Journal of Computer Science 4 (9): 741-743, 2008
ISSN 1549-3636

The Relationship between Exclusive-Or and the Unique Existential Quantifier

Maher A. Nabulsi and Ayman M. Abdalla
Department of Computer Science, Al-Zaytoonah University of Jordan,
Amman 11733, Jordan

Abstract: Problem Statement: The efficiency of computer architecture design is affected by the components used. Establishing a relationship between exclusive-or (XOR) and the unique existential quantifier provides alternative system implementations. **Approach:** Applications of XOR and the unique existential quantifier were explained. Then, prepositional logic was used to establish the relationship between them. **Results:** Different quantified assertions with two variables that use the unique existential quantifier were represented without quantifiers by using XOR. **Conclusions:** The unique existential quantifier and XOR were helpful in some computer architecture systems such as multiplexers, decoders and bus systems. The unique existential quantifier and XOR may be used interchangeably in some situations, but not always.

Key words: Exclusive-or, multiplexer, decoder, unique existential quantifier, universal quantifier

INTRODUCTION

This study establishes a relationship between exclusive-or (XOR) and the unique existential quantifier (∃!). The representations of the unique existential quantifier in computer architecture may be done using any of several combinations of XOR gates, multiplexers, decoders, AND gates and OR gates. This is closely related to propositional logic, where computer architecture circuits may be represented with logical formulas.

MATERIALS AND METHODS

Several applications in computer architecture implicitly apply ∃!xP(x), mainly in situations that use a bus system and need to take information from one source while all other sources are inhibited. For example, consider the predicate P(x) defined over the finite set {1, 2, 3, 4}. Then, ∃!xP(x) is logically equivalent to:

$$[P(1) \wedge \sim P(2) \wedge \sim P(3) \wedge \sim P(4)] \vee$$
$$[\sim P(1) \wedge P(2) \wedge \sim P(3) \wedge \sim P(4)] \vee$$
$$[\sim P(1) \wedge \sim P(2) \wedge P(3) \wedge \sim P(4)] \vee$$
$$[\sim P(1) \wedge \sim P(2) \wedge \sim P(3) \wedge P(4)].$$

This means that only one of the four possible elements of P(x) can be true, while the others must all be false. This idea can be seen in various applications.

A 4×1 multiplexer (MUX), also known as a data selector, is illustrated in Fig. 1. The value of the output; U, is equal to one of the inputs; I_0 through I_4, selected based on the controls; S_0 and S_1. These values are shown in Table 1.

A decoder with three-state buffers, seen in Fig. 2, selects one of the data inputs; I_0 through I_7, corresponding to the control inputs, D_0 through D_7. These control inputs are determined by a decoder controlled by three variables, as shown in Table 2. Only one data input is transferred to the single output at any given time.

Decoders and multiplexers are important building blocks of computer systems[4,5]. For example, a bus system may be constructed using a system of decoders with three-state buffers or using multiplexers. The single output of each multiplexer (or decoder system) represents one line in the bus. This bus has various applications, such as the system seen in Fig. 3. In this example, the system uses four multiplexers to choose a 4-bit word from one of four memories (A, B, C and D). The selection lines, S_0 and S_1, control the multiplexers so that one memory word is selected, as shown in Table 3.

XOR has applications in logic programming[1,2] and in experimental psychology[3]. In addition, XOR may be used to implement a one-selecting variant of a transition system[2], which may also be represented using the unique existential quantifier.

Corresponding Author: Maher A. Nabulsi, Department of Computer Science, Al-Zaytoonah University of Jordan, Amman 11733, Jordan

Table 1: Output of multiplexer selected from the inputs

S_1	S_0	U
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Table 2: Output of buffer system selected from the inputs

S_2	S_1	S_0	Active output of decoder	U
0	0	0	D_0	I_0
0	0	1	D_1	I_1
0	1	0	D_2	I_2
0	1	1	D_3	I_3
1	0	0	D_4	I_4
1	0	1	D_5	I_5
1	1	0	D_6	I_6
1	1	1	D_7	I_7

Table 3: Memory word selected for bus lines based on the controls

S_1	S_0	Selected memory word
0	0	From Memory A
0	1	From Memory B
1	0	From Memory C
1	1	From Memory D

Table 4: Truth table for the unique existential quantifier and XOR with two elements

P(1)	P(2)	∃!xP(x)	P(1) ⊕ P(2)
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Table 5: Truth table for the unique existential quantifier and XOR with three elements

P(1)	P(2)	P(3)	∃!xP(x)	P(1) ⊕ P(2) ⊕ P(3)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	1

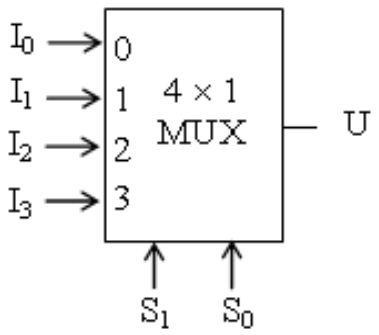

Fig. 1: A 4×1 Multiplexer

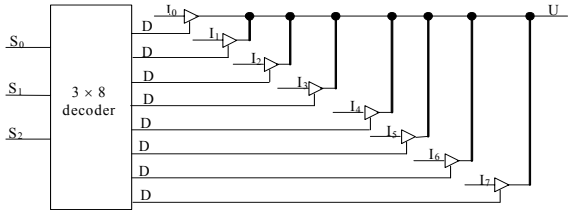

Fig. 2: A decoder with three-state buffers, where the outputs of the decoder (from the top down) are D_0 through D_7

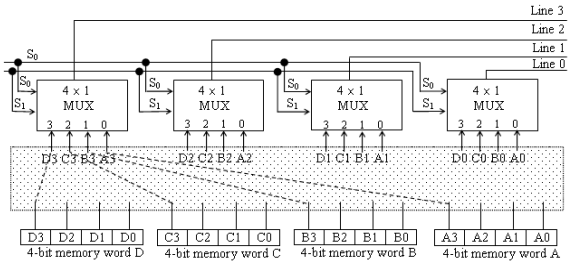

Fig. 3: A memory selection system using multiplexers

The relationship between XOR and the unique existential quantifier can be seen by examining truth tables. First, consider a predicate P defined over a finite universe of one variable (x) with two elements {1, 2}. As seen in Table 4, ∃!xP(x) is equivalent to P(1) ⊕ P(2). However, with a finite universe of three elements {1, 2, 3}, ∃!xP(x) is not equivalent to P(1) ⊕ P(2) ⊕ P(3). As seen in Table 5, they do not have the same value when all three elements are true. Furthermore, ∃!xP(x) is not equivalent to XOR with a number of elements of four or more. This is because XOR is an odd function that has a value of 1 when there is an odd number of elements (possibly three or more) with the value 1. The unique existential quantifier, on the other hand, has a value of 1 only when there is exactly one element with the value 1.

With a function of two variables; x and y, and a universe of two elements; {1, 2}, quantified assertions can be represented using XOR. For example, consider the following two assertions.

$$\forall x \exists! y P(x,y) \Leftrightarrow ([P(1,1) \wedge \sim P(1,2)] \vee [P(1,2) \wedge \sim P(1,1)]) \wedge ([P(2,1) \wedge \sim P(2,2)] \vee [P(2,2) \wedge \sim P(2,1)]),$$

$$\exists! y \forall x P(x,y) \Leftrightarrow ([P(1,1) \wedge P(2,1)] \wedge \sim [P(1,2) \wedge P(2,2)]) \vee ([P(1,2) \wedge P(2,2)] \wedge \sim [P(1,1) \wedge P(2,1)]).$$

Since A ⊕ B is equivalent to (~A∧B)∨(A∧~B), the above assertions can be written as:

$$\forall x\exists!yP(x,y) \Leftrightarrow [P(1,1)\oplus P(1,2)] \wedge [P(2,1)\oplus P(2,2)],$$
$$\exists!y\forall xP(x,y) \Leftrightarrow [P(1,1)\wedge P(2,1)] \oplus [P(1,2)\wedge P(2,2)].$$

RESULTS

The values of ∃!xP(x) and XOR are equivalent for two elements, but not for three or more elements.

The quantified assertions ∀x∃!yP(x,y) and ∃!y∀xP(x,y) on the universe of two elements can be written through XOR and AND functions because the universal quantifier (∀) is equivalent to AND, and the unique existential quantifier is equivalent to XOR.

DISCUSSION

XOR and the unique existential quantifier are interchangeable for two elements. This helps redesign some architecture systems to increase their efficiency. In addition, some logical equations that use quantified assertions can be simplified using AND and XOR.

The idea of the unique existential quantifier is clearly seen in a decoder where only one output is active at any given time, while all other outputs are inactive. The decoder is a part of the multiplexer, and therefore, the idea of the unique existential quantifier can be used in different applications that use decoders and multiplexers. This includes multiple processor and multiple memories connected to bus systems.

CONCLUSIONS

The relationship between ∃!xP(x) and XOR was discussed, and some applications of ∃!xP(x) in computer architecture were presented.

There are several hardware applications for ∃!xP(x) in computer architecture, such as in a multiplexer, in a decoder with three state buffers and in a data bus with multiple memories.

REFERENCES

1. Cho, S.J., U.S. Choi, Y.H. Hwang and H.D. Kim, 2008. Design of new XOR-based hash functions for chache memories. Computers and Math. Appl., 55: 2005-2011. DOI: 10.1016/j.camwa.2007.07.008
2. Fecher, H. and H. Schmidt, 2008. Comparing disjunctive modal transition systems with an one-selecting variant. J. Logic and Algebraic Programming, 77: 20-39. DOI: 10.1016/j.jlap.2008.05.003
3. Grand, C. and R.C. Honey, 2008. Solving XOR. J. Exp. Psy. Anim. Behavior Process., 34: 486-493. http://www.ncbi.nlm.nih.gov/pubmed/18954232
4. Khan, M., 2007. Reversible realization of quaternary decoder, multiplexer, and demultiplexer circuits. Eng. Lett., 15: 203-207. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=27952764&site=ehost-live
5. Rakes, C.D., 1998. CMOS ICsdecoders and multiplexers. Popular Electr., 15: 66-49. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=232579&site=ehost-live