Microelectronics Journal 51 (2016) 99–111

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

High speed multiplexer design using tree based decomposition algorithm

Mohamed Asan Basiri M, Noor Mahammad Sk

Indian Institute of Information Technology Design and Manufacturing, Kancheepuram, Chennai, Tamil Nadu, India

ARTICLE INFO

Article history:
Received 13 February 2015
Received in revised form
22 September 2015
Accepted 23 February 2016
Available online 22 March 2016

Keywords:
Barrel shifter
Multiplexer
Multiply accumulate circuit

ABSTRACT

This paper proposes an effective algorithm to design a larger multiplexer using a tree of smaller multiplexers for a particular user defined library. The proposed algorithm outputs the larger multiplexer into a tree based structure, which gives the scope to pipeline the larger multiplexer as per the requirements of data path design. The experimental result shows that the proposed algorithm gives an efficient multiplexer design with less delay compared to existing algorithms. For example, 48-to-1, 31-to-1 multiplexer designs using the proposed algorithm achieve an improvement of 30.4% and 21.95% in delay compared with designs using 2-to-1 multiplexer tree for 45 nm technology respectively.

1. Introduction

A multiplexer (MUX) is a combinational data path element that selects one of the several inputs as an output using select lines. The number of select lines is n for 2^n inputs. The multiplexer is also called as a data selector. Multiplexers have two sets of inputs. They are data inputs $(D_1, D_2, D_3, \ldots, D_n)$ and selection inputs $(S_1, S_2, S_3, \ldots, S_m)$, where $m = \lceil \log_2 n \rceil$ [1]. A complete multiplexer is the one with $n = 2^m$. Each data input is selected by one distinct binary code in the selection lines. Fig. 1 shows the basic multiplexer structure with n data inputs and m select lines.

An incomplete multiplexer is the one with $2^{m-1} < n < 2^m$. A multiplexer can be designed with several levels. The level of multiplexer M is $1 + \text{max\_}level$, where max_*level* is the maximum level of the multiplexer whose outputs are connected to the inputs of the multiplexer M. A multiplexer which has all the inputs as direct data inputs is said to be in level 1, where max_*level* $= 0$. Fig. 2 shows the multiplexer design with level 1. For example, 64-to-1 multiplexer can be designed with $m = 5$ and $i = 32$ and $j = 64$. The normal approach to design an efficient multiplexer would be enumerating all the possibilities of a given n-to-1 multiplexer and calculate the delay of each possibility to select the best one. Table 1 shows the number of possibilities for various multiplexers. If the n value for n-to-1 multiplexer increases, then the number of possible designs for n-to-1 multiplexer also increased exponentially and hence this will turn out to be NP-complete [9]. For example, the number of possible cases to design the 6-to-1 multiplexer is

(1) using 2-to-1 multiplexers alone
(2) using 3-to-1 multiplexer and others
(3) using 4-to-1 multiplexer and others
(4) using 5-to-1 multiplexer and others
(5) direct 6-to-1 multiplexer

The 6-to-1 multiplexer can be designed in 5 ways by using the above-mentioned possible cases, where 2-to-1, 3-to-1, 4-to-1, and 5-to-1 multiplexers are considered as the auxiliary components for 6-to-1 multiplexer. Similarly, 2-to-1, 3-to-1, 4-to-1, and 5-to-1 multiplexers can be designed in 1, 2, 3, and 4 number of ways respectively. If 6-to-1 multiplexer design using 2-to-1 multiplexers alone (case 1) and direct 6-to-1 (case 5) are not considered, then the number of possible cases will be 3. Here every possible case for all the auxiliary components is considered to design each possible design of 6-to-1 multiplexer. Therefore, the total number of possible designs for the 6-to-1 multiplexer is $2 \times 3 \times 4 = 24 = 4!$, which means that $(n-2)!$ for n-to-1 multiplexer. If the case 1 and 6 is included for 6-to-1 multiplexer design, then total number of possible designs is $4! + 2$. So in general, the number of possible designs of n-to-1 multiplexer is $(n-2)! + 2$. According to the stirling's [2] approximation $n! = \sqrt{2n\pi}(n/e)^n(1 + \Theta(1/n)) = \Theta(k^n)$, where e and k are positive integers. Therefore, the number of possible designs for n-to-1 multiplexer can be written as $\Theta(k^n)$.

If the last possible case of n-to-1 multiplexer is considered, then the direct n-to-1 multiplexer will give worst performance according to the increase in the value of n. The output will be produced after all the n inputs are available. If any one of the input of the direct n-to-1 multiplexer has largest arrival time, then the whole system will be delayed due to that particular input. This can be solved by allowing multiple levels for n-to-1 multiplexer, where the signal line with maximum arrival time is pushed into the last level of the multiplexer. Another problem with direct n-to-1 multiplexer (level is 1) is

http://dx.doi.org/10.1016/j.mejo.2016.02.009
0026-2692/© 2016 Elsevier Ltd. All rights reserved.

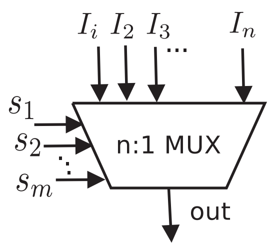

Fig. 1. Basic design of multiplexer.

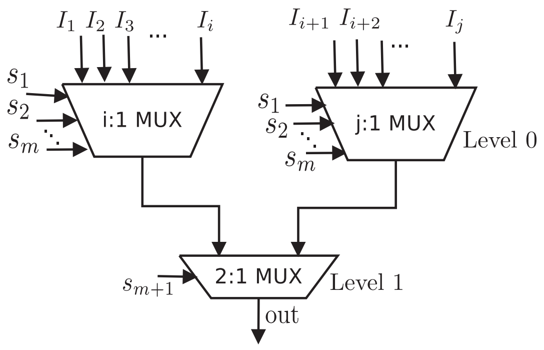

Fig. 2. Design of multiplexer with level 1.

Table 1
Number of possible multiplexer designs.

Multiplexer	Number of possible designs
3-to-1	2
4-to-1	4
5-to-1	8
6-to-1	26

largest clock period, and this can be reduced by pipeline the n-to-1 multiplexer which is made up of smaller multiplexers with multiple levels. Similarly, if n-to-1 multiplexer is designed using 2-to-1 multiplexers alone, then the delay of the design will be increased due to the increase in the level of the 2-to-1 multiplexers tree.

The major objective of this work is to improve the performance of the n-to-1 multiplexer. In general, the performance of a circuit depends on the circuit delay, circuit size (area) i.e., the number of combinational elements and net power requirement. Here circuit delay means that the path from input to output with maximum (worst path) delay in the circuit. The careful optimization in these parameters will ensure the high performance. In this work, the delay minimization is considered as the primary goal.

2. The literature review

Generally, multiplexers are used in the data paths [3] of circuits to route the data. Multiplexers are used in the reconfigurable [4] digital systems to perform different modes. In low power digital system design, multiplexers play the major role. For example, a multiplexer based full adder for low power requirement [5]. The paper [6] discusses about multiplexer based block enabling technique for multiply accumulate circuit design, where the multiplexer is used for low power implementation. Barrel shifters [7] are the efficient shifters used in the digital systems. Here the multiplexers are the elementary components. Shashidhar et al. [8] proposed the decomposition algorithm of a multiplexer. Here the decomposition tree is made up of 2-to-1 multiplexers only. The time complexity to build the decomposition tree for n-to-1 multiplexer with 2-to-1 multiplexers alone is $O(2^s)$, where s is the number of select lines with $n=2^s$. This paper suggests to push the select lines with maximum arrival time towards the root (output node) of the tree. The total number of 2-to-1 multiplexer nodes in n-to-1 multiplexer $=1+2+2^2+\cdots+2^{s-1}=2^s-1$.

Subhasish et al. [9,10] discuss the tree based area minimization algorithm to implement an efficient multiplexer design. Here the library of components is considered to build the required multiplexer. The required n-to-1 multiplexer is decomposed into tree form of pairs and $\lceil n/2 \rceil$ pairs will be produced. The decomposition process for n-to-1 multiplexer will be continued from $(1, n-1)$ to $(n-\lceil n/2\rceil-1, \lceil (n+1)/2 \rceil)$, where n is odd. In the case of n as even, the decomposition can be started from $(1, n-1)$ to $(n-\lceil n/2 \rceil, \lceil (n+1)/2 \rceil)$. Here the reverse order of the pairs $(n-i, i)$ is not considered, because the pair $(n-i, i)$ will give the same design as the pair $(i, n-i)$. In the next step, selection of pair will be carried. If both the components $n1$ and $n2$ of the pair $(n1, n2)$ are available in the library, then the corresponding multiplexers will be selected. If any one of the components is not available in the library, then the corresponding component will be decomposed further. In each selection, the local cost and the global cost of the design are calculated. Fig. 3 shows the decomposition tree for 6-to-1 multiplexer, where the dark nodes are called as OR nodes and white nodes are considered as AND nodes. The maximum height of the tree includes $(n-2)$ levels of AND/OR nodes for a n-to-1 multiplexer. In this case, maximum height of the decomposition tree of 6-to-1 multiplexer includes 4 levels of AND/OR nodes. The number of AND nodes in the tree will give the total number of possible designs of the particular n-to-1 multiplexer. Hence the time complexity of this algorithm depends on the total number of AND nodes.

Fig. 4 shows the number of AND nodes in the decomposition tree form of n-to-1 multiplexer using [9], where the maximum height of the tree includes $(n-2)$ levels of AND nodes. The n-to-1 multiplexer is decomposed into $(\lceil n/2 \rceil)$ AND nodes in the 1-st level. In the next step, AND nodes are changed into the OR nodes according to the algorithm. Then the OR nodes are further decomposed. So the number of AND nodes in level $2(T_{and,2})$ is

$$\begin{aligned}
&= \left\lceil \frac{n}{4} \right\rceil + \left(\left\lceil \frac{n}{4} \right\rceil + \frac{1}{2} \right) + \left(\left\lceil \frac{n}{4} \right\rceil + \frac{1}{2} + \frac{1}{2} \right) + \cdots + \left\lceil \frac{n-3}{2} \right\rceil + \left\lceil \frac{n-2}{2} \right\rceil + \left\lceil \frac{n-1}{2} \right\rceil \\
&\approx \frac{n}{4} + \left(\frac{n}{4} + \frac{1}{2} \right) + \left(\frac{n}{4} + \frac{1}{2} + \frac{1}{2} \right) + \cdots + \left(\frac{n-3}{2} \right) + \left(\frac{n-2}{2} \right) + \left(\frac{n-1}{2} \right) \\
&= \frac{n}{4} + \left(\frac{n}{4} + \frac{1}{2} \right) + \left(\frac{n}{4} + \frac{1}{2} + \frac{1}{2} \right) + \cdots + \left(\frac{n}{4} + \frac{n-6}{4} \right) + \left(\frac{n}{4} + \frac{n-4}{4} \right) + \left(\frac{n}{4} + \frac{n-2}{4} \right)
\end{aligned}$$

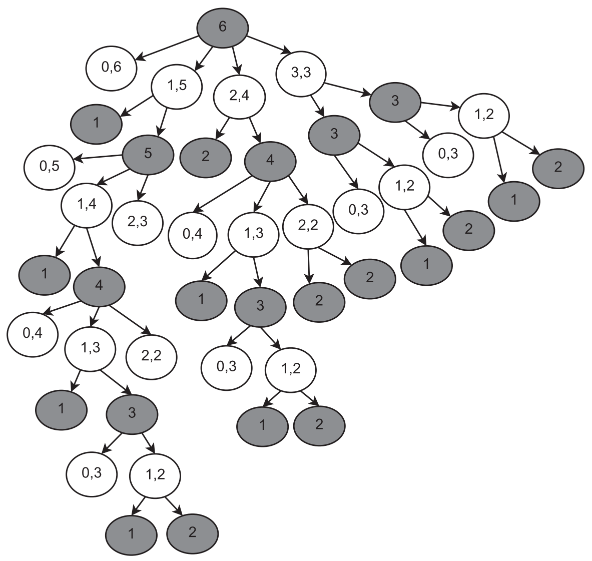

Fig. 3. Decomposition tree form of 6-to-1 multiplexer using [9].