Artificial Results From Hardware Synthesis

Ahmed Alharbi[1] and Charles Bouillaguet[1]

Sorbonne Université, CNRS, LIP6, F-75005 Paris, France,
{ahmed.alharbi,charles.bouillaguet}@lip6.fr

Abstract. In this paper, we revisit venerable lower bounds on the AT or AT^2 performance metric of hardware designs. A series of works started in the late 1970's has established that if a hardware circuit of area A computes a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ in T clock cycles, then AT^2 is asymptotically larger than the square of the communication complexity of f. These lower bounds ignore the active components of the circuit such as the logic gates and only take into account the area of the wiring connecting them.

It seems that it has become a common practice to report the performance characteristics of hardware designs after synthesis, namely after having "compiled" the design into the topological description of a hardware circuit made of standard cells. The area of the cells can be be determined with certainty, whereas the area occupied by the wires cannot. This may leads to optimistic performance figures, that may even violate the old lower bounds.

In this paper, we take the case of the Möbius transform as a case study, following the work of Banik and Regazzoni in TCHES, 2024(2) who presented hardware designs that implement it. We first determine the communication complexity of the Möbius transform. Then, following the old methodology, we derive lower-bounds on the area (in μm^2) of any circuit that implement the operation using several open Process Design Kits for ASIC production. For large enough instances, the wires provably occupy more area than the logic gates themselves. This invalidate previous theoretical claims about the performance of circuits implementing the Möbius transform.

Fundamentally, the root cause of the contradiction between "VLSI-era" lower bounds and current performance claims is that the lower bounds apply to a geometric description of the circuit where the lengths of the wires are known, while it is common to report performance results on the basis of hardware synthesis alone, where a topological description of the circuit has been obtained but the actual lengths of wires is unknown.

Keywords: Möbius transform · Area Lower bound · Communication Complexity · circuits

1 Introduction

Academic researchers proposing hardware designs often have to argue about their performance before an actual transistor is etched onto a silicium wafer. Consider for instance recent proposals of low-latency Pseudo-Random Functions such as Orthros [BIL+21], Speedy [LMMR21], QARMAv2 [ABD+23] or Gleeok [ABC+24]. In all cases, the area (μm^2), power (mW), energy (pJ) and latency (ns) of the corresponding hardware designs are reported for various industrial chip production processes. However, no actual chip was manufactured for the purpose of obtaining these performance results experimentally. The corresponding hardware designs have been synthetized for various cell libraries, such as the Nangate 45nm library ("FreePDK45"), the Nangate 15nm ("FreePDK15"), the TSMC 90nm library or the STM 90nm library. This synthesis is almost universally performed

Table 1: Characteristic of an OpenCore double-precision FPU circuit after synthesis, place and route using two different process design kits. Data: [MMR+15].

	FreePDK45	FreePDK15
#Cells	36,583	31,442
Cell area (µm^2)	68,566.29	15,437.17
Total area (µm^2)	211,068.87	21,706.92
Ratio	×3.08	×1.4

using (closed-source) commercial tools. The performance metrics are derived from the result of the synthesis process.

Starting from a higher-level description of the hardware design (for instance in VHDL or Verilog), *synthesis* produces a lower-level description: the design is realized by assembling logic cells taken from a cell library. Synthesis produces a *netlist*, namely a description of the electrical connection between pins of the cells that compose the circuit. It is a topological description of the final chip.

Placement and *routing* take the synthetized netlist and produce the actual physical layout of the chip. The result is a geometric description of the circuit that is ready for industrial production. At this point, it is possible to test if the geometric description of the initial design satisfies constraints imposed by the manufacturing process (thickness of wires, spacing ensuring electrical isolation, etc.). Foundries require this kind of geometric description to produce actual chips (using a cell library that correspond to their manufacturing process).

After placement and routing have been completed and a geometric description of a chip has been obtained, the silicim area of the chip that would be manufactured is known with certainty. Power and timing simulations can be performed with a high degree of realism as well.

However, all these steps (synthesis, placement, routing) are complex and computationally intensive. Therefore, it is common to take a shortcut and to only perform synthesis, not placement nor routing. The performance characteristics of the chip that could be obtained by industrial production is then *estimated* from the netlist. The underlying assumption is that it possible to estimate, with a reasonable degree of certainty, characteristics of the geometric description of the circuit from its topological description alone. This is what is done for the evaluation of the low-latency PRFs mentionned above [BIL+21, LMMR21, ABD+23, ABC+24] and it seems to be common in the literature.

With regards to the area of the circuit, once a netlist has been synthesized, the area of the cells that compose the circuit is known. This is what is reported as "area" in these works. The underlying assumption is then that adding electrical wires to connect these cells does not significantly increase the area of the final hardware chip, and that most of the silicium area is occupied by the cells.

In this article, we revisit this assumption and show that it is bound to be false, at least in some cases. As a preliminary observation, we note that the article introducing the FreePDK15 open cell library [MMR+15] provides the example of a double-precision FPU core that has been synthetized, placed and routed using both the FreePDF45 and FreePDF15 cell library and corresponding processes. The results are shown in Table 1. Clearly, the final area of the circuit after placement and routing is larger than the area of the cells. In this case, the expansion is nevertheless limited. It is smaller with the newer 15nm process than with the 45nm process, because the latter allows more layers of wires (13 vs 10). In this article, we investigate another "real-life" example where the effect is more dramatic.

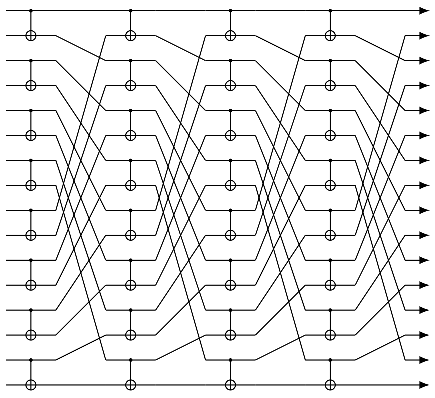


Figure 1: “Butterfly diagram” that computes the Möbius transform for inputs of size 16.

1.1 A Case Study: the Möbius Transform

To make our point, we use a recently published article by Banik and Regazzoni [BR24a] and the follow-up work [BR24b]. It describes a hardware design that solves systems of Boolean polynomial equations using the “memory-efficient” Möbius transform invented by Dinur [Din21]. This is motivated by cryptanalytic applications. Any Boolean function

$$\begin{array}{rccc} f: & \{0,1\}^n & \to & \{0,1\} \\ & (x_0, \ldots, x_{n-1}) & \mapsto & f\left(x_0, \ldots, x_{n-1}\right) \end{array}$$

on n variables can be completely described by providing its *truth table*, namely the array of 2^n bits that contain its value on each of the possible values of the n input variables. It can also be described as a multivariate polynomial over $\mathbb{F}_2$ with 2^n terms:

$$f(x) = \sum_{m \in \{0,1\}^n} a_m x_1^{m_1} \ldots x_n^{m_n}$$

The function f is completely described by the values of the 2^n coefficients $a_0, a_1, \ldots$ This representation is the Algebraic Normal Form (ANF) of f.

The *Möbius transform* computes the ANF of a function given its truth table, and vice-versa (the operation is involutive). The interested reader may consult [Jou09] for more details. The Möbius transform is a linear function described by the block-matrix

$$M_n = \begin{pmatrix} M_{n-1} & 0 \\ M_{n-1} & M_{n-1} \end{pmatrix} \qquad \text{with } M_0 = (1)$$

It can be implemented efficiently, using the following observation. Suppose that the input vector x of size 2^n is split in two halves:

$$\begin{pmatrix} y^\uparrow \\ y^\downarrow \end{pmatrix} = M_n \cdot \begin{pmatrix} x^\uparrow \\ x^\downarrow \end{pmatrix} = \begin{pmatrix} M_{n-1} & 0 \\ M_{n-1} & M_{n-1} \end{pmatrix} \cdot \begin{pmatrix} x^\uparrow \\ x^\downarrow \end{pmatrix} = \begin{pmatrix} M_{n-1} \cdot x^\uparrow \\ M_{n-1} \cdot x^\uparrow + M_{n-1} \cdot x^\downarrow \end{pmatrix}$$

This suggests a simple in-place recursive algorithm to compute $M_n \cdot x$: set $y^\uparrow \leftarrow M_{n-1} \cdot x^\uparrow$ and $y^\downarrow \leftarrow M_{n-1} \cdot x^\downarrow$, then do $y^\downarrow \leftarrow y^\downarrow + y^\uparrow$. This requires $\mathcal{O}\left(n \log n\right)$ operations. From a hardware point of view, the operation can be implemented by a depth-n circuit that has a regular shape as illustrated by Fig. 1. On inputs of size 2^n, n identical rounds with 2^{n-1} XOR gates and a permutation of wires are sufficient.

Two hardware designs called **Expmob1** and **Expmob2** to compute the Möbius transform are described in [BR24a]. They are used as subcomponents of a broader design. The **Expmob1** circuit is fully unrolled and computes the Möbius transform of an input

Table 2: Experimental performance results given in [BR24a, Appendix D]. σ denotes the area of a XOR gate.

	Expmob1		**Expmob2**	
n	A (μm^2)	$A/(\sigma n 2^{n-1})$	Area (μm^2)	$A/(\sigma 2^{n-1})$
6	95.600	1.13	157.041	11.1
7	233.128	1.18	300.958	10.7
8	572.473	1.27	573.702	10.2
9	1333.936	1.31	1138.754	10.1
10	3227.910	1.43	2255.831	10.0
11	7855.473	1.58	4486.152	9.9
12	18784.420	1.73	9046.868	10.0
13	46181.007	1.96	17860.706	9.9
14	110026.212	2.17	35611.803	9.8

of size $N = 2^n$ in a single clock cycle by using $\log_2 N$ layers of "butterflies" as in Fig. 1 and has $0.5N \log_2 N$ XOR gates. The second circuit, **Expmob2**, uses a single layer of butterlies and runs in $\log_2 N$ clock cycles with just $0.5N$ XOR gates. These two circuits are claimed in to have area proportional to the number of XOR gates they contain. They have been synthetized for various input sizes, starting from VHDL code, using a commercial tool and the FreePDK15 open cell library. The area, power and energy consumption, as well as the latency of the corresponding hardware implementations are reported (part of this data is reproduced in Table 2). The reported areas are indeed nearly proportional to the number of XOR gates contained in the circuits, as suggested by Table 2.

It follows that the two proposed designs, **Expmob1** and **Expmob2** both offer an area-time tradeoff given by $AT^2 = \mathcal{O}(N \log N)$ and $AT^2 = \mathcal{O}\left(N \log^2 N\right)$, respectively. This is explicitly claimed in [BR24a]. Both circuits offer the tradeoff $AT = \mathcal{O}(N \log N)$.

However, the algorithm to compute the Möbius transform is quite reminiscent of the most common way to implement the Discrete Fourier Transform: the radix-2 decimation in time that is ususally called "the Fast Fourier Transform". From the point of view of hardware implementations, it was shown more than 45 years ago [Tho79] that any circuit of area A running in time T that computes the Discrete Fourier Transform of a input of size N is subject to a lower-bound of the kind $AT^2 = \Omega\left(N^2\right)$. Because any circuit that computes the discrete Fourier transform must be able to contain the whole input data, there is an area lower-bound of $A = \Omega(N)$, and this gives a lower-bound on the area-time ratio $AT = \Omega\left(N^{1.5}\right)$.

The same kind of lower-bounds have subsequently been proved for various other operations: cyclic shift and convolution [Vui83], sorting [BP86], integer multiplication [BK80], matrix product [Sav81], etc. These bounds have generally been matched by corresponding designs, and hence are tight. The common argument is based on *communication complexity*. The general argument is now standard and it is described in the *Handbook of Theoretical Computer Science* [Len90]. In short, if the (planar) circuit is cut in two in a certain way, then a given amount of information must flow across the cut. This in turn requires a minimum number of wires to be cut if the flow is fast enough, and hence a minimum area for these wires.

The two proposed designs in our case study, **Expmob1** and **Expmob2**, apparently escape the kind of lower-bound that applies to a very similar operation, namely the Fast Fourier Transform. So, two outcomes are possible: either the Möbius transform, despite its resemblance with the FFT, is not subject to a comparable $AT^2 = \Omega\left(N^2\right)$ lower-bound. Or the usual assumption that the area of wiring can be ignored led the authors of [BR24a] to state erroneous theoretical performance claims and present experimental results about their hardware designs that may not representative of the performance of real chips that

Table 3: Effective lower-bounds, with A given in μm^2. T denotes the number of clock cycles required to evaluate $f(\cdot)$. $CC(f)$ is the communication complexity of f with the natural input partition, and $CC(f)$ is the communication complexity of f for the "best" input partition.

Process	Thm. 2	Thm. 3
FreePDK15	$70.4\sqrt{A} + 149 \geq CC(f)/T$	$458\sqrt{A} + 35394 \geq CC_{best}(f)/T$
FreePDK45	$19.7\sqrt{A} + 124 \geq CC(f)/T$	$196\sqrt{A} + 26604 \geq CC_{best}(f)/T$
GF 180nm	$4.68\sqrt{A} + 184 \geq CC(f)/T$	$98\sqrt{A} + 31284 \geq CC_{best}(f)/T$

could be manufactured from their design.

1.2 Our Contribution

We answer the above question by rigorously proving that any hardware circuit that implements the Möbius transform is subject to a lower-bound of the kind $AT^2 = \Omega\left(N^2/\log^2 N\right)$. This invalidates the theoretical claims made in [BR24a]. We also show that if the circuit meets certain conditions, namely receiving its input from a bus, then in fact $AT^2 = \Omega\left(N^2\right)$. We do this by determining the communication complexity of the Möbius transform: if Alice owns the first half of x and Bob owns the second half, then computing $y = M_n \cdot x$ requires that Alice and Bob exchange at least $CC(M_n) \geq 2^{n-1} - 1$ bits. If Alice owns an arbitrary subset of half the bits of x (and Bob owns the complement), then we show that they need to exchange $CC_{best}(M_n) \geq 2^{n-1}/n - 1$ bits to complete the computation.

Because the square of the communication complexity is an (asymptotic) lower-bound on the AT^2 metric of any hardware circuit that implement the corresponding function, this implies that any hardware implementation of **Expmob1** must have area $\Omega\left(N^2/\log^2 N\right)$ (remember that it operates in a single clock cycle), while **Expmob2** has area $\Omega\left(N^2/\log^4 N\right)$. This is asymptotically larger than the area of the cells.

The root cause of the contradiction between these lower bounds and the results presented in [BR24a] is the following: the lower bound apply to the area occupied by wires in the circuit whereas [BR24a] provides the area of the cells (obtained after a netlist has been synthesized), ignoring the area of the wires. We assume that the total length of the wires cannot be know with certainty without completing placement and routing.

These lower bounds on wiring area have been known for decades but they are asymptotic in nature. While it is clear that they contradict the theoretical claims of [BR24a] about the asymptotic grow of A, AT or AT^2 for the **Expmob1** and **Expmob2** families of circuits, it is difficult to compare them with the accompanying concrete experimental data. Surely the area of any actual piece of silicon that implement the Möbius transform must grow faster than initially announced, but does that mean that the area reported in [BR24a] for, say, $n = 10$ is physically impossible?

To answer this question, we derive an "effective" version of the now classic asymptotic lower bound $AT^2 = \Omega\left(CC^2\right)$, where CC denotes the "best-case" communication complexity of the computed function. In other terms, we explicitly determine the constants hidden in the "big Omega", and express it in terms of simple properties of the cell library and of the underlying industrial process. This yields a lower-bound, in μm^2, on the area of any circuit that computes a function f on a given technological process. Table 3 shows the result for several open process design kits. Theorem 2 offers a better lower-bound but requires an additional assumption (the circuit reads its inputs from a bus), while theorem 3 does not. More advanced technological processes offer weaker bounds because they offer more routing layers or thinner wires, so that the area they require is lower.

We finally compare this concrete lower bound with the experimental results given in [BR24a] using the FreePDK15 cell library. We combine our "effective" $AT^2 = \Omega\left(CC^2\right)$

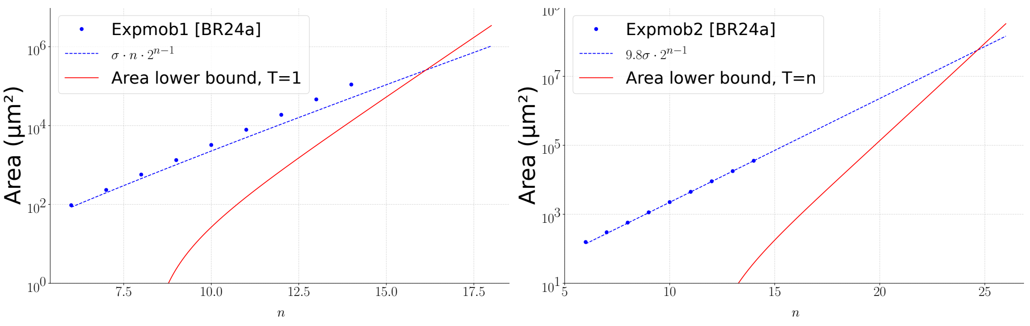


Figure 2: Extending the numbers given by [BR24a] versus the lower bound from theorem 2. σ denotes the area of the smallest XOR cell available in the cell library.

Table 4: Smallest value of n such that the area lower bound of theorem 2 becomes larger than the projected cell area ($\sigma n 2^{n-1}$ for **Expmob1** and $9.8\sigma 2^{n-1}$ for **Expmob2**, where σ denotes the area of a XOR cell).

Process	**Expmob1**	**Expmob2**
FreePDK15	17	25
FreePDK45	15	23
GF 180nm	15	23

with our lower-bound on the communication complexity of the Möbius transform. The results are shown in Figure 2. Clearly, the lower bounds we obtain do not invalidate the experimental data. In other terms, our lower bound on the area occupied by the wiring stays below the area of the cells determined after synthesis has been performed for $n = 6 \ldots 14$. We therefore extrapolated the size of the cells for larger values of n and determined the crossing point where the wiring would provably be larger than the XOR cells required to implement the function, for various cell libraries and production processes. The results are shown in Table 4.

We conclude that if the authors of [BR24a] had used the older FreePDK45 process design kit along with the Nangate 45nm open cell library instead of FreePDK15, and synthetized their design for $n = 15$, then they would unfortunately have given an area figure that is physically impossible to reach in the real world. This provides a concrete example of a situation where assuming that the size of the hardware chip is just the size of the cells may be dangerous.

In addition, the lower bounds given in this article are not tight. For instance, in this paper, for the sake of simplicity, we assume that the circuits under study are rectangular, and these lower bound consider the area of the enclosing rectangle. It the circuit is not rectangular, but has a convex shape, then there is an enclosing rectangle that has at most twice the surface (this loses a factor of at least two in the lower bound).

We cannot draw conclusions about the other performance metrics mentionned in [BR24a] (power, energy, latency). It seems to us that, without having performed placement and routing, and thus without knowing the geometry of the final chip, it is difficult to estimate them without making strong assumptions (for instance about the propagation time in wires of unknown length).

Lastly, lower bounds on AT are usually derived from separate lower bounds on A and on AT^2 (multiplying them gives a lower-bound on A^2T^2 and it remains to take the square root). It is sometimes easy to establish that any circuit computing a function f must have area $A = \Omega(n)$ because it has to memorize its entire input. This is for instance the case when all output bits potentially depend on each input bits. The circuit then cannot start emitting the output before reading its whole input, and therefore it must have area at least $\mathcal{O}(n)$ to store it. This argument is sufficient for the Fast Fourier Transform, integer multiplication modulo a power of two, cyclic shifts, etc. However, this simple argument does not apply when the circuit reads its input progressively and may emits part of its output "on the fly", before reading the entire input. This is in particular the case of the Möbius transform, where the i-th output bit depends only on the first i input bits.

We provide an area lower-bound based on one-way communication complexity. Intuitively, the one-way communication complexity is a lower-bound on the number of memory bits that the circuit must have to compute the function. This strategy (space lower bounds from communication complexity) is well-known. Our lower bound on the communication complexity of the Möbius transform then shows that a circuit for the Möbius transform that reads its input in the natural order must have area $\Omega(2^n)$, regardless of the number of clock cycles it requires to complete the computation. If it is allowed to read its input in an arbitrary order, then it has area $\Omega(2^n/n)$.

These last results are given in appendix.

2 Integrated Circuits

We consider a circuit model that is not too far removed from the reality of contemporary design and production processes. At a high level, a circuit consists of transistors connected by electrical wires. To simplify the design process, the *standard-cell methodology* is used almost universally. The final circuit is formed by assembling copies of ready-made *cells* taken from a *cell library.* Cells implement boolean operations (inverter, AND, XOR, ...) and flip-flops or latches that can store a single bit. A cell contains transistors and wires, tightly packed, with a well-defined interface. A circuit can then be obtained by placing cells and adding wires to connect their ports.

Cell libraries are specific to a production process. In their relationships with circuit designers, commercial foundries usually impose non-disclosure agreements that cover the precise decsription of their production process as well as the corresponding cell libraries. For this reason, free "Process Design Kits" (PDKs) have been made available for research and education. While they cannot be used to manufacture actual circuits, they provide an as-good-as-possible approximation of the process. The FreeDPK45 process and the associated Open Cell library (also known as the Nangate45 cell library), released in 2008, is one such widely-used cell library, simulating a 45nm production process. There has been an upgraded version to a improved 15nm in 2015, termed FreePDK15 [BD15]. The associated cell open cell library [MMR+15] is also called the FreePDK15 library. It can be obtained either from the Silicon Integration Initiative [Fre14a] or from the North Carolina State University [Fre14b]. All the information we present below is extracted from the former, because the latter can only be exploited using commercial tools that we do not own.

In these cell libraries, each cell may come in a number of variants. For instance, in FreePDK15 there are $\texttt{AND}_2$, $\texttt{AND}_3$ and $\texttt{AND}_4$ cells with 2, 3 and 4 inputs respectively. However, there is only a two-input XOR cell. Some cells may also be available in different versions, depending on their drive strength (tolerated output electrical load). For instance, in FreePDK15, the $\texttt{AND}_2$ cell comes in two versions, while the $\texttt{XOR}_2$ only has a single version. In this paper, we disregard the issue of drive strength.

We assume that all the cells of a library are rectangles of the same height (768nm in FreePDK15) and they are aligned in rows. There are two wide power tracks at the bottom and top of each row. The cells do not all have the same width: the $\texttt{AND}_2$ show in Fig. 3 has width 384nm, whereas the $\texttt{XOR}_2$ cell has width 575nm and the smallest Flip-flop cell has width 1664nm.

It follows that a circuit in FreePDK15 technology that is capable of storing n memory bits must have area at least $n \times 0.768\mu m \times 1.664\mu m = 1.277 n\mu m^2$ (this is the size of its

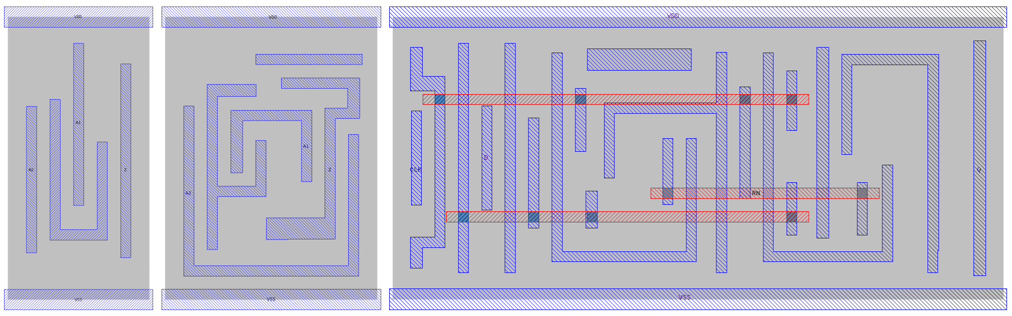


Figure 3: Three cells from the FreePDK15 library: `AND2_X1`, `XOR2_X1`, `DFFRNQ_X1`. Wires in the `M1` layer are shown in blue. Wires in the `Mint1` layer are shown in red. The vias between the two layers are visible. The images have been generated by the KLayout software [Köf24].

flip-flop cells).

Each cell has input and output *ports* (not all of the same size and shape). All the inputs of a circuit must be connected to different input ports of different gates (otherwise they would either be shorted or ignored). In the FreePDK15 cell library, the cell with the highest ratio of inputs to area are the `AOI22` and `OAI22` (that compute $\overline{(a \wedge b) \vee (c \vee d)}$ and $\overline{(a \vee b) \wedge (c \vee d)}$, respectively). They have four inputs and width 448nm. Therefore a circuit in this technology that has n inputs must have area at least $n/4 \times 0.448\mu m \times 0.768\mu m \approx 0.086 \cdot n \cdot \mu m^2$ just for the cells that are connected to its inputs.

The ports of the cells must be connected by electrical wires (thin metal layers laid out on an insulator material). Two wires transporting different signals cannot touch, as otherwise they would be electrically connected. This imposes a minimum amount of space between wires. The wires themselves have a minimum width. Several layers of wires can however be stacked up vertically. Wires in different layers can be connected by "vias" that go through the insulating material. Because wires cannot cross, each layer has a preferred direction (horizontal or vertical).

A Process Design Kit specifies how many layers of wires can be used and specifies physical characteristics of the wires (minimum wire width, minimum space between the wires, resistivity, capacitance, etc.). For instance the FreePDK15 library assumes up to 13 layers of metal wires summarized in Table 5. For the sake of completeness, let us mention that there are also other layers of semi-conductor materials below the first metal layer, that are not relevant here. The cells have many wires in these semiconductor layers, as well as in the first metal layer. A few of the most complex cells such as the flip-flop shown on the right of Fig. 3 also have a few wires in the second metal layer. It follows that the first metal layer is mostly unavailable for routing wires between cells.

In the contemporary design process, it is common to do "over-the-cells routing", namely to stack the cells as tightly as possible and to connect them by routing wires in all layers above the cells. This justifies the common assumption that "cells occupy all the area and wires do not require any extra space", because there is sufficient space in all layers above the cells to accommodate the wiring.

The `M1` layer itself is so densely used inside the cells themselves that it is mostly unavailable for routing. Wires in M1 cannot cross the VDD and VSS power lines, so they could only be used to connect adjacent cells — for instance, if two XOR cells were adjacent, the Z output of the left one could be connected to the A_2 input of the right one. Note that the A_1 input of a XOR cell is inaccessible using the M1 layer. Therefore, we assume that at most one horizontal wire can be added by the router between the VDD and VSS

Table 5: Characteristics of the routing layers in FreePDF15.

Layer	Routing	min. width (nm)	min. spacing (nm)
M1	$\emptyset$	28	36
Mint1	$\leftrightarrow$		
Mint2	$\updownarrow$		
Mint3	$\leftrightarrow$		
Mint4	$\updownarrow$		
Mint5	$\leftrightarrow$		
Msmg1	$\updownarrow$	56	56
Msmg2	$\leftrightarrow$		
Msmg3	$\updownarrow$		
Msmg4	$\leftrightarrow$		
Msmg5	$\updownarrow$		
Mg1	$\leftrightarrow$	112	112
Mg2	$\updownarrow$		

power rails in the M1 layer.

It follows from Table 5 in the `Mint1` layer, at most 15.6 horizontal wires can cross a vertical section of $1\mu m$. If we add layers `M1`, `Mint3`, `Mint5`, `Msmg2`, `Msmg4` and `Mg1`, we find a total of at most 70.5 horizontal wires per μm (and 62.5M vertical wires per μm).

The process of producing a circuit from a high-level description is usually broken into steps. The *synthesis* step takes a description of the circuit in a high-level language as well as the description of a cell library and outputs a *netlist*, namely a description of the circuit in terms of cells and *nets* (that connect the ports of the cells). Given a netlist, the total area occupied by the gates can be determined. This provides a *lower bound* on the final area of the circuit. The common assumption is that this lower bound is tight.

After synthesis, obtaining a physical description of the circuit usually involves two subsequent steps. The *placement* step decides the physical location of the cells. The cells in each row must be adjacent, so the placer decides the number of rows and their total width. It may also add "filler" cells that serve no other purpose than making room for wires on upper layers (in the FreePDF15 library, the smallest filler cell has width 128nm). Finally, the *routing* step draws the wires specified by the netlist in all layers and places the vias. Routing may fail if the cells are too densely packed; in this case, placement is redone and more filler cells are added to make the layout more amenable to routing.

3 Communication Complexity

The notion of communication of complexity was introduced by Yao in 1979 [Yao79]. meterReaders interested in more details could read well-known textbooks [NK96, RY20]. Informally, two parties cooperate to compute a function f, each party has a separate part of the input and has unlimited computational power. How many bits do they need to exchange in order to compute f?

3.1 Two-Way Communication Complexity

Consider a function $f : X \times Y \to Z$. Informally, (deterministic) two-way communication complexity is when two parties, Alice and Bob, wish to compute $f(x, y)$. Alice's input is $x \in X$ and Bob's input is $y \in Y$. They can exchange information back and forth until being able to compute the function.

In the original model given in [Yao79], the output $f(x, y)$ must be completely determined by the transcript of the protocol (e.g. the bits exchanged by Alice and Bob). This can be seen as a way to formalize the fact that both parties learn the output of the computation. In any case, if (say) Alice computes $f(x, y)$, then she can send this value to Bob. When the function f is a predicate, i.e. when its output is a single bit, this increases the total size of the transcript by one — a quantity usually considered to be asymptotically negligible. The existing literature almost exclusively deals with this specific case.

However, when the output of the function is large, i.e. when the range Z is larger than $\{0, 1\}$, the situation is not so simple. One has to decide what exactly is the output of the protocol and who emits it. The answers to these questions lead to different notions of communication complexity. Fontes, Laplante, Laurière and Nolin [FLLN23] recently explored the relationships between different such models.

We are interested in the situation where not only the input, but also the output is split between the two parties, who have to emit their part of the output at the end of the protocol. This is essentially the same model as in [Len90], that was proposed in the context of deriving area-time lower bounds on circuits.

We thus consider a function $f : X \times Y \to U \times V$ where both the input and the output are split between the two parties. If $f(x, y) = (u, v)$, then we consider communication protocols for f where Alice (resp. Bob) has local input x (resp. y) and must emit u (resp. v). To some extent, this can be seen as the simultaneous evaluation of two functions $f_a : X \times Y \to U$ (for Alice) and $f_b : X \times Y \to V$ (for Bob).

A *protocol* π is specified by a rooted binary tree. Each internal node v has two children, and is *owned* either by Alice or by Bob. Each internal node v is also associated with a function g_v that maps the input of the owner of v to $\{0, 1\}$. We interpret the output of g_v as one of the children of v in the tree by associated 0 with the left child and 1 with the right child. The outcome of the protocol π on input $(x, y) \in X \times Y$ is a leaf of the tree, computed as follows. The parties first set the current vertex to the root of the tree. If this vertex is owned by Alice, she announces the bit $g_v(x)$, otherwise Bob announces $g_v(y)$. Both parties set the new current vertex to be the child of v indicated by the announced value of g_v. This process is repeated until the current vertex is a leaf, and this leaf is the outcome of the protocol.

The input (x, y) induces a path from the root of the protocol tree to the leaf $\pi(x, y)$. This path corresponds to the transcript of the conversation between the parties. However, the tree itself encodes all possible transcripts. We say that the protocol π computes a function f if there are two maps ϕ and ψ such that $f(x, y) = (\phi(x, \pi(x, y)), \psi(y, \pi(x, y)))$. Intuitively, ϕ produces Alice's output (given Alice's input and the transcript) while ψ produces Bob's output.

The *length* of the protocol π is the height of the binary tree, namely the length of the longest possible transcript that may occur when the protocol is executed. The *communication complexity* of a function f is the minimum length achieved by a protocol that computes f.

We will use several times the well-known fact that if Alice owns an n-bit string x and Bob owns another n-bit string y, the communication complexity of testing whether $x = y$ is exactly $n + 1$ bit (Alice sends x to Bob, and Bob sends back the Boolean that tells whether $x = y$, so that the result can be obtained from the transcript of the protocol). More formally, consider the predicate $\texttt{EQ}_n : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ that evaluates to 1 when its two arguments are equal.

Theorem 1 ([MS82])**.** *The communication complexity of* $\texttt{EQ}_n$ *is* $n + 1$.

3.2 Best-Case Communication Complexity

We are primarily interested in the scenario where the inputs of a single function $h : \{0, 1\}^{2n} \to \{0, 1\}^m$ are equally divided between the two parties. In other terms, h is a function of $2n$ Boolean variables, and Alice (resp. Bob) possesses n input bits.

Definition 1. Let $h : \{0,1\}^{2n} \rightarrow \{0,1\}^m$ be a function and write $h(x_1, \ldots, x_{2n}) = (y_1, \ldots, y_m)$. Let X and Y be a partition of the input variables $x_1, \ldots, x_{2n}$ into two disjoints sets. Also let U and V be an arbitrary partition of the outputs $y_1, \ldots, y_m$ into two disjoints sets. The *communication complexity* of h for the partition (X, Y, U, V) is denoted as $CC^{X,Y,U,V}(h)$.

The communication complexity of a given function is very dependent on the choice of the input/output partition. For instance, consider the identity function over $2n$ bits, where Alice receives the first half of the input and Bob the second half. If both parties have to emit *their* input, then the communication complexity is zero. If both parties have to emit *each other*'s input, then they have to exchange their inputs and the communication complexity is $2n$. This leads to notions of "best-case" communication complexity with varying amounts of freedom in the input/output partition.

Definition 2. The *best-case communication complexity of h for the input partition* (X, Y) is the minimum of $CC^{X,Y,U,V}(h)$ over all (arbitrary) partitions U, V of $y_1, \ldots, y_m$. It is denoted as $CC^{X,Y}_{best}(h)$.

The *best-case communication complexity of h* is the minimum of $CC^{X,Y,U,V}(h)$ over all partitions X, Y of $x_1, \ldots, x_{2n}$ with $|X| = |Y|$ and all (arbitrary) partitions U, V of $y_1, \ldots, y_m$. It is denoted as $CC_{best}(h)$.

3.3 AT^2 Lower Bounds from Communication Complexity

Suppose that a circuit computing a function h is sliced in two halves. The cut partitions the input and output bits in two categories (those that are read/emitted on each side of the cut). The communication complexity with this specific input/output partition is a lower bound on the amount of information that the two parts must exchange to compute the output. If the number of wires between the two halves is large, this information flow can be fast, but the area will also be large. If the number of wires between the two halves is small, the circuit can be compact but the amount of time needed to transfer the information flow will be large. This argument is well-known [Len90].

In this section we derive an "effective" version where the constants are made explicit. We use the following notations: h denotes the common height of all cells, w denotes the width of the largest cell, ν the minimum wire width, β the minimum space between two different wires and γ the maximum number of wires that can cross a line of length 1m. We begin with a simple (but relevant) case.

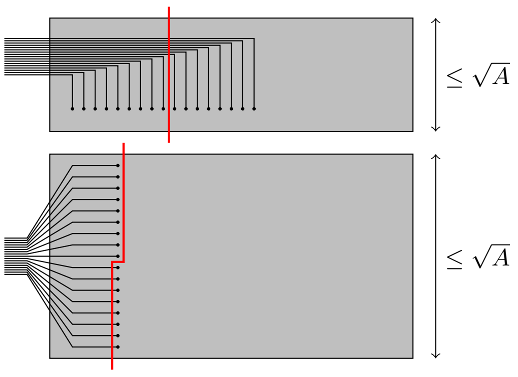
Theorem 2. *Consider a (rectangular) circuit of area A that evaluates a function $f : \{0,1\}^{2n} \rightarrow \{0,1\}^m$ in T clock cycles and receives its input in a single clock cycle from a bus (this means that the input ports are aligned horizontally or vertically as in Fig. 4a). Let (I, J) denote the input partition where Alice receives the first n bits of the input and Bob receive the last n ones. Then*

$$\max(h, w) + \sqrt{A} \geq \frac{CC^{I,J}_{best}(f)}{\gamma T}.$$

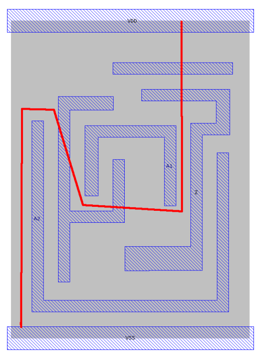
Lifting the restriction that input bits must arrive on a bus leads to the following result, in terms of the best-case communication complexity.

Theorem 3. *Consider a (rectangular) circuit of area A where each input port receives the same number of input bits, that evaluates a function $f : \{0,1\}^{2n} \rightarrow \{0,1\}^m$ in T clock cycles. Then*

$$\left(1 + 2\frac{w}{h}\right)\sqrt{A} + hw\left(\frac{6}{\beta} + \frac{4}{\nu}\right) \geq \frac{CC_{best}(f)}{\gamma T}.$$



(a) Cutting a rectangular circuit that receives its input from a bus.



(b) Cutting the XOR$_2$ cell of FreePDK15 to put A_1 on the left and A_2 on the right.

Figure 4: Cuts that separate input bits.

Table 6: Constants for the lower bounds, with A in μm^2 and all physical measures in μm. γ is given in wires / μm.

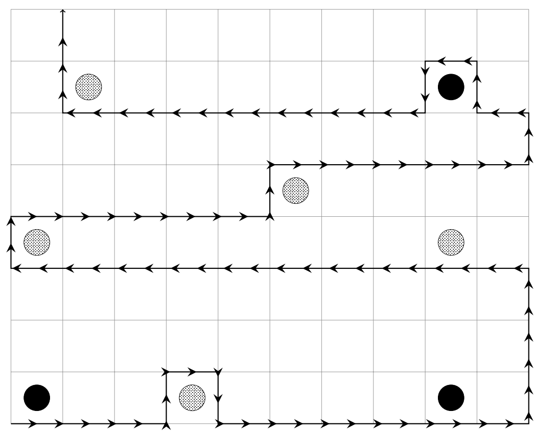
Process	h	w	ν	β	γ	Thm. 2		Thm. 3	
						ϕ	ψ	ϕ	ψ
FreePDK15	0.768	2.112	0.028	0.036	70.5	70.5	149	458	35394
FreePDK45	1.4	6.27	0.065	0.065	19.7	19.7	124	196	26604
GF 180nm	3.92	39.2	0.23	0.23	4.7	4.68	184	98	31284

We compiled the relevant information for some common open Process Design Kits. Theorem 2 yields a lower bound of the kind $\phi\sqrt{A}+\psi \geq CC^{I,J}_{best}(f)/T$ and theorem 3 yields $\phi\sqrt{A}+\psi \geq CC_{best}(f)/T$. Table 6 shows the values of the relevant parameters as well as the constants ϕ and ψ for some PDKs. The bound provided by theorem 2 is better (ϕ is necessarily lower), and it is also better because it involves the communication complexity for a fixed input partition, not the "best-case" communication complexity that can be lower.

Proof of theorem 2. Without loss of generality, assume that the width of the circuit is the longer side. If the input ports are aligned horizontally, draw a vertical line that separates the n left ones from the n right ones (as in Fig. 4a, top). This line has length less than $\sqrt{A}$. If the input ports are aligned vertically, draw a line with a "dogleg" in the middle to separate the top n ones from the bottom n ones (as in Fig. 4a, bottom). The horizontal segment of the dogleg can have the maximum length of a single cell to make sure that none of the above input port are touched by the cut. The cut has length $\sqrt{A}+h$. It follows that, in all cases (even if the longer side is the height), the cut has length less than $L := \sqrt{A}+\max(h, w)$.

The two parts of the circuit may communicate using all the wires that cross the cut. By definition of γ, there are at most $L\gamma$ such wires, so that in T clock cycles the two parts of the circuit can exchange at most $TL\gamma$ bits. Together they form a two-party protocol that evaluates f. Therefore they have to exchange at least $CC^{I,J}_{best}(f)$ bits. □

In general, we need argue that there is a short curve that cuts the circuit in two while separating one half of the input from the other half. Consider the case of a single cell. Consider a partition of the input ports of a cell in two categories L and R. We claim that the cell can always be cut in two by a continuous line that starts at the bottom of the cell, ends at the top of the cell, and separates L and R. (this claim is proved below). See for


(a) Black circles represent the ports that need to be on the left of the cut.

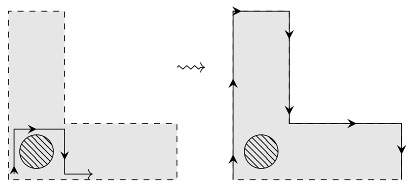

(b) Expanding a port point to its original size expands the cut line

Figure 5: Cutting a single cell to separate input ports.

instance Fig.4b. Consider the shortest possible cut, for any possible partition of the input ports, and denote by ℓ the maximum length over all cells of the library. In other terms, the input ports of any cell of the libray can be separated arbitrarily in two categories by a cut of length at most ℓ.

Lemma 1. $h \leq \ell \leq hw\left(\frac{6}{\beta}+\frac{4}{\nu}\right)$.

Proof. That $h \leq \ell$ follows from the fact that the cut must cross the entire cell from bottom to top.

Divide the cell into squares of length $\beta/2$ as in figure 5a and shrink all input ports to their bottom-left point. If for an input port p, its representative point touches an edge of the grid, replace it with a nearby point inside p's region that does not touch any edge. Since the distance between two points is at least β, then there is always a full square from the grid, of length β, between any two points of two different input ports.

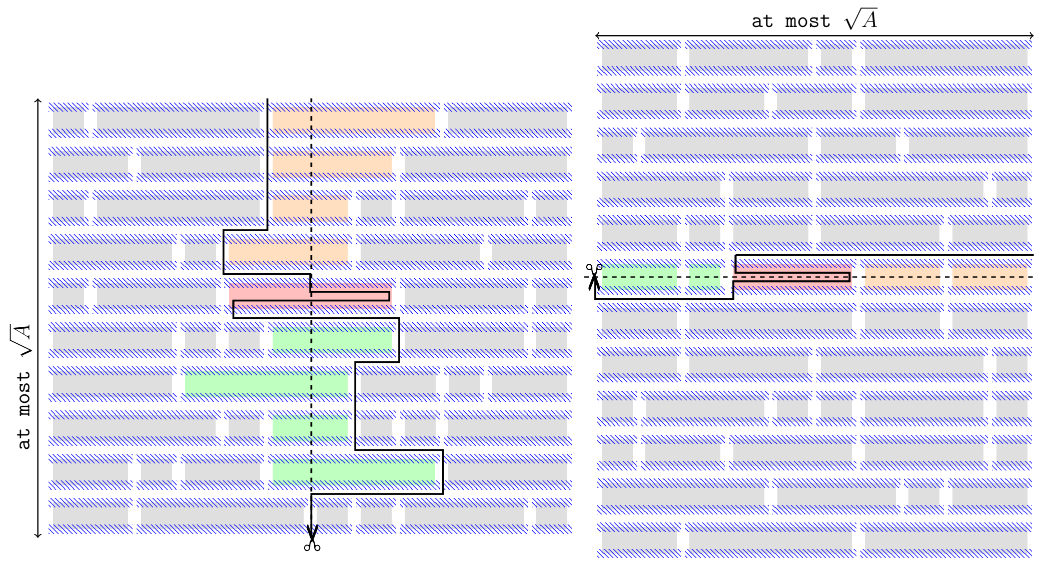
We cut the points that are on the bottom (at level $y = 0$), if any, then points at level $y = 1$ (if any), and so on. At each level, the cut either starts from the rightmost or the leftmost x position.

If the cut line starts at the leftmost x position at level y, then steer it to the right tracing the bottom edges of the grid at level y. When it encounters a port point, go above it if the port point should be to the right of the cut, otherwise go below it. After separating a port point that it is not the rightmost point in this level, steer the cut down to level y, then continue moving to the right. After the cut separates the rightmost point, steers it up to the nearest y' that has port points.

If the cut line starts at the rightmost x position, repeat the above procedure with reversing directions except steering the cut down after passing a port point, and steering the cut up after passing all port points. Figure 5a shows an example of this algorithm. Separating the points requires at most traversing 3 sides of all the squares inside the grid. Write $\sigma = hw$ the area of the cell. Then the cut has length less than $6\sigma/\beta$.

After separating the ports shrunk to a single point, expand them to their original shape while pushing the cut in all directions it touches a port, see Figure 5b. This increases the length of the cut by the sum of the perimeters of the input ports in the worst case. Denote this sum by P.

Each input port is an assembly of non-overlapping rectangles. Number these rectangles from 1 to n and say that the i-th rectangle has dimension $w_i \times h_i$. All the rectangles are



(a) Vertical cut. (b) Horizontal cut.

Figure 6: Cuts along the shortest side (goes through the red cell).

contained in the cell, therefore $\sum w_i h_i \leq \sigma$. At least one side of the rectangles must be larger than the minimum width of wires (ν), which brings:

$$\nu \sum_{i=1}^{n} \max(w_i, h_i) \leq \sum_{i=1}^{n} \min(w_i, h_i) \max(w_i, h_i) = \sum_{i=1}^{n} w_i h_i \leq \sigma.$$

The perimeter of the i-th rectangle is $2(h_i + w_i)$, and we find:

$$P = 2\sum_{i=1}^{n}(h_i + w_i) \leq 4\sum_{i=1}^{n} \max(h_i, w_i) \leq 4\frac{\sigma}{\nu}$$

This establishes the lemma. □

We need the following setup before proving theorem 3.

Write $[n] = \{1, \ldots, n\}$. Consider a collection of n weighted intervals of $\mathbb{R}$. The i-th interval is $[\ell_i, r_i]$ and it has weight w_i. Given a subset $I \subseteq [n]$, write $W(I) = \sum_{i \in I} w_i$ the total weight of the intervals in I. Write $W := W([n])$ the total weight of all the intervals. For any α, define:

$$\begin{aligned} L_\alpha(I) &= \{i \in I \ : \ r_i < \alpha\} \\ R_\alpha(I) &= \{i \in I \ : \ \alpha < \ell_i\} \\ M_\alpha(I) &= \{i \in I \ : \ \ell_i \leq \alpha \leq r_i\} \end{aligned}$$

$L_\alpha(I)$ contains the intervals of I that are strictly on the left of the line $x = \alpha$. $R_\alpha(I)$ and $M_\alpha(I)$ contain the intervals that are strictly on the right or that intersect the line, respectively.

Lemma 2. *For any subset of intervals I and any $0 \leq X \leq W(I)$, there is an α such that $W(L_\alpha(I)) \leq X$ and $W(R_\alpha(I)) \leq W(I) - X$.*

Proof. Take all right ends of the intervals in I and order them: $\{r_i \ : \ i \in I\} = \{\alpha_1 < \cdots < \alpha_r\}$. It is not difficult to see that $W(L_\alpha(I))$ is an increasing function of α that stays constant whenever $\alpha \in (\alpha_i, \alpha_{i+1}]$.

As α passes α_i, several intervals may switch from M to L. It follows that $L_{\alpha_{i+1}}(I) \subseteq L_{\alpha_i}(I) \cup M_{\alpha_i}(I)$.

Now, take the largest i such that $W(L_{\alpha_i}(I)) \leq X$. By definition, this means that $W(L_{\alpha_{i+1}}(I)) \geq X$. By the above remark, We have $X \leq W(L_{\alpha_i}(I)) + W(M_{\alpha_i}(I)) = W - W(R_{\alpha_i}(I))$. It follows that the lemma holds with $\alpha = \alpha_i$.

□

Proof of theorem 3. If there is only one input port, then the circuit needs $T \geq 2n$ cycles to read its input. Also recall that $n \geq CC_{best}(f)$. Finally, we claim that $\ell\gamma \geq 1$. Indeed, $h \leq \ell$ by lemma 1 and there is at least one wire that can cross a cell horizontally (otherwise it would be impossible to connect the cells). Therefore, $\ell \geq 1/\gamma$ and we have:

$$\left(1 + \frac{w}{h}\right)\sqrt{A} + hw\left(\frac{6}{\beta} + \frac{4}{\nu}\right) \geq \ell \geq \frac{1}{2\gamma} \geq \frac{CC_{best}\,(f)}{\gamma T}.$$

So the theorem is established in this case.

Assume now that there are at least two input ports. Without loss of generality, assume the height of the circuit is the shortest side as in Figure 6a. Because the circuit is assumed to be rectangular, its height is less than $\sqrt{A}$.

Project all the cells of the circuit onto the x axis. Because cells are rectangles, their projections are closed intervals. Set the weight of one of these intervals to the number of input bits received by the corresponding cell. The total weight of the intervals is $2n$. By lemma 2 there is an x_0 such that L input bits are received by cells that are strictly on the left of the line $x = x_0$, R input bits are received by cells that are strictly on the right, and M input bits are received by cells that intersect the line, with $L, R \leq n$ and $L + M + R = 2n$.

Next, project the cells that intersect the line $x = x_0$ onto the y axis. This again defines a weighted collection of intervals with total weight C. Use lemma 2 again with $X = n - L$ to find an y_0 such that U input bits are received by cells strictly above the line $y = y_0$ (they are orange in Fig. 6a), D input bits are received by cells below the line (green in the same figure) and C input bits are received by the single cell that possibly intersect the line (red). We thus have $L + U \leq n$ and $R + D \leq n$ and $(L + U) + C + (D + R) = 2n$.

If $C \neq 0$, use lemma 1 to cut the single cell that intersect both lines in two parts A and B such that A receives $n - L - U$ input bits. Then the input ports in $L + U + A$ (resp. $R + D + B$) receive exactly n input bits.

We now draw a curve that splits the circuit in two parts, where the n input bits received by $L + U + A$ are on the left of the curve. Starting from the bottom row of cells at $x = x_0$, we go round the green cells that intersect the line to the right. Passing a single cell requires moving to the right by at most w, up by h and back left by at most w. We go up until we hit the red cell (that intersect the line $y = y_0$). We cut this cell using lemma 1, which add at most ℓ to the length of the curve. Finally, we move up the circuit going round the orange cells to the left. This adds $2w + h$ per row of cells passed in this way. The total length of the curve is at most $L := \sqrt{A}/h(2w + h) + \ell$.

If the width of the circuit is the shortest side (as in Fig. 6b), then a similar reasoning leads to a (shorter) cut of lenght $L' := \sqrt{A} + \ell$. It follows that in both cases, there is a cut of length less than L.

The cut defines a balanced partition of the input bits, and an arbitrary partition of the output bits. Say that Alice has the inputs on the left and emits the output on the left. Bob receives the inputs on the right and emits the outputs on the right. The number of bits that must cross the cut is at least the communication complexity for this specific input/output partition. However, at most γL wires may cross the cut. Therefore, a certain number of clock cycles is needed for the required number of bits to cross the cut. Combining all above inequalities, we obtain $T\gamma L \geq CC_{best}\,(f)$. Easy manipulations (and the value of ℓ from lemma 1) conclude the proof. □

Table 7: Known upper-bounds on the best-case communication complexity of the Möbius transform.

n	1	2	3	4	5	6	7	8
$CC_{best}(M_n)$	1	2	3	6	≤ 10	≤ 22	≤ 40	≤ 94
$2^{n-1}/n$	1	1	≈ 1.3	2	3.2	≈ 5.3	≈ 9.1	16

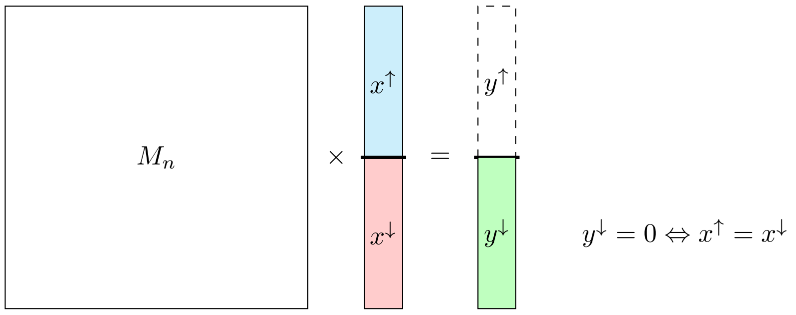


Figure 7: How to compare bit strings using the Möbius transform. Blue bits and red bits are compared in the order they occur. Green bits are the comparison bits.

4 Communication Complexity of the Möbius Transform

In this section, we prove a lower bound on the best-case communication complexity of the Möbius transform. The main tool we use is a reduction to equality testing: given a protocol that computes the Möbius transform, we build a a protocol that tests equality between bit strings. By theorem 1, there is a lower bound of the communication complexity of equality testing, and the reduction extends this lower bound to the computation of the Möbius transform. The main results of this section are the following:

Theorem 4. *Let (I, J) denote the input partition where Alice owns the first half of x and Bob owns the second half. Then $CC^{I,J}_{best} \geq 2^{n-1} - 1$.*

Theorem 5. $CC_{best}(M_n) \geq 2^{n-1}/n - 1$.

The bound provided by theorem 4 is tight, because Alice can simply send her 2^{n-1} output bit to Bob, who then does the whole computation and emits the result. This means that for the "natural" input partition where Alice has the first half, the communication complexity is maximal. On the other hand, the "best-case" bound of theorem 5 seems quite loose, as illustrated by Table 7. We obtained the numbers presented in this table by finding input/output partitions and determining the actual communication complexity (using techniques that are beyond the scope of this article).

4.1 Equality Testing With the "Natural" Input Partition

In this section we prove theorem 4. Consider the setting Alice owns the first half of the input vector and Bob owns the second half (this is the "natural" input partition). In this case, a two-party protocol to compute the Möbius transform directly leads to an equality testing protocol, for which a lower bound on communication complexity is readily available. The main tool used in the reduction is the following simple argument.

Lemma 3. *Let $x^\uparrow, x^\downarrow, y^\uparrow$ and $y^\downarrow$ denote vectors of length 2^n, with*

$$M_{n+1} \begin{pmatrix} x^\uparrow \\ x^\downarrow \end{pmatrix} = \begin{pmatrix} y^\uparrow \\ y^\downarrow \end{pmatrix}.$$

Then $x^\uparrow = x^\downarrow$ if and only if $y^\downarrow = 0$.

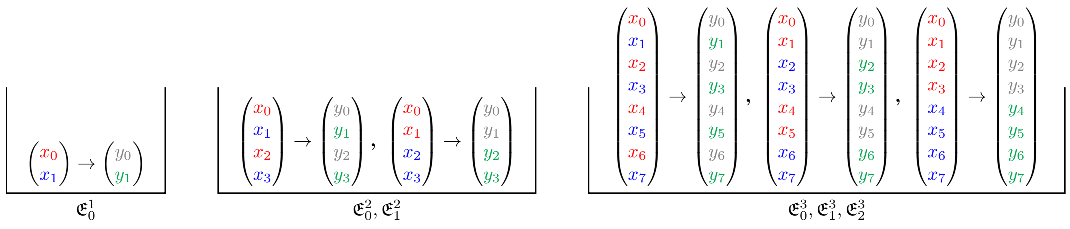

$\mathfrak{E}_0^1$
$\mathfrak{E}_0^2, \mathfrak{E}_1^2$
$\mathfrak{E}_0^3, \mathfrak{E}_1^3, \mathfrak{E}_2^3$

Figure 8: The equality-testing configurations of M_3. Blue bits are compared to red bits in the order they appear in. Green bits are the all zero if they match.

Proof. Because $M_{n+1} = \begin{pmatrix} M_n & 0 \\ M_n & M_n \end{pmatrix}$, we find that $y^{\downarrow} = M_n(x^{\uparrow} + x^{\downarrow})$. Because M_n is invertible, $y^{\downarrow} = 0$ is equivalent to $x^{\uparrow} + x^{\downarrow} = 0$, namely $x^{\uparrow} = x^{\downarrow}$. □

Consider a protocol π that computes M_n with the natural input partition (Alice owns $x^{\uparrow}$ and Bob owns $x^{\downarrow}$ using the notations of lemma 3 and Fig. 7). The output partition can be arbitrary. We build a protocol π' that tests if Alice's input is equal to Bob's input. Alice and Bob first run π. The lower half of the result ($y^{\downarrow}$) is then split between them. Both parties locally compute the Boolean OR of the bits of $y^{\downarrow}$ that they own and broadcast the result. This completes the description of π'. By lemma 3, $x^{\uparrow} = x^{\downarrow}$ if and only if $y^{\downarrow} = 0$ and this condition is realized if and only if the last bit sent by each party is zero. Therefore π' correctly tests if $x^{\uparrow} = x^{\downarrow}$ and the result can be inferred from the transcript of its execution. By theorem 1, this shows that:

$$CC_{best}^{I,J}(M_n) \geq 2^{n-1} - 1,$$

where (I, J) denote the natural input partition.

4.2 Equality Testing for Some Other Input Partitions

In order to prove theorem 5, we first generalize the previous reasoning to more input partitions that we call "equality-testing configuration". Figure 8 illustrates some of them.

Definition 3. For $0 \leq \ell < n$, the ℓ-th *equality-testing configuration* for M_n, is $\mathfrak{E}_\ell^n = (\mathcal{I}_\ell^n, \mathcal{C}_\ell^n)$, with:

$$\mathcal{I}_\ell^n = \bigcup_{j=0}^{2^{n-\ell-1}-1} \left\{ \{i, i+2^\ell\} \ : \ j2^{\ell+1} \leq i < j2^{\ell+1} + 2^\ell \right\},$$

$$\mathcal{C}_\ell^n = \bigcup_{j=0}^{2^{n-\ell-1}-1} \left\{ k + 2^\ell \ : \ j2^{\ell+1} \leq i < j2^{\ell+1} + 2^\ell \right\}.$$

The interest of these input partitions is that they allow the comparison of the 2^{n-1} bits of Alice with the 2^{n-1} bits of Bob. This is what lemma 5 below claims, but we first need a technical result about their structure.

Lemma 4. *For all $n \geq 1$ and all $0 \leq \ell < n$, we have:*

$$\mathcal{I}_\ell^{n+1} = \mathcal{I}_\ell^n \cup \left\{ \{i + 2^n, j + 2^n\} \ : \ \{i, j\} \in \mathcal{I}_\ell^n \right\}$$

$$\mathcal{C}_\ell^{n+1} = \mathcal{C}_\ell^n \cup \left\{ k + 2^n \ : \ k \in \mathcal{C}_\ell^n \right\}$$

In addition, all integers $0 \leq u < 2^n$ appear in $\mathcal{I}_\ell^{n+1}$.

Proof. By expanding definition 3, and separating $j = 0$ to $2^{n-l-1} - 1$ and $j = 2^{n-l-1}$ to $2^{n-l} - 1$:

$$\begin{aligned}
\mathcal{I}_\ell^{n+1} &= \mathcal{I}_\ell^n \cup \bigcup_{j=2^{n-\ell-1}}^{2^{n-\ell}-1} \Big\{ \{i, i+2^\ell\} \ : \ j2^{\ell+1} \le i < j2^{\ell+1} + 2^\ell \Big\} \\
&= \mathcal{I}_\ell^n \cup \bigcup_{r=0}^{2^{n-\ell-1}-1} \Big\{ \{i, i+2^\ell\} : \ (2^{n-\ell-1}+r)2^{\ell+1} \le i < (2^{n-\ell-1}+r)2^{\ell+1} + 2^\ell \Big\} \quad \text{(1a)} \\
&= \mathcal{I}_\ell^n \cup \bigcup_{r=0}^{2^{n-\ell-1}-1} \Big\{ \{k+2^n, \ (k+2^\ell)+2^n\} \ : \ r2^{\ell+1} \le k < r2^{\ell+1} + 2^\ell \Big\} \quad \text{(1b)} \\
&= \mathcal{I}_\ell^n \cup \bigcup_{r=0}^{2^{n-\ell-1}-1} \Big\{ \{k+2^n, \ k'+2^n\} \ : \ \{k, k'\} \in \mathcal{I}_\ell^n \Big\}
\end{aligned}$$

In line (1a), $j = r + 2^{n-\ell-1}$, and in line (1b), $i = k + 2^\ell$. The exact manipulation proves the statement for $\mathcal{C}^{n+1}$.

The second point of the lemma is established by induction on n. If $n = 1$ (and $\ell = 0$), then $\mathcal{I}_\ell^n = \{\{0, 1\}\}$ and all integers strictly less than 2 occur. If $n > 1$, then by induction hypothesis all integers less than 2^n occur in $\mathcal{I}_\ell^n$, and therefore in $\mathcal{I}_\ell^{n+1}$ given the inclusion proved above. Take an integer $2^n \le k < 2^{n+1}$. Then $k - 2^n$ occur in $\mathcal{I}_\ell^n$, so that k occur in $\mathcal{I}_\ell^{n+1}$ given the first point of the lemma. ☐

Lemma 5. *Let $x \in \mathbb{F}_2^n$ and $y = M_n \cdot x$. For all $0 \le \ell < n$, we have:*

$$x_i = x_j \text{ for all } \{i, j\} \in \mathcal{I}_\ell^n \iff y_k = 0 \text{ for all } k \in \mathcal{C}_\ell^n$$

Proof. We first observe that if $\ell = n - 1$, then:

$$\begin{aligned}
\mathcal{I}_n^{n+1} &= \Big\{ \{i, i+2^n\} \ : \ 0 \le i < 2^n \Big\} \\
\mathcal{C}_n^{n+1} &= \Big\{ k \ : \ 2^n \le k < 2^{n+1} \Big\}.
\end{aligned}$$

In this case, the theorem holds because it reduces to lemma 3. We now assume that $\ell < n - 1$ and we prove the result by induction on n. If $n = 1$ and $\ell = 0$, then the result holds as argued above. If $n \ge 2$, let us decompose x and y into their two halves by writing

$$\begin{pmatrix} y^\uparrow \\ y^\downarrow \end{pmatrix} = \begin{pmatrix} M_{n-1} & 0 \\ M_{n-1} & M_{n-1} \end{pmatrix} \cdot \begin{pmatrix} x^\uparrow \\ x^\downarrow \end{pmatrix}$$

Also write $z = M_{n-1} \cdot x^\downarrow$ so that $y^\downarrow = y^\uparrow + z$. Denote by (A) and (B) the conditions on the left-hand side and right-hand side of the statement of the theorem, respectively. We consider the following equivalences: 1) Lemma 4 states that (A) holds if and only if both $x_i^\uparrow = x_j^\uparrow$ and $x_i^\downarrow = x_j^\downarrow$ for all $\{i, j\} \in \mathcal{I}_\ell^{n-1}$. 2) By induction hypothesis, $y_k^\uparrow = 0$ for all $k \in \mathcal{C}_\ell^{n-1}$ if and only if $x_i^\uparrow = x_j^\uparrow$ for all $\{i, j\} \in \mathcal{I}_\ell^{n-1}$. 3) Also by induction hypothesis, $z_k = 0$ for all $k \in \mathcal{C}_\ell^{n-1}$ if and only if $x_i^\downarrow = x_j^\downarrow$ for all $\{i, j\} \in \mathcal{I}_\ell^{n-1}$. 4) Lemma 4 grants us that $y_k^\uparrow = y_k^\downarrow = 0$, for all $k \in \mathcal{C}_\ell^{n-1}$ if and only if (B) holds.

Now, suppose (A). Then points 1,2 and 3 combined with the definition of z prove that $y_k^\downarrow = 0$ for all $k \in \mathcal{C}_\ell^{n-1}$. Point 4 then implies (B). Conversely, assume (B). Point 4 along with the definition of z implies that $z_k = 0$ for all $k \in \mathcal{C}_\ell^{n-1}$. Then points 1,2 and 3 imply (A). ☐

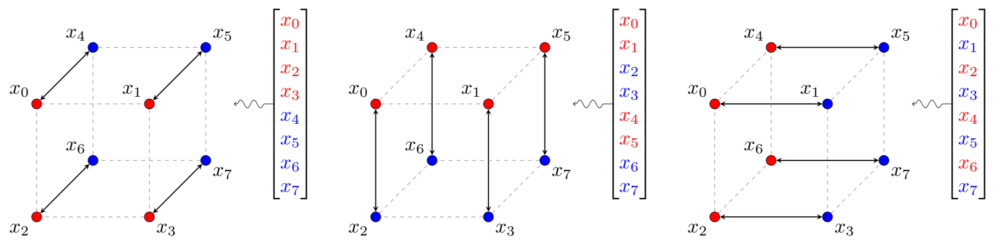


Figure 9: Equality-testing configuration graph for M_3.

4.3 Lower Bound for the Best-Case Communication Complexity

In this section we prove theorem 5. The results presented in sections 4.1 and 4.2 show that the communication complexity of the Möbius transform is high for some specific input partitions. However, to obtain a lower bound on the best-case communication complexity, we need to cover all possible input partitions. While in general we cannot test if Alice's input is equal to Bob's input, we can always test if a $1/n$ fraction of their input bits are equal. This happens because any arbitrary input partition is (relatively) close to at least one of the equality-testing configurations of the previous section.

Definition 4. The *equality-testing configuration graph* for M_n is the (undirected) graph $G_n = (V, E)$, where $V := \{0, \dots, 2^n - 1\}$, and $E = \cup_{\ell=0}^{n-1} \mathcal{I}_\ell^n$. In other terms, $i \leftrightarrow j$ is an edge in G_n if and only if x_i and x_j can be compared together by one of the equality-testing configurations.

The equality-testing configuration graph for M_3 is shown in figure 9. The n-hypercube graph Q_n is the graph whose vertices are n-bit strings where u and v are adjacent if and only if they differ in exactly one bit position.

Lemma 6. *The equality-testing configuration graph G_n for M_n is exactly the n-hypercube graph Q_n.*

Proof. First, we know that the n-hypercube graph has $n2^{n-1}$ edges, while by definition G_n has at most $n2^{n-1}$ edges. Both graph have the same set of vertices. We now show that Q_n is contained in G_n. This proves the theorem.

Pick an edge $u \leftrightarrow v$ in the hypercube graph. The two n-bit strings u and v differ in a single position, say the ℓ-th. Without loss of generality, assume that $u < v$, which means that $v = u + 2^\ell$. Because u and v agree on all bits except the ℓ-th one, we can write $u = j2^{\ell+1} + r$ and $v = j2^{\ell+1} + 2^\ell + r$ with $0 \le j < 2^{n-\ell-1}$ and $0 \le r < 2^\ell$. By the definition of $\mathcal{I}_\ell^n$, this shows that $\{u, v\} \in \mathcal{I}_\ell^n$ and therefore that there is an edge $u \leftrightarrow v$ in G_n. □

We are now ready to prove theorem 5.

Consider that $y = M_n \cdot x$ and let (I, J) denote an arbitrary input partition: Alice owns $(x_i)_{i \in I}$ and Bob owns $(x_j)_{j \in J}$. We will show that $CC_{best}^{I,J} \ge 2^{n-1}/n$, and the theorem follows.

The two bits x_i and x_j (with $i \in I$ and $j \in J$) can be compared by one of the equality-testing configurations if $i \leftrightarrow j$ is an edge in the n-hypercube graph. Given an arbitrary subset A of the vertices, the *boundary* $\partial A := \{\{i, j\} : i \in A, j \notin A\}$ is the set of edges connecting vertices of A to vertices of its complement. Hypercube graphs enjoy a nice *edge-isoperimetric inequality* [Til00]: $|\partial A| \ge \log_2\left(\frac{|V|}{|A|}\right)|A|$.

The input partition (I, J) in fact partitions in two the set of vertices of Q_n. By the edge-isoperimetric inequality, we find that there are at least 2^{n-1} edges between Alice's input bits and Bob's input bits. Because the graph is the assembly of n subgraphs corresponding to the n equality-testing configuration $(\mathfrak{E}_\ell^n)_{0 \leq \ell < n}$, it follows that there is ℓ_0 such that $\mathfrak{E}_{\ell_0}^n$ contains at least $2^{n-1}/n$ edges between I and J. Define $\Delta := \mathcal{I}_{\ell_0}^n \cap I \times J$ and number its elements $\Delta = \{(i_0, j_0), (i_1, j_1), \dots\}$. By the previous reasoning, $|Delta| \geq 2^{n-1}/n$.

Let π denote a two-party protocol to compute $y = M_n \cdot x$ with the (arbitrarily chosen) input partition (I, J) and an arbitrary output partition. We construct an equality-testing protocol π' for inputs of size $|\Delta|$. In this protocol, Alice (resp. Bob) receives a bitstring u (resp. v) of size $|\Delta|$. Alice builds a vector x by setting $x_{i_k} \leftarrow u_k$ for $0 \leq k < |\Delta|$ and sets all other coordinates to zero. Bob does the same with $x_{j_k} \leftarrow v_k$. Alice and Bob execute the protocol π to compute $y = M_n \cdot x$. Alice then computes the Boolean OR of all the bits of y with index in $\mathcal{C}_{\ell_0}^n$ that she owns and broadcasts it. Bob does the same.

We claim $u = v$ if and only if the last bit emitted by each party is zero. This follows from lemma 5. The protocol π' decides equality of bit strings of size $|\Delta|$, so by theorem 1 its communication complexity is at least $|\Delta| + 1$. It follows that the communication complexity of π is at least $|\Delta| - 1$, and the theorem is proved by the lower bound on $|\Delta|$.

Conclusion

We have shown that the area of circuits that compute the Möbius transform is asymptotically dominated not by the area of the cells, but by the area occupied by the wires. We have shown that the generic lower bound on wiring area can become quite close to the area of the cells in actual circuits, using actual process design kits.

This shows that the usual assumption that "the area of the cells is the area of the circuit" should be used with caution. At the very least, if only synthesis of a hardware design has been performed, as opposed to placement and routing, then reporting the area *of the cells* is free of any assumption. Other performance metrics (power, energy, latency) are probably affected in the some way or some other.

Pushing this line of research further could be done by taking various hardware design of cryptographic relevance (block ciphers, low-latency PRFs for instance, or the Möbius transform), performing synthesis, placement and routing, then comparing the "extrapolated" results obtained after synthesis with the "real" results obtained after routing.

Acknowledgements

Interesting and lively discussions with Subhadeep Banik and Francesco Regazzoni triggered this work. We are indebted to Roselyne Chotin, Franck Wajsbürt, Jean-Paul Chaput and Gabriel Rocherolle for their thorough explanations. The authors are grateful to Jan Vondrák for his lecture notes on polyhedral techniques in combinatorial optimization and to Alexandra Elbakyan for enabling us to read the works of our colleagues.

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations.

We acknowledge financial support from the French *Agence Nationale de la Recherche* under project "GORILLA" (ANR-20-CE39-0002) and "CRYPTANALYSE" (ANR-22-PECY-0010).

References

[ABC+24] Ravi Anand, Subhadeep Banik, Andrea Caforio, Tatsuya Ishikawa, Takanori Isobe, Fukang Liu, Kazuhiko Minematsu, Mostafizar Rahman, and Kosei Sakamoto. Gleeok: A family of low-latency PRFs and its applications to authenticated encryption. *IACR TCHES*, 2024(2):545–587, 2024.

[ABD$^+$23] Roberto Avanzi, Subhadeep Banik, Orr Dunkelman, Maria Eichlseder, Shibam Ghosh, Marcel Nageler, and Francesco Regazzoni. The QARMAv2 family of tweakable block ciphers. *IACR Trans. Symm. Cryptol.*, 2023(3):25–73, 2023.

[BD15] Kirti Bhanushali and W. Rhett Davis. Freepdk15: An open-source predictive process design kit for 15nm finfet technology. In Azadeh Davoodi and Evangeline F. Y. Young, editors, *Proceedings of the 2015 Symposium on International Symposium on Physical Design, ISPD 2015, Monterey, CA, USA, March 29 - April 1, 2015*, pages 165–170. ACM, 2015.

[BIL$^+$21] Subhadeep Banik, Takanori Isobe, Fukang Liu, Kazuhiko Minematsu, and Kosei Sakamoto. Orthros: A low-latency PRF. *IACR Trans. Symm. Cryptol.*, 2021(1):37–77, 2021.

[BK80] R. P. Brent and H. T. Kung. The chip complexity of binary arithmetic. In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pages 190–200, New York, NY, USA, 1980. ACM.

[BP86] Gianfranco Bilardi and Franco P. Preparata. Area-time lower-bound techniques with applications to sorting. *Algorithmica*, 1(1):65–91, 1986.

[BR24a] Subhadeep Banik and Francesco Regazzoni. Compact circuits for efficient Möbius transform. *IACR TCHES*, 2024(2):481–521, 2024.

[BR24b] Subhadeep Banik and Francesco Regazzoni. Faster and more energy-efficient equation solvers over GF(2). In Johann Knechtel, Urbi Chatterjee, and Domenic Forte, editors, *Security, Privacy, and Applied Cryptography Engineering - 14th International Conference, SPACE 2024, Kottayam, India, December 14-17, 2024, Proceedings*, volume 15351 of *Lecture Notes in Computer Science*, pages 16–39. Springer, 2024.

[Din21] Itai Dinur. Cryptanalytic applications of the polynomial method for solving multivariate equation systems over GF(2). In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 374–403. Springer, Cham, October 2021.

[FLLN23] Lila Fontes, Sophie Laplante, Mathieu Laurière, and Alexandre Nolin. The communication complexity of functions with large outputs. In Sergio Rajsbaum, Alkida Balliu, Joshua J. Daymude, and Dennis Olivetti, editors, *Structural Information and Communication Complexity*, pages 427–458, Cham, 2023. Springer Nature Switzerland.

[Fre14a] 15nm open-cell library and 45nm freepdk, 06 2014. v0.1. Available online at https://si2.org/open-cell-and-free-pdk-libraries/.

[Fre14b] Freepdk15, 2014. Available online at https://eda.ncsu.edu/freepdk15/.

[Jou09] Antoine Joux. *Algorithmic cryptanalysis*. CRC Press, 2009.

[Köf24] Matthias Köfferlein. Klayout — high performance layout viewer and editor. *Version 0.29.1*, 2024.

[Len90] Thomas Lengauer. Chapter 16 - vlsi theory. In Jan Van Leeuwen, editor, *Algorithms and Complexity*, Handbook of Theoretical Computer Science, pages 835–868. Elsevier, Amsterdam, 1990.

[LMMR21] Gregor Leander, Thorben Moos, Amir Moradi, and Shahram Rasoolzadeh. The SPEEDY family of block ciphers engineering an ultra low-latency cipher from gate level for secure processor architectures. *IACR TCHES*, 2021(4):510–545, 2021.

[MMR$^+$15] Mayler G. A. Martins, Jody Maick Matos, Renato P. Ribas, André Inácio Reis, Guilherme Schlinker, Lucio Rech, and Jens Michelsen. Open cell library in 15nm freepdk technology. In Azadeh Davoodi and Evangeline F. Y. Young, editors, *Proceedings of the 2015 Symposium on International Symposium on Physical Design, ISPD 2015, Monterey, CA, USA, March 29 - April 1, 2015*, pages 171–178. ACM, 2015.

[MS82] Kurt Mehlhorn and Erik M. Schmidt. Las vegas is better than determinism in vlsi and distributed computing (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 330–337, New York, NY, USA, 1982. Association for Computing Machinery.

[NK96] Noam Nisan and Eyal Kushilevitz. *Communication Complexity*, volume 1996. Cambridge University Press, 1996.

[RY20] Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications.* Cambridge University Press, 2020.

[Sav81] John E. Savage. Area—time tradeoffs for matrix multiplication and related problems in vlsi models. *Journal of Computer and System Sciences*, 22(2):230 – 242, 1981.

[Tho79] C. D. Thompson. Area-time complexity for vlsi. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 81–88, New York, NY, USA, 1979. ACM.

[Til00] Jean-Pierre Tillich. Edge isoperimetric inequalities for product graphs. *Discrete Mathematics*, 213(1):291–320, 2000.

[Vui83] J. Vuillemin. A combinatorial limit to the computing power of vlsi circuits. *IEEE Trans. Comput.*, 32(3):294–300, March 1983.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing(preliminary report). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, page 209–213, New York, NY, USA, 1979. Association for Computing Machinery.

A Computational Model

Once a circuit has been synthetized and a netlist has been obtained, the behavior of the circuit can be simulated.

A netlist is composed of cells and wires. It is essentially a hypergraph $G = (V, E)$ that describes the wires. We assume that the cells used in the netlist have distinct identifiers (for instance, consecutive integers) and that each of them has a type (`AND2_X1`, `DFFNRQ_X1`, ...) in accordance with the cell library. The type of a cell determines what input/output ports it exposes. Each port is either an input or an output port. For instance, the `AND` gate on the left of figure 3 has two input ports labelled `A1` and `A2` and an output port labelled `Z`. The `DFFRNQ` flip-flop gate shown in the right of the same figure has three input ports `D`, `RN` and `CLK` as well as one output port `Q`.

Informally, in the hypergraph that describes a netlist, a node $v \in V$ is a port in some gate. The nodes are therefore labelled with a cell identifier and the name of a port (that is exposed by this cell). A wire $e \in E$ is in fact a hyperedge $e \subseteq V$ connecting a subset of the vertices. We expect that each wire is adjacent to at most one output port.

At each point in time, a wire is either in state 0, or in state 1 (we disregard transient states). The state of a wire e in the i-th clock cycle is denoted by e_i. The output ports of cells are in the state dictated by the functional specification of the cell (AND gate, inverter, etc.). Wires connected to an output port are in the state imposed by this output port. We assume that an external mechanism provides the inputs to the circuit. If the circuit is supposed to read n input bits $x_1, \dots, x_n$, then we are given an input specification, namely a list of n pairs $(e_1, t_1), \dots, (e_n, t_n)$ with the following semantics: the external mechanism forces the state of the wire e_i to x_i in the t_i-th clock cycle. If $t_1 = \cdots = t_n$, then the circuit reads its whole input in a single clock cycle. We assume that these "input wires" are not connected to an output port. In the same way, there is an output specification, which is a list of pairs (e_i, t_i) such that the i-th output bit is set to the state of the wire e_i at the t_i-th clock cycle. These wires must be connected to an output port. If a wire is not connected to any output port and if its state is not forced by the input mechanisme, we assume that it is in state 0.

An external mechanism provides a clock signal that triggers the next cycle. The relationship between the states of the input and output ports of combinatorial cells (AND, OR, XOR, ...) is straightforward. For instance, for the AND cells we have $\texttt{Z}_i = \texttt{A1}_i \land \texttt{A2}_i$. For flip-flow cells, the specification is slighlty more complex, because these cells have an internal state s. We assume that initially $s_0 = 0$. The state of the output $\texttt{Q}_i$ is always equal to the internal state s_i. If the `RN` input is 0, then the internal state (and the output port) are forced in state 0; otherwise if the `RN` input is 1, then the next value of the state s_{i+1} is set to the current state of the input port $\texttt{D}_i$. Note that flip-flop cells need to be fed the clock signal.

It follows that in a given clock cycle, the state of each wire is completely determined by the input bits provided in this clock cycle and the current state of each flip-flop cell.

This natural computational model enables us to talk about the function that a given circuit, provided as a netlist, computes.

B One-way Communication Complexity

The concept of (two-way, deterministic) communication complexity can naturally be restricted to the setting of *one-way communications* where only Alice announces bits to Bob and Bob remain silent. At the end of the protocol, both parties emit their local output.

It must be noted that while there is always a two-way protocol for any partition of the input and any partition of the output (Alice and Bob could just exchange their inputs), it is not necessarily the case if communication is restricted from Alice to Bob. For example, consider the case where Alice's output is Bob's input: she has no way of receiving it. In the case where no such protocol exist, we define the one-way communication complexity to be $+\infty$.

Definition 5. Let $h : \{0,1\}^{2n} \to \{0,1\}^m$ be a function and write $h(x_1, \dots, x_{2n}) = (y_1, \dots, y_m)$. Let X and Y be a partition of the input variables $x_1, \dots, x_{2n}$ into two disjoints sets. Also let U and V be a partition of the outputs $y_1, \dots, y_m$ into two disjoints sets. The *one-way communication complexity* of h for the partition (X, Y, U, V) is denoted as $owCC^{X,Y,U,V}(h)$. The *best-case one-way communication complexity* of h is the minimum of $owCC^{X,Y,U,V}(h)$ over all partitions X, Y of $x_1, \dots, x_{2n}$ with $|X| = |Y|$ and all (arbitrary) partitions U, V of $y_1, \dots, y_m$. It is denoted as $owCC_{best}(h)$. The variant where only the input partition is fixed is denoted as $owCC_{best}^{X,Y}(h)$.

Note that the best-case one-way communication complexity is upper-bounded by n.

Indeed, given any balanced partition of the input, there is always a partition of the output that is feasible: Alice emits nothing and transmits her input to Bob, who computes the function and emits the full result.

It is also fairly obvious that the (two-way) communication complexity is always lower than the one-way communication complexity, both for fixed input/output partitions and in the best case.

C Area Lower-Bounds from One-Way Communication Complexity

Consider a circuit computing a functions f taking $2n$ bits as input. If the circuit reads its entire input in a single clock cycle, then as argued in section 2 it must have area $\Omega(n)$. But what if the circuit reads its input one bit at a time? It is sometimes easy to establish that any circuit computing f must have area at least $\Omega(n)$ because it has to memorize its entire input. This is for instance the case when all output bits potentially depend on each input bits. The circuit then cannot start emitting the output before reading its whole input, and therefore it must have area at least $\mathcal{O}(n)$ to store it. This argument is sufficient for the Fast Fourier Transform, integer multiplication modulo a power of two, cyclic shifts, etc.

However, this simple argument does not apply when the circuit reads its input progressively and may emits part of its output "on the fly", before reading the entire input. This is in particular the case of the Möbius transform, where the i-th output bit depends only on the first i input bits.

In this specific case, we provide an area lower-bound based on one-way communication complexity. Intuitively, the one-way communication complexity is a lower-bound on the number of memory bits that the circuit must have to compute the function. This strategy (space lower bounds from communication complexity) is well-known.

Theorem 6. *Consider a circuit that evaluates a function $h : \{0,1\}^{2n} \to \{0,1\}^m$, in a technological process where storing a memory bit requires area μ and receiving an input bit requires area σ.*

i) If A denotes the area of the circuit, then

$$A \geq \min(\sigma, \mu) \cdot owCC_{best}(h).$$

ii) If in addition $n = 2\ell$ (for $k, \ell \in \mathbb{N}$) and the circuit reads either 0 or k input bits in each clock cycle, then

$$A \geq \mu \cdot owCC_{best}(h).$$

Proof. Assume that we are given a circuit, given by a netlist and an input/output specification as specified in Section A. Its input specification is $I = (e_1, t_1), \dots, (e_{2n}, t_{2n})$. Each input bit is provided only once to the circuit. We denote by n_i the number of input bits that are provided in the i-th clock cycle, so that $\sum n_i = 2n$.

Define $S_{-1} = 0$ and $S_j = \sum_{i=0}^{j} n_k$ for all $j \geq 0$. S_j is the total number of input bits received by the circuit between the beginning of its execution and cycle j (included). Denote by k the smallest j such that $S_j > n$ (so that at cycle k, the circuit has received at least half of its input).

Denote by $\texttt{left}(I)$ the set of S_{k-1} input bits received by the circuit during the first $k-1$ clock cycles. In the same vein, denote by $\texttt{middle}(I)$ the set of n_k bits received during the k-th clock cycle. It follows from definition of k that $|\texttt{left}(I)| \leq n$ and

$|\texttt{left}(I)| + |\texttt{middle}(I)| > n$. Let $\texttt{extra}(I)$ denote an arbitrary subset of $\texttt{middle}(I)$ such that $|\texttt{left}(I)| + |\texttt{extra}(I)| = n$.

Next, we consider the following protocol with one-way communication to evaluate f.

- Alice receives the inputs bits in $\texttt{left}(I) \cup \texttt{extra}(I)$.
- Bob receives all the other input bits.
- Alice simulates the execution of the circuit for the first $k - 1$ cycles. If the circuits emits some output bits, then these are part of Alice's output.
- Alice transmits to Bob the internal state of all flip-flop cells, along with the input bits in $\texttt{extra}(I)$.
- Bob simulates the execution of the circuit starting from the k-th clock cycle. All the output bits emitted by the circuit are part of Bob's output.

It is straightforward that this protocol correctly computes the function h. If the circuit has ℓ flip-flop gates, then the volume of data communicated by this protocol is $\ell + n - S_{k-1}$. Both Alice and Bob receives exactly n inputs bits, therefore $\ell + n - S_{k-1} \geq owCC_{best}(h)$.

In the special case given by item *ii*), we have $S_{k-1} = n$. The number of memory bits of the circuit is then lower-bounded by $owCC_{best}(h)$ and the area is lower-bounded by the size of the corresponding number of flip-flop cells. This establishes the second point of the theorem.

Otherwise, it is easy to check that $n - S_{k-1} \leq n_{k-1}$. It follows that

$$\ell \geq owCC_{best}(h) - n_{k-1}. \tag{2}$$

Pick $0 \leq \alpha \leq 1$. Then, one of the following holds:

- Either $n_{k-1} \geq \alpha \cdot owCC_{best}(h)$. In this case, because of the number of inputs accepted by the circuit, it has area greater than $\sigma n_{k-1} \geq \alpha\sigma \cdot owCC_{best}(h)$.
- Otherwise, it follows from eq. (2) that the number of flip-flop gates is lower-bounded by $(1-\alpha) \cdot owCC_{best}(h)$, and the area of the circuit is at least $(1-\alpha)\mu \cdot owCC_{best}(h)$.

It follows that the area of the circuit is at least $\min_\alpha(\alpha\sigma, (1-\alpha)\mu) \cdot owCC_{best}(h)$. The theorem follows from the fact that $\min_{0\leq\alpha\leq1}(\alpha\sigma, (1-\alpha)\mu) = \min(\sigma, \mu)$. □

If we restrict our attention to circuits with a particular input/output specification, then the results can be strenghtened a little. The previous theorem applies to any circuit that computes h, regardless of the order in which it receives its inputs and emits its outputs. In practice, circuits often read their input and emit their results in a specific order that makes sense from the application's point of view. For instance, a 64-bit multiplier may take 2 cycles to read its two 64-bit operands $a_0 \dots a_{63}$ and $b_0 \dots b_{63}$ using 64 input ports. But is it likely that it is going to read first a and then b, or first $a_0 \dots a_{31}$ and $b_0 \dots b_{31}$, while it seems quite unlikely that it is going to read $a_0 a_1, a_2, a_3, b_4 a_5, b_6 a_7, b_8, b_9, b_{10} a_{11}, b_{12}, a_{13}, b_{14}, b_{15}, b_{16} a_{17}, b_{18}, a_{19}, b_{20}, b_{21}, b_{22}, a_{23}, \dots$ on the first cycle and the complement on the second cycle...

If a circuit reads its input sequentially in the "natural" order, then the best-case one-way communication complexity can be replaced by the best-case one-way communication complexity *for a specific input partition.*

Corollary 1. *Consider a circuit that evaluates a function $h : \{0,1\}^{2n} \to \{0,1\}^m$, in a technological process where storing a memory bit requires area μ and receiving an input bit requires area σ.*

Suppose that the circuit conforms to the input specification $I = (e_1, t_1), \ldots, (e_{2n}, t_{2n})$ *with* $e_i \leq e_{i+1}$ *for* $1 \leq i < 2n$. *If in addition* $n = k\ell$ *(for* $k, \ell \in \mathbb{N}$*) and the circuit reads either 0 or* k *input bits in each clock cycle, then its area is lower-bounded by*

$$A \geq \mu \cdot owCC_{best}^{X,Y}(f).$$

where X *contains the first* n *input bits read by the circuit and* Y *contains the* n *last ones.*

The proof is almost exactly the same as that of the previous theorem.