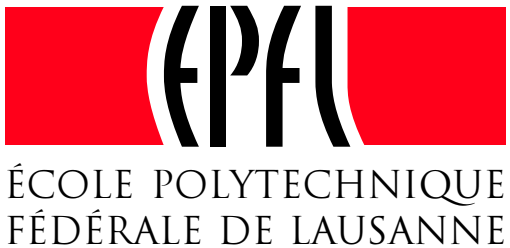
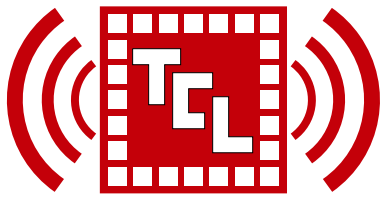
On Metric Sorting for Successive Cancellation List Decoding of Polar Codes

Alexios Balatsoukas-Stimming,[1] Mani Bastani Parizi,[2] Andreas Burg[1]

1: Telecommunications Circuits Laboratory, EPFL
2: Information Theory Laboratory, EPFL





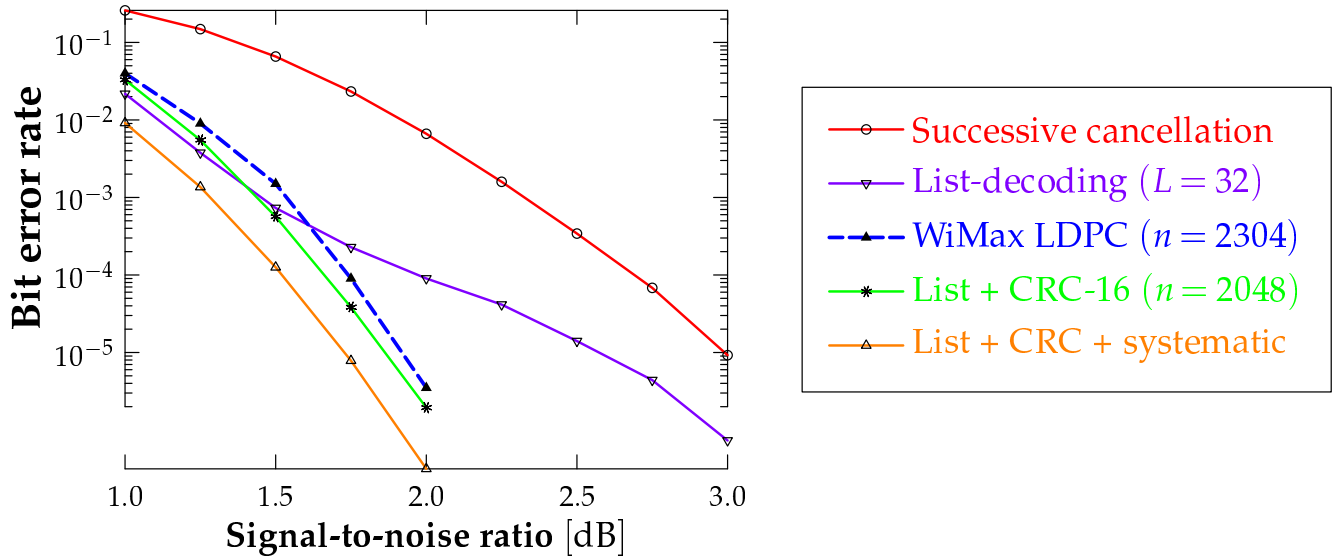
ISCAS 2015, Lisbon

Motivation

- Polar codes provably achieve the capacity of any binary symmetric channel under low-complexity successive cancellation decoding *asymptotically*.

Motivation

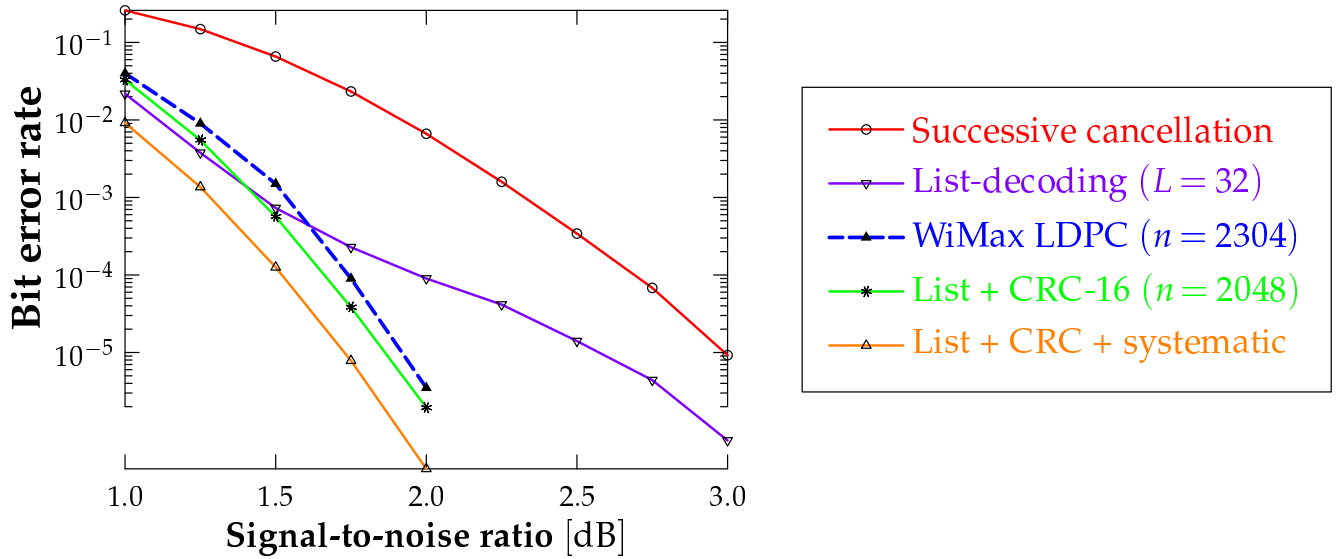
- Polar codes provably achieve the capacity of any binary symmetric channel under low-complexity successive cancellation decoding *asymptotically*.
- Successive cancellation list decoding to improve their performance at low-to-moderate block-lengths.



Tal, I., Vardy, A. "List Decoding of Polar Codes", *IEEE Trans. Inf. Theory*, May 2015.

Motivation

- Polar codes provably achieve the capacity of any binary symmetric channel under low-complexity successive cancellation decoding *asymptotically*.
- Successive cancellation list decoding to improve their performance at low-to-moderate block-lengths.

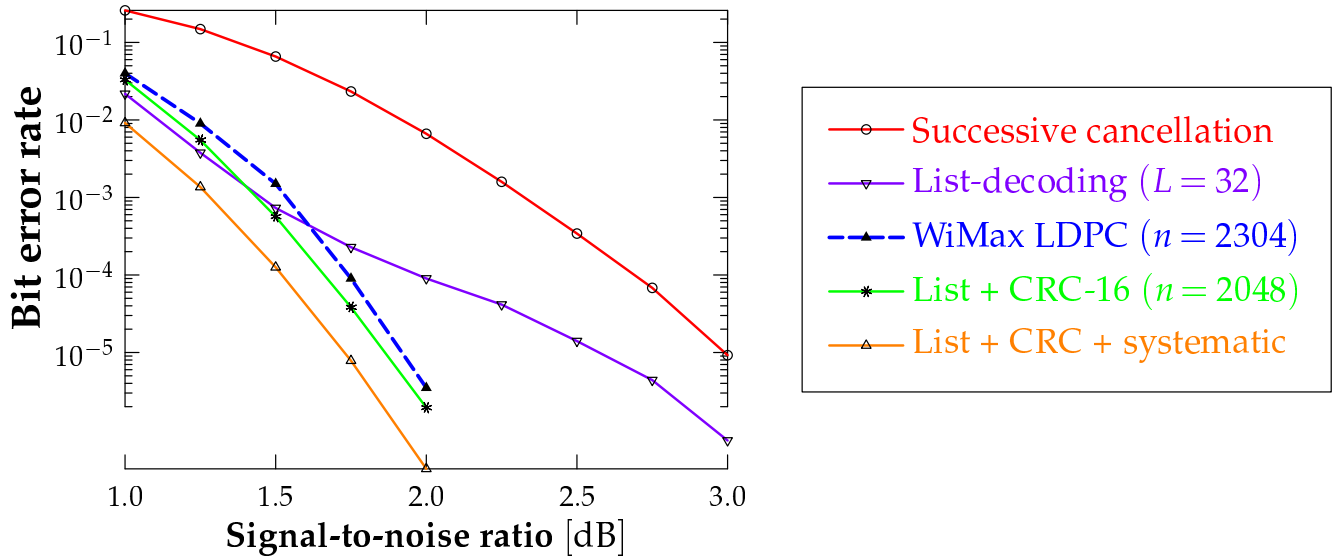


Tal, I., Vardy, A. "List Decoding of Polar Codes", *IEEE Trans. Inf. Theory*, May 2015.

- Main step of SCL decoding: choose L best paths out of $2L$ possible ones:

Motivation

- Polar codes provably achieve the capacity of any binary symmetric channel under low-complexity successive cancellation decoding *asymptotically*.
- Successive cancellation list decoding to improve their performance at low-to-moderate block-lengths.

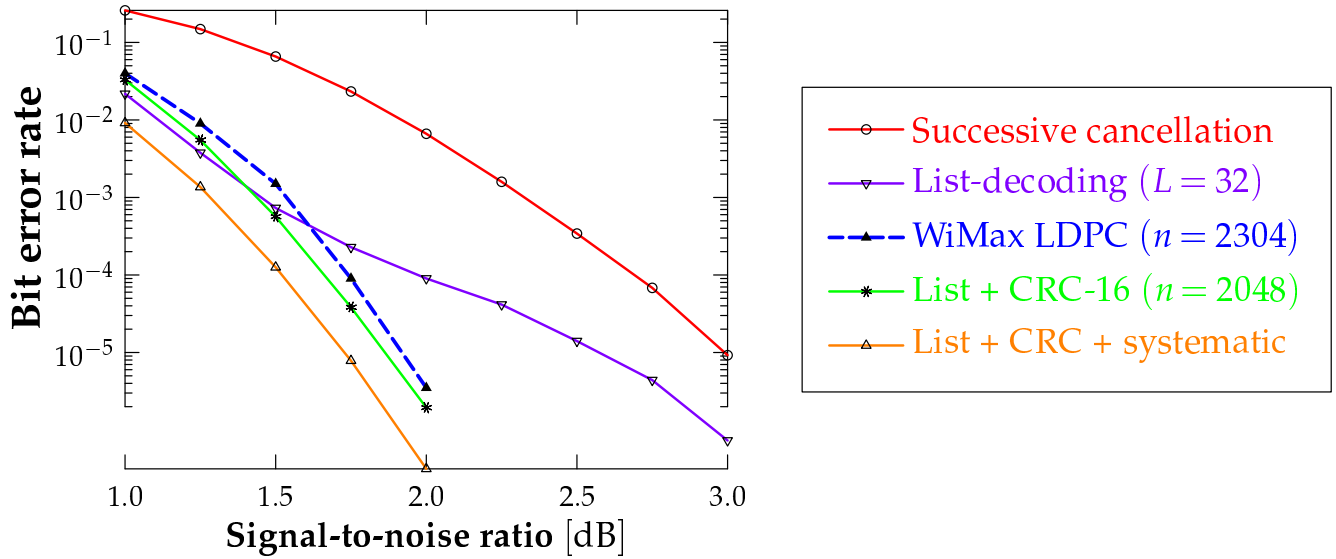


Tal, I., Vardy, A. "List Decoding of Polar Codes", *IEEE Trans. Inf. Theory*, May 2015.

- Main step of SCL decoding: choose L best paths out of $2L$ possible ones:
  - Each tentative path is associated with a metric

Motivation

- Polar codes provably achieve the capacity of any binary symmetric channel under low-complexity successive cancellation decoding *asymptotically*.
- Successive cancellation list decoding to improve their performance at low-to-moderate block-lengths.



Tal, I., Vardy, A. "List Decoding of Polar Codes", *IEEE Trans. Inf. Theory*, May 2015.

- Main step of SCL decoding: choose L best paths out of $2L$ possible ones:
  - Each tentative path is associated with a metric
  - The list of $2L$ path metrics is sorted.
  - First L metrics in the sorted list correspond to the surviving paths.

Contribution

- The metric sorter is a **crucial** component which quickly becomes the critical path as L is increased.

Contribution

- The metric sorter is a **crucial** component which quickly becomes the critical path as L is increased.

- We leverage the properties of an LLR-based formulation of SCL decoding to propose various simplified metric sorters:

Contribution

- The metric sorter is a **crucial** component which quickly becomes the critical path as L is increased.

- We leverage the properties of an LLR-based formulation of SCL decoding to propose various simplified metric sorters:
  1. Radix-$2L$ sorter $\rightarrow$ **Pruned** radix-$2L$ sorter.

Contribution

- The metric sorter is a **crucial** component which quickly becomes the critical path as L is increased.

- We leverage the properties of an LLR-based formulation of SCL decoding to propose various simplified metric sorters:
  1. Radix-$2L$ sorter $\rightarrow$ **Pruned** radix-$2L$ sorter.
  2. Bitonic sorter $\rightarrow$ **Pruned** bitonic sorter.

Contribution

- The metric sorter is a **crucial** component which quickly becomes the critical path as L is increased.

- We leverage the properties of an LLR-based formulation of SCL decoding to propose various simplified metric sorters:
  1. Radix-$2L$ sorter $\rightarrow$ **Pruned** radix-$2L$ sorter.
  2. Bitonic sorter $\rightarrow$ **Pruned** bitonic sorter.
  3. An efficient implementation of **bubble sorter**.

Contribution

- The metric sorter is a **crucial** component which quickly becomes the critical path as L is increased.

- We leverage the properties of an LLR-based formulation of SCL decoding to propose various simplified metric sorters:
  1. Radix-$2L$ sorter $\rightarrow$ **Pruned** radix-$2L$ sorter.
  2. Bitonic sorter $\rightarrow$ **Pruned** bitonic sorter.
  3. An efficient implementation of **bubble sorter**.

Significant improvement in operating frequency and area

Contribution

- The metric sorter is a **crucial** component which quickly becomes the critical path as L is increased.

- We leverage the properties of an LLR-based formulation of SCL decoding to propose various simplified metric sorters:
  1. Radix-$2L$ sorter $\rightarrow$ **Pruned** radix-$2L$ sorter.
  2. Bitonic sorter $\rightarrow$ **Pruned** bitonic sorter.
  3. An efficient implementation of **bubble sorter**.

Significant improvement in operating frequency and area

Different list size $\rightarrow$ **Different** optimal sorter choice

Problem Statement

- Let

$$m_0, m_1, m_2, m_3, \ldots, m_{2L-2}, m_{2L-1}$$

be the list of $2L$ metrics to be sorted.

Problem Statement

- Let

$$\mu_0, m_1, \mu_1, m_3, \ldots, \mu_{L-1}, m_{2L-1}$$

  be the list of $2L$ metrics to be sorted.
- In an LLR-based formulation of the SCL decoder,
  - half of the metrics previously existing path metrics which are already sorted,

$$\mu_0 \leq \mu_1 \leq \cdots \leq \mu_{L-1}.$$

Problem Statement

- Let

$$\mu_0, \mu_0 + a_0, \mu_1, \mu_1 + a_1, \ldots, \mu_{L-1}, \mu_{L-1} + a_{L-1}$$

  be the list of $2L$ metrics to be sorted.
- In an LLR-based formulation of the SCL decoder,
  - half of the metrics previously existing path metrics which are already sorted,

$$\mu_0 \leq \mu_1 \leq \cdots \leq \mu_{L-1}.$$

  - the other half are obtained by adding non-negative numbers $a_0, a_1, \ldots, a_{L-1}$ to $\mu_0, \mu_1, \ldots, \mu_{L-1}$.

Problem Statement

- Let

$$[\mu_0, \mu_0 + a_0, \mu_1, \mu_1 + a_1, \ldots, \mu_{L-1}, \mu_{L-1} + a_{L-1}]$$
$$\triangleq [m_0, m_1, m_2, m_3, \ldots, m_{2L-2}, m_{2L-1}]$$

  be the list of $2L$ metrics to be sorted.
- In an LLR-based formulation of the SCL decoder,
  - half of the metrics previously existing path metrics which are already sorted,

$$\mu_0 \leq \mu_1 \leq \cdots \leq \mu_{L-1}.$$

  - the other half are obtained by adding non-negative numbers $a_0, a_1, \ldots, a_{L-1}$ to $\mu_0, \mu_1, \ldots, \mu_{L-1}$.

- **Known relations:**

1. $m_{2\ell} \leq m_{2(\ell+1)}$

   for all $\ell \in \{0, 1, \ldots, 2L-2\}$.

2. $m_{2\ell} \leq m_{2\ell+1}$

Problem Statement

- Let

$$[\mu_0, \mu_0 + a_0, \mu_1, \mu_1 + a_1, \ldots, \mu_{L-1}, \mu_{L-1} + a_{L-1}]$$
$$\triangleq [m_0, m_1, m_2, m_3, \ldots, m_{2L-2}, m_{2L-1}]$$

  be the list of $2L$ metrics to be sorted.
- In an LLR-based formulation of the SCL decoder,
  - half of the metrics previously existing path metrics which are already sorted,

$$\mu_0 \leq \mu_1 \leq \cdots \leq \mu_{L-1}.$$

  - the other half are obtained by adding non-negative numbers $a_0, a_1, \ldots, a_{L-1}$ to $\mu_0, \mu_1, \ldots, \mu_{L-1}$.

- **Known relations:**
  1. $m_{2\ell} \leq m_{2(\ell+1)}$
  2. $m_{2\ell} \leq m_{2\ell+1}$

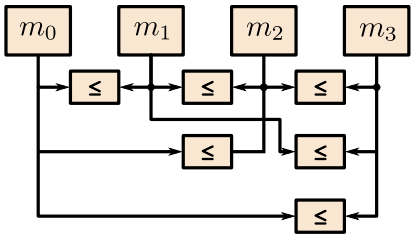
  for all $\ell \in \{0, 1, \ldots, 2L-2\}$.
- We don't need to sort the entire list.
  It is sufficient to find a sorted list of the L smallest elements of the list.

Radix-$2L$ and Pruned Radix-$2L$ Sorters

- **Radix-$2L$ sorter**

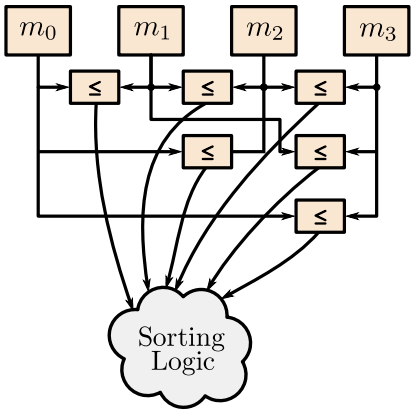
Radix-$2L$ and Pruned Radix-$2L$ Sorters

- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators



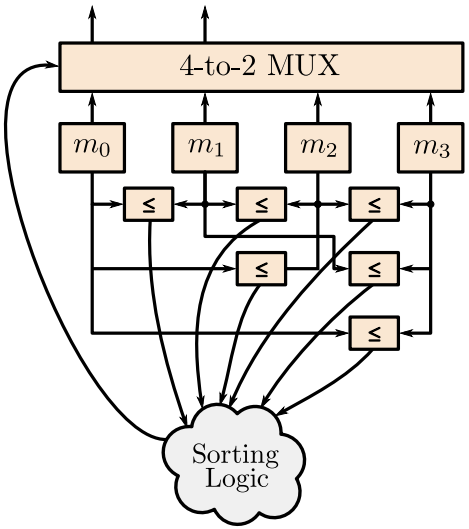
Radix-$2L$ and Pruned Radix-$2L$ Sorters

- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function



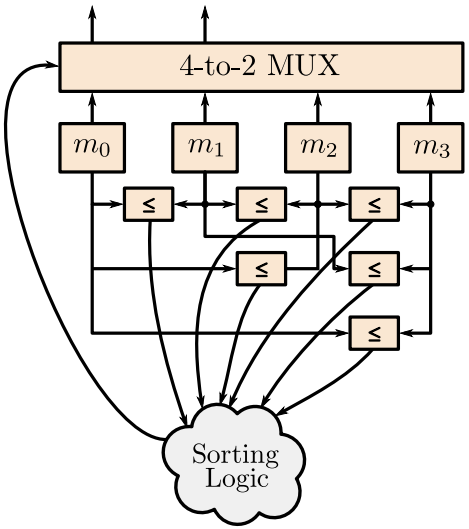
Radix-$2L$ and Pruned Radix-$2L$ Sorters

- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



Radix-$2L$ and Pruned Radix-$2L$ Sorters

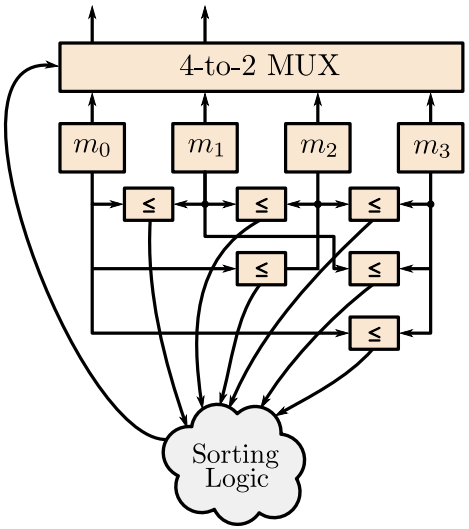
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Radix-$2L$ and Pruned Radix-$2L$ Sorters

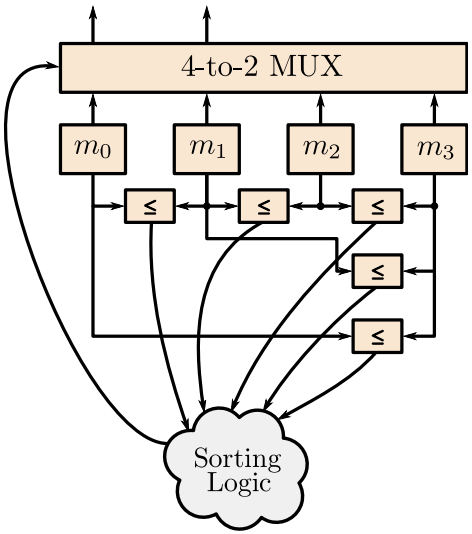
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)} \Longrightarrow m_0 \leq m_2$
2. $m_{2\ell} \leq m_{2\ell+1}$

Radix-$2L$ and Pruned Radix-$2L$ Sorters

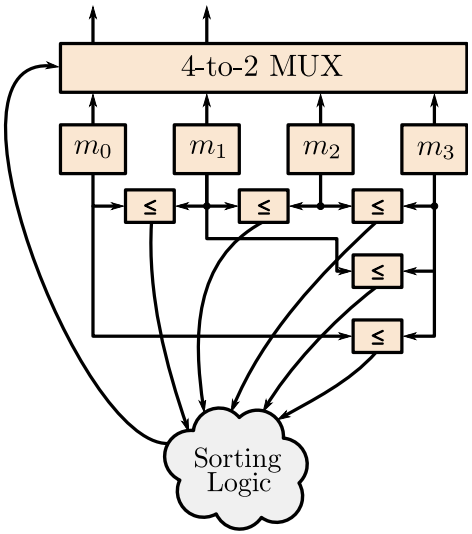
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Radix-$2L$ and Pruned Radix-$2L$ Sorters

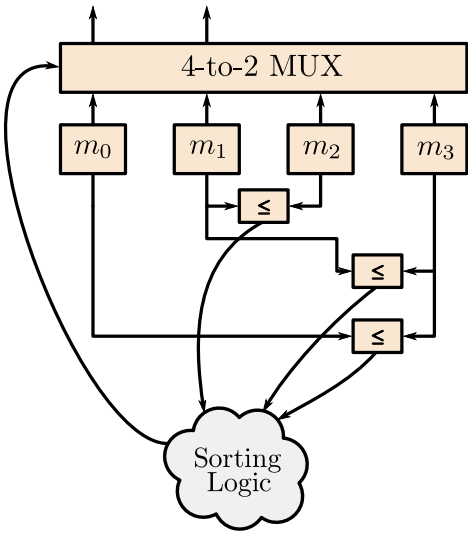
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1} \Longrightarrow m_0 \leq m_1,\ m_2 \leq m_3$

Radix-$2L$ and Pruned Radix-$2L$ Sorters

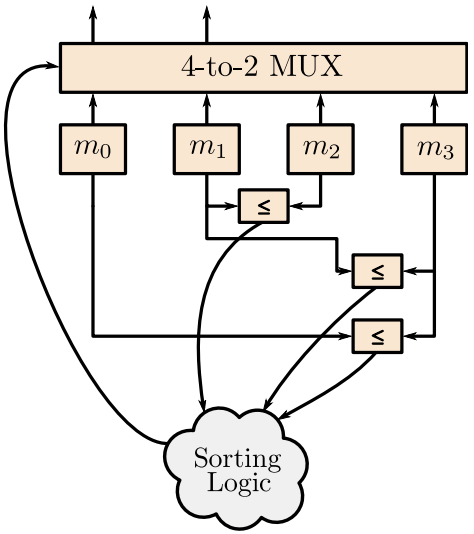
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Radix-$2L$ and Pruned Radix-$2L$ Sorters

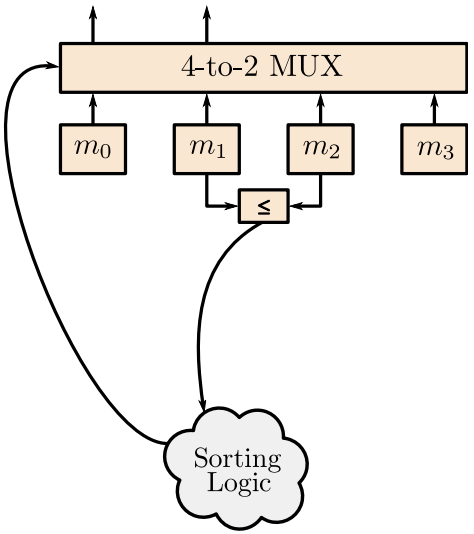
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} is never in L smallest

Radix-$2L$ and Pruned Radix-$2L$ Sorters

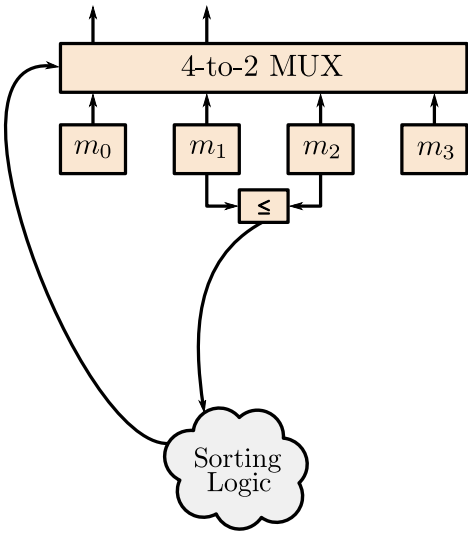
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} is never in L smallest

Radix-$2L$ and Pruned Radix-$2L$ Sorters

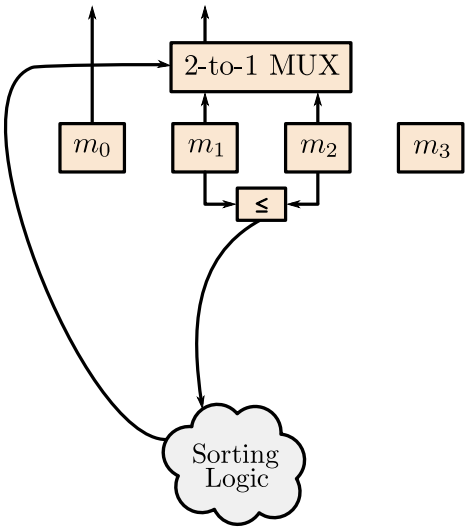
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} is never in L smallest
   - m_0 is always smallest

Radix-$2L$ and Pruned Radix-$2L$ Sorters

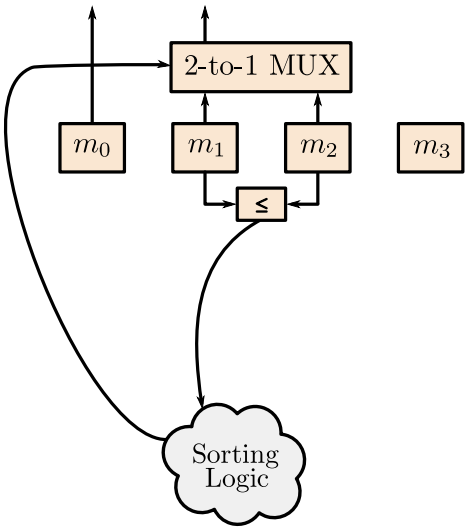
- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} is never in L smallest
   - m_0 is always smallest

Radix-$2L$ and Pruned Radix-$2L$ Sorters

- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes

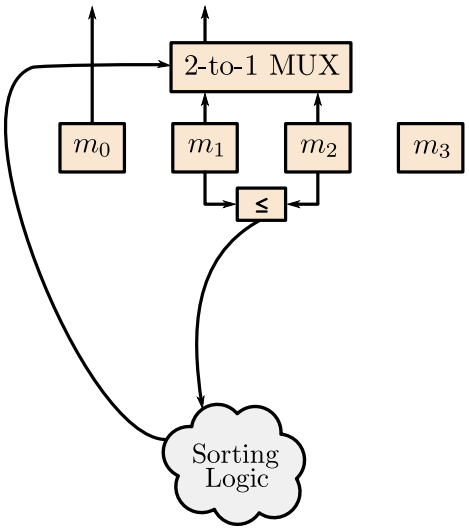


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} is never in L smallest
   - m_0 is always smallest

- **Pruned Radix-$2L$ sorter** requires:
  - $(L-1)^2 \approx L^2$ comparators

Radix-$2L$ and Pruned Radix-$2L$ Sorters

- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes

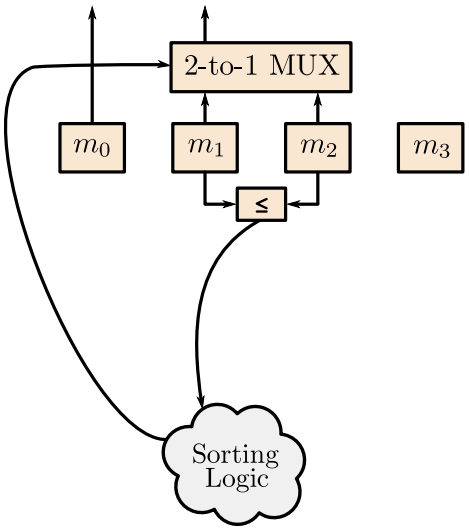


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} is never in L smallest
   - m_0 is always smallest

- **Pruned Radix-$2L$ sorter** requires:
  - $(L-1)^2 \approx L^2$ comparators
  - $(L-1)^2$-input $L-1$-output logic function

Radix-$2L$ and Pruned Radix-$2L$ Sorters

- **Radix-$2L$ sorter** requires:
  - $L(2L-1) \approx 2L^2$ comparators
  - $L(2L-1)$-input L-output logic function
  - L $2L$-to-1 MUXes



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} is never in L smallest
   - m_0 is always smallest

- **Pruned Radix-$2L$ sorter** requires:
  - $(L-1)^2 \approx L^2$ comparators
  - $(L-1)^2$-input $L-1$-output logic function
  - $(L-1)$ $(2L-2)$-to-1 MUXes

Radix-$2L$ Sorters – Synthesis Results

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

Radix-$2L$ Sorters – Synthesis Results

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Radix-$2L$		Pruned Radix-$2L$	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	2128	3007	4545	608
$L = 4$	1111	12659	2083	3703
$L = 8$	526	50433	1031	18370
$L = 16$	229	238907	372	70746

Radix-$2L$ Sorters – Synthesis Results

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Radix-$2L$		Pruned Radix-$2L$	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	2128	3007	4545	608
$L = 4$	1111	12659	2083	3703
$L = 8$	526	50433	1031	18370
$L = 16$	229	238907	372	70746

- **At least** 62% improvement in **maximum frequency**.

Radix-$2L$ Sorters – Synthesis Results

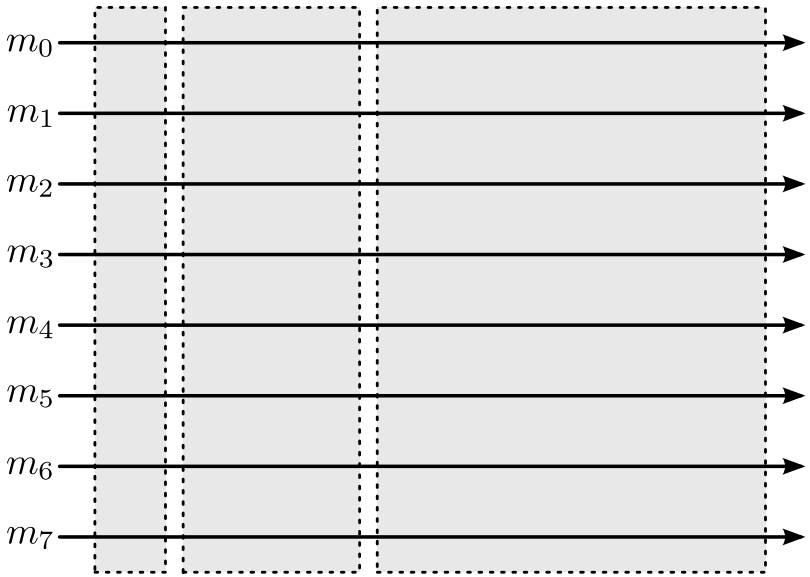
- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Radix-$2L$		Pruned Radix-$2L$	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	2128	3007	4545	608
$L = 4$	1111	12659	2083	3703
$L = 8$	526	50433	1031	18370
$L = 16$	229	238907	372	70746

- **At least** 62% improvement in **maximum frequency**.
- **At least** 63% improvement in **area**.

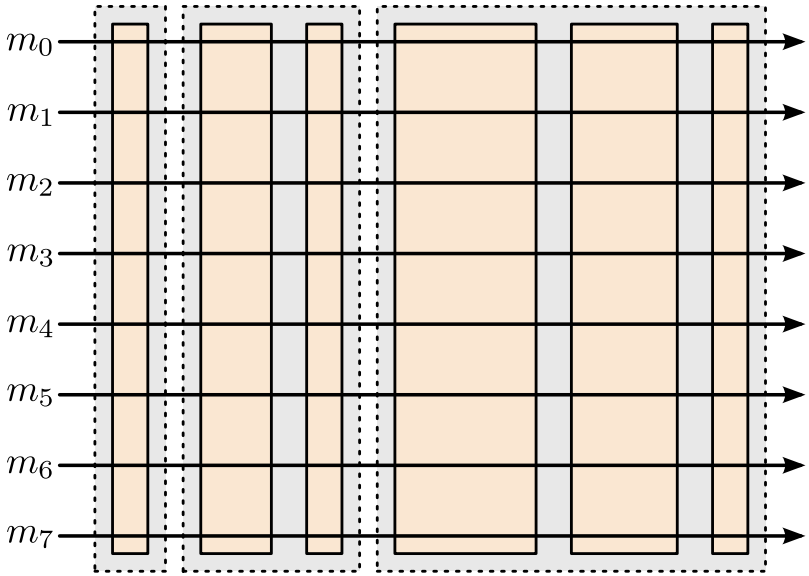
Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.



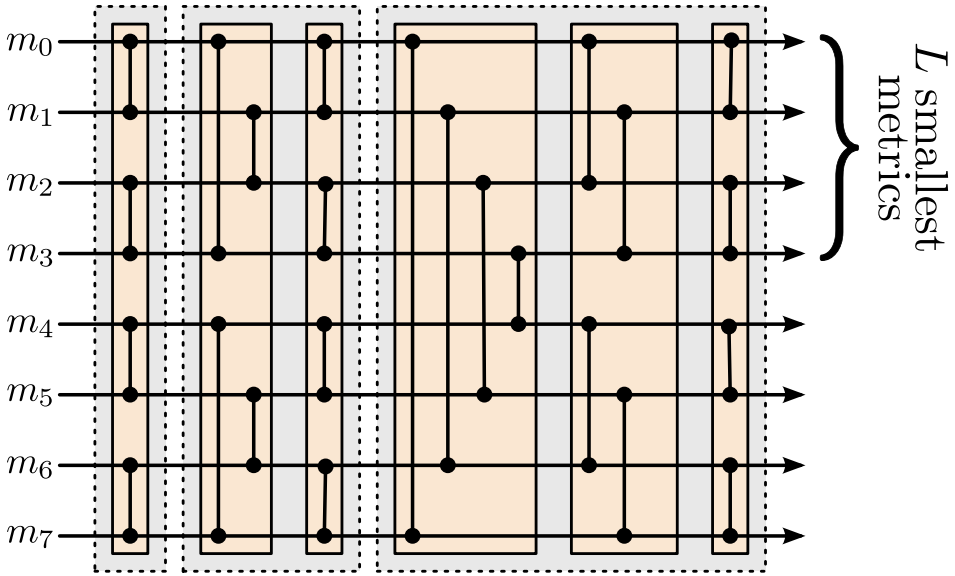
Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.



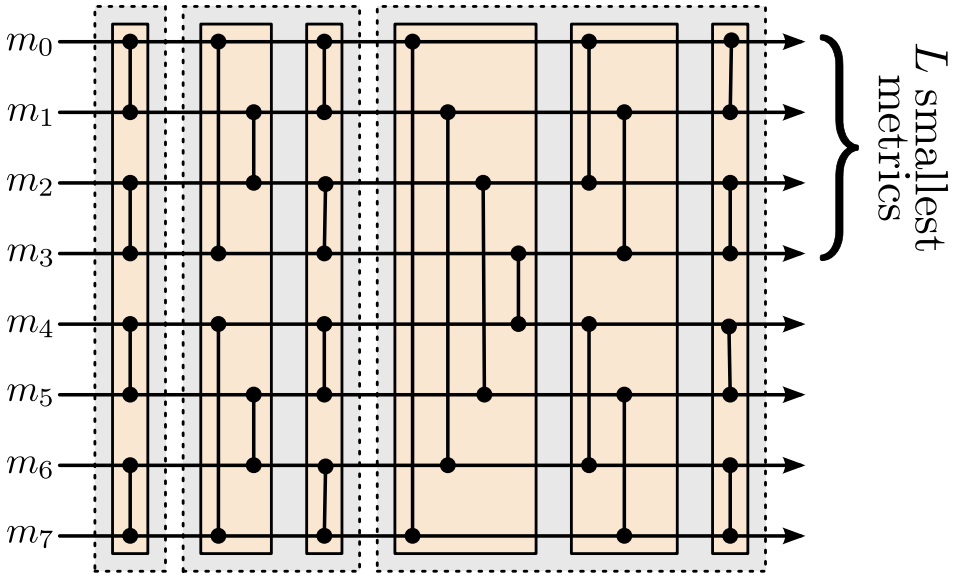
Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.



Bitonic Sorter and Pruned Bitonic Sorter

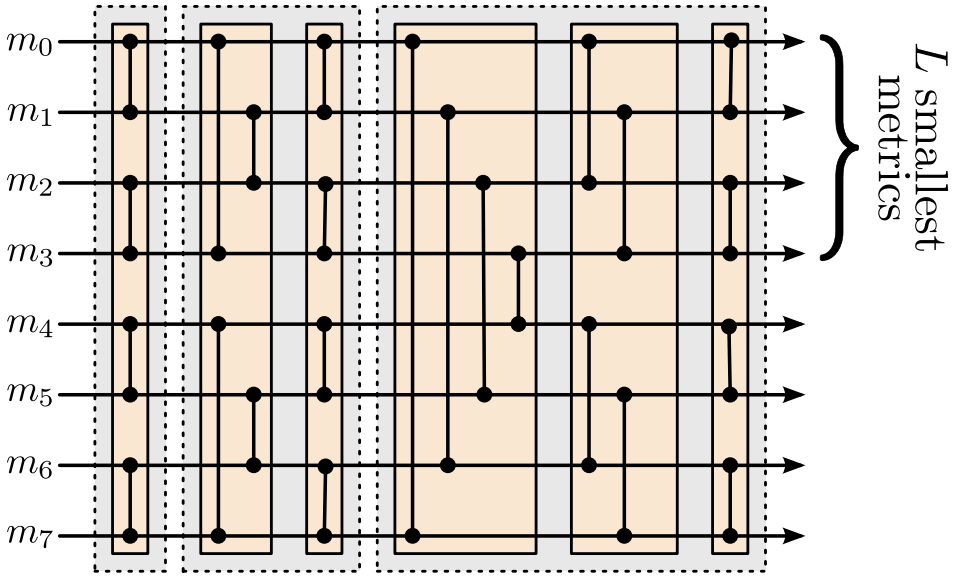
- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.



- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

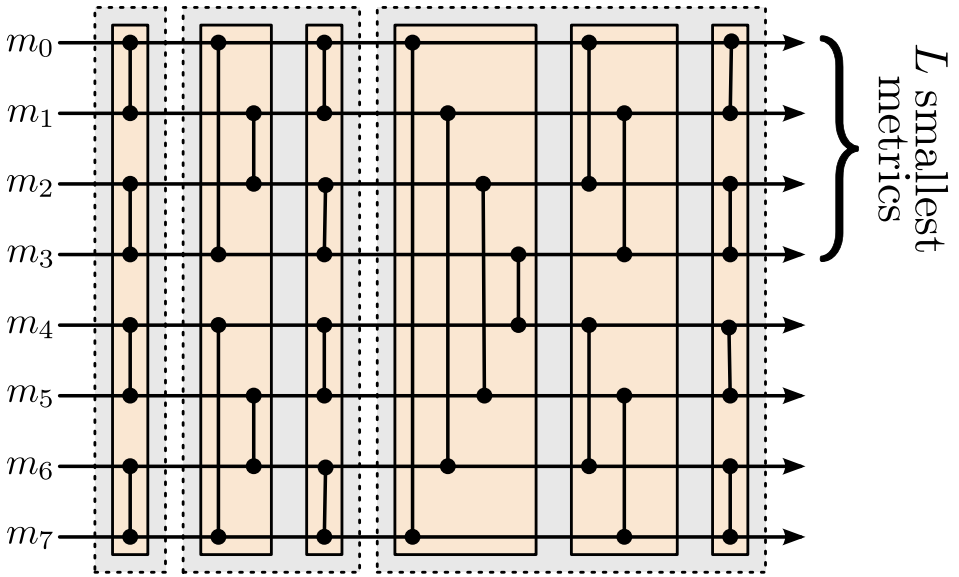
- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.



- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

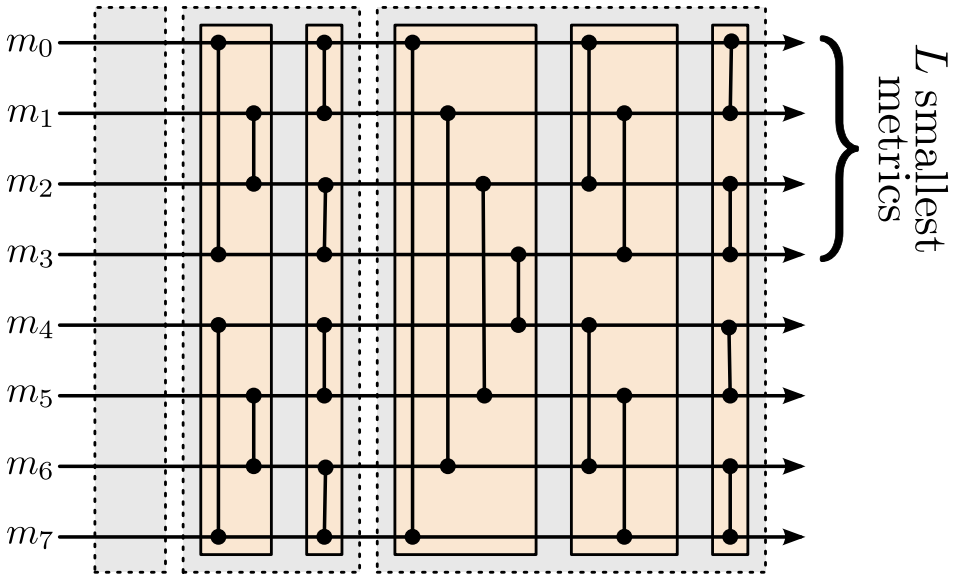


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

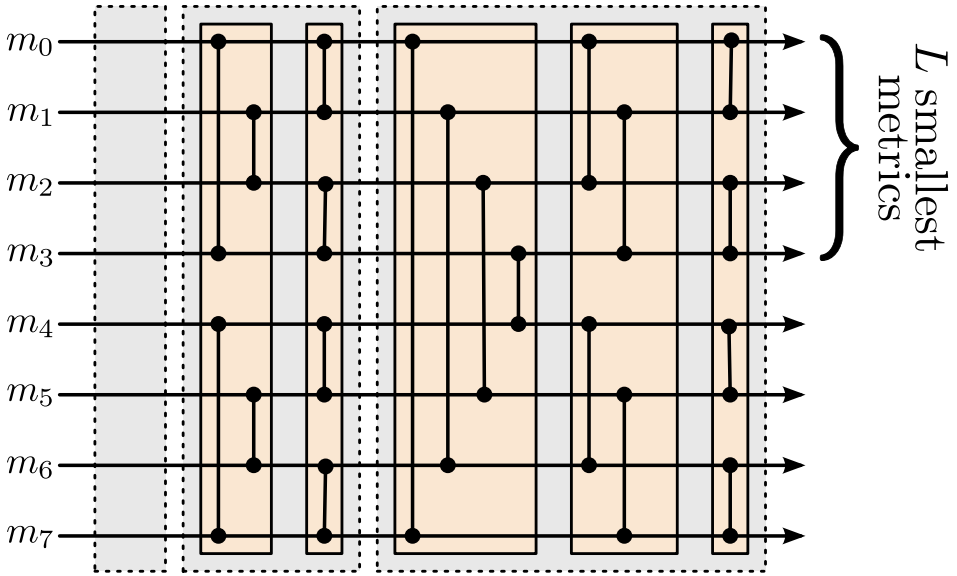


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

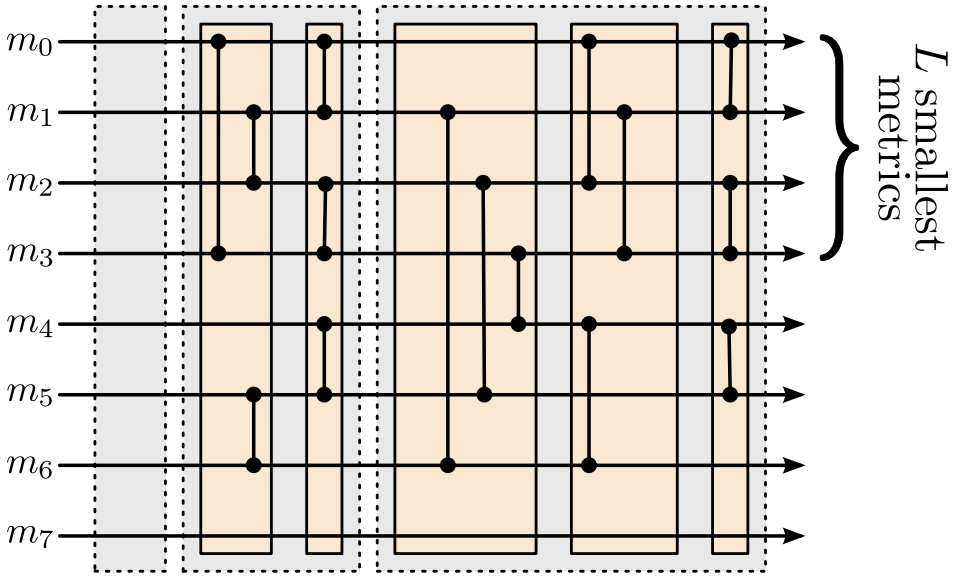


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} never in L smallest

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

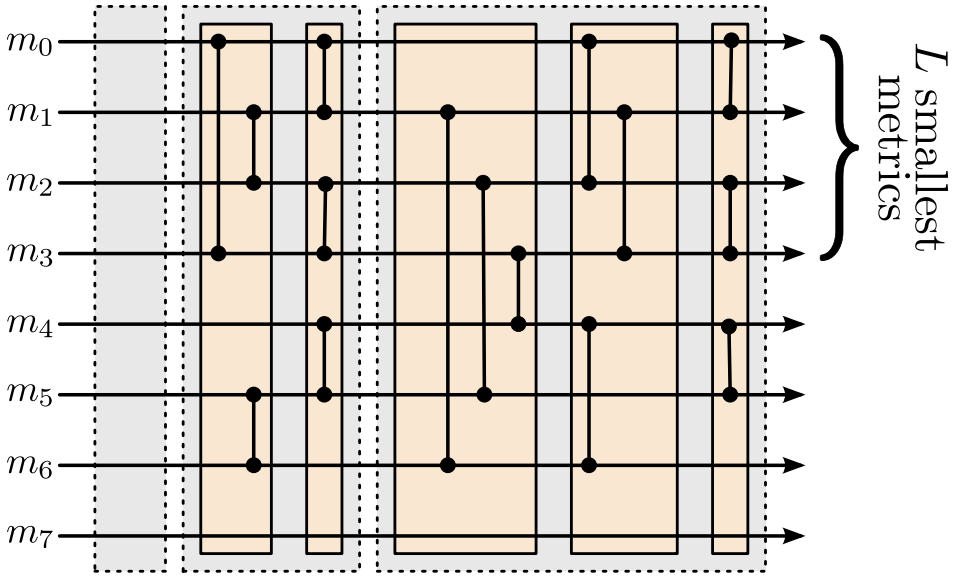


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} never in L smallest

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

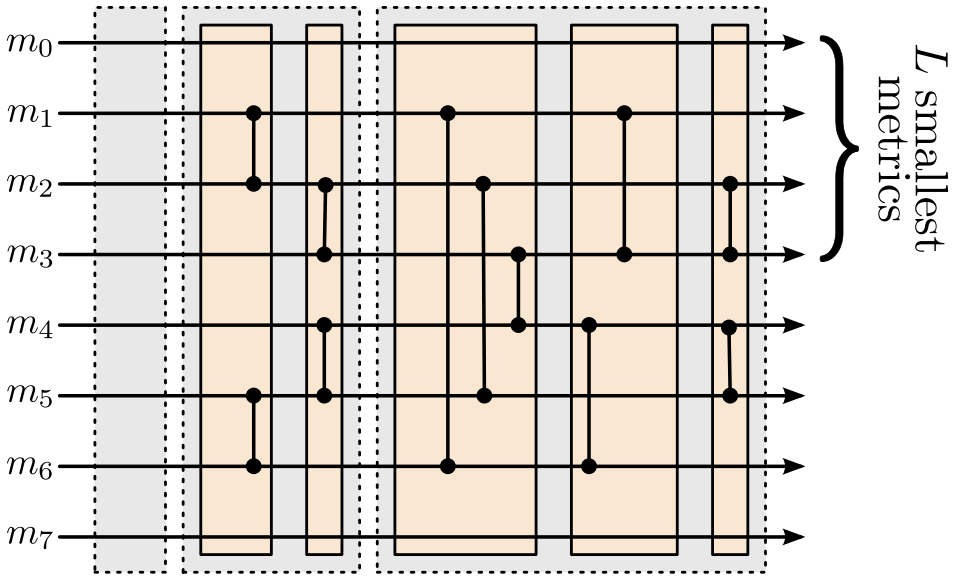


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} never in L smallest
   - m_0 always smallest

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

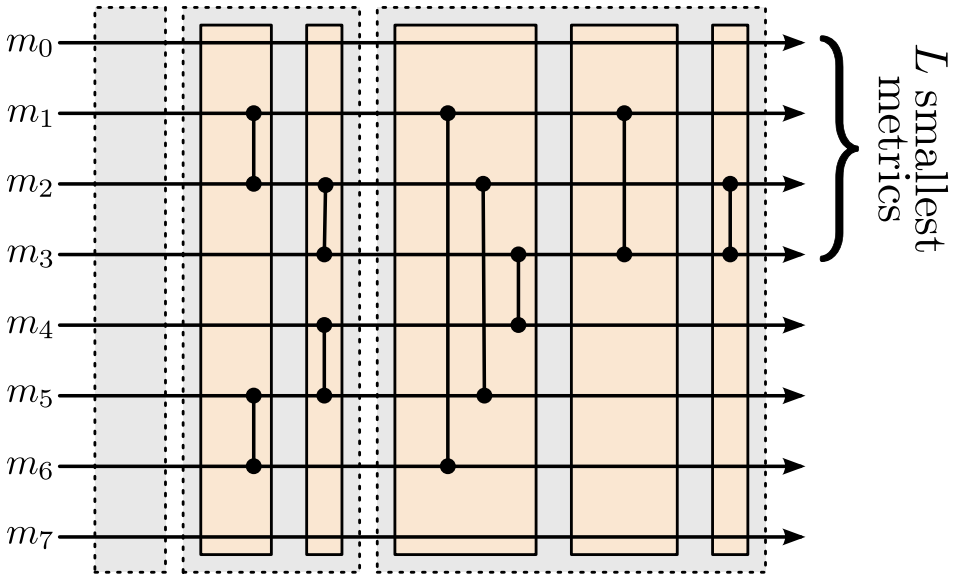


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} never in L smallest
   - m_0 always smallest

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

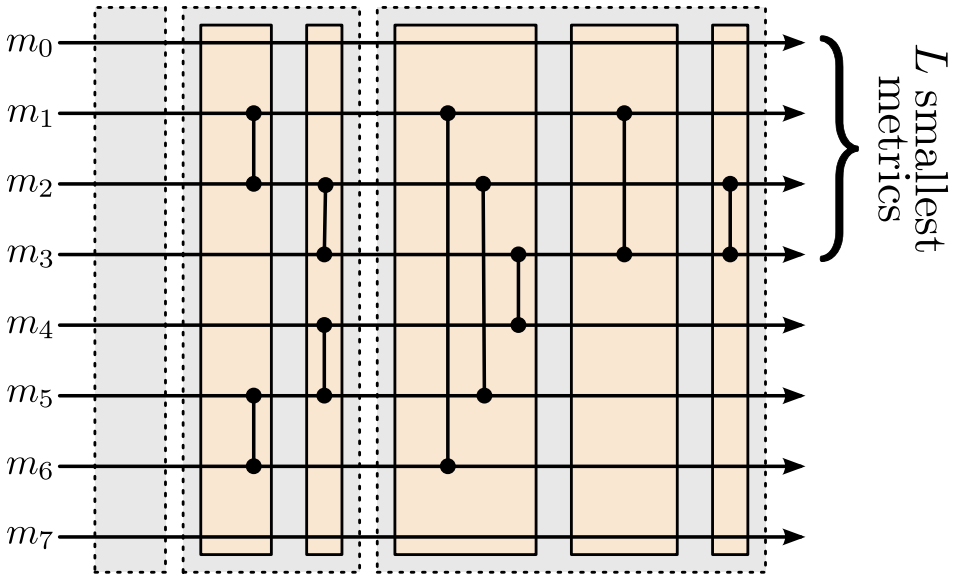


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} never in L smallest
   - m_0 always smallest

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2)$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.

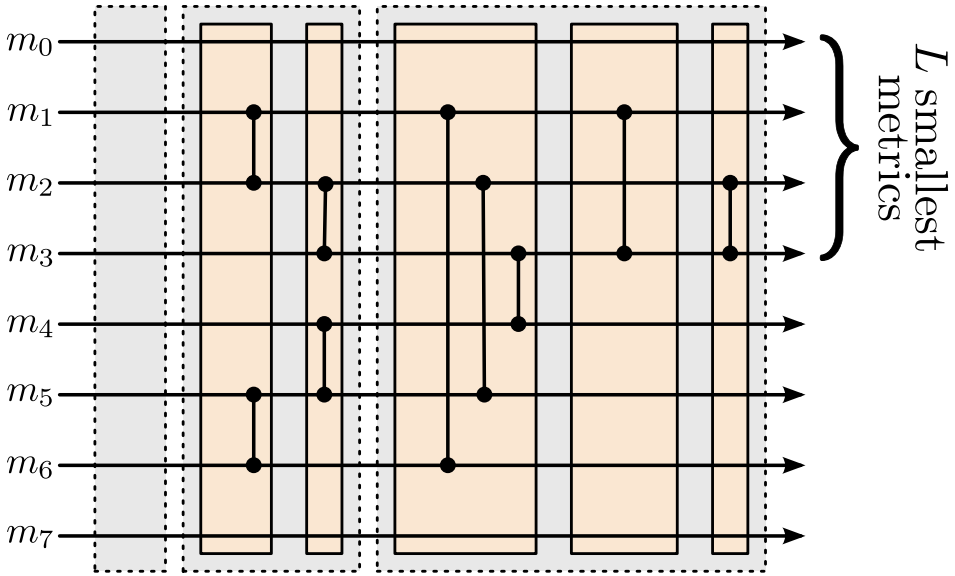


1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} never in L smallest
   - m_0 always smallest

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2) - 1$.
- **Area** proportional to total CAS units: $\frac{L}{2}(\log L + 1)(\log L + 2)$.

Bitonic Sorter and Pruned Bitonic Sorter

- Consists of $(\log L + 1)$ **superstages**.
- Each super-stage s contains s **stages**.
- Each stage contains L **compare-and-select (CAS) units**.



1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$
   - m_{2L-1} never in L smallest
   - m_0 always smallest

- **Critical path** proportional to total stages: $\frac{1}{2}(\log L + 1)(\log L + 2) - 1$.
- **Area** proportional to total CAS units: $(\frac{L}{2} - 1)(\log L)(\log L + 2) + 1$.

Operating frequency and area improvement **diminishes** as L is increased.

Bitonic Sorters – Synthesis Results

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

Bitonic Sorters – Synthesis Results

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Bitonic		Pruned Bitonic	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	1370	2109	4545	608
$L = 4$	676	8745	952	3965
$L = 8$	347	27159	478	20748
$L = 16$	214	82258	256	69769
$L = 32$	157	238721	166	205478

Bitonic Sorters – Synthesis Results

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Bitonic		Pruned Bitonic	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	1370	2109	4545	608
$L = 4$	676	8745	952	3965
$L = 8$	347	27159	478	20748
$L = 16$	214	82258	256	69769
$L = 32$	157	238721	166	205478

- **At least** 6% improvement in **maximum frequency**.

Bitonic Sorters – Synthesis Results

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Bitonic		Pruned Bitonic	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	1370	2109	4545	608
$L = 4$	676	8745	952	3965
$L = 8$	347	27159	478	20748
$L = 16$	214	82258	256	69769
$L = 32$	157	238721	166	205478

- **At least** 6% improvement in **maximum frequency**.
- **At least** 14% improvement in **area**.

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
    for $\ell = 2L - 1$ **to** 1 **do**
        if $m_{\ell-1} > m_\ell$ **then**
            Swap m_ℓ and $m_{\ell-1}$;

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
    for $\ell = 2L - 1$ **to** 1 **do**
        if $m_{\ell-1} > m_\ell$ **then**
            Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L - 1$ **to** 1 **do**
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
    for $\ell = 2L-1$ **to** 1 **do** /* $\boxed{\ell = 7}$ */
        if $m_{\ell-1} > m_\ell$ **then**
            Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 6$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
    for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 6$ */
        if $m_{\ell-1} > m_\ell$ **then**
            Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 5$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 4$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 4$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
    for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 3$ */
        if $m_{\ell-1} > m_\ell$ **then**
            Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
    for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 2$ */
        if $m_{\ell-1} > m_\ell$ **then**
            Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 2$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 1$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$

  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L - 1$ **to** 1 **do**
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$

  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$

  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L - 1$ **to** 1 **do**
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- The execution of the **if** block depends only on the state of the list at the beginning of the **for** loop.

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L - 1$ **to** 1 **do**
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- The execution of the **if** block depends only on the state of the list at the beginning of the **for** loop.
- Each element participates in at most one **swap** operation.

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do**
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- The execution of the **if** block depends only on the state of the list at the beginning of the **for** loop.
- Each element participates in at most one **swap** operation.
- The execution of the **for** loop can be parallelized.

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 7\&3$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- The execution of the **if** block depends only on the state of the list at the beginning of the **for** loop.
- Each element participates in at most one **swap** operation.
- The execution of the **for** loop can be parallelized.

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do** /* $\ell = 7\&3$ */
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 10 \leq 11 \leq 17 \leq 18 \leq 19 \leq 20$

1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- The execution of the **if** block depends only on the state of the list at the beginning of the **for** loop.
- Each element participates in at most one **swap** operation.
- The execution of the **for** loop can be parallelized.

Bubble Sorter

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L - 1$ **to** 1 **do**
    if $m_{\ell-1} > m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

Example $(3, 11, 9, 17, 10, 20, 18, 19)$

- $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 20 \not\leq 18 \leq 19$
  $3 \leq 11 \not\leq 9 \leq 17 \not\leq 10 \leq 18 \leq 20 \not\leq 19$
  $3 \leq 11 \not\leq 9 \leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 11 \not\leq 10 \leq 17 \leq 18 \leq 20 \not\leq 19$
- $3 \leq 9 \leq 10 \leq 11 \leq 17 \leq 18 \leq 19 \leq 20$

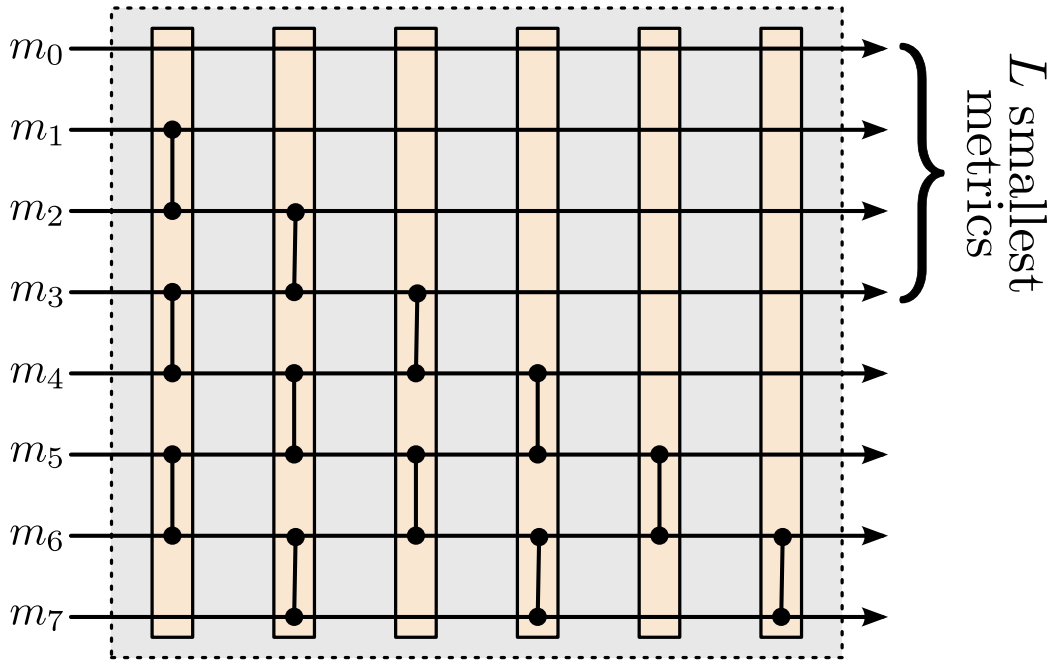
1. $m_{2\ell} \leq m_{2(\ell+1)}$
2. $m_{2\ell} \leq m_{2\ell+1}$

- The execution of the **if** block depends only on the state of the list at the beginning of the **for** loop.
- Each element participates in at most one **swap** operation.
- The execution of the **for** loop can be parallelized.
- At odd (even) rounds of the **while** loop, CAS operations only take place for even (odd) ℓ.

Hardware Implementation of Bubble Sort

- $2L-2$ stages implementing (at most) $2L-2$ iterations of the **while** loop.

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do**
    if $m_{\ell-1} \geq m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

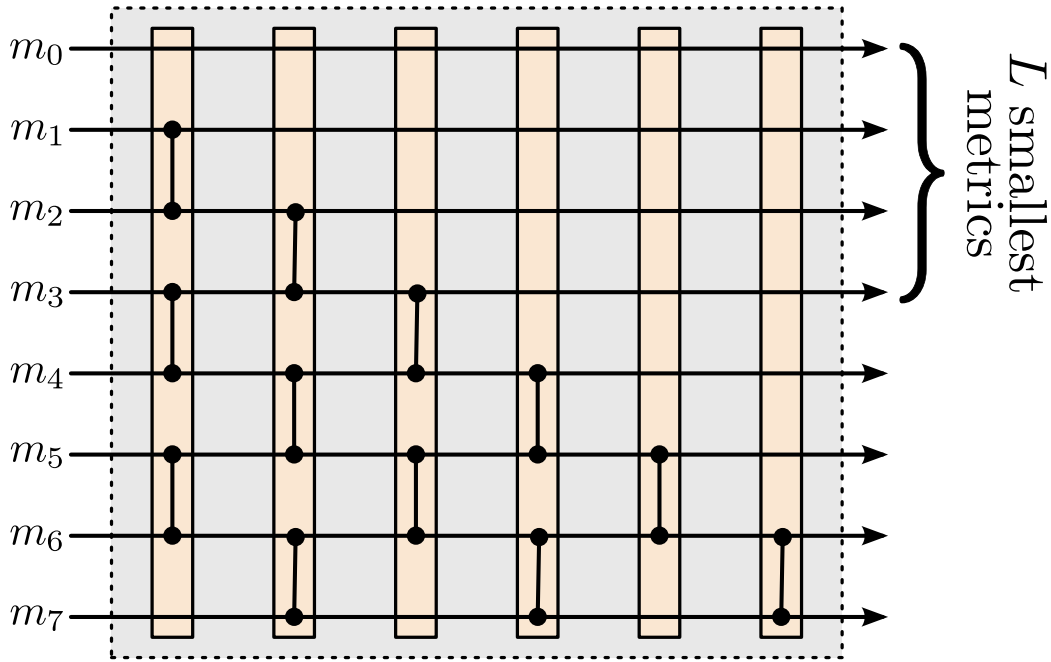


Hardware Implementation of Bubble Sort

- $2L-2$ stages implementing (at most) $2L-2$ iterations of the **while** loop.
- Stage $t \in \{1, 2, \ldots, 2L-2\}$ implements the **for** loop in parallel using $L - \lceil \frac{t}{2} \rceil$ CAS units

---

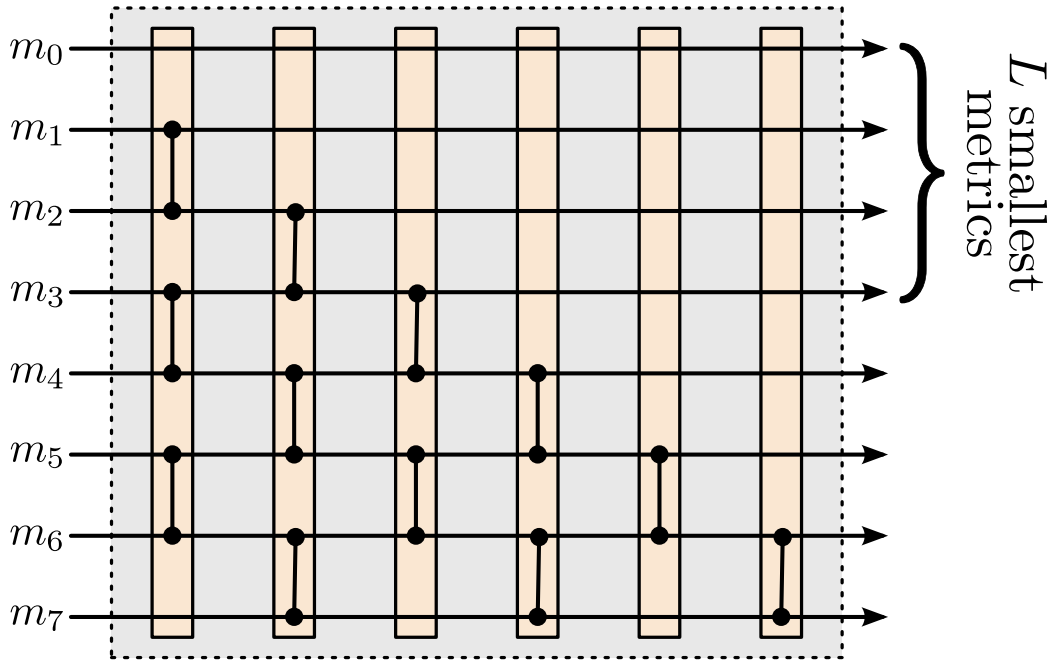
while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do**
    if $m_{\ell-1} \geq m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;



Hardware Implementation of Bubble Sort

- $2L-2$ stages implementing (at most) $2L-2$ iterations of the **while** loop.
- Stage $t \in \{1, 2, \ldots, 2L-2\}$ implements the **for** loop in parallel using $L - \lceil \frac{t}{2} \rceil$ CAS units $\Rightarrow$ $L(L-1)$ CAS units in total.

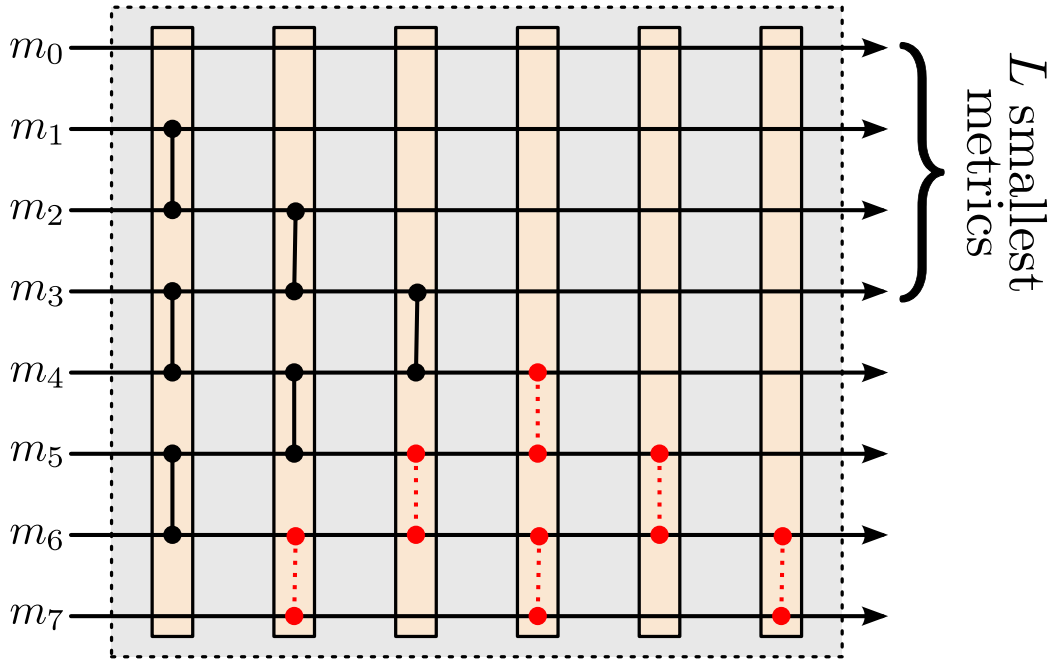
while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do**
    if $m_{\ell-1} \geq m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;



Hardware Implementation of Bubble Sort

- $2L-2$ stages implementing (at most) $2L-2$ iterations of the **while** loop.
- Stage $t \in \{1, 2, \ldots, 2L-2\}$ implements the **for** loop in parallel using $L - \lceil \frac{t}{2} \rceil$ CAS units $\Rightarrow$ $L(L-1)$ CAS units in total.

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do**
    if $m_{\ell-1} \geq m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

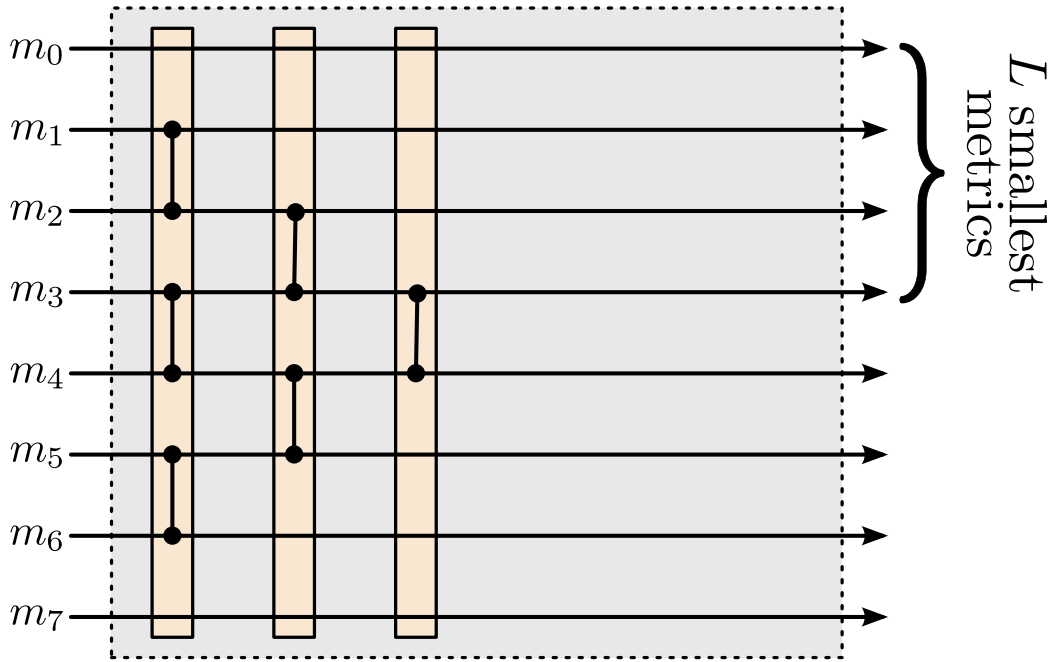


- Since we only need the first L ordered elements of the list half of the CAS units are unnecessary.

Hardware Implementation of Bubble Sort

- $2L-2$ stages implementing (at most) $2L-2$ iterations of the **while** loop.
- Stage $t \in \{1, 2, \ldots, 2L-2\}$ implements the **for** loop in parallel using $L - \lceil \frac{t}{2} \rceil$ CAS units $\Rightarrow$ $L(L-1)$ CAS units in total.

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do**
    if $m_{\ell-1} \geq m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;

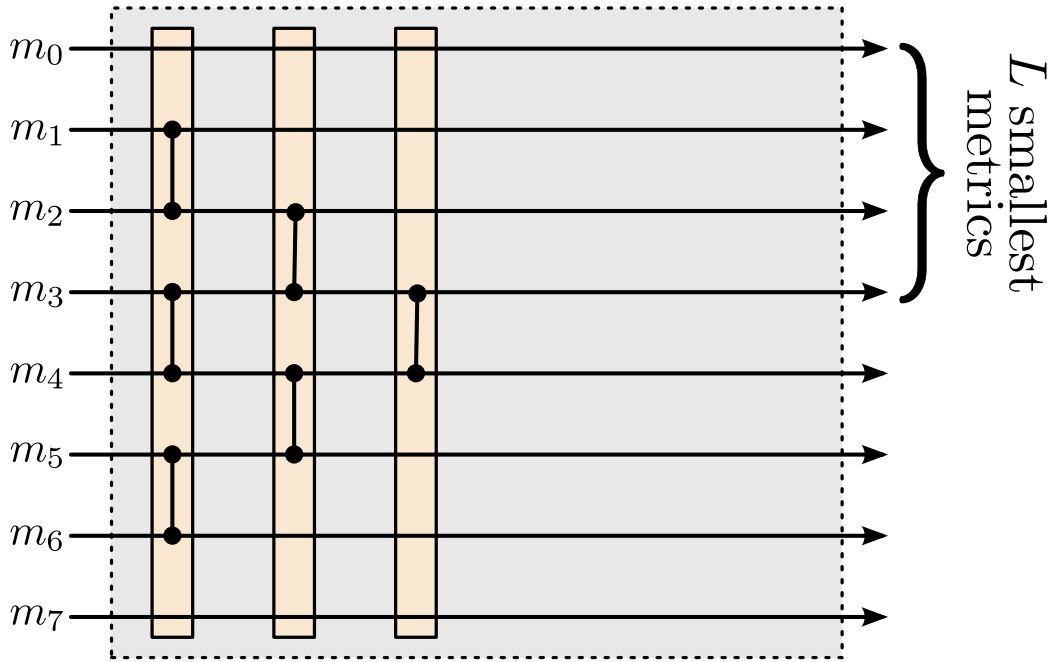


- Since we only need the first L ordered elements of the list half of the CAS units are unnecessary.

Hardware Implementation of Bubble Sort

- $2L-2$ stages implementing (at most) $2L-2$ iterations of the **while** loop.
- Stage $t \in \{1, 2, \ldots, 2L-2\}$ implements the **for** loop in parallel using $L - \lceil \frac{t}{2} \rceil$ CAS units $\Rightarrow$ $L(L-1)$ CAS units in total.

while *exists* ℓ *such that* $m_\ell > m_{\ell+1}$ **do**
  for $\ell = 2L-1$ **to** 1 **do**
    if $m_{\ell-1} \geq m_\ell$ **then**
      Swap m_ℓ and $m_{\ell-1}$;



- Since we only need the first L ordered elements of the list half of the CAS units are unnecessary.

$\Rightarrow \frac{1}{2}L(L-1)$ CAS units in $L-1$ stages.

Which sorter to choose?

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

Which sorter to choose?

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Pruned Radix-$2L$		Pruned Bitonic		Simplified Bubble	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	4545	608	4545	608	4545	608
$L = 4$	2083	3703	952	3965	1388	2756
$L = 8$	1031	18370	478	20748	534	11726
$L = 16$	372	70746	256	69769	247	51159
$L = 32$	145	376945	166	205478	127	212477

Which sorter to choose?

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Pruned Radix-$2L$		Pruned Bitonic		Simplified Bubble	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	4545	608	4545	608	4545	608
$L = 4$	2083	3703	952	3965	1388	2756
$L = 8$	1031	18370	478	20748	534	11726
$L = 16$	372	70746	256	69769	247	51159
$L = 32$	145	376945	166	205478	127	212477

- For $L = 2$ it can be shown that all three sorters are **equivalent**.

Which sorter to choose?

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Pruned Radix-$2L$		Pruned Bitonic		Simplified Bubble	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	4545	608	4545	608	4545	608
$L = 4$	2083	3703	952	3965	1388	2756
$L = 8$	1031	18370	478	20748	534	11726
$L = 16$	372	70746	256	69769	247	51159
$L = 32$	145	376945	166	205478	127	212477

- For $L = 2$ it can be shown that all three sorters are **equivalent**.
- For $L \leq 16$:
  - The pruned radix-$2L$ sorter is the **fastest**.

Which sorter to choose?

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Pruned Radix-$2L$		Pruned Bitonic		Simplified Bubble	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	4545	608	4545	608	4545	608
$L = 4$	2083	3703	952	3965	1388	2756
$L = 8$	1031	18370	478	20748	534	11726
$L = 16$	372	70746	256	69769	247	51159
$L = 32$	145	376945	166	205478	127	212477

- For $L = 2$ it can be shown that all three sorters are **equivalent**.
- For $L \leq 16$:
  - The pruned radix-$2L$ sorter is the **fastest**.
  - The simplified bubble sorter is the **smallest**.

Which sorter to choose?

- TSMC 90nm, typical timing library (25° C, 1.2 V), 8-bit metrics.

	Pruned Radix-$2L$		Pruned Bitonic		Simplified Bubble	
	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)	Freq. (MHz)	Area (μm^2)
$L = 2$	4545	608	4545	608	4545	608
$L = 4$	2083	3703	952	3965	1388	2756
$L = 8$	1031	18370	478	20748	534	11726
$L = 16$	372	70746	256	69769	247	51159
$L = 32$	145	376945	166	205478	127	212477

- For $L = 2$ it can be shown that all three sorters are **equivalent**.
- For $L \leq 16$:
  - The pruned radix-$2L$ sorter is the **fastest**.
  - The simplified bubble sorter is the **smallest**.
- For $L = 32$:
  - The pruned bitonic sorter is **fastest** and **smallest**.

Thank you!

Questions?

VHDL code for all sorters available at: `http://tcl.epfl.ch`