Optimal Fredkin Gate Designs for Logical Operations with Two and Three Inputs

Weston Beebe and J.W. Bruce
Department of Electrical and Computer Engineering

1. INTRODUCTION

Energy consumption and heat are crucial limiting factors in modern digital systems. Landauer [1] showed that any irreversible logic gate dissipates kT ln 2 Joules of energy with each irreversible bit . Therefore, reversible logic is required for ultimate energy efficiency in computational systems. Conservative logic gates [2] – a gate whose Hamming weights for inputs and outputs are equal – are theoretically energy neutral in that output signal energy can be derived from input signal energy.

2. BACKGROUND

A well-known conservative reversible logic (CRL) gate is the Fredkin gate [2]. The Fredkin gate (FG) is a "controlled-swap" whereby one signal determines whether two other signals are passed "straight-through" or "swapped". The FG is often used as an operator in many quantum computing algorithms.

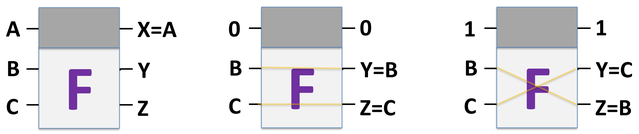

Fig. 1: Fredkin gate (L) and its two operational states

Previous research, including [2] and [3], have proposed FG implementations of common digital logic operators. Later, additional FG designs for logical primitives were proposed and used in the design of more complex logical operations [3].

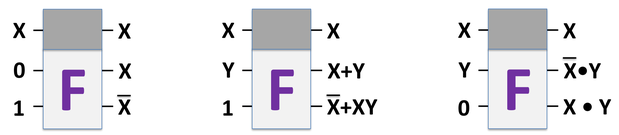

Fig. 2: FG implementations of digital logic primitive operations (L) to (R): NOT/fan-out, OR2, and AND2

To date, proposed FG designs were determined in ad-hoc manner. No systematic study of FG implementations for traditional digital logic operations has been undertaken. It is not known if the prior proposed FG designs are unique and/or optimal.

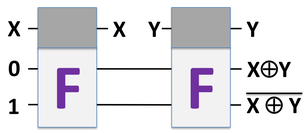

Fig. 3: FG implementations of digital logic primitive XOR2/XNOR2

3. OBJECTIVES

- Determine if existing FG implementations of traditional digital logical operations are unique
- Find more efficient FG implementations of traditional logical operations, if they exist

4. METHODOLOGY

To find FG implementations of traditional logical operations, circuits were simulated and circuit outputs were recorded for different combinations of gate inputs and gate connections.

The number of variables for each circuit were specified, but variable placement can be varied. At least one function input is required to impinge on the first gate. Additional function inputs can be applied at any other location in the circuit.

Like functional inputs, Boolean constants can be placed at any point in the circuit.

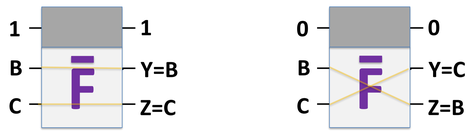

Fig. 4: FG with control signal "inverted"

The FG may be implemented in two ways – based on the interpretation of the "control" signal, A. See Fig. 1. The signals B and C may also be exchanged with an active-low A, as shown in Fig 4. This study considered both FG implementations.

At each additional stage appended to the circuit, at least one, and up to all three, of the inputs of a FG can be driven by the outputs of the previous gate. The inputs of the first gate can be three variables, two variables and a constant, or one variable and two constants. This study considered all possible ways to connect subsequent FG stages.

At each stage, the gate output is computed according to its inputs and behavior. If the desired digital function output is not found, an additional FG stage is added. For each stage, unused variables, constants, and the previous gate's outputs are combined in sets of three. For each combination, the inputs are considered in each of the six possible ways to order them. Both FG implementations (Figs. 1 and 4) were considered.

In this study, combinations of variables, constants, gates, and wirings between gates were built to a specified depth of gates. Additional gates were added in the search until logical operations of interest in this search had been found.

5. RESULTS AND DISCUSSION

All two-input logical functions can be implemented in one or two FGs. Four two-input logic functions require two FGs with the balance capable of being formed with a single FG. This study found that the previously published FG forms for primitive logical operation AND2 and OR2 seen in Figs. 2 and 3 to be optimal.

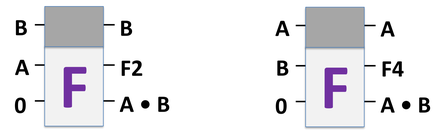

Fig. 5: FG circuits: AND2 with F2 (L) and AND2 with F4 (R)

For the two-input logical functions F2 and F4, four unique FG designs were discovered. These functions show up in conjunction with AND2, and shown in Fig. 5.

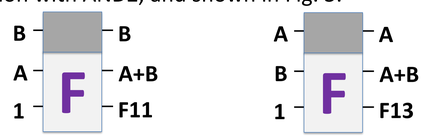

Fig. 6: FG circuits: OR2 with F11 (L), and OR2 with F13 (R)

For two-input logical functions F11 and F13, four unique FG designs were discovered. Functions F11 and F13 are created simultaneously with OR2, and shown in Fig. 6.

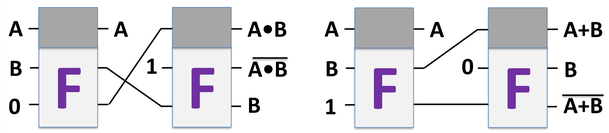

Fig. 7: FG circuits: AND2 with NAND2 (L) and OR2 with NOR2 (R)

The NAND2 and NOR2 functions, like XOR2 and XNOR2 in Fig. 3 require two FGs. These newly discovered FG designs are seen in Fig. 7. Furthermore, the AND2 and OR2 gates can be used with the FG inverter implementation to create NAND2 and NOR2 simultaneously.

A search of all possible FG implementations for all two-input logical functions was performed. Every one of the possible two-input logical function possess multiple FG implementations. Table 1 shows the number of FG designs for the non-trivial two-input logical operations.

Function	*Fn*	Number
AND2	F1	8
NAND2	F14	96
OR2	F7	8
NOR2	F8	96
XOR2	F6	32

Function	*Fn*	Number
XNOR2	F9	32
Implication	F11	4
	F13	4
Inhibition	F2	4
	F4	4

Table 1: Number of distinct FG implementations of the two-input logical operations. Trivial and 1-input functions F0 (null), F15 (identity), F3 (A), F5 (B), F10 (NOT B) and F12 (NOT A) are not considered.

There are 256 possible logical functions over three inputs. This work also found all possible FG implementations of three-input functions. Table 2 shows the number of FG designs for common three-input logical operators.

Function	*Fn*	Number
AND3	F1	192
NAND3	F254	2688
OR3	F127	192
NOR3	F128	2688
XOR3	F105	384
XNOR3	F150	384

Function	*Fn*	Number
MAJ3	F23	1536
FA_CARRY3	F23	1536
FA_SUM3	F105	384
OAI21	F234	1408
AOI21	F168	1408

Table 2: Number of distinct FG implementations of the common three-input operations.

Of the functions listed in the Table 2, only AND3 and OR3 can be implemented in two FGs, the remainder require three FGs. 32 of the 256 possible three-input operators require one FG, 122 require two FGs, and 85 require three FGs. F22, F41, F73, F97, F104, F107, F109, F121, F134, F146, F148, F151, F158, F182, F214, F232 and F233 require more than three FGs.

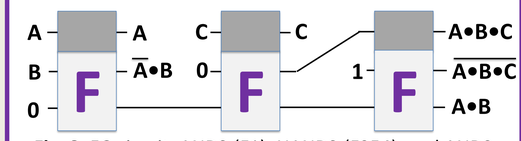

Fig. 8: FG circuit: AND3 (F1), NAND3 (F254), and AND2

6. CONCLUSIONS

This work describes the first-ever known systematic study of FG implementations of two-input and three-input logical operators. All possible FG circuit designs with one through three FGs were derived and examined. Previously published logical primitive operations in FGs were shown to be optimal. Several new optimal FG designs for logical primitive function were discovered.

No published work to date has claimed to have found optimal FG designs for three-input logical operators. This study has identified the optimal FG designs for nearly all of the 256 possible three-input logical operations, including many three-input functions commonly used in nearly all digital designs.

The study also found that FG designs for all logical operators examined exist using both variations of the Fredkin gate.

7. REFERENCES

[1] R. Landauer, "Irreversibility and Heat Generation in the Computing Process", *IBM J. Research & Development*, v.3, pp. 183-191, July 1961.
[2] E. Fredkin and T. Toffoli, "Conservative Logic", *Intl. J. Theoretical Physics*, v.21, n.3-4, pp. 219-253, 1982.
[3] J.W. Bruce et al, "Efficient Adder Circuits Based on a Conservative Reversible Logic Gate, *Proc. IEEE ISVLSI 2002*, pp.83-88, 2002.