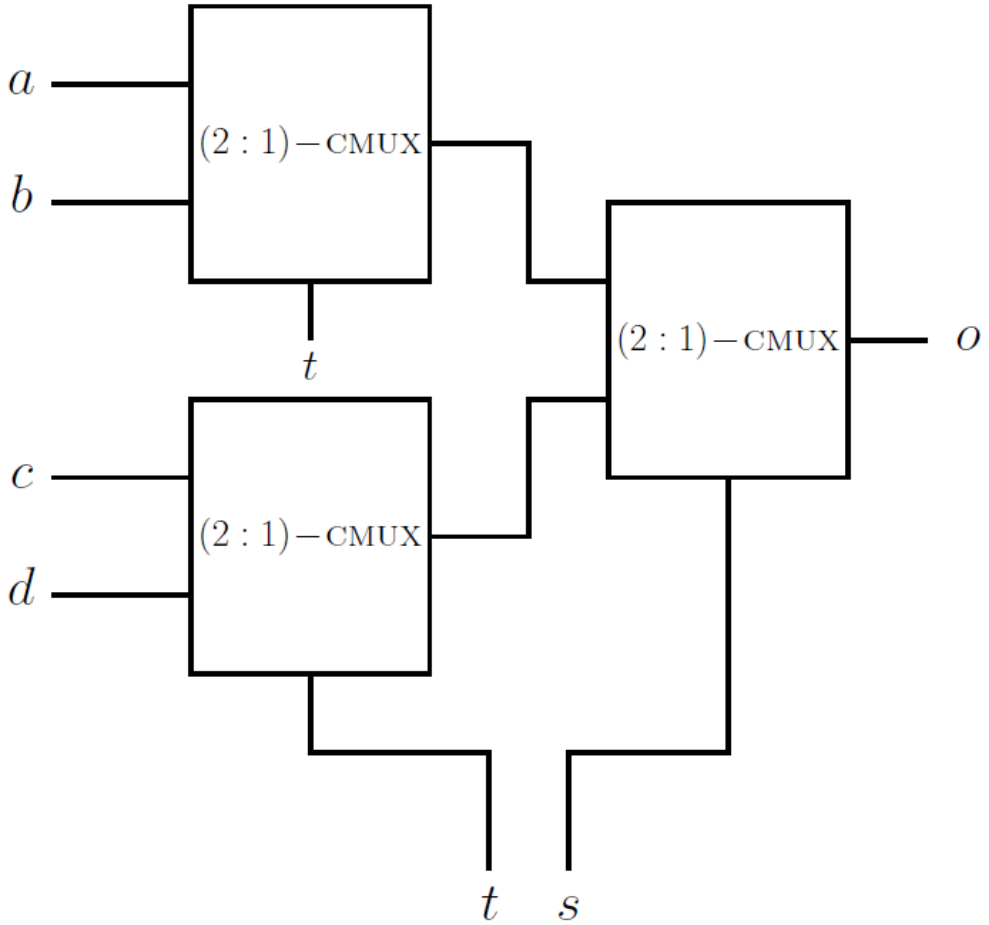
Ms-C Multiplexers

- $(2\colon 1)$-CMUX
  - Problem: output is Ms even if there is no ``choice''.
  - Solution: keep invariant s.t. if $a = b$ then we ``don't care'' about s





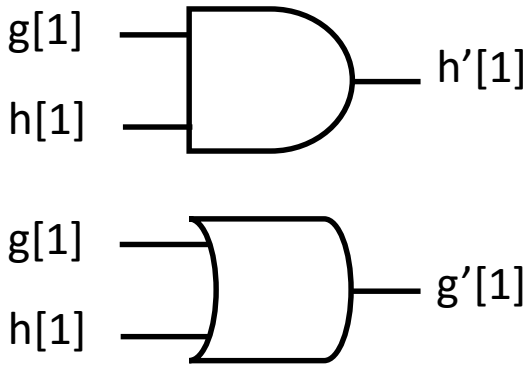
- $(2^k\colon 1)$-CMUX
  - The recursive extension keeps the invariant.



$+ = OR$
$\cdot = AND$

$2 - sort(B)$ Efficient & Ms-C Design: High Level Idea

- Recursive scheme.
- Directly operates on the Gray code representation.
- Simply branching on the first bit of the code → not Ms-C.



- Observation
  - 1st bit Ms→ stable bits are the max number.
- Observation + Ms-C Mux
  - both branching options are the same
  - → no additional Ms bits!
- Base case: $B = 1, u, v \in \{0,1\}$
  - $\max_{rg}\{u, v\} = u + v$
  - $\min_{rg}\{u, v\} = u \cdot v$

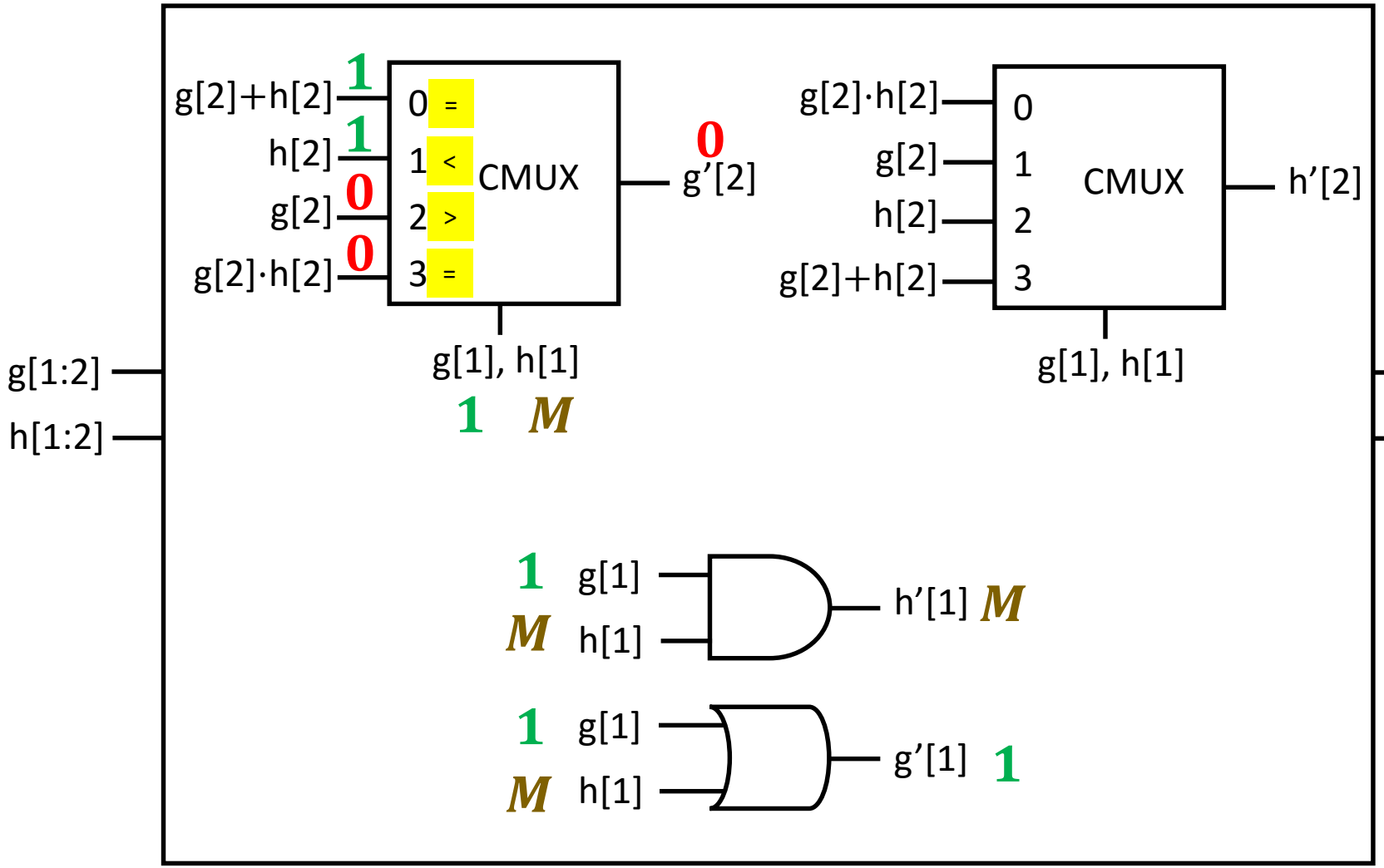
$2-sort(B)$ Design & Proof (by example)

00
01
11
10

$+ = OR$
$\cdot = AND$

g[1]	g[2]
0	0
1	0
1	0

h[1]	h[2]
1	1
M	1
1	M

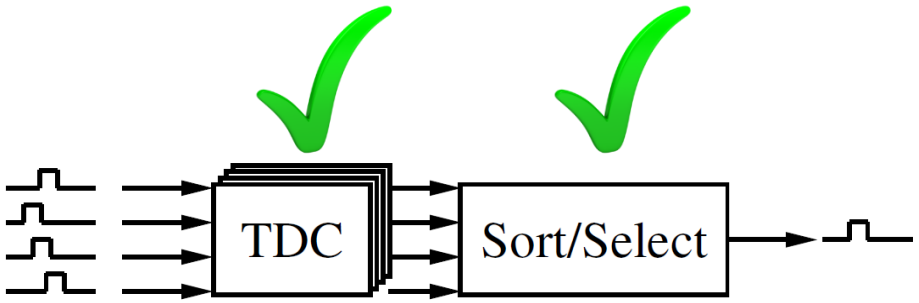


g'[1]	g'[2]
1	1
1	0
1	0

h'[1]	h'[2]
0	0
M	1
1	M

Conclusion

- Allowing for computations to take place **before** metastability is resolved.
- Analog measurement **precision is kept**.
- Building an asynchronous system **without synchronizers**.
  - By implementing the LW algorithm



Further research & Open questions

- More efficient $2 - sort(B)$.

NEW! $\log B$ delay $B\log B$ cost!

- Design a $2 - sort(B)$ that operates on Redundant Binary encoding .
- Super linear delay lower bound for Ms-C combinational circuits?
- Replace “Waiting” with Ms-C circuits.


That's all Folks!