Comparative Analysis of 8-bit Adders for Embedded Application

1Kunjan D. Shinde  
2nd Year, M.Tech. in Digital Electronics,  
Dept. of Electronics & Communication Engineering  
SDMCET, Dharwad - 02  
Karnataka, INDIA  
Kunjan18m@gmail.com

2Jayashree C. Nidagundi  
Assistant Professor  
Dept. of Electronics & Communication Engineering  
SDMCET, Dharwad - 02  
Karnataka, INDIA  
jayaprajwal@rediffmail.com

Abstract—Digital computations and processing is involved in each and every embedded and non-embedded device, such applications and devices have arithmetic logic unit, adders are most important and essential block of these system, design and selection of adders plays a very important role, this paper presents a comparative study of five different adders like Ripple Carry Adder, Carry Skip Adder, Carry Lookahead Adder, Kogge Stone Adder, with performance metrics as delay and area. From the results it is clear that Kogge Stone Adder provides a less delay with a compromise in area.

Keywords—Ripple Carry Adder, Carry Skip Adder, Carry Lookahead Adder, Kogge Stone Adder, Delay, No. of slices, No. of LUTs.

I. INTRODUCTION

Adders are fundamental and most essential blocks in every digital system, design of effective and reliable adders plays an important role[1], as the technology is scaled down the complexity in the design increases with some reduction is the performance of the adders, in this paper a different set of adders like Ripple Carry Adder, Carry Skip Adder, Carry Lookahead Adder, Kogge Stone Adder are designed using Verilog coding and the performance of the adders are tabulated.

A simple adder performs the addition of given two numbers and the result is sum of those two numbers. Adders can be implemented in different ways using different technologies at different levels of architectures. [2] Design of high speed and reliable adders is the prime objective and requirement for embedded applications as the technology is scaled down. Binary addition is a fundamental operation in most digital circuits. There are a variety of adders, each has certain performance. Adder is selected depending on where the adder is to be used.

Ripple Carry Adder (RCA)

A combinational circuit that adds two bits is called a half adder, A combinational circuit that adds two bits along with the carry generated/ carry in bit is called as full adder. The ripple carry adder is constructed by cascading full adder blocks in series, Fig-1.a and Fig. 1.b. shows the block diagram and gate level schematic of 1-bit full adder. To obtain the 8-bit Ripple Carry Adder it requires 8 stages of 1-bit full adders, the carryout of one stage is fed directly to the carry-in of the next stage which is shown in Fig. 2. For an n-bit parallel adder, it requires n full adder[2][5].

Drawbacks of Ripple Carry Adder

- Not very efficient when large bit numbers are used
- Delay increases linearly with the bit length
**Equation of Ripple Carry Adder**

\[ S_i = A_i \oplus B_i \]  

(1)

The worst case of delay might happen when the inputs at 1st stage are at logic ‘1’ and/or any one of the input is at logic ‘1’ an at this time the carry in is also at logic ‘1’, at this state the carry propagated is carried out till the last stage and the delay in generating proper result is large and hence it the largest carry propagation path that can occur in the RCA.

**CARRY SKIP ADDER (CSA)**

The design of a carry-skip adder is based on the classical definition propagate signals as follows

\[ P_i = A_i \oplus B_i \]  

(2)

where \( P_i \) is the propagate signal and \( A_i \) and \( B_i \) are the input operands to the \( i \)th adder cell, \( C_i \) is the carry input to the \( i \)th cell. The carry output from the \( i \)th adder cell is expressed as

\[ C_{i+1} = (A_i \text{ AND } B_i) \text{ OR } (P_i \text{ AND } C_i) \]  

(3)

Figure 3 shows the block diagram of 8-bit Carry Skip Adder, the block diagram is obtained from the equations mentioned above.

**Equations of Carry Lookahead Adder**

\[ P_i = A_i \text{ OR } B_i \]  

(4)

\[ G_i = A_i \text{ AND } B_i \]  

(5)

\[ S_i = A_i \oplus B_i \oplus C_i \]  

(6)

\[ C_{i+1} = G_i \text{ OR } (P_i \text{ AND } C_i) \]  

(7)

Where \( P_i \) is propagate signal and \( G_i \) is generate signal and \( S_i \) is the final sum output. The carry output of \( i \)th adder is obtained by equation (7). [3]

The above two equations can be written in terms of two new signals and which are shown in Figure 4:

![Figure 4. Processing Stages of Carry Lookahead Adder](image)

**KOGGE STONE ADDER**

KSA is a parallel prefix form carry look ahead adder [1] It generates carry in O (logn) time and is widely considered as the fastest adder and is widely used in the industry for high performance arithmetic circuits. [1] In KSA, carries are computed fast by computing them in parallel at the cost of increased area. [1]

- Prefix: The outcome of the operation depends on the initial inputs.
- Parallel: Involves the execution of an operation in parallel. This is done by segmentation into smaller pieces that are computed in parallel.
As mentioned earlier the Kogge Stone Adder is a parallel prefix form of Carry Look ahead Adder and the figure 4 shows the processing stages of Carry Look ahead adder and its equation for implementing/designing the same.

The complete functioning of KSA can be easily comprehended by analyzing it in terms of three distinct parts [4][7][8]:

1. **Pre processing**
   
   $P_i = A_i \text{ XOR } B_i$  
   
   $G_i = A_i \text{ AND } B_i$  

2. **Carry look ahead network**
   
   $P_{i:j} = P_{i:k+1} \text{ AND } P_{k:j}$  
   
   $G_{i:j} = G_{i:k+1} \text{ OR } (P_{i:k+1} \text{ AND } G_{k:j})$

3. **Post processing**
   
   $S_i = p_i \text{ XOR } C_{i-1}$

**RESULTS AND DISCUSSIONS.**

Figure 8 shows the simulation results for all the adders give same inputs to every adder & the result of which is also the same and hence verifying the functionality of the adders mentioned above in this paper.

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>No. of Slices</th>
<th>No. of 4 input LUTs</th>
<th>No. of Bounded IOBs</th>
<th>Delay</th>
<th>Macro statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple Carry Adder (Data flow)</td>
<td>10/768</td>
<td>17/1536</td>
<td>26/98</td>
<td>17.102ns</td>
<td>1-bit Xor3-08</td>
</tr>
<tr>
<td>Ripple Carry Adder (Structural)</td>
<td>10/768</td>
<td>17/1536</td>
<td>26/98</td>
<td>17.102ns</td>
<td>1-bit Xor3-08</td>
</tr>
<tr>
<td>Carry Skip Adder (Data flow)</td>
<td>10/768</td>
<td>17/1536</td>
<td>26/98</td>
<td>16.941ns</td>
<td>1-bit Xor - 13</td>
</tr>
<tr>
<td>Carry Skip Adder (Structural)</td>
<td>10/768</td>
<td>17/1536</td>
<td>26/98</td>
<td>16.941ns</td>
<td>1-bit Xor - 16</td>
</tr>
<tr>
<td>Carry Lookahead Adder (Data flow)</td>
<td>09/768</td>
<td>15/1536</td>
<td>26/98</td>
<td>15.858ns</td>
<td>1-bit Xor2-15</td>
</tr>
<tr>
<td>Kogge Stone Adder (Data flow)</td>
<td>20/768</td>
<td>34/1536</td>
<td>26/98</td>
<td>17.065ns</td>
<td>1-bit Xor2-16</td>
</tr>
<tr>
<td>Kogge Stone Adder (Structural)</td>
<td>20/768</td>
<td>34/1536</td>
<td>26/98</td>
<td>15.776ns</td>
<td>1-bit Xor2-16</td>
</tr>
</tbody>
</table>
Table 2. Comparative analysis of different adders with performance measure as delay

<table>
<thead>
<tr>
<th>Adder</th>
<th>Logic Delay</th>
<th>Route Delay</th>
<th>Total Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple Carry Adder (Data flow)</td>
<td>09.456 ns</td>
<td>07.424 ns</td>
<td>16.880 ns</td>
</tr>
<tr>
<td>Ripple Carry Adder (structural)</td>
<td>09.456 ns</td>
<td>07.646 ns</td>
<td>17.102 ns</td>
</tr>
<tr>
<td>Carry Skip Adder (Data flow)</td>
<td>09.456 ns</td>
<td>07.485 ns</td>
<td>16.941 ns</td>
</tr>
<tr>
<td>Carry Skip Adder (Structural)</td>
<td>09.456 ns</td>
<td>07.485 ns</td>
<td>16.941 ns</td>
</tr>
<tr>
<td>Carry Lookahead Adder (Data flow)</td>
<td>08.977 ns</td>
<td>06.881 ns</td>
<td>15.858 ns</td>
</tr>
<tr>
<td>Kogge Stone Adder (Data flow)</td>
<td>09.456 ns</td>
<td>07.609 ns</td>
<td>17.065 ns</td>
</tr>
<tr>
<td>Kogge Stone Adder (Structural)</td>
<td>08.977 ns</td>
<td>06.799 ns</td>
<td>15.776 ns</td>
</tr>
</tbody>
</table>

Figures from Fig. 9 to Fig. 15 give the Top Level Schematic Implementation result of various adders mentioned above.
Figure 1. Technology Schematic of Carry Skip Adder (Structural Modeling)

Figure 2. Technology Schematic of Carry Lookahead Adder (Data Flow Modeling)

Figure 3. Technology Schematic of Kogge Stone Adder (Data Flow Modeling)

Figure 4. Technology Schematic of Kogge Stone Adder (Structural Modeling)
CONCLUSION
From the simulation results the functional verification of the adders is done and from the synthesis report it is observed that, the Kogge Stone Adder gives better results compared to the other adders implemented in this paper, the area for the KSA is larger than RCA, CSA and CLA, from the top level schematic of all the above adders it is clear that the area for RCA, CSA and CLA the device utilization is same but the way they are implemented is different and the same is observed KSA adder design.

ACKNOWLEDGMENT
The Authors would like to thank the management and the Principal of SDMCET Dharwad for providing all the support and we would also like to thank Dept. of Electronics and Communication Engineering, SDMCET, for all the resources & support provided.

REFERENCES

AUTHORS
Mr. Kunjan D. Shinde received B E degree in Electronics & Communications Engineering from university of Visveswaraya Technology, Belgaum and pursuing M.Tech. in Digital Electronics from University of Visveswaraya, Belgaum. His research interests include VLSI design, Error Control Coding, Robotics and Digital system design.

Mrs. Jayashree C. Nidagundi is with SDMCET, Dharwad, Karnataka, India. She is serving as Assistant Professor in the Department of Electronics & Communications Engineering. Her research interests include VLSI design, Error Control Coding and Digital Circuit Design.