A Survey on The Multiplexer (MUX)

Ngu Dang *

June 2025

Abstract

In this survey, we revisit the complexity of the multiplexer function MUX, that is, the $2N - 2$ lower bound (Paul, 1975) and the $2N + O(\sqrt{N})$ upper bound (Klein & Patterson, 1980) providing two main contributions.

- First, we refine known upper bounds by giving exact summation formulas for our recursive constructions, not only eliminating the $O(\cdot)$ terms and making the counts fully explicit, but also providing a clean and easy to analyze construction that helps provide better understanding on how MUX-function works. At the same time, we also refine and modernize the proof of its known lower bound under the DeMorgan basis (which was done in a slightly different basis in Paul's work).
- Second, we examine the limits of the classical Gate Elimination method in this setting and identify why even a small improvement over Paul's lower bound would require insights beyond existing techniques.

Contents

*Boston University, ndang@bu.edu

1 Introduction

1.1 Overview

Understanding the circuit complexity of natural Boolean functions is a fundamental pursuit in theoretical computer science for over five decades. Among existing techniques, Gate Elimination — a robust method for proving circuit lower bounds by analyzing the effects of input substitution — has emerged as one of the most tractable and intuitive methods for proving circuit lower bounds. In particular, current known lower bounds in the DeMorgan, $\mathcal{U}_2$ [Red73; Sch74; Zwi91; IM02; ILMR02], and $\mathcal{B}_2$ [Sch74; Sto77; DK11; FGHK16; LY22] basis were proven via Gate Elimination. However, despite its appeal and the long-standing interest, progress in proving strong lower bounds remains limited. To date, no super-linear lower bounds are known for general Boolean circuit classes over different bases, and breaching this barrier via Gate Elimination alone remains elusive [Weg87].

One insight from lower bound proofs is that they often reveal partial structure about optimal circuits computing the function itself. Thus, understanding the structure of optimal circuits is also a fundamental and parallel goal. Unfortunately, even for seemingly simple functions, the space of structurally distinct but functionally equivalent circuits is vast and not trivial to analyze [Sat81; BS84; Weg87].

In this survey, we aim to make this challenge more explicit by focusing on a canonical Boolean function, the Multiplexer (MUX). Given its structure — selecting a bit from a list of data bits based on an address encoded in binary — the MUX-function captures an essential form of computation: indirect addressing or memory lookup. Despite its simple nature, proving tight lower and upper bounds for its circuit complexity has remained a challenging open problem for decades. This makes MUX a prime candidate for advancing our understanding of Boolean circuit complexity as well as a testbed for the Gate Elimination technique. To get a better understanding of how MUX works, we also study another elementary Boolean function that we call the Positional Decoder (DEC) which, on the binary encoded address, outputs a one-hot vector indicating the position of the selected data bit in the list of all bits. These functions are not only fundamental in logic design but also present rich structures worth analyzing. We hope our clarifies the limitations of current techniques and motivates the search for stronger methods as well as further studies on optimal circuit structures.

1.2 Related Work

The Complexity for Multiplexer Function in Switching Circuits [LK21] Lozhkin and Khzmalyan studied the complexity of the MUX-function in the setting of switching circuits, a model that measures complexity by the number of binary switches rather than logic gates. They used the notion of *noneliminable* variable sets to argue baseline lower bounds under substitutions. In particular, they derive combinatorial constraints on cycles of data-input switches and bound how many data variables can appear with exactly two switches, then propagate these bounds through a sequence of circuit simplifications under substitutions to reach the global lower bound and achieved a tighter bound for MUX under the switching circuits setting compared to known results in Boolean circuits setting.

On the Depth of a Multiplexer Function with a Small Number of Select Lines [Loz24]. Lozhkin characterizes the depth complexity of the standard multiplexer function in the DeMorgan basis. For sufficiently large n, it is shown that the minimum depth of any circuit computing MUX_n is exactly $n+2$ using techniques such as substitution on *unstrikable* variable sets. While depth is not the main focus in our work, these results underscore the inherent rigidity of MUX- circuits supporting our motivation to study recursive decompositions in the size model, where structure can be made more explicit.

Linear-Size Boolean Circuits for Multiselection [HR24]. Holmgren and Rothblum study the multi-output variant of MUX, where instead of selecting a single data value, the circuit selects k of them based on a set of selection indices. They show that multiselection can be implemented by Boolean circuits of slight bigger than linear size in n, the total bit-length of the data and selectors, even when k is large. This is accomplished via combinatorial routing networks and conditional move gadgets, bypassing the need for multiple sequential MUX computations. Although their work targets a more general functionality, it provides further evidence that modular designs rooted in MUX-like components can be size-efficient even in more expressive models.

1.3 Our Contribution

This work revisits two fundamental functions in circuit complexity: the Multiplexer (MUX) and the Decoder (DEC), focusing on studying their upper and lower bounds within the DeMorgan basis ($\{\wedge, \vee, \neg\}$) where $\neg$-gates are for free. We re-express known lower bound from the works of Paul [Pau75] which was done under a different basis $\{\wedge, \oplus, \neg\}$ (Theorem 13). We also recount the known upper-bound by Klein and Paterson [KP80] while making certain implicit steps in their work fully explicit and accessible (Lemma 11).

Furthermore, to simplify analysis and avoid irregularities arising in generic even or odd input lengths, we define and analyze restricted variants of these functions—Exponent Decoder (eDEC) and Exponent Multiplexer (eMUX) where the address length is always a power of two. This constraint enables recursive constructions that are clean, easily analyzable, and asymptotically optimal while remaining representative of the general case. In particular, we provided a fully formal and expanded version of the upper bound construction from [KP80] by first analyzing the construction for eDEC (Theorem 5) and then lift that to eMUX (Theorem 14).

Lastly, we highlight subtle difficulties in identifying optimal circuit structures even for small-input instances in the family of eDEC-functions (Observation 9). These challenges demonstrate the inherent limitations of classical techniques like Gate Elimination and suggest the need for stronger variants or even alternative methods. By isolating and articulating these obstacles, this work contributes to a better understanding of the structural complexity underlying even basic Boolean functions.

1.4 Discussion & Next Steps

While this survey is a bit short of a new insights towards a tighter lower bound, it clarifies the structural barriers and outline concrete pathways for progress. In particular, a tight bound for eMUX_ℓ remains elusive, let alone MUX_n. Even a small, provable gap above $2N-2$ would be significant, as it would demonstrate a new technique capable of handling substructures that classical elimination leaves untouched. Whether through combinatorial innovation, fusion-based covers, or solver-driven structure search, cracking this problem would advance our toolkit for Boolean circuit lower bounds. Building on this foundation, two promising directions emerge:

Applying other lower-bound frameworks. As surveyed by Wigderson [Wig93], the Fusion Method turns the computation into a static cover problem over a set of "fusing functionals." By choosing functionals tailored to the address/data separation in eMUX_ℓ (or more generally, MUX_n), it may be possible to prove that small covers are impossible and thereby derive new bounds. Recently, Calavar and Oliveira recast Fusion Method using a modern set-theoretic language that works for various settings including non-monotone Boolean circuits [CO25]. They mention that reproving known circuit lower bounds via Gate Elimination, or better yet improving these lower bounds, for non-monotone functions using their framework remains an open problem. We discuss this direction in more details in Section 4.2.3.

Formalizing conjectures for solver-aided verification. Tools like Programming **Z3** by Bjørner, de Moura, Nachmanson, and Wintersteiger [BMNW18] can symbolically explore all candidate circuits up to a certain size, ruling out counterexamples and potentially suggesting new elimination patterns. Given the combinatorial explosion of structurally distinct circuits, solver-aided search could be crucial for finding patterns or tightening conjectures.

2 Preliminaries

2.1 Minterms, Disjunctive Normal Form (DNF) & Boolean Circuits

Throughout this paper, we use the notion of a *minterm* in the variables in $X = \{x_1, x_2, \ldots, x_n\}$ which is the AND of each variable or its negation. For instance, when $n = 4$, $x_1 \wedge x_2 \wedge \overline{x_3} \wedge \overline{x_4}$ is a minterm, and it has value 1 exactly when $x_1x_2x_3x_4 = 1100$. We observe the following useful fact that each input string $x \in \{0,1\}^n$ corresponds to exactly one minterm in n variables or its negation. We say the *disjunctive normal form* (DNF) of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is the OR of the minterms of f.

We study general Boolean circuits over the *DeMorgan basis* $\mathcal{B} = \{\wedge, \vee, \neg, 0, 1\}$ of Boolean functions: binary $\wedge$ and $\vee$, unary $\neg$ and zero-ary (constants) 1 and 0. Circuits take zero-ary variables in $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots y_m\}$ for some fixed n and m as inputs. The standard formulation of circuits consists of single-sink DAGs with nodes labeled by function symbols or variables and edges as "wires" between the gates. We write V_C and E_C to denote the set of nodes and the set of edges of a circuit C respectively, omitting the subscript when it is clear which circuit is being referenced. Boolean circuits compute Boolean functions through *substitution* followed by *evaluation*.

An *assignment* of input variables is a mapping from the set of inputs to $\{0, 1\}$. To substitute into a circuit according to an assignment, each input x_i is replaced by the constant. To evaluate a circuit, the values of interior nodes labeled by function symbols are computed in increasing topological order. For each interior node labeled by a function, it's value is obtained by applying it's function to the value of the node's incoming wires. The output of the circuit overall is the value of its sink.

Let $\mathcal{F}_n$ be the family of Boolean functions on n variables. We say a circuit G on n variables computes $g \in \mathcal{F}_n$ if for all $\alpha \in \{0,1\}^n$, $G(\alpha) = g(\alpha)$. The size of a circuit G, denoted $|G|$, is the number of $\wedge$ and $\vee$ gates in the circuit, and $CC(g)$, the circuit complexity of a Boolean function g, is the minimum size of any circuit computing g. Since only $\wedge$ and $\vee$ gates contribute to circuit size, we refer to these gates as *costly*. We say a circuit G computing g is optimal if $|G| = CC(g)$.

2.2 Positional Decoder & Multiplexer

Definition 1 (Multiplexer)**.** We define the multiplexer $\mathsf{MUX}_n : \{0,1\}^n \times \{0,1\}^{2^n} \to \{0,1\}$ as follows. For all (a, x) where $a \in \{0,1\}^n, x \in \{0,1\}^{2^n}$, we have

$$\mathsf{MUX}_n(a, x) = x_{(a)}$$

where $x_{(a)}$ is the (a)-th bit of x (0-indexed) when a is interpreted as a base-2 number.

For example, if $n = 2$, $a = 01$, and $x = x_0x_1x_2x_3 = 0101$, then $(a) = 1$, thus $\mathsf{MUX}_n(a, x) = x_1 = 1$.

Throughout this paper, we will address $a_1, \ldots a_n$ as the *address bits*, and $x_1, \ldots, x_{2^n}$ as the *data bits*. To better understand the construction for MUX-circuits later, we define a binary-to-positional decoder as follows

Definition 2 (Positional Decoder)**.** A function $\mathsf{DEC}_n : \{0,1\}^n \to \{0,1\}^{2^n}$ is called a binary-to-positional decoder if the following holds:

$$\mathsf{DEC}_n(a) = (sel_0(a), \ldots, sel_{2^n-1}(a))$$

where $sel_i(a) = 1$ iff $i = (\mathbf{a})$

Thus we can write the definition of MUX as a DNF using the output of DEC_n as follows

$$\mathsf{MUX}_n(a, x) = \bigvee_{i=0}^{2^n-1} sel_i(a) \wedge x_i$$

Using the example above, we have $\mathsf{DEC}_2(a) = (0, 1, 0, 0)$, and thus

$$\mathsf{MUX}_2(a, x) = (0 \wedge x_0) \vee (1 \wedge x_1) \vee (0 \wedge x_2) \vee (0 \wedge x_3) = x_1 = 1$$

For our results, we consider a restricted family of DEC and MUX where the length of input strings is strictly a power of 2. Namely,

Definition 3 (Exponent Positional Decoder and Multiplexer)**.** Let $\ell \in \mathbb{N}$, we define the classes of functions of Exponent Positional Decoders $\mathsf{eDEC}_\ell : \{0,1\}^{2^\ell} \to \{0,1\}^{2^{2^\ell}}$ and Exponent Multiplexers $\mathsf{eMUX}_\ell : \{0,1\}^{2^\ell} \times \{0,1\}^{2^{2^\ell}} \to \{0,1\}$ as follows

$$\mathsf{eDEC}_\ell(a) = (sel_0(a), \ldots, sel_{2^{2^\ell}-1}(a))$$

where $sel_i(a) = 1$ iff $i = (\mathbf{a})$

$$\mathsf{eMUX}_\ell(a, x) = \bigvee_{i=0}^{2^{2^\ell}-1} sel_i(a) \wedge x_i$$

Here, notice the subscript of ℓ used in the definition of Binary Positional Decoder and Multiplexer which no longer indicates the number of input variables. Instead, it serves as an "index" for the function in this particular family. So for example, eDEC_0 is the first and "smallest" one in the eDEC-function family and it corresponds to DEC_1 in the standard decoder family. Similarly, eDEC_1 is the second one in the family and corresponds to DEC_2, eDEC_2 to DEC_4, and so on. The same logic applies for the eMUX-function family.

Finally, note that from the definitions, eDEC and eMUX (as well as DEC and MUX) must depend on all of its input variables.

3 On (Exponent) Positional Decoder

In this section, we study the upper bound and lower bound for the Binary Positional Decoder (eDEC_ℓ). We begin with recounting the previous work on the generic Decoder DEC_n for $n \in \mathbb{N}$ in [KP80; SC98]. Then, we present our upper bound and conjecture for the lower bound for eDEC_ℓ where, as a reminder, input lengths are only powers of 2.

3.1 Previous Work on DEC

To begin with, we will look at a naive construction for a circuit computing DEC_n. Observe that for n address bits, there are $2^n = N$ possible minterms, and each minterm corresponds to exactly one output of DEC_n. Thus, our circuit construction is to simply compute all possible minterms composed by the address bits $a_1, \ldots, a_n$. To construct a minterm, we need $n - 1 = \log N - 1$ $\wedge$-gates. Therefore, this construction yields an upper bound $CC(\mathsf{DEC}_n) \leq (\log N - 1)N$ which is $O(N \log N)$ costly gates.

However, observe that we can achieve a better upper-bound by constructing the circuit for DEC_n recursively. Namely, we notice that by definition, minterms have a recursive property, i.e. a minterm on n variables is an AND of 2 smaller minterms, one is on half of the variables, and the other one is on the other half of the variables. That said, a better way to construct a DEC_n-circuit is to use two subcircuits, $\mathsf{DEC}_{\lfloor n/2 \rfloor}$-circuit and $\mathsf{DEC}_{\lceil n/2 \rceil}$-circuit, and combine every minterm generated by the first $\mathsf{DEC}_{\lfloor n/2 \rfloor}$-circuit with every minterm generated by the second $\mathsf{DEC}_{\lceil n/2 \rceil}$-circuit with $\wedge$-gates. We present Figure 1 regarding this construction in our upper bound proof in Section 3.2.1 below. Thus, this construction yields a circuit complexity that is at most twice the complexity of $\mathsf{DEC}_{n/2}$-circuit plus N $\wedge$-gates that are used to combine the minterms of the two DEC-subcircuits.

Lemma 4 ([KP80; SC98])**.** *$CC(\mathsf{DEC}_n) \leq N + CC(\mathsf{DEC}_{\lfloor n/2 \rfloor}) + CC(\mathsf{DEC}_{\lceil n/2 \rceil}) = N + O(\sqrt{N})$, where $N = 2^n$*

Notice that each output entry of DEC_n corresponds to exactly one minterm the variables $\{a_1, a_2, \ldots, a_n\}$. Each minterm computes a distinct non-constant function which requires at least one costly gate to compute, and there are 2^n possible minterms. Thus, an obvious lower bound for the circuit complexity of DEC_n is $CC(\mathsf{DEC}_n) \geq N$ [SC98]. However, this lower bound only gives us information regarding the output layer on the top of the circuit as n gets large which leaves rooms for improvement if one can witness a restriction that is guaranteed to eliminate gates in the intermediate levels of the circuit.

3.2 On the upper bound and lower bound of eDEC

In this section, we present an upper bound and a lower bound for eDEC that are nearly tight to one another. We note that our construction for the upper bound is based on that of [KP80; SC98], but we provide an exact count of gates for the case of eDEC. Then, we show the optimal structure of circuits computing eDEC_ℓ, for any $\ell \in \mathbb{N}$.

3.2.1 Upper Bound

Theorem 5. *Let $\ell \in \mathbb{N}$ and $N = 2^{2^\ell}$, then $CC(\mathsf{eDEC}_\ell) \leq N + \mathcal{S}(N)$ where $\mathcal{S}(N) = \sum_{i=1}^{\log\log N - 1} 2^i \cdot N^{1/2^i}$*

Proof. Let $r = s = \frac{2^\ell}{2} = 2^{\ell-1}$, then for some $a = b \circ c$, where $b \in \{0,1\}^r$, $c \in \{0,1\}^s$, and $i = 0, \ldots, 2^r - 1$, $j = 0, \ldots 2^s - 1$, we have

$$sel_{i \cdot 2^r + j}(a) = sel_{i \cdot 2^r + j}(b \circ c) = sel_i(b) \wedge sel_j(c)$$

which implies the following equation computing eDEC_ℓ defined in Definition 3

$$\mathsf{eDEC}_\ell(a) = \mathsf{eDEC}_\ell(b \circ c) = (sel_i(b) \wedge sel_0(c), \ldots, sel_i(b) \wedge sel_{2^j - 1}(c) \quad \text{for } i = 0, \ldots, 2^r - 1)$$

In words, for every $i = 0, \ldots, 2^s - 1$, we compute $sel_i(b) \wedge sel_j(c)$ for every $j = 0, \ldots, 2^r - 1$, so $sel_0(a)$ corresponds to $sel_i(b) \wedge sel_j(c)$ where $(i, j) = (0, 0)$, and $sel_1(a)$ corresponds to the term where $(i, j) = (0, 1)$, and so on.

We observe the following recursive construction for D_ℓ based on the equation above. In particular, let D_ℓ be the circuit computing eDEC_ℓ, then the construction of D_ℓ uses two $D_{\ell-1}$ sub-circuits and the output of D_ℓ are given by connecting the outputs of the two $D_{\ell-1}$ in the following manner: for each output of one $D_{\ell-1}$, we connect it to each output of the other $D_{\ell-1}$ using an $\wedge$-gate which requires a total of N $\wedge$-gates. Thus, we obtain

$$CC(\mathsf{eDEC}_\ell) \leq 2 \cdot CC(\mathsf{eDEC}_{\ell-1}) + N$$

We can repeat this process to construct $D_{\ell-1}$ using two sub-circuits $D_{\ell-2}$, and so on which suggests a recurrence for the construction of eDEC_ℓ-circuits. In particular, observe that the base case of the recursive construction is when $\ell = 0$ which is a simple decoder that has $2^{2^0} = 2$ outputs. This decoder requires zero costly gate since we have only two possible minterms, i.e. either the variable itself or its negation, so $CC(\mathsf{eDEC}_0) = 0$. Thus, we can express the upper-bound of $CC(\mathsf{eDEC}_\ell)$ via the following recurrence with $N = 2^{2^\ell} \implies \sqrt{N} = 2^{2^\ell/2} = 2^{2^{\ell-1}}$ (meaning decreasing ℓ by 1 accounts for taking the square root on the number of outputs)

$$T(N) = \begin{cases} 0, & \text{if } N = 2 \\ 2 \cdot T(N^{1/2}) + N, & \text{if } N > 2 \end{cases}$$

Now, we compute the number of steps to reach the base case. Let this number be k, then we have $1 = \frac{\ell}{2^k} \implies 2^k = \log N \implies k = \log\log N$. Next, we unroll the recurrence for k steps.

$$\begin{aligned} T(N) &= 2 \cdot T(N^{1/2}) + N = 2(2 \cdot T(N^{1/4}) + N^{1/2}) + N \\ &= 4 \cdot T(N^{1/4}) + 2 \cdot N^{1/2} + N \\ &= 8 \cdot T(N^{1/8}) + 4 \cdot N^{1/4} + 2 \cdot N^{1/2} + N \\ &\cdots \\ &= 2^k \cdot T(2) + \sum_{i=0}^{k-1} 2^i \cdot N^{1/2^i} \\ &= \log N \cdot 0 + N + \sum_{i=1}^{\log\log N - 1} 2^i \cdot N^{1/2^i} \\ &= N + \sum_{i=1}^{\log\log N - 1} 2^i \cdot N^{1/2^i} \end{aligned}$$

Setting $\mathcal{S}(N) = \sum_{i=1}^{\log\log N - 1} 2^i \cdot N^{1/2^i}$, we have $CC(\mathsf{eDEC}_\ell) \leq N + \mathcal{S}(N)$ as desired. □

Remark 6. *Our upper bound for eDEC_ℓ is consistent with the lower bound of DEC_n in Lemma 4. In particular, the first term in $\mathcal{S}(N)$, $2 \cdot N^{1/2}$, dominates the others, and thus, we believe one can show that $\mathcal{S}(N) = O(\sqrt{N})$. Furthermore, a useful thing to try is to derive the exact count for the general case of DEC_n where n is even or odd, each may differ by a few gates, but should be in $O(\sqrt{N})$ still.*

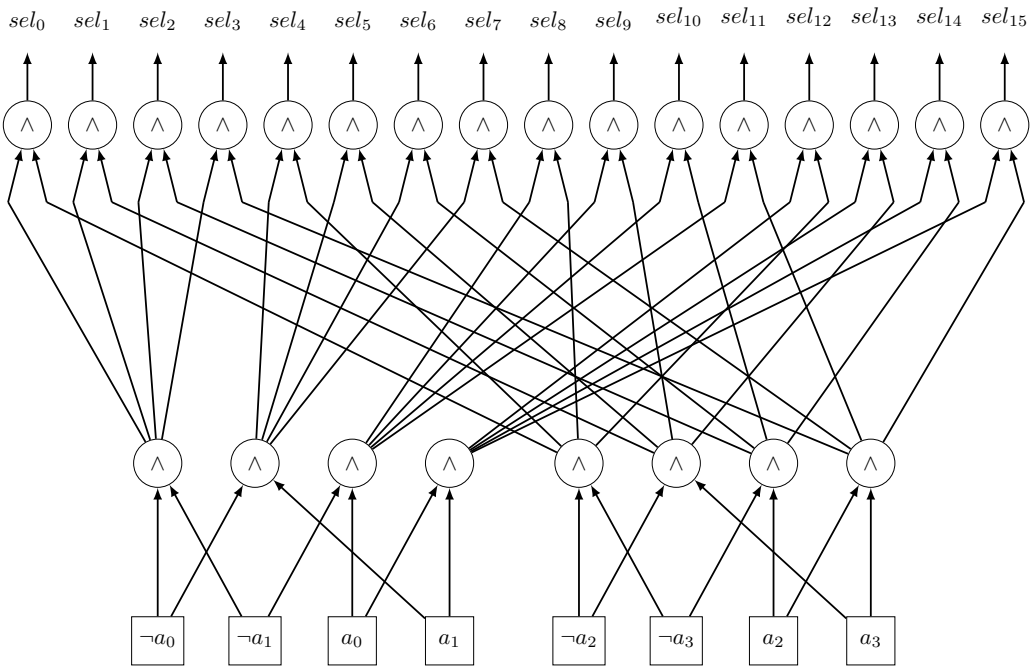


Figure 1: The figure shows the construction of DEC_4 that uses two DEC_2 subcircuits. The total complexity of this construction is $16 + 2 \cdot 16^{1/2^1} = 16 + 8 = 24$ costly gates. For readability, we simplify the negations of the input variables by treating each negation as its own input node.

3.2.2 Conjecture on the Lower Bound and Optimal Circuit Structure

Now, we study the lower bound of eDEC. Notice that just like DEC_n, we have $CC(\mathsf{eDEC}_\ell) \geq N$ where $N = 2^{2^\ell}$. However, this lower bound eludes the potential costly gates internally which leaves room for improvement. We conjecture that the upper bound is tight, and better yet, the circuit construction in Theorem 5 is optimal. In particular, we believe the following statement holds

Conjecture 7. *Any optimal circuit computing* eDEC_2 *requires* 2 *subcircuits computing* eDEC_1.

To achieve this, we believe it is crucial to characterize the optimal circuits computing eDEC_1 (i.e. $\ell = 1$) and generalize the pattern for any $\ell \geq 2$ using Gate Elimination. In particular, the recursive decomposition from the upper bound construction indicates that each minterm for eDEC_2 can be expressed as an AND between minterms from two independent eDEC_1 instances (one for the first half and one for the second half of address bits). Then, suppose there exists an optimal eDEC_2-circuit without two distinct eDEC_2 subcircuits clearly computing separate halves. Then, one might be able to argue that minterm computation would have redundant computations that might not optimally share intermediate results. This might involve *explicitly enumerating* all possible cases and gate counts for *any deviation* from the conjectured pattern of using two explicit eDEC_1-subcircuits, and then show via a detailed gate-count analysis (i.e. counting how many gates are unnecessarily duplicated) that such a structure cannot be optimal. Finally, one can then generalize via an inductive argument to obtain.

Conjecture 8. *For any* $\ell > 2$, *optimal circuits computing* eDEC_ℓ *requires* 2 *subcircuits computing* $\mathsf{eDEC}_{\ell-1}$.

However, we notice the case $\ell = 1$, characterizing the optimal circuits computing eDEC_1 in our setting can yield multiple "shapes". In paricular,

Observation 9. *Let* $\mathsf{eDEC}_1 : \{0,1\}^{2^1} \to \{0,1\}^{2^{2^1}}$ *be the binary positional decoder for* 2 *address bits* a_0, a_1. *Then, any optimal circuit* C *computing* eDEC_2 *satisfies:*

1. $CC(\mathsf{eDEC}_1) = |C| = 4$,

2. *Each costly gate corresponds to uniquely one minterm in* $\{a_0a_1, a_0\overline{a_1}, \overline{a_0}a_1, \overline{a_0a_1}\}$

3. *There is one address bit that is read at least* 3 *times, and all address bits must be read at least twice.*

Proof. Let C be an optimal circuit computing eDEC_2. To prove statement (1), we show $|C| \leq 4$ and $|C| \geq 4$. Apply Lemma 5 for one iteration of the recurrence with $\ell = 2$, we obtain $|C| \leq 4$. We now show $|C| \geq 4$. To this end, note that the functions computing the minterms are disjoint, and each requires at least one costly gate to compute. In other words, a single costly gate cannot be the output gate of more than one minterm, ensuring no sharing of costly gates among the outputs. This also implies statement (2). For statement (3), we first argue that all address bits must be read at least twice. Assume otherwise, then there exists an assignment that disconnects a_i from the circuit via a fixing rule on its costly successor which contradicts the definition of eDEC which depends on all input variables.

Now, to prove that there exists one address bit that must be read at least 3 times, we first show the following claim regarding the input of each costly gate

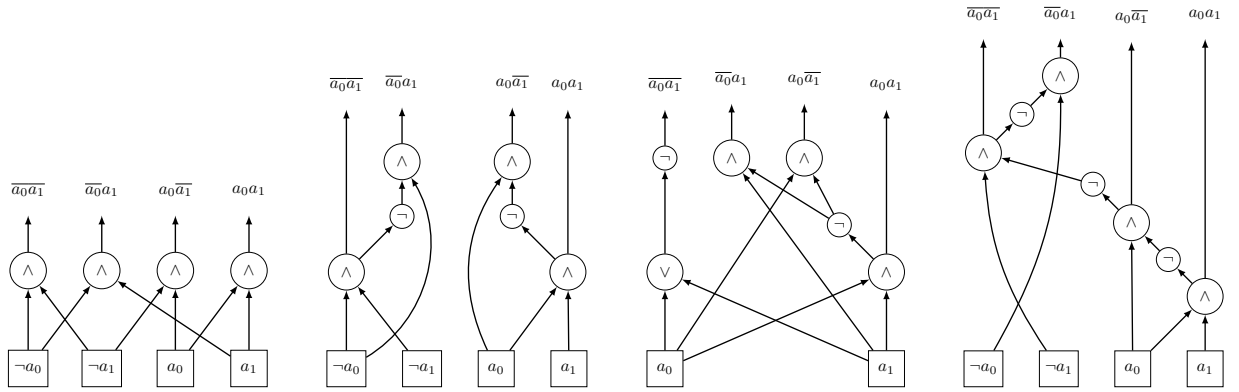
Claim 10. *Each costly gate in* C *must read at least one input variable or its negation.*

Proof. Assume towards contradiction that a costly gate g_i reads the outputs of two other costly gate g_j and g_k. Note that $g_j \neq g_k$ as otherwise, we have $g_i \equiv g_j$ or g_i becomes a constant, and in whichever case, we can simplify g_i contradicting the optimality of C.

Note that there exists an assignment α, β such that $g_j(\alpha) = g_k(\alpha) = 0$ and $g_j(\beta) = g_k(\beta) = 0$ which results in $g_i(\alpha) = g_i(\beta) = 0$ regardless of g_i's gate-type. Indeed, since each of the 4 costly gates must correspond to a distinct minterm, a witness of such assignments α, β are two that make g_i and the last costly gate evaluates to 1 respectively. However, $g_i(\alpha) = 1$ in this case and thus $g_i(\alpha) \neq g_i(\beta)$, a contradiction. □

At this point, we know that each of the 4 costly gates must read at least a literal or its negation as one of its inputs. Now, since C is finite, there exists at least one (bottom) costly gate that reads both input variables as its inputs. Since each costly gate has fan-in 2, then there are at least $2 + 1 + 1 + 1 = 5$ literals that is read, and thus by the pigeonhole principle, one of the input variables must be read at least 3 times as desired. □

Some optimal shapes of C that satisfy the 3 conditions above are shown in Figure 2 below.



(a) Both inputs are read 4 times, each is read negatively 2 times and positively 2 times

(b) a_0 are read 4 times and a_1 are read only twice

(c) Both inputs are read 3 times positively

(d) a_0 are read 3 times and a_1 are read only twice

Figure 2: Different shapes of optimal eDEC_1 circuits. For readability, we simplify the negation on the input variables and treat each negation like a separate input.

Having multiple "shapes" for eDEC_1 yields certain difficulties for proving our conjectures via Gate Elimination alone.

- The observation above shows that optimal circuits computing eDEC_1 does not necessarily have the "nice" output layer composed of $\wedge$-gates. Thus, it is unclear to us whether this only happens for eDEC_1 or not which makes it difficult to enumerate all possible cases for Gate Elimination argument.
- We had hoped that eDEC_1 circuits compute each minterm independently (i.e. no minterm can be used to compute other minterms in the output) which can be a useful "pattern" for generalizing how the structure of the circuits for cases of $\ell \geq 2$ should look like, but it is unfortunately not the case as in Figure 2b, 2c, and 2d.
- It seems reasonable to us that eDEC_ℓ for $\ell \geq 2$ also contain the same issue, and thus, it is unclear whether the optimal way to compute eDEC_ℓ in this case is to have to compute each "half-minterm" and AND-ing them together just by using Gate Elimination.

4 On (Exponent) Multiplexer

4.1 Previous work on MUX_n

In this section, we aim to recount the elementary lower bound by Paul [Pau75] and upper bound [KP80] by Klein and Paterson for MUX_n which motivates our line of work. We present a detailed and refined presentation for both proofs of the upper bound and lower bound respectively.

4.1.1 Upper Bound

We begin with establishing an upper bound for MUX_n, and we start with the naive circuit construction. In particular, the construction follows from the alternative definition of MUX_n via Definition 2 of DEC_n. As a reminder,

$$\mathsf{MUX}_n(a, x) = \bigvee_{i=0}^{2^n-1} sel_i(a) \wedge x_i$$

In particular, it uses a total of $N - 1$ $\vee$-gates and N $\wedge$-gates, where $N = 2^n$ (see Figure 3). Thus, $CC(\mathsf{MUX}_n) \leq 2N - 1 + CC(\mathsf{DEC}_n) \leq 3N + O(\sqrt{N})$.

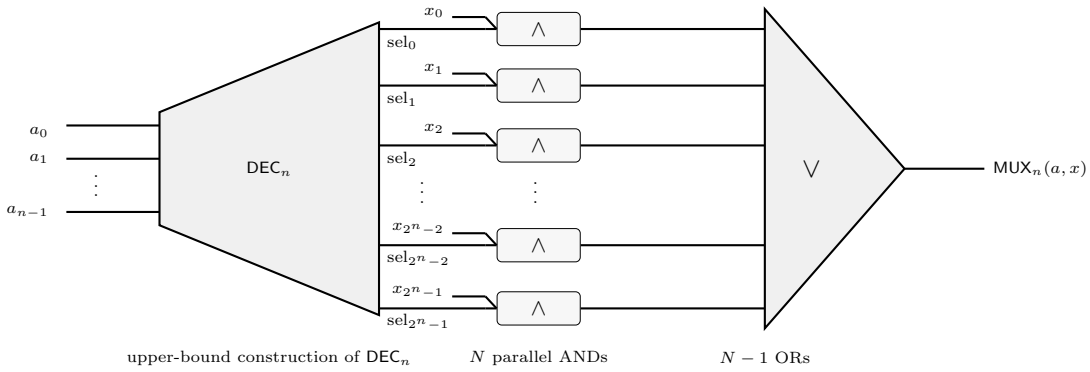


Figure 3: Naïve MUX_n-circuit construction (on input address bits $a_0, \ldots, a_{n-1}$ and data bits $x_0, \ldots, x_{2^n-1}$). The decoder DEC_n (on inputs $a_0, \ldots, a_{n-1}$) emits a one-hot vector $(sel_0, sel_1, \ldots, sel_{2^n-1})$. Each x_i and sel_i are fed into an AND-gate, and the results are combined by an OR-circuit to produce the final output.

The "overhead" of this construction is upper bounded by $2N - 1$, but it turns out that we can also apply a recursive construction for MUX_n to obtain a better upper bound for the overhead. A better construction proposed by Klein and Paterson [KP80] takes advantage of the downward self-reducibility of MUX. In particular, the "overhead" of N costly gates can be reduced by using two smaller decoders *in sequence*, selecting part of the address with respect to different blocks of arguments in turn (see Figure [REF]). Namely, we have the following upper bound

Lemma 11 ([KP80]). *$CC(\mathsf{MUX}_n) \leq 2N + O(\sqrt{N})$, where $N = 2^n$*

Proof. Let $r = \lfloor \frac{n}{2} \rfloor$, $s = \lceil \frac{n}{2} \rceil$, and let D_r, D_s denote the circuits computing DEC_r, DEC_s respectively. We have the following sequence of equations that uses D_r and D_s in sequence.

$$\begin{aligned}
\mathsf{MUX}_n(a, x) = \mathsf{MUX}_n(b \circ c, x) &= \bigvee_{i=0}^{2^r-1} \bigvee_{j=0}^{2^s-1} (sel_i(b) \wedge sel_j(c) \wedge x_{i \cdot 2^r + j}) \\
&= \bigvee_{i=0}^{2^r-1} \left(sel_i(b) \wedge \left(\bigvee_{j=0}^{2^s-1} (sel_j(c) \wedge x_{i \cdot 2^r + j}) \right) \right) \\
&= \mathsf{MUX}_r \left(b, \left(\bigvee_{j=0}^{2^s-1} (sel_j(c) \wedge x_{i \cdot 2^r + j}) \text{ , for } i = 0, \cdots, 2^r - 1 \right) \right)
\end{aligned}$$

The circuit construction in this case is as follows (which is suggested by the middle equation): we first use D_s to select the appropriate j given by its binary presentation c. Then, for this "fixed" j, the term $sel_j(c) \wedge x_{i \cdot 2^r + j}$ can "filter out" the input indices that do not correspond to j. In other words, at this stage, we are asking the question: for all possible prefixes $b \in \{0,1\}^r$, what is the only correct suffix c? Thus, the output of this stage is the remaining $2^{\lfloor n/2 \rfloor}$ data variables x's whose suffix of the its index in binary is the string c. That said, we can ignore the inputs that do not meet this condition and proceed with selecting the other half b. Finally, we use D_r to select the other half b and the expression $b + 2^r \cdot c$ yields the correct index. Thus, we obtain a recursive computation of MUX_n in terms of MUX_r as suggested by the last equation.

We will now analyze the circuit complexity of this construction. In particular,

- the first stage of selecting the suffix c for the data bit's index in binary requires $2^r \cdot 2^s = 2^n$ $\wedge$-gates and $2^r(2^s - 1) = 2^n - 2^r$ $\vee$-gates. Specifically, there are 2^r possible prefixes $b \in \{0,1\}^r$, and for each b, we use 2^s $\wedge$-gates and $2^s - 1$ $\vee$-gates to select the correct c. Taking the complexity of D_s into account (i.e. $CC(\mathsf{DEC}_s)$), the total number of gates used in this stage is

$$CC(\mathsf{DEC}_s) + 2 \cdot 2^n - 2^r$$

- the second stage of selecting the prefix b for the data bit's index in binary, given a "fixed" suffix c requires 2^r $\wedge$-gates and $2^r - 1$ $\vee$-gates. Taking the complexity of D_r into account (i.e. $CC(\mathsf{DEC}_r)$), the total number of gates used in this stage is

$$CC(\mathsf{DEC}_r) + 2 \cdot 2^r - 1$$

  Note that the construction of this stage is exactly like the naive approach but for MUX_r.

Therefore, by Lemma 4 and substituting $N = 2^n$ and $r = \lfloor n/2 \rfloor$, $s = \lceil n/2 \rceil$, the total complexity of this construction is upper bounded by

$$CC(\mathsf{MUX}_n) \leq CC(\mathsf{DEC}_r) + CC(\mathsf{DEC}_s) + 2 \cdot 2^n + 2^r - 1 = 2N + O(\sqrt{N})$$

□

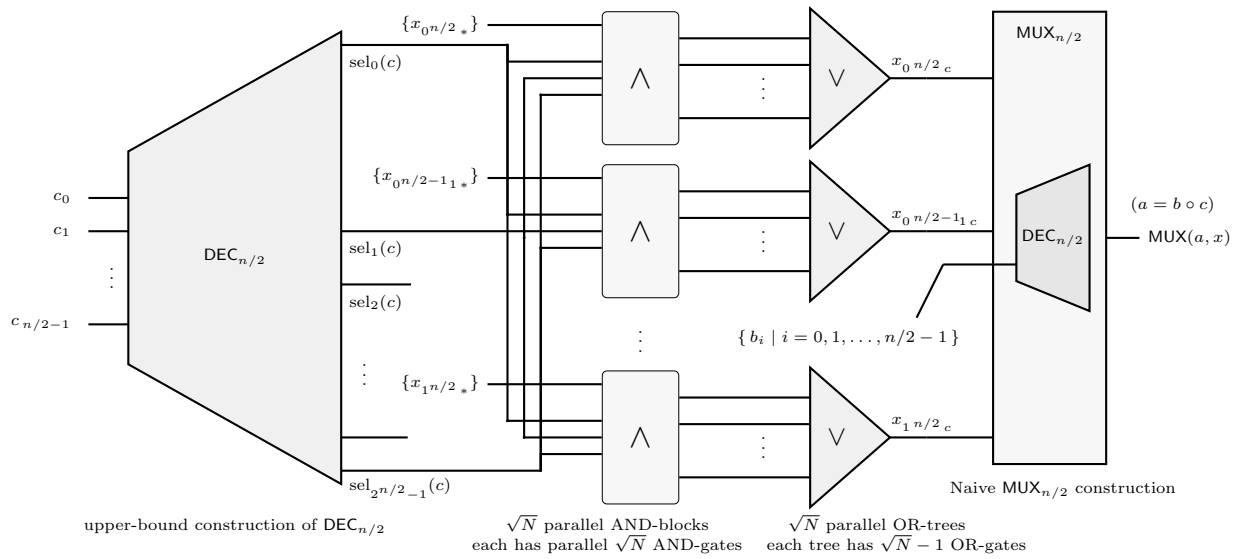


Figure 4: One-level expansion of MUX_n. The construction of $\mathsf{MUX}_{n/2}$-circuit on the right side follows the naive construction as in Figure 3.

The same analysis can be used to show an upper bound for eMUX_ℓ with an exact count rather than bigO. In particular, apply the upper bound in Theorem 5 for $\mathsf{eDEC}_{\ell-1}$, we obtain

$$\begin{aligned} CC(\mathsf{eMUX}_\ell) &\le 2N + \sqrt{N} - 1 + 2 \cdot CC(\mathsf{eDEC}_{\ell-1}) \\ &= 2N + \sqrt{N} - 1 + 2(\sqrt{N} + \mathcal{S}(\sqrt{N})) \\ &= 2N + 3\sqrt{N} + 2 \cdot \mathcal{S}(\sqrt{N}) - 1, \quad \text{where } \mathcal{S}(\sqrt{N}) = \sum_{i=1}^{\log\log\sqrt{N}-1} 2^i \cdot (\sqrt{N})^{1/2^i} \end{aligned}$$

which is consistent with [KP80]. However, Klein and Patterson pointed out that the analysis of Lemma 11 suggests a recursive construction which can give further improvement to the $O(\sqrt{N})$ term, and in particular, decrease the coefficient of the term $\sqrt{N}$ for eMUX_ℓ which we show in Section 4.2.1.

4.1.2 Lower Bound

In this section we will recount Paul's lower bound for the multiplexing function (MUX) [Pau75]. We note that Paul's basis was $\{\wedge, \oplus, \neg\}$ (with free $\neg$-gates and costly $\wedge, \oplus$-gates) which is different from our basis. Still, the core ideas for the argument remain the same, an argument via Gate Elimination. Here, we simply provide our refined presentation of Paul's lower bound that aligns with our basis.

To this end, let us first quickly recall the main idea behind gate elimination: we take a circuit, substitute a subset of inputs with constants, and then eliminate gates according to a set of basic simplification rules.

We categorize most of the rules into four categories below.

$0 \wedge \gamma \to 0$	$1 \wedge \gamma \to \gamma$	$\gamma \wedge \neg\gamma \to 0$	$\gamma \wedge \gamma \to \gamma$
$\gamma \wedge 0 \to 0$	$\gamma \wedge 1 \to \gamma$	$\neg\gamma \wedge \gamma \to 0$	
$1 \vee \gamma \to 1$	$0 \vee \gamma \to \gamma$	$\gamma \vee \neg\gamma \to 1$	$\gamma \vee \gamma \to \gamma$
$\gamma \vee 1 \to 1$	$\gamma \vee 0 \to \gamma$	$\neg\gamma \vee \gamma \to 1$	
$\neg 0 \to 1$			$\neg\neg\gamma \to \gamma$
$\neg 1 \to 0$			
(fixing)	(passing)	(resolving)	(pruning)

Now, we define the set of i index selector functions as follows

Definition 12. Let $f : \{0,1\}^n \times \{0,1\}^{2^n}$ be a Boolean function with two distinguished sets of input variables: $a_1, \ldots a_n$ the set of *address bits* and $x_1, \ldots x_{2^n-1}$ be the set of *data bits*.

We define $A(f) = \{\alpha \in \{0,1\}^n : f\restriction_{a=\alpha} \equiv x_{(\alpha)}\}$ to be the set of *selectable addresses* of f.

We define S_i, the set of i index selector functions, as:

$$S_i = \{f : \{0,1\}^n \times \{0,1\}^{2^n} \to \{0,1\} : |A(f)| \geq i\}$$

In words, the set S_i contains Boolean functions f on $(n+2^n)$ input variables such that there are at least i addresses a satisfying $f(a,x) = x_{(a)}$. Thus, it is easy to see that when $i = 2^n$, we have $f \equiv \mathsf{MUX}_n$. With this observation, we obtain the following lower bound on MUX.

Theorem 13. *For the DeMorgan basis $\{\wedge, \vee, \neg\}$ where $\neg$-gates are for free, and for all $f \in S_i$, we have $CC(f) \geq 2i - 2$. Furthermore, $CC(\mathsf{MUX}_n) \geq 2N - 2$ where $N = 2^n$.*

Proof. Let C be an optimal circuit for some $f_i \in S_i$ (as in Definition 12) with $i \geq 1$, we proceed with proving the lower-bound by induction on i. The base case $i = 1$ is vacuous as $2 \cdot 1 - 2 = 0$ and the circuit complexity of all functions is non-negative.

For $i = 2$, we wish to show that $CC(f_2) \geq 2 \cdot 2 - 2 = 2$ for some $f_2 \in S_2$. By Definition 12, there are two selectable addresses α_1, α_2. By definition, f_2 is a Boolean function such that there exists at least two addresses $\alpha_1, \alpha_2 \in \{0,1\}^\ell$ such that $f_2(\alpha_1, x) = x_{(\alpha_1)}$ and $f_2(\alpha_2, x) = x_{(\alpha_2)}$. Since α_1 and α_2 corresponds to two different data bits, we need at least one $\wedge$-gate to select the appropriate appropriate data bit and one $\vee$-gate to combine the results of the selection (either one of them). Thus, $CC(f_2) \geq 2$.

For some $i = k \geq 1$, assume that $CC(f) \geq 2k - 2$ for any $f \in S_k$. Fix $f_{k+1} \in S_{k+1}$, we will show that $CC(f_{k+1}) \geq 2(k+1) - 2$. To this end, let C be an optimal circuit computing f_{k+1}. By definition of S_{k+1} we know $|A(f_{k+1})| \geq k + 1$. Consider any $\alpha \in A$ and it's corresponding input $x_{(\alpha)}$. We analyze the following cases on the fan-out of $x_{(\alpha)}$ and argue the desired lower-bound for $|C|$ on all cases. First, we note that $\text{fanout}(x_{(\alpha)})$ cannot be 0. Assume towards contradiction that $\text{fanout}(x_{(\alpha)}) = 0$. Since f_{k+1} depends on $x_{(\alpha)}$ then $x_{(\alpha)}$ must be the output of the circuit (i.e. $f_{k+1} \equiv x_{(\alpha)}$ regardless of the value of a). Since $k \geq 1$, $|A(f_{k+1})| \geq 2$ and thus there exists an $a' \in A$ that is distinct from A. However by definition, $f_{k+1}\restriction_{a=\alpha'} \equiv x_{(\alpha')} \not\equiv x_{(\alpha)}$, a contradiction. We now analyze the remaining cases.

- Case 1: $\text{fanout}(x_{(\alpha)}) > 1$. In other words, $x_{(\alpha)}$ is feeds into at least two other costly gates in C. Fix $b \in \{0,1\}$ and consider C', the circuit obtained by moving any wires originating from $x_{(\alpha)}$ in C to the constant b and then performing standard gate elimination. We note that $|C'| \leq |C| - 2$ since $x_{(\alpha)}$ had at least two costly neighbors which would have been eliminated after our constant substitution.

  Let $f' : \{0,1\}^\ell \times \{0,1\}^N$ be the function C' computes. Observe that $A(f') = A(f) \setminus \{\alpha\}$. By the inductive hypothesis, $2^k - 2 \leq CC(f') \leq |C'| \leq |C| - 2 \leq CC(f_{k+1}) - 2$. Rearranging yields $CC(f_{k+1}) \geq 2k - 2 + 2 = 2(k+1) - 2$.

- Case 2: $\text{fanout}(x_{(\alpha)}) = 1$. This means that $(\neg)x_{(\alpha')}$ is fed into exactly one costly gate in C. Let us call this costly gate g. We first show that it is not the output of the circuit. If it were, notice that there is some constant we can substitute in for $x_{(\alpha')}$ that would eliminate g via a fixing rule and leaving the circuit constant. In other words, there exists a constant $b \in \{0,1\}$ such that $f_{k+1}\restriction_{x_{(\alpha)}=b}$ is degenerate with respect to all $x_{(\alpha')}$ for any $\alpha' \neq \alpha$ and $\alpha \neq \alpha'$. However, since $|A(f_{k+1})| \geq 2$, there is at least one other α'' such that $f_{k+1}\restriction_{a=\alpha'', x_{(\alpha)}=b} \equiv x_{(\alpha'')} \neq b$.

  Therefore, g is not the output gate and thus $(\neg)g$ feeds another costly gate $h \in \{\wedge, \vee\}$. Substituting the constant for $x_{(\alpha)}$ which eliminates g via a fixing rule will then also eliminate h as well. Thus, we obtain $CC(f_{k+1}) = |C| \geq 2(k+1) - 2$ via the same argument as in case 2.

Since $CC(f_{k+1}) \geq 2 \cdot 2^{k+1} - 2$ for both cases, the statement is true by the principle of mathematical induction. Set $i = 2^n = N$, we obtain $CC(\mathsf{MUX}_n) \geq 2N - 2$. □

4.2 On The Upper Bound and Lower Bound of eMUX

In this section, we establish a tighter upper bound for eMUX_ℓ on top of Klein and Patterson's construction, and then propose our conjecture on the lower bound and explain some difficulties proving it using Gate Elimination alone. We begin with the upper bound.

4.2.1 Upper Bound

The analysis in Lemma 11 suggests a recursive construction for the circuit which can yield an improvement for the $3\sqrt{N}$ term. In particular, we notice that we can split the string b into two halves and continue "filtering" the remaining variables and repeat the process. With this observation, we establish the following upper bound for eMUX_ℓ.

Theorem 14. *Let $\ell \in \mathbb{N}$, $CC(\mathsf{eMUX}_\ell) \leq 2N + \mathcal{S}(N) + 3$, where $N = 2^{2^\ell}$, and $\mathcal{S}(N) = \sum_{i=1}^{\log\log N - 1} 2^i \cdot N^{1/2^i}$*

Proof. Let $r = s = 2^\ell/2$. We apply one more iteration of the recursive construction and observe the improvement. In particular, we expand the expression for eMUX_ℓ similar to Lemma 11 further by splitting b into b_1 and c_1 where each has length $r/2 = 2^\ell/4$, then use two subcircuits computing $\mathsf{eDEC}_{\ell-2}$, one to fix c_1 and the other to process b_1 accordingly.

$$
\begin{aligned}
\mathsf{eMUX}_\ell(a, x) &= \mathsf{eMUX}_\ell(b_1 \circ c_1 \circ c_0, x) \\
&= \bigvee_{i=0}^{2^{r/2}-1} \bigvee_{j=0}^{2^{r/2}-1} \bigvee_{k=0}^{2^s-1} (sel_i(b_1) \wedge sel_j(c_1) \wedge sel_j(c_0) \wedge x_{i\cdot 2^r + j\cdot 2^{r/2}+k}) \\
&= \bigvee_{i=0}^{2^{r/2}-1} \left(sel_i(b_1) \wedge \left(\bigvee_{j=0}^{2^{r/2}-1} \left(sel_j(c_1) \wedge \left(\bigvee_{k=0}^{2^s-1} (sel_k(c_0) \wedge x_{i\cdot 2^r + j\cdot 2^{r/2}+k}) \right) \right) \right) \right) \\
&= \mathsf{eMUX}_{\ell-2} \left(b_1, \left(\bigvee_{j=0}^{2^{r/2}-1} \left(sel_j(c_1) \wedge \left(\bigvee_{k=0}^{2^s-1} (sel_k(c_0) \wedge x_{i\cdot 2^r + j\cdot 2^{r/2}+k}) \text{ , for } i = 0, \ldots, 2^{r/2}-1 \right) \right) \right) \right)
\end{aligned}
$$

We will now analyze the circuit complexity of this construction. In particular,

- The first stage of fixing c_0 is the same as the analysis in Lemma 11 which requires
$$CC(\mathsf{eDEC}_{\ell-1}) + 2 \cdot 2^{2^\ell} - 2^{2^\ell/2}$$
- The second stage of selecting the c_1-part for the index requires $2^{\ell/2}$ $\wedge$-gates and $2^{r/2}(2^{r/2}-1)$ $\vee$-gates since this stage goes through all $2^{\ell/2}$ indices to select the c_1-part and combine. Taking $CC(\mathsf{eDEC}_{\ell-2})$ into account, the total number of gates used in this stage is
$$CC(\mathsf{eDEC}_{\ell-2}) + 2^{2^{\ell/2}} + 2^{2^{\ell/4}}(2^{2^{\ell/4}} - 1) = CC(\mathsf{eDEC}_{\ell-2}) + 2 \cdot 2^{2^{\ell/2}} - 2^{2^{\ell/4}}$$
- The third stage of selecting the b_1-part for the index, given fixed c, c_1-parts requires $2^{\ell/4}$ $\wedge$-gates and $2^{\ell/4} - 1$ $\vee$-gates. Taking the complexity $CC(\mathsf{eDEC}_{\ell-2})$, the total number of gates used in this stage is
$$CC(\mathsf{eDEC}_{\ell-2}) + 2 \cdot 2^{2^{\ell/4}} - 1$$

Substituting $N = 2^{2^\ell}$ and apply Theorem 5 for $CC(\mathsf{eDEC}_{\ell-1})$ and $CC(\mathsf{eDEC}_{\ell-2})$, we obtain

$$
\begin{aligned}
CC(\mathsf{eMUX}_\ell) &\leq CC(\mathsf{eDEC}_{\ell-1}) + 2 \cdot 2^{2^\ell} - 2^{2^{\ell/2}} + CC(\mathsf{eDEC}_{\ell-2}) + 2 \cdot 2^{2^{\ell/2}} - 2^{2^{\ell/4}} + CC(\mathsf{eDEC}_{\ell-2}) + 2 \cdot 2^{2^{\ell/4}} - 1 \\
&= 2 \cdot 2^{2^\ell} + 2^{2^{\ell/2}} + 2^{2^{\ell/4}} + CC(\mathsf{eDEC}_{\ell-1}) + 2 \cdot CC(\mathsf{eDEC}_{\ell-2}) - 1 \\
&\leq 2N + N^{1/2} + N^{1/4} + N^{1/2} + \mathcal{S}(N^{1/2}) + 2N^{1/4} + 2 \cdot \mathcal{S}(N^{1/4}) - 1 \\
&= 2N + 2N^{1/2} + 3N^{1/4} + \mathcal{S}(N^{1/2}) + 2 \cdot \mathcal{S}(N^{1/4}) - 1
\end{aligned}
$$

where $\mathcal{S}(M) = \sum_{i=1}^{\log\log M - 1} 2^i \cdot M^{1/2^i}$. We can repeat the process for $\ell = \log\log N$ times and express the upper bound of $CC(\mathsf{eMUX}_\ell)$ as the following recurrence

$$T(N) = \begin{cases} 3, & \text{if } N = 2 \\ T(N^{1/2}) + 2N + \mathcal{S}(N^{1/2}), & \text{if } N > 2 \end{cases}$$

For the base case $N = 2$ (i.e. $\ell = 0$), we have eMUX_0 which is equivalent to multiplexing 2 data bits x_0, x_1. We can construct the circuit by using a eDEC_0-circuit which requires zero costly gate because it is simply the input variable itself (a_0) or its negation ($\overline{a_0}$). Then, we construct $\overline{a_0} \wedge x_0$, $a_0 \wedge x_1$ and combine their results with an $\vee$-gate which yields a total of 3 costly gates.

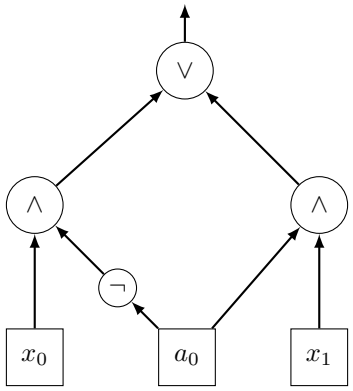


Figure 5: A circuit computing eMUX_0

Justification for the case of $N > 2$ is based on the analysis of performing one recursive step as in Lemma 11 as shown below

$$\begin{aligned} CC(\mathsf{eMUX}_\ell) &\leq \underbrace{CC(\mathsf{eDEC}_{\ell-1}) + (2N - \sqrt{N})}_{\text{selecting data bits based on an address slice stage}} + CC(\mathsf{eMUX}_{\ell-1}) \\ &\leq \underbrace{\sqrt{N} + \mathcal{S}(\sqrt{N})}_{\text{apply Theorem 5 for } CC(\mathsf{eDEC}_{\ell-1})} + (2N - \sqrt{N}) + \underbrace{CC(\mathsf{eMUX}_{\ell-1})}_{T(N^{1/2})} \\ &= 2N + \mathcal{S}(N^{1/2}) + T(N^{1/2}) = T(N) \end{aligned}$$

Now we unroll the recurrence for $\log\log N$ steps (until the data bits domain shrink to the base case)

$$\begin{aligned} T(N) &= T(N^{1/2}) + 2N + \mathcal{S}(N^{1/2}) \\ &= T(N^{1/4}) + 2N^{1/2} + \mathcal{S}(N^{1/4}) + 2N + \mathcal{S}(N^{1/2}) \\ &= T(N^{1/8}) + 2N^{1/4} + \mathcal{S}(N^{1/8}) + 2N^{1/2} + \mathcal{S}(N^{1/4}) + 2N + \mathcal{S}(N^{1/2}) \\ &\cdots \\ &= T(2) + \sum_{i=0}^{\log\log N - 1} (2N^{1/2^i} + \mathcal{S}(N^{1/2^{i+1}})) \\ &= 3 + \sum_{i=0}^{\log\log N - 1} (2N^{1/2^i} + \mathcal{S}(N^{1/2^{i+1}})) \end{aligned}$$

Note that we can simplify the expression further by simplifying $\sum_{i=0}^{\log\log N-1}(2N^{1/2^i} + \mathcal{S}(N^{1/2^{i+1}}))$. In particular, for each i, we expand $\mathcal{S}(N^{1/2^{i+1}}) = \sum_{j=1}^{\log\log N - i - 2} 2^j N^{1/2^{i+1+j}}$. For readability, set $L = \log\log N$, then, we have

$$\sum_{i=0}^{L-1}(2N^{1/2^i} + \mathcal{S}(N^{1/2^{i+1}})) = \sum_{i=0}^{L-1} 2N^{1/2^i} + \sum_{i=0}^{L-1}\sum_{j=1}^{L-i-2} 2^j N^{1/2^{i+1+j}}$$

The index of the inner sum runs up to $L-1-2$ because by definition $\mathcal{S}(M) = \sum_{i=1}^{\log\log M - 1} 2^i \cdot M^{1/2^i}$, then for $M = N^{1/2^{i+1}}$, we have $\log\log M - 1 = L - (i+1) - 1 = L - i - 2$.

Re-index the double sum with $u = i + j + 1$, then since $j \geq 1$, we have $u - 1 - i \geq 1 \iff i \leq u - 2$. Thus,

$$\sum_{i=0}^{L-1} \sum_{j=1}^{L-i-2} 2^j N^{1/2^{i+1+j}} = \sum_{u=1}^{L-1} \sum_{i=0}^{u-2} 2^{u-i-1} N^{1/2^u}$$

Rename the index of the single sum with u and factor out the first term of the sum, we have

$$\sum_{i=0}^{L-1} 2N^{1/2^i} = 2N + \sum_{u=1}^{L-1} 2N^{1/2^u}$$

Putting everything together, we have

$$\begin{aligned}\sum_{i=0}^{L-1} 2N^{1/2^i} + \sum_{i=1}^{L-1} \sum_{i=1}^{L-i-2} 2^j N^{1/2^{i+1+j}} &= 2N + \sum_{u=1}^{L-1} 2N^{1/2^u} + \sum_{u=1}^{L-1} \sum_{i=0}^{u-2} 2^{u-i-1} N^{1/2^u} \\ &= 2N + \sum_{u=1}^{L-1} \left(2 + \sum_{i=0}^{u-2} 2^{u-i-1}\right) N^{1/2^u}\end{aligned}$$

Finally, we evaluate $2 + \sum_{i=0}^{u-2} 2^{u-i-1}$ to determine the coefficient of $N^{1/2^u}$. To this end, re-index the sum with $k = u - i - 1$, and since $i = 0, 1, \ldots, u-2$, we have $k = u-1, u-2, \ldots, 1$. In other words,

$$2 + \sum_{i=0}^{u-2} 2^{u-i-1} = 2 + \sum_{k=1}^{u-1} 2^k = 2 + \frac{2(2^{u-1} - 1)}{2 - 1} = 2^u \quad \text{(apply geometric series formula)}$$

Thus, substitute $L = \log\log N$ and $\mathcal{S}(N) = \sum_{i=1}^{\log\log N - 1} 2^i \cdot N^{1/2^i}$, we obtain the desired upper bound

$$T(N) = 3 + 2N + \sum_{u=1}^{\log\log N - 1} 2^u N^{1/2^u} = 2N + \mathcal{S}(N) + 3$$

□

If we unroll the first few terms in the sum $\mathcal{S}(N)$, we get $2N + 2\sqrt{N} + O(\sqrt[4]{N})$ which is a slight improvement over Lemma 11 which is $2N + 3\sqrt{N} + O(\sqrt[4]{N})$

Remark 15. *We think it can be beneficial for gaining a better understanding on how* MUX_n *works for general cases where* n *is either even or odd by getting an exact count of the upper bound for those cases. We think each may differ by a few gates, but the general form of the upper bound should still be* $2N + 2\sqrt{N} + O(\sqrt[4]{N})$.

4.2.2 Conjecture on the Lower Bound and the Optimal Circuit Structure

It is obvious that Paul's $2N - 2$ lower bound for MUX_n also holds for eMUX_ℓ. However, we conjecture that the lower bound can be made tighter and even close to the upper bound in Theorem 14. In particular, Paul's technique also leaves room for improvement as they only targeted the restrictions on the data bits x's in the argument, and no restriction on the address bits a's. Still, one sticky point with the approach of gate elimination via restricting the address bits is that it is not obvious whether any address bit a_i shares a costly gate with a data bit x_i.

Thus, we believe that one possible step to get around this is to show that an optimal eMUX-circuit requires an embedded optimal eDEC-circuit. If it is true, then restrictions on address bits could wipe out more than two costly gates

Conjecture 16. *Let* $\ell \in \mathbb{N}$*, then* $CC(\mathsf{eMUX}_\ell) \geq 2N + O(\sqrt{N})$.

One could also try showing the recursive construction of eMUX in Theorem 14 is optimal. But obviously, proving that *every* optimal circuit has to be structured this way is non-trivial as it is something that standard Gate Elimination does not capture very well. Gaining even a slight improvement over $2N-2$ without digging too deep into structural characterization would be a meaningful contribution, because it would require new insights beyond Paul's method. In particular, we can gain an improvement by showing that certain number of address bits and data bits cannot be too "close" to one another in the circuit, i.e. they cannot share costly gates or at least not too many, which one can exploit and argue, for the very least, a larger-than-2 constant number of gates are eliminated.

4.2.3 Fusion Method - A Promising Direction?

As surveyed by Wigderson [Wig93], the Fusion Method turns the computation into a static cover problem over a set of "fusing functionals." By choosing functionals tailored to the address/data separation in MUX_n, it may be possible to prove that small covers are impossible and thereby derive new bounds. Recently, Calavar and Oliveira recast Fusion Method using a modern set-theoretic language that works for various settings including non-monotone Boolean circuits [CO25]. They mention that reproving known circuit lower bounds via Gate Elimination, or better yet improving these lower bounds, for non-monotone functions using their framework remains an open problem.

Intuition on proving Lower Bound via Fusion Method. Informally speaking, the fusion method asks a simple question: if a circuit for f were really small, could we "mix and match" pieces of many true examples to accidentally accept a false one? Think of each input bit (or its negation) as a basic condition, and think of an AND as saying "you must satisfy all these conditions at once." A small circuit would impose only a few such "all-at-once" requirements. Fusion formalizes a tester that collects all conditions satisfied by a known true input and tries to keep combining them in the specific ways the small circuit would—hoping to land on a contradiction when we apply them to a false input. Our job, then, is to specify a short checklist of required combinations (or *a cover*) that forces any such attempt to end in a contradiction. The size of the smallest checklist tells you how many ANDs you truly need. In short, if every "mix and match" plan with too few AND-steps breaks, the circuit must use more AND gates—giving the lower bound. Thus, the minimal cover size exactly measures the AND cost needed to compute f. In this way, fusion cleanly yields DeMorgan circuit lower bounds by showing that any attempt to compute f with too few AND-gates is inevitably incorrect on certain inputs.

Circuits Model in Negation Normal Forms (NNF) as Set Theoretic. We briefly explain circuits in NNF and its cover complexity in a high level idea. Imagine every string $x \in \{0,1\}^n$ as a point in a universe of all length-n strings Γ. Each literal (a variable or its negation) denotes a "region" in that universe, i.e. x_i is the region of strings where the i^{th}-bit of is 1, and $\overline{x_i}$ is the complementary region, i.e. strings where the i^{th}-bit is 0. In negation normal form (NNF), every internal gate is either an AND-gate or an OR-gate, and we read those gates as pure set operations, i.e. an AND-gate corresponds to set intersection ($\cap$) and an OR-gate corresponds to set union ($\cup$). This means that an AND-gate keeps the overlap of two regions, whereas an OR-gate takes the combined regions.

We evaluate the circuit from the leaf-level upward, and as proceed, we construct the regions defined by the internal gates, step by step, and the region defined by the output gate is exactly the set A of length-n strings where the circuit evaluate to 1. Thus, counting gates now matches counting operations: each AND is one intersection, each OR is one union, and total circuit size is the total number of these set operations used to assemble A from the literals.

Then, for a high level idea, *a cover* preserves some propeties and conditions on the computation of the circuit expressed as set-theoretic, and a *cover complexity* is a measurement on the size of the "checklist" of required combinations/conditions. Calaver and Oliveira proved that the number of intersections required is "sandwiched" between the cover complexity and its square.

Theorem 17. *(Informal of Theorem 22 and 24 of [CO25]) Let $D_\cap$ denote the intersection complexity of a Boolean circuit expressed in set theoretic, and let ρ denote its cover complexity, then $\rho \leq D_\cap \leq \rho^2$.*

Potential next steps towards this direction. Even though what we have learned so far about Fusion Method applies for DeMorgan circuits in NNF, rather than generic DeMorgan circuits, it is reasonable to think that this approach is helpful for gaining more knowledge on the lower bound of DEC and MUX for various reasons.

- Under the NNF restriction, we can observe that the construction shows in Figure 1 or Figure 2a is the only option for the upper-bound since we cannot have negations in the middle of the circuit. This construction shows a layer of AND-gates only which is compatible with how Fusion Method is useful for determining the AND-complexity (i.e. the lower bound on the number of AND-gates). Thus, if we can extend the framework in [CO25] for multi-output functions, then it is reasonable to believe that DEC-circuits in NNF must use *many* AND-gates and thus a tighter bound.
- Along this line, we observe that the full recursive upper bound construction for MUX-circuits as proposed in Theorem 14 is already in NNF assuming the NNF construction of DEC-circuits. Looking further in the construction, we observe that it *mainly* uses AND-gates as when we shrink the overhead to just MUX_1 in this full recursive construction, the number of OR-gates used is a constant. Thus, it is also reasonable to believe that Fusion Method can help us show that the AND-complexity of MUX in NNF is high, and thus, a tighter lower bound. Gaining more knowledge even under NNF restriction is a meaningful contribution.
- Lastly, to our knowledge, it seems that proving lower bound via Fusion Method does not require us to dig too deep into the local structural characterization of parts of the circuit. As surveyed by Wigderson, this method essentially tries to answer the static cover problem asking how many intersections/AND-gates one would need to maintain the correctness of the circuit without the need to know how these intersections/AND-gates are put together [Wig93].

References

[BMNW18] Nikolaj Bjørner, Leonardo de Moura, Lev Nachmanson, and Christoph M Wintersteiger. "Programming Z3". In: *International Summer School on Engineering Trustworthy Software Systems*. Springer, 2018, pp. 148–201.

[BS84] Norbert Blum and Martin Seysen. "Characterization of all optimal networks for a simultaneous computation of AND and NOR". In: *Acta informatica* 21.2 (1984), pp. 171–181.

[CO25] Bruno Cavalar and Igor Oliveira. "Boolean Circuit Complexity and Two-Dimensional Cover Problems". In: *ACM Transactions on Computation Theory* 17.2 (2025), pp. 1–23.

[DK11] Evgeny Demenkov and Alexander S. Kulikov. "An Elementary Proof of a 3n - o(n) Lower Bound on the Circuit Complexity of Affine Dispersers". In: *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*. Ed. by Filip Murlak and Piotr Sankowski. Vol. 6907. Lecture Notes in Computer Science. Springer, 2011, pp. 256–265. DOI: 10.1007/978-3-642-22993-0_25. URL: https://doi.org/10.1007/978-3-642-22993-0%5C_25.

[FGHK16] Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. "A Better-Than-3n Lower Bound for the Circuit Complexity of an Explicit Function". In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. 2016, pp. 89–98. DOI: 10.1109/FOCS.2016.19.

[HR24] Justin Holmgren and Ron Rothblum. "Linear-size boolean circuits for multiselection". In: *39th Computational Complexity Conference (CCC 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2024, pp. 11–1.

[ILMR02] Kazuo Iwama, Oded Lachish, Hiroki Morizumi, and Ran Raz. "An Explicit Lower Bound of 5n - o(n) for Boolean Circuits". In: (2002).

[IM02] Kazuo Iwama and Hiroki Morizumi. "An Explicit Lower Bound of 5n - o(n) for Boolean Circuits". In: *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*. Ed. by Krzysztof Diks and Wojciech Rytter. Vol. 2420. Lecture Notes in Computer Science. Springer, 2002, pp. 353–364. DOI: 10.1007/3-540-45687-2_29. URL: https://doi.org/10.1007/3-540-45687-2%5C_29.

[KP80] Klein and Paterson. "Asymptotically optimal circuit for a storage access function". In: *IEEE Transactions on Computers* 100.8 (1980), pp. 737–738.

[LK21] SA Lozhkin and DE Khzmalyan. "The Complexity of the Standard Multiplexer Function in a Class of Switching Circuits". In: *Computational Mathematics and Modeling* 32.4 (2021), pp. 478–489.

[Loz24] Sergei Andreevich Lozhkin. "On the depth of a multiplexer function with a small number of select lines". In: *Mathematical Notes* 115.5 (2024), pp. 748–754.

[LY22] Jiatu Li and Tianqi Yang. "3.1n - $o(n)$ circuit lower bounds for explicit functions". In: *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*. Ed. by Stefano Leonardi and Anupam Gupta. ACM, 2022, pp. 1180–1193. DOI: 10.1145/3519935.3519976. URL: https://doi.org/10.1145/3519935.3519976.

[Pau75] Wolfgang J Paul. "A 2.5 n-lower bound on the combinational complexity of Boolean functions". In: *Proceedings of the seventh annual ACM symposium on Theory of computing*. 1975, pp. 27–36.

[Red73] NP Red'kin. "Proof of minimality of circuits consisting of functional elements". In: *Systems Theory Research: Problemy Kibernetiki* (1973), pp. 85–103.

[Sat81] Jürgen Sattler. "Netzwerke zur simultanen Berechnung Boolescher Funktionen (Ausführliche Kurzfassung)". In: *Theoretical Computer Science: 5th GI-Conference Karlsruhe, March 23–25, 1981*. Springer. 1981, pp. 32–40.

[SC98] JE Savage and Models Of Computation. *Exploring the power of Computing*. 1998.

[Sch74] Claus-Peter Schnorr. "Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen". In: *Computing* 13.2 (1974), pp. 155–171. DOI: 10.1007/BF02246615. URL: https://doi.org/10.1007/BF02246615.

[Sto77] Larry J. Stockmeyer. "On the Combinational Complexity of Certain Symmetric Boolean Functions". In: *Math. Syst. Theory* 10 (1977), pp. 323–336. DOI: 10.1007/BF01683282. URL: https://doi.org/10.1007/BF01683282.

[Weg87] Ingo Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.

[Wig93] Avi Wigderson. "The fusion method for lower bounds in circuit complexity". In: *Combinatorics, Paul Erdos is Eighty* 1.453-468 (1993), p. 68.

[Zwi91] Uri Zwick. "A 4n lower bound on the combinational complexity of certain symmetric boolean functions over the basis of unate dyadic Boolean functions". In: *SIAM Journal on Computing* 20.3 (1991), pp. 499–505.