Computers, which have become a ubiquitous staple of modern society, consume nearly 10% of all the energy produced worldwide. While computing implementation technology has been made more energy efficient over the years, the energy required to operate a gate logically has become an increasingly large proportion of the total energy required. A systematic improvement of this power use would result in significant power savings, which would grow even more appreciable as overall efficiency improves. One method for achieving these power savings would be the use of conservative reversible logic (CRL) gate systems for design system. However, to date, only a few designs using these types of gates have been developed. This sporadic development is primarily due to the lack of a systematic method for representing CRL gates.

### 1. INTRODUCTION

#### A Binary Decision Diagram (BDD) is a systematic method for representing digital circuits using a directed acyclic graph [1]. In a typical BDD, a truth table is used to create the diagram, as demonstrated in Fig. 1.

![Truth Table and BDD for AND2 operation](image)

In the BDD, dashed arrows indicate the input it originates from is a ‘0’, while a solid arrow indicates a ‘1’. Starting at the first input ‘A’, the arrows are followed depending on the value of the input, and ultimately terminate at either a ‘0’ or ‘1’, which indicates the output.

While CRL gates can be represented using traditional BDDs, their truth tables grow exponentially with the number of inputs. A CRL gate is ultimately just a permutation of its inputs.

Due to this, the representation of these gates can be condensed using a modified BDD, designed specifically for permutations, known as a nDD [2]. In a nDD, any permutation can be written as an ordered list of two element swaps on certain items.

For example, if an initial ordered list of items is written as:

\[ \pi_{\text{initial}} = (1,2,3,4) \]

and a permutation of that list of items is written as:

\[ \pi_{\text{permutation}} = (1,3,2) \]

The permutation can be as the original list of items followed by a swap of the second and third items, which yields the nDD pictured in Fig. 2.

![Example nDD](image)

In a nDD, a solid arrow indicates the swap occurs and the dashed arrow indicates it does not, and the ‘0’ and ‘1’ termination blocks indicate whether the resulting permutation is in the desired list of permutations or not.

### 2. OBJECTIVES

- Present two methods of adapting nDDs to represent CRL gates
- Demonstrate examples of design manipulation and analysis

### 3. BACKGROUND

#### One Efficient Design Representation of Conservative Reversible Logic Gates

Tyler McCormick, J.W. Bruce, Ph.D.
Department of Electrical and Computer Engineering

The simplest CRL gate, the Fredkin gate [3], has been well studied and used in large digital designs [4]. However, larger CRL gates have not been systematically examined.

Figs. 3 and 4 show two representative 4-input, 4-output CRL gates and their operation with respect to their “control” inputs, A.

![A 4x4 CRL gate and its two operational states](image)

For the first approach, each output column of the original truth table is mapped to a specific permutation, and each permutation is represented in the final nDD. First the control line, column A, is defined as an ordered list of unique items:

\[ \pi_{\text{A}} = (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16) \]

where items 1-8 are defined as 0’s and items 9-16 are defined as 1’s. Therefore, the resulting nDD for CRL gate ‘X’ will contain the four distinct permutations:

\[ \{\pi_{\text{X}}, (1,2,3,4,9,10,11,12,7,8,13,14,15,16), \}

\[ (1,2,9,11,5,6,13,15,10,4,12,7,14,8,16), \]

\[ (1,9,10,5,11,7,2,4,8,13,14,15,16) \]

and the final nDDs for CRL gates ‘A’ and ‘B’, pictured in Fig. 5, can be constructed.

![Representation #1 of 4x4 CRL gates A (left) and B using nDD](image)

While CRL ‘B’ will contain the permutations:

\[ \{\pi_{\text{B}}, (1,2,3,4,9,10,11,12,7,8,13,14,15,16), \}

\[ (1,2,9,11,5,6,13,15,10,4,12,7,14,8,16), \]

\[ (1,9,10,5,11,7,2,4,8,13,14,15,16) \]

and the final nDDs for CRL gates ‘A’ and ‘B’, pictured in Fig. 5, can be constructed.

![Representation #2 of 4x4 CRL gate A (left) and gate B using nDD](image)

5. RESULTS

Using Approach 2 for depicting CRL gates using nDDs, CRL gates ‘A’ and ‘B’, are denoted with nDDs in Fig. 6. Using the precise element transposition notation for nDDs, the resulting permutation from ‘A’ followed by ‘B’ can be written as the transpositions of CRL gate ‘A’ followed by the transpositions of CRL gate ‘B’.

\[ r(\tau_{\text{A}}) \circ r(\tau_{\text{B}}) = r(\tau_{\text{AB}}) \]

This series of transpositions can then be re-ordered using an algorithm for swapping transpositions [2], first, the last two transpositions are compared:

\[ r(\tau_{\text{2B}}) \circ r(\tau_{\text{1A}}) \]

The first two transpositions violate the fixed order of a nDD, so they must be adjusted and swapped:

\[ r(\tau_{\text{2A}}) \circ r(\tau_{\text{1B}}) \]

Lastly, the last two transpositions can be eliminated because they are swapping the same two items, yielding the following remaining transposition:

\[ r(\tau_{\text{2B}}) \]

This remaining transposition, is then depicted as the nDD pictured in Fig. 8.

![The 4x4 CRL gate followed by CRL gate B](image)

This single nDD yields the output column permutation ‘1324’, which is identical to the nDD created from the resulting permutation from cascading CRL gates ‘A’ and ‘B’, as depicted in Fig. 7.

![nDD denoting the cascaded CRL gates in Fig. 6](image)

6. CONCLUSION

Using a nDD, two different methods for representing CRL gates were created and then analyzed. Not only are the methods a compact and efficient way to represent CRL gates, they can also be used for designs consisting of cascaded CRL gates.

### REFERENCES


---

**Fig. 1:** Truth Table and BDD for AND2 operation

**Fig. 2:** Example nDD

**Fig. 3:** A 4x4 CRL gate and its two operational states

**Fig. 4:** A 4x4 CRL gate and its two operational states

**Fig. 5:** Representation #1 of 4x4 CRL gates A (left) and B using nDD

**Fig. 6:** Representation #2 of CRL gate A (left) and gate B using nDD

**Fig. 7:** The 4x4 CRL gate followed by CRL gate B

**Fig. 8:** nDD denoting the cascaded CRL gates in Fig. 6

---

**Example**

\[ (\langle \pi_{\text{A}} \rangle, (1,2,3,4,9,10,11,12,7,8,13,14,15,16), \]

\[ (1,2,9,11,5,6,13,15,10,4,12,7,14,8,16), \]

\[ (1,9,10,5,11,7,2,4,8,13,14,15,16) \]

and the final nDDs for CRL gates ‘A’ and ‘B’, pictured in Fig. 5, can be constructed.