# The Three-Wire Mutex Comparator

a primitive below the comparator, and what it buys

Original Result by **Heath Hunnicutt***,
with **Mavdil**†
Ruach Tov Collective

**Abstract**

We introduce a sorting primitive on three wires. Given that $x \le z$ already holds, the two comparators $(x, y)$ and $(y, z)$ are **mutually exclusive**: at most one can swap. They may therefore be evaluated on the same state and applied on the same clock edge without a write conflict, and counted as a single operation. Searching exhaustively over the extended vocabulary gives programs using 4 swap units for $n = 4$ and 6 for $n = 5$, against the classical 5 and 9; and a depth-4 program for $n = 5$, against the classical depth 5. Counting comparisons and swap units separately shows the trade: the mutex buys swap hardware at the price of comparisons. Every result is verified over all $n^n$ value tuples and all $n!$ permutations, with zero write conflicts.

## 1 How to count

A comparator is one comparison and one swap unit. A mutex is **two** comparisons and **one addressable** swap unit: after measuring $(x, y)$ and $(y, z)$, a branch decides whether that single swap-logic unit is connected to wires $\{x, y\}$, to $\{y, z\}$, or to neither.

*Remark* 1 (The mutex buys swap hardware, not comparisons). Counting operations alone hides the trade. Under the three-column model:

| | comparisons | swap units | depth |
|---|---|---|---|
| $n = 3$ classical | 3 | 3 | 3 |
| $n = 3$ mutex | 3 | **2** | **2** |
| $n = 4$ classical | 5 | 5 | 3 |
| $n = 4$ mutex | 6 | **4** | 3 |
| $n = 5$ classical | 9 | 9 | 5 |
| $n = 5$ mutex, fewest swaps | 10 | **6** | 6 |
| $n = 5$ mutex, least depth | 12 | 8 | **4** |

The mutex programs use **more** comparisons and **fewer** swap units. That is the real trade, and it is the right one wherever a swap costs more than a comparison — a wide datum, a memory move, a write port. At $n = 3$ the comparison count is unchanged and the saving is pure.

## 2 The primitive

**Definition 2** (Mutex comparator). For distinct wires $x, y, z$, the operation $\mathrm{mux}(x, y, z)$ evaluates both comparators $(x, y)$ and $(y, z)$ against the **same** input state and applies whichever one is out of order:

$$\mathrm{mux}(x, y, z): \quad \text{if } v_x > v_y \text{ then swap } (x, y); \quad \text{else if } v_y > v_z \text{ then swap } (y, z).$$

**Lemma 3** (Mutual exclusion). *If $v_x \le v_z$ then at most one of the two halves fires.*

*Proof.* Both fire only when $v_x > v_y$ and $v_y > v_z$, hence $v_x > v_z$, contradicting the hypothesis. □ □

The condition is not decoration. Both halves write $y$, so a simultaneous application would be a write conflict; the hypothesis $v_x \le v_z$ is exactly what makes the operation single-valued. We call it the **straddling fact**, since $(x, z)$ straddles the pair of comparators.

*Remark* 4 (The primitive is not two comparators relabelled). A mutex expands to two comparators, so counting it as one is a claim about **cost**, not about structure. The claim is that mutual exclusion makes it a single write port and a single clock edge. Measured against that, the $n = 5$ program below uses ten comparators when expanded — worse than the classical nine — and six operations when counted. The saving lives entirely in the exclusion.

## 3 The base case: $n = 3$

The classical $n = 3$ network is $(A, C)\,(A, B)\,(B, C)$: three comparators, and depth three, because $(A, B)$ and $(B, C)$ share wire $B$ and cannot occupy one layer.

With the mutex, $(A, C)$ establishes $A \le C$ — precisely the straddling fact for $\mathrm{mux}(A, B, C)$ — so the program is

$$(A, C); \quad \mathrm{mux}(A, B, C)$$

two operations, and two layers. Verified: after $(A, C)$, of all $64$ tuples on four values, **zero** have both halves wanting to fire.

## 4 Legality must be checked on permutations

*Remark* 5 (A trap worth naming). Both halves fire exactly when $v_x > v_y > v_z$ — a **strict three-chain**. Over $0$–$1$ vectors a strict three-chain is impossible:

| inputs with $v_0 > v_1 > v_2$ | count |
|---|---|
| $0$–$1$ vectors | **0** |
| permutations | 1 of 6 |

So a search that carries $0$–$1$ states will judge **every** mutex legal, including one placed before anything is established. This produced an apparently valid depth-4 program for $n = 5$ whose first layer contained a mutex; it double-writes on 20 of 120 permutations. The $0$–$1$ principle is sound for **deciding whether a network sorts**; it is not sound for this. All searches below carry permutations.

## 5 Fewest swap units

Exhaustive search over the vocabulary — comparators, plus mutexes legal in the current state — with iterative deepening on cost.

### 5.1 $n = 4$: four swap units, against $S(4) = 5$

$$(w_0, w_1); \quad \mathrm{mux}(w_0, w_2, w_1); \quad (w_1, w_3); \quad \mathrm{mux}(w_0, w_1, w_2)$$

### 5.2 $n = 5$: six swap units, against $S(5) = 9$

$(w_0, w_1);\ \mathrm{mux}(w_0, w_2, w_1);\ \mathrm{mux}(w_0, w_3, w_1);\ (w_1, w_4);\ \mathrm{mux}(w_0, w_1, w_3);\ \mathrm{mux}(w_1, w_2, w_3)$

| program | tuples tested | unsorted | write conflicts |
|---|---|---|---|
| $n = 4$, 4 swap-units | $4^4 = 256$ | 0 | 0 |
| $n = 5$, 6 swap-units | $5^5 = 3125$ | 0 | 0 |

Both also verified over all permutations.

## 6 Optimal depth for $n = 5$

A layer is a set of operations on pairwise disjoint wires. A mutex occupies three wires and a comparator two, so at $n = 5$ the richest layer is one of each — two mutexes would need six wires.

| layer | operations |
|---|---|
| 1 | $(w_0, w_1)$, $(w_2, w_3)$ |
| 2 | $\mathrm{mux}(w_0, w_2, w_1)$, $(w_4, w_3)$ |
| 3 | $\mathrm{mux}(w_0, w_4, w_2)$, $(w_3, w_1)$ |
| 4 | $\mathrm{mux}(w_0, w_1, w_4)$, $(w_2, w_3)$ |

**Depth 4, against the classical depth 5.** Zero unsorted and zero conflicts over all $3125$ tuples and all $120$ permutations. Each mutex was checked against the state that actually reaches it:

| layer | mutex | states that would double-fire |
|---|---|---|
| 2 | $\mathrm{mux}(0, 2, 1)$, straddles $(0, 1)$ | 0 |
| 3 | $\mathrm{mux}(0, 4, 2)$, straddles $(0, 2)$ | 0 |
| 4 | $\mathrm{mux}(0, 1, 4)$, straddles $(0, 4)$ | 0 |

*Remark* 6 (Layer one holds no mutex). A mutex is legal only where its straddling fact is established, and at the start nothing is. So layer 1 is comparators only — as the search independently found. This is why the depth does not fall to the counting bound of 3: a comparator offers 2 outcomes and a mutex 3, so the richest layer carries $\log_2 6 \approx 2.585$ bits against $\log_2 120 \approx 6.907$ needed, permitting 3; the first layer's restriction to $2$ bits costs the fourth.

*Remark* 7 (The two optima are different programs). The depth-4 program uses 8 swap units and 12 comparisons; the 6-swap program has depth 6 and 10 comparisons. As with classical networks, the objectives are not simultaneously attainable.

## 7 What remains

The straddling fact is exactly a cell of an ordering calculus: **is $x \le z$ established at this point?** A verifier that carries the intermediate ordering state is therefore a legality checker for this primitive, and the search above is that checker run constructively.

Two directions are open. Whether the size and depth advantages persist beyond $n = 5$ is unmeasured, and two sizes is not a trend. And the primitive admits a sharper machine realisation — comparisons hoisted to a single wide issue, with the results carried through an exchange rather than recomputed — which is the subject of the next chapter.