

High Speed Gate Level Synchronous Full Adder Designs

PADMANABHAN BALASUBRAMANIAN[§] and NIKOS E. MASTORAKIS[¶]

[§]School of Computer Science,
The University of Manchester,
Oxford Road, Manchester M13 9PL,
UNITED KINGDOM.

padmanab@cs.man.ac.uk

[¶]Department of Computer Science,
Military Institutions of University Education, Hellenic Naval Academy,
Piraeus 18539,
GREECE.

mastor@hna.gr

Abstract: - Addition forms the basis of digital computer systems. Three novel gate level full adder designs, based on the elements of a standard cell library are presented in this work: one design involving **XNOR** and **multiplexer** gates (XNM), another design utilizing **XNOR**, **AND**, **Inverter**, **multiplexer** and **complex gates** (XNAIMC) and the third design incorporating **XOR**, **AND** and **complex gates** (XAC). Comparisons have been performed with many other existing gate level full adder realizations. Based on extensive simulations with a 32-bit carry-ripple adder implementation; targeting three process, voltage and temperature (PVT) corners of the high speed (low V_D) 65nm STMicroelectronics CMOS process, it was found that the XAC based full adder is found to be delay efficient compared to all its gate level counterparts, even in comparison with the full adder cell available in the library. The XNM based full adder is found to be area efficient, while the XNAIMC based full adder offers a slight compromise with respect to speed and area over the other two proposed adders.

Key-Words: - Combinational logic, Full adder, High performance, Standard cells, and Deep submicron design.

1 Introduction

A binary full adder is often found in the critical path of microprocessor and digital signal processor data paths, as they are fundamental to almost all arithmetic operations. It is the core module used for many essential operations like multiplication, division and addresses computation for cache or memory accesses and is usually present in the arithmetic logic unit and floating point units. Hence, their speed optimization carries significant potential for high performance applications.

A 1-bit full adder module basically comprises of three input bits (say, a , b and cin) and produces two outputs (say, sum and $cout$), where ' sum ' refers to the summation of the two input bits, ' a ' and ' b ', and cin is the carry input to this stage from a preceding stage. The overflow carry output from this stage is labeled as ' $cout$ '.

Many efficient full custom transistor level solutions for full adder functionality have been proposed in the literature [1] – [10], optimizing any or all of the design metrics viz. speed, power and area. In this paper, our primary focus is on realizing

high performance full adder functionality using readily available off-the-shelf components of a standard cell library [11]. Hence, our approach is semi-custom rather than being full custom. This article primarily focuses on the novel design of full adders at the logic level and also highlights a comparison with many other existing gate level solutions, from performance and area perspectives. The inferences from this work may be used for further improvement of full adder designs at the transistor level. Apart from this, this article is also intended to provide pedagogical value addition.

The remaining part of this paper is organized as follows. Section 2 describes the various existing gate level realizations of a 1-bit binary full adder. The three newly proposed full adder designs are mentioned in section 3. Section 4 gives details about the simulation mechanism and results obtained. Finally, we conclude in the next section.

2 Existing Gate Level Adder Designs

The truth table of a binary 1-bit full adder is given in Table 1 and the fundamental equations governing

the full adder's sum and carry outputs are given below. The sum output corresponds to the exclusive-OR operation performed on all the input bits, while the carry output basically conforms to majority logic, in that, if any of the two inputs have a similar logic state, then the carry output is assigned the same state.

$$sum = a \oplus b \oplus cin \quad (1)$$

$$cout = a \cdot b + b \cdot cin + a \cdot cin \quad (2)$$

Table 1. Truth table of a full adder

Inputs			Outputs	
a	b	cin	sum	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2.1 Full adder based on minimum sum-of-products form

A minimum sum-of-products (MSOP) form of a Boolean function can be obtained using a standard two-level logic minimizer: Espresso [12]. Though the logic realization corresponds to an AND-OR two-level logic format, inverting buffers required for the primary inputs have to be separately accounted for. A 1-bit full adder, based on MSOP forms for sum and carry outputs, is shown in fig. 1.

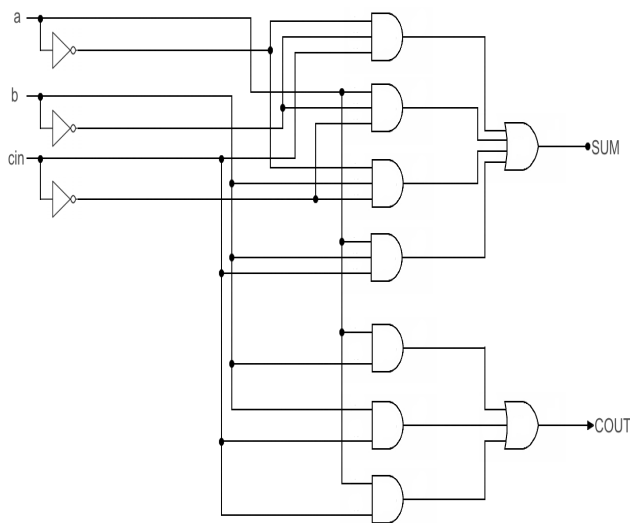


Fig. 1. Full adder realization corresponding to conventional AND-OR logic based on two-level logic minimization, with input inverters

2.2 Full adder implementation using two half adder modules

A traditional full adder implementation based on two half adder modules [13] is shown in fig. 2.

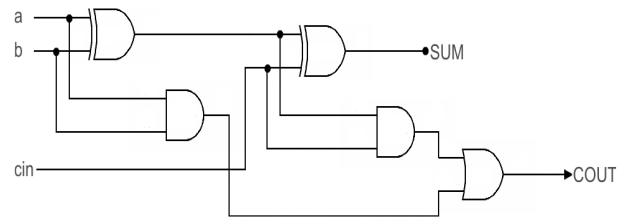


Fig. 2. 1-bit full adder based on two half adders

2.3 Full adder design incorporating logic sharing

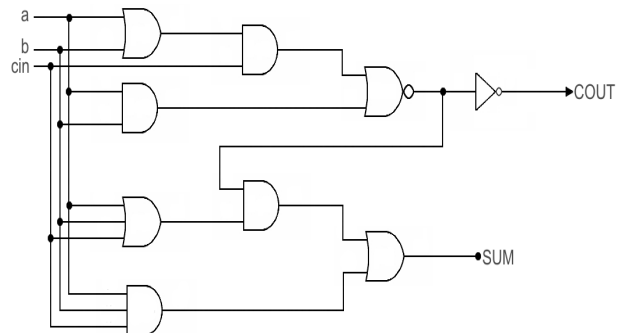


Fig. 3. Full adder design with logic sharing

The full adder design incorporating logic sharing, as described in [14] is depicted by fig. 3. A slight modification is done, in that the NOR gate followed by the NOT gate to realize the sum output in the original circuit has been replaced by an OR gate.

2.3 Full adder realization predominantly using 2:1 MUXes

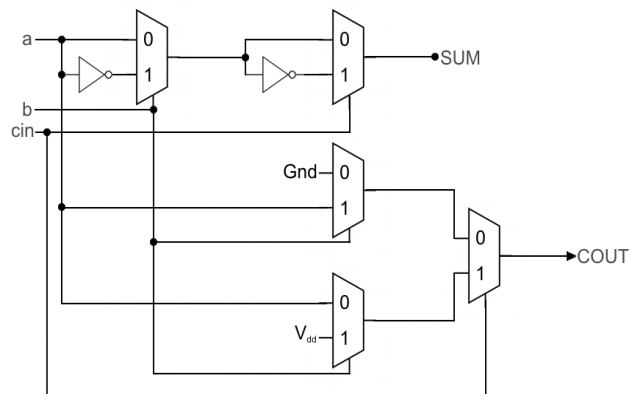


Fig. 4. Full adder realization predominantly based on 2:1 MUXes

