REALIZATION OF TREE MULTIPLIERS AND THEIR PERFORMANCE EVALUATION

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF TECHNOLOGY
IN
VLSI DESIGN & EMBEDDED SYSTEMS

Submitted by:

RUPENDRA SINGH

2K18/VLS/14

Under the supervision of

Dr. NEETA PANDEY
Professor

ELECTRONICS & COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

2018-2020

ELECTRONICS &COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Rupendra Singh, Roll No. 2K18/VLS/14 student of M.Tech (VLSI & Embedded systems), hereby declare that the work presented in this thesis designated "**Realization of Tree Multiplier and Their Performance Evaluation"** is done by me and submitted to the Department of Electronics and Communication Engineering, Delhi Technological University, Delhi in fractional fulfillment of the prerequisite for the award of the degree of Master of Technology.

This is an original research work and not copied from any source without acknowledge them with proper citation and has not previously published anywhere for the award of any Degree, Diploma Associate ship, Fellowship or other similar title or recognition.

Place: Delhi **(Rupendra Singh)**

Date:

ELECTRONICS & COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "**Realization of Tree Multiplier and Their Performance Evaluation**" which is submitted by **Rupendra Singh, 2K18/VLS/14**, to the Department of Electronics & Communication Engineering, Delhi Technological University, Delhi in partial fulfillment of the prerequisite for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date**:**

PROF. NEETA PANDEY

Department of ECE

Delhi Technological University

ACKNOWLDEMENTS

It gives me immeasurable pleasure to express my deepest sense of gratitude and sincere appreciation to my supervisor, Professor Neeta Pandey, who has the substance of a intellect. She persuasively encouraged and guided me to be professional and do the work in a proper manner. The objective of this project would not have been completed without her persistent help. Her useful suggestions during this whole work and supportive behavior are sincerely acknowledged.

I would like to acknowledge the support of my family and friends. They helped me a lot directly and indirectly during this project work.

Date: **RUPENDRA SINGH**

2K18/VLS/14

ABSTRACT

Multipliers are the most vital part of any computational applications for real time data processing systems. Hence designers tries to make an efficient multiplier design on the basis of trade-off between the design constraints i.e. speed, power and area. In this project we realize tree multipliers (Wallace Tree and Dadda multiplier) and evaluate their performance using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The Tree multiplier architectures are designed in three stages which are, partial product generation, their reduction and the final addition stages. Here in partial product reduction stage, for the reduction of partial products, m:n compressors are used. For the final addition stage different adder designs are used. The main objective of this work is to implement different designs of adders and use them in Tree multiplier to investigate the better design between Wallace tree and Dadda multiplier. After that 4:2 compressor are used in reduction stage and implement different designs by using this reduction stage. A new design of tree multiplier is proposed and compare with other existing designs.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

RCA – Ripple Carry Adder

CLA – Carry Look Ahead Adder

KSA – Kogge Stone Adder

BKA – Brent Kung Adder

PPA – Parallel Prefix Adder

PPs – Partial Products

CHAPTER 1

Introduction

1.1 Background

Nowadays, there is tremendous demand for low-power, less area and high speed computational applications for real time data processing systems. These computational applications require sophisticated data acquisition systems and hardware implementing complex arithmetic operations, in particular fast adders and multipliers. The design constraints (area, power and delay) are the limiting factors for a design. Though, several options for adders and multipliers are available in literature, there is still scope to increase the performance of these arithmetic blocks. The multiplier may be realized simply by performing shift and add operations. Such multipliers are serial in nature, use smaller hardware and have slow response. Tree multipliers give faster response and primarily use three stages to compute the multiplication. These three stages are partial product generation, partial product reduction and final addition, Dadda [1] and Wallace [2] multipliers belong to the latter category. In Partial Product Reduction Stage, Wallace Tree used "As soon as Add" strategy i.e. add as many partial products as possible to reduce the levels whereas Dadda Multiplier relies on "As late as Possible" strategy i.e. use minimum reduction essential at each level. Therefore, Dadda multiplier needs lesser additions as compare to Wallace Tree [1] leading to reduced half adders and full adders. However, the designing of Dadda Multiplier [1] is complex as compared to Wallace tree [1]. The levels of reduction are same in both multipliers.

Researchers have proposed various realizations of Dadda multipliers [3-16]. Modified Booth Algorithm is used in the generation of partial products of Dadda multiplier [3] and the comparison with Modified Booth Wallace Multiplier shows that former has reduced area and better speed. The Carry Select Adder with binary excess 1 converter is used in [4] in reduction stage while decomposition logic is used in [5] wherein partial products are divided into smaller sub groups that are used as smaller multipliers and the collective outputs of these multipliers gives final results. Here, additional circuitry is needed for final accumulation leading to larger implementation area. A hybrid multiplier design founded on decomposition logic in reduction stage uses Dadda and Wallace algorithm for different sub groups [6]. An Efficient Charge Recovery Logic (ECRL) is used for implementation

of full adders based Vedic-Dadda hybrid multiplier design [7]. Swing Restored Complementary Pass-transistor Logic (SR-CPL) and Dual Pass-Transistor Logic (DPL) are used in reduction and final stage [8]. A pipelined Dadda multiplier design with pipelined carry look ahead adder in final stage is presented in [9] which possesses less delay of each pipelined stage and less latency of entire multiplier.

To reduce the partial products in reduction stage of tree multiplier, different types of m:n compressors can be used. Reduction stage consumes most of the power of multiplier, to reduce it compressors are used. Compressors can reduce delay of reduction stage also.

A 4:2 compressor is one of the famous compressor used in Dadda multiplier. There are different implementation of compressor design i.e. XOR-XNOR based compressors, XOR and Mux based compressors, MUX based compressors etc.[17-19]. A fully Mux based 4:2 compressors utilized in reduction stage of dada multiplier design demonstrate 89% improvement in PDP [10].

A 8x8 Dadda multiplier design with high speed whch used 4:2 compressors in Dadda reduction stage is suggested in [11], exhibits less delay and power delay product (PDP). In [12] approximate 4:2 compressors were used in 32-bit Dadda multiplier depicts less power consumption and delay.

Parallel prefix adders (PPAs) have less area than other addr designs like RCA and CLA. These adders faster in nature. Therefore researchers tried to improve the Dadda multiplier design by introducing parallel prefix adders namely BKA [13], Sklansky tree adder [14] and KSA [15] in final addition stage. In [15] different designs of dada multiplier are compared and it shows that design with KSA has highest speed and BKA has less power dissipation and area.

A 16x16 Dadda multiplier design based on 4:2 compressors with KSA in final stage shows better speed as compare to other dadda multiplier designs [16].

1.2 Objective

The objective of this work is to realize tree multipliers using different elements in Partial Product Reduction and Final Addition stage and evaluate the performance.

1.3 Organization

The work in this thesis is organized in five chapters including this chapter.

Chapter 2 comprises of discussion on various adders namely Ripple carry adder, carry

look ahead adder, Brent-Kung and Kogge-Stone adders.

Chapter 3 describes Wallace tree and Dadda multipliers. The performance of these multipliers is compared on the basis of adders employed in the final addition stage. The addition of partial products is performed using half and full adders for fairer comparison.

Chapter 4 gives realization of Dadda multiplier which uses 4:2 compressors in the reduction stage. Subsequently a new realization is given that uses Brent-Kung adder in final addition stage. Its performance is compared with Dadda multipliers employing Ripple carry adder (RCA), Carry look ahead (CLA) adder, Kogge-Stone adder and Brent-Kung adder) for final stage addition.

Chapter 5 consists of the conclusion and the future scope of the work.

All the simulations are carried out using Vivaldo software and performance is compared on the basis of number of Look Up Tables (LUTs) and slices used, power consumption, delay and Power Delay Product (PDP).

CHAPTER-2

Traditional Adders

Adders are used in almost every processing task and unit such as in Arithmetic Logic Unit (ALU) in computers [22], ECC Processor Design [23], Communication System and many other applications.

Various realizations of adders are available in literature which differ in the way carry is propagated among stages. In this chapter the Ripple Carry Adder (RCA) and Carry Look Ahead (CLA) adder are described first followed by Parallel Prefix Adders namely Brent-Kung adder (BKA) and Kogge-Stone Adder (KSA). All the adders are implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The performance is measured in terms of the area, delay and power consumption of the synthesized design.

2.1 Ripple Carry Adder

A full adder adds two 1-bit numbers so to add N-bit numbers, full adders are to be cascaded in serial fashion. Therefore, there must be N number of full adders for N-bit parallel adder. The full adders are cascaded in a structured form so that the carry out of a full adder behaves as the carry in of the succeeding next most significant full adder. Thus carry out of full adder is rippled into the next stage and the adder so obtained is known as Ripple Carry adder. An N-bit RCA is shown in Fig. 2.1 and the governing Boolean equations are given by (2.1) where A_i and B_i represent i^{th} bit of the input numbers. Cin_i and $Cout_i$ represent carry input to i^{th} full adder. It may be noted that $Cin_{i+1} = Cout_i$ (i = 0,..,N-2).

$$S_i = A_i \oplus B_i \oplus Cin_i \quad (2.1)$$

$$Cout_i = Ai.Bi + Ai.Cini + Bi.Cin_i \quad (2.2)$$

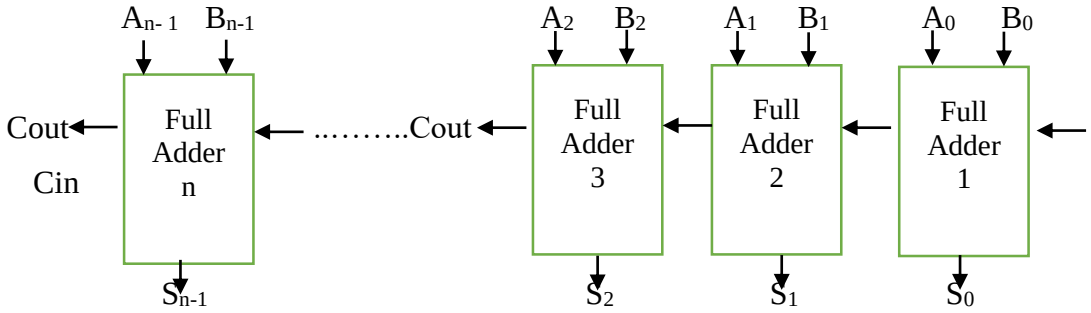


Fig. 2.1 N-bit Ripple Carry Adder

Ripple carry adder utilizes the lowest chip area if compare to other designs. It is a structured design hence designing for different word size is easy. It suffers from a very long propagation chain, that makes the worst case delay largest if compared with other implementations. So there is a trade-off between area and delay. The advantage, however,

is simplest design.

A 32 bit RCA is implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The RTL schematic and synthesized designs are shown in Figs. 2.2 and 2.3 respectively. The design uses 30 LUTs. The delay, power and PDP are found to be 13.4 ns, 0.243 W and 3.256 nJ respectively.

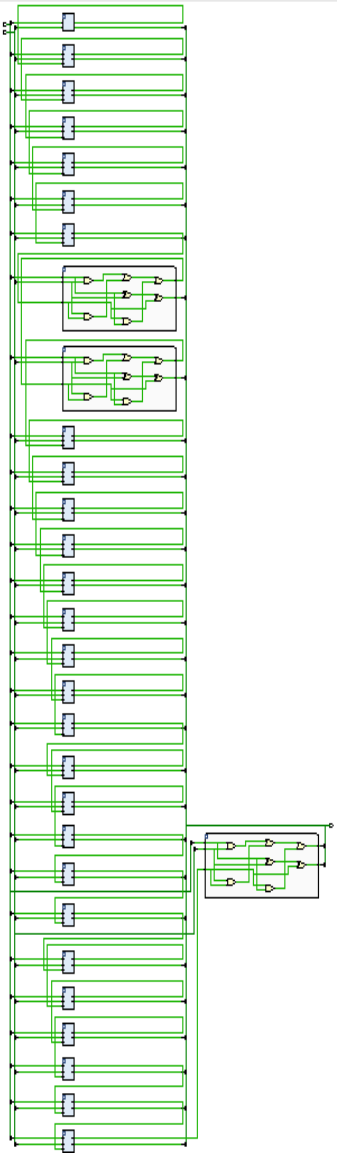


Fig. 2.2 RTL Schematic of 32 bit RCA

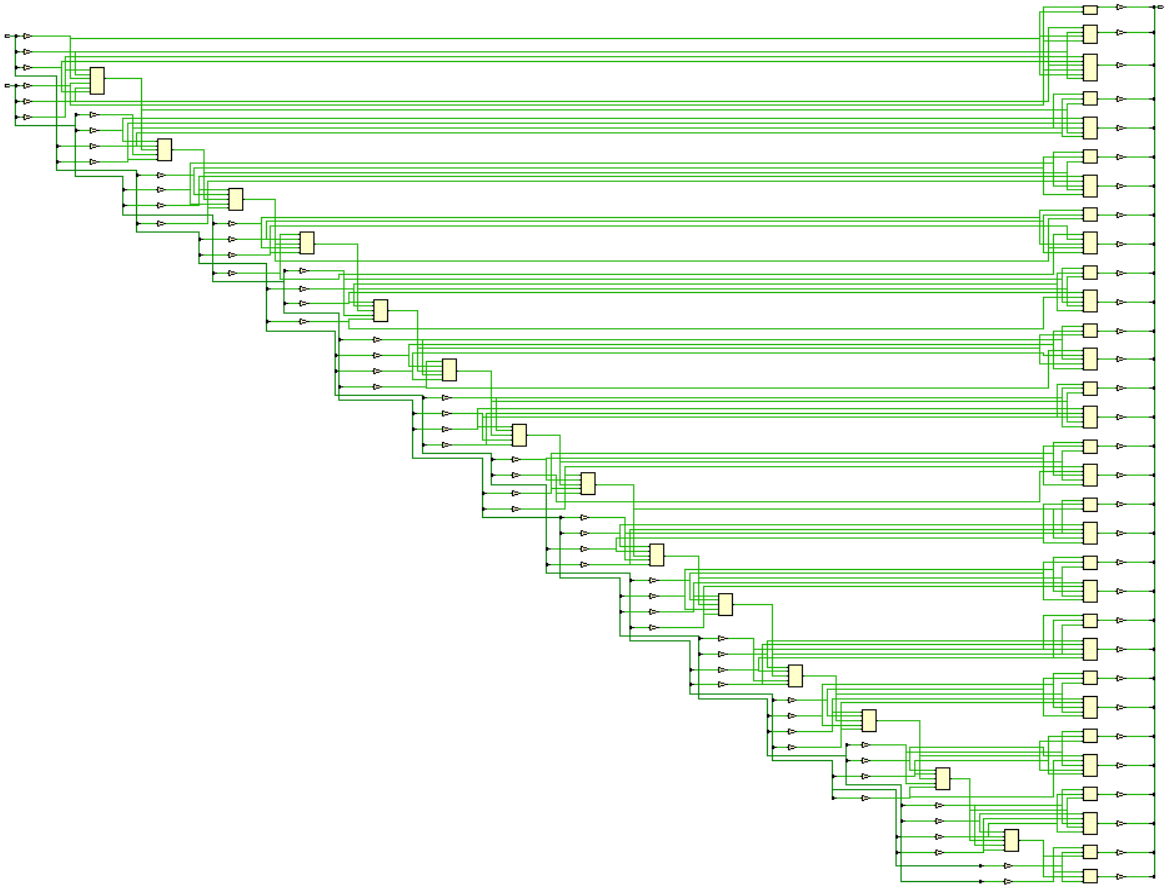


Fig. 2.3 Synthesized design of RCA

2.2 Carry Look Ahead Adder

Carry look ahead adder (CLA) is a parallel adder which has high speed than ripple carry adder. It reduces the propagation delay on the cost of hardware complexity. Hence it takes more area than RCA and also costlier. In carry look ahead adder, carries are generating early of time therefore delays produced in the computation process is less as compare to RCA.

In carry look ahead adder the calculation process can also be presented in two level logic as shown in Fig. 2.4. Here two variables are define as carry Propagation (P) and Generation (G) for each bit. If P = 1, the Cout of present bit is the same as its Cin, while if P = 0, the Cout won't propagate its Cin, that mean Cout is independent of Cin. G = 1 means the carry will be generated regardless of its Cin. The calculation can be formulated as follows:

$$Pi = Ai \oplus Bi \quad (2.3)$$

$$Gi = Ai \oplus Bi \quad (2.4)$$

$$Si = Pi \oplus Gi \quad (2.5)$$

$$Ci+1 = Gi + (Pi \oplus Ci) \quad (2.6)$$

Where Pi is a carry propagator and it is related with the propagation of carry from Ci to

Ci+1. Gi is a carry generate which produces the carry when both Ai, Bi are one. Here Ci+1 is the carry out for present addition.

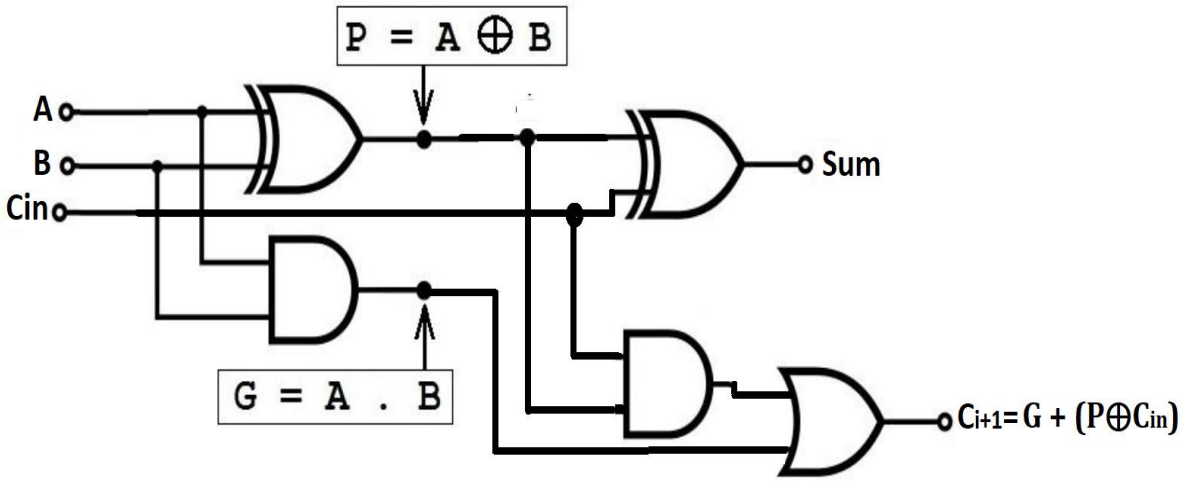


Fig. 2.4 Full Adder with Generate and Propagate signal

Let's assume that the Inputs are A_0, B_0, C_{in}, A_1, B_1, A_2, B_2, A_3 and B_3 the subsequent carries C_1, C_2 and C_3 are computed using the generation and propagation signals as given below.

$C_1 = G_0 + P_0.C_0$ (2.7)

$C_2 = G_1 + P_1.G_0 + P_1.P_0.C_0$ (2.8)

$C_3 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.C_0$ (2.9)

If a 4 bit CLA Generator is used then the Group Generate (G3:0) and Group Propagate (P3:0) can be expressed as given in equation 2.10 and 2.11.

$G3:0 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0$ (2.10)

$P3:0 = P_3.P_2.P_1.P_0$ (2.11)

These carries are produced with help of a CLA Block. The generated carries are send to the relevant Full Adders. A 4 bit CLA is Shown in Fig. 2.5

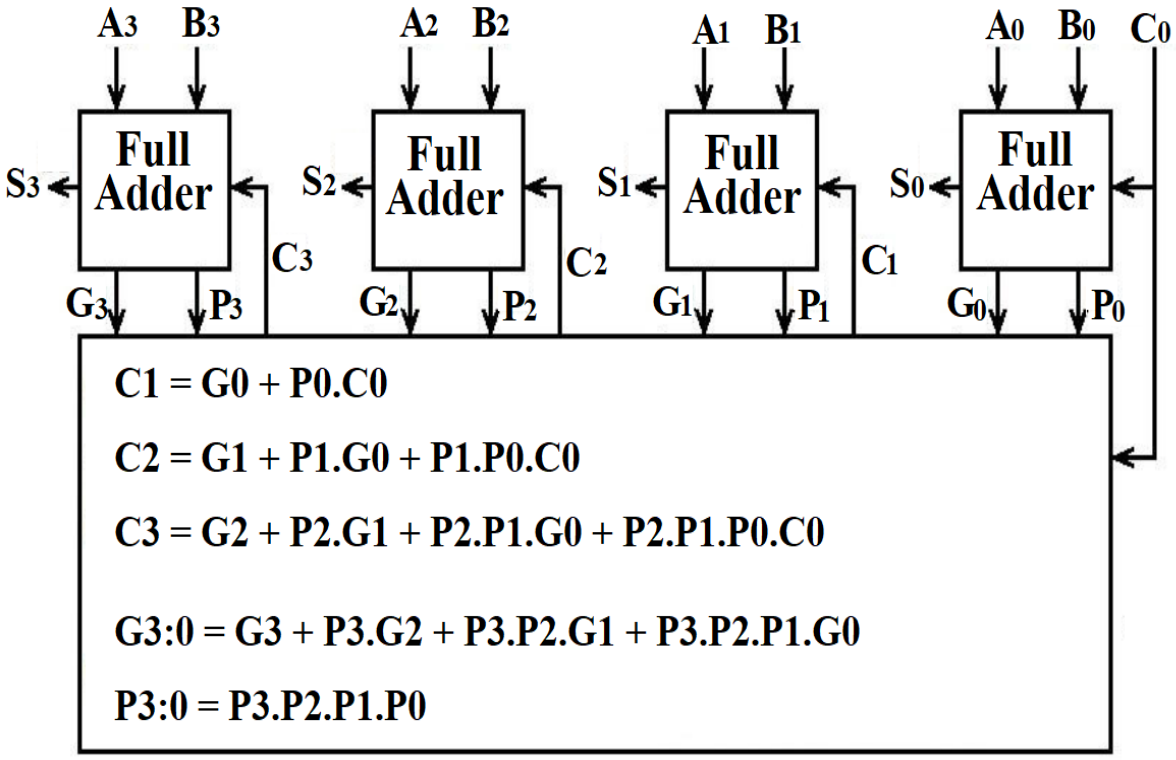


Fig. 2.5 4-bit Carry Look-Ahead Adder Block

The propagation delay of a CLA design can be greatly reduced using the P and G computation network. Therefore, CLAs are the most natural approaches in high performance adders. Carry look ahead adder uses tree configuration topologies to formalize the representation of divide-and-conquer approach of adder design. Hence, multi-level CLA, which exploits parallel carry computation, is achievable.

A 32 bit CLA is implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The RTL schematic and synthesized designs are shown in Figs. 2.6 and 2.7 respectively. The design uses 68 LUTs and. The delay, power and PDP are found to be 8.852 ns, 0.249 W and 2.204 nJ respectively.

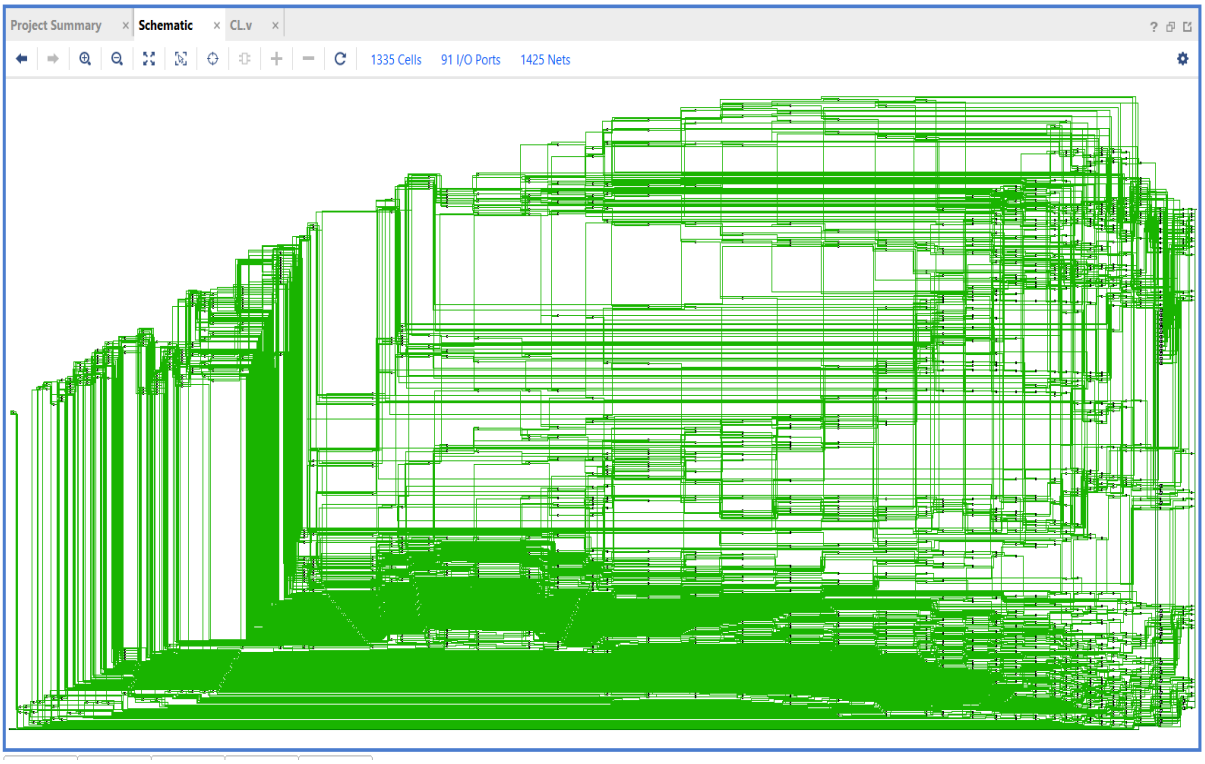


Fig. 2.6 RTL Schematic of 32 bit CLA

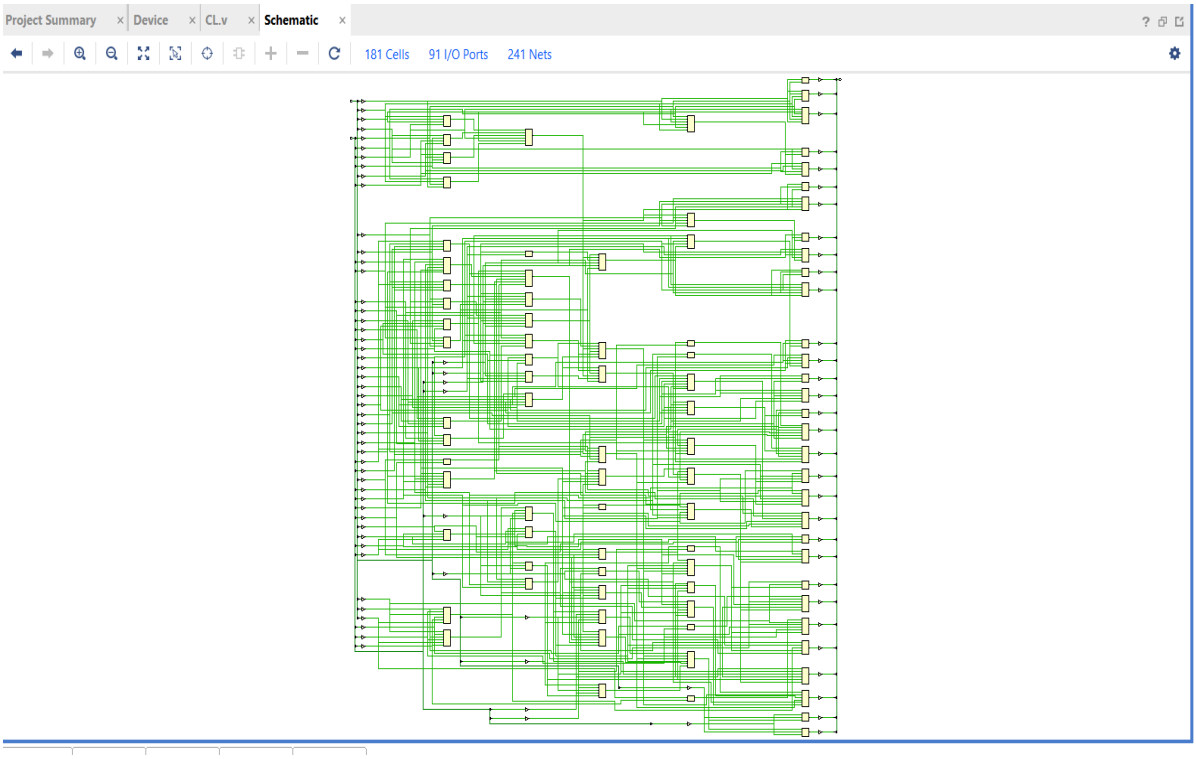


Fig. 2.7 Synthesized Design of 32 bit CLA

2.3 Parallel Prefix Adder

When the carry look ahead adders is design with more number of bits for example N > 16 bits, critical delay becomes dominant in the adders due to the carry delay propagation through the blocks of look ahead increase. This increased delay can easily diminished by surpassing the blocks look ahead. Usually a multi stage tree of look ahead structures can be created in order to achieve a delay that is limited with $\log_2 N$, where N is the number of bits. These multi stage structured adders are generally known as tree adders or multi stage look ahead adders or logarithmic adders or parallel prefix adders.

There are many ways to build parallel prefix adders based on the number of wiring between the levels, number of the fan out from each gate, total logic gates involved and the number of levels of the logic. Parallel prefix adders (PPA) are mostly used in high performance computational circuits since these are faster adders. The two elementary tree adders are BKA and KSA.

BKA [25] uses an optimum number of stages but it has disproportionate loading on all transitional stages. It uses less number of propagate and generate signals compared to KSA. Therefore the cost of complexity is less but the gate level of depth is high. It consumes less power but delay is more than KSA with high speed.

The process of parallel prefix adder is done in three stages, which are:

1. Pre processing
2. Carry generation stage
3. Post Computation

Carry propagation and carry generation signals are processed in pre computation stage for each pair of input bits. In carry generation stage, carry signals are computed in parallel using the Group Generate and Group Propagate corresponding to each bit. In the final stage, sum bits are obtained from the computation of the carry bit and the propagate signal. These stages are shown in the fig. 2.8.

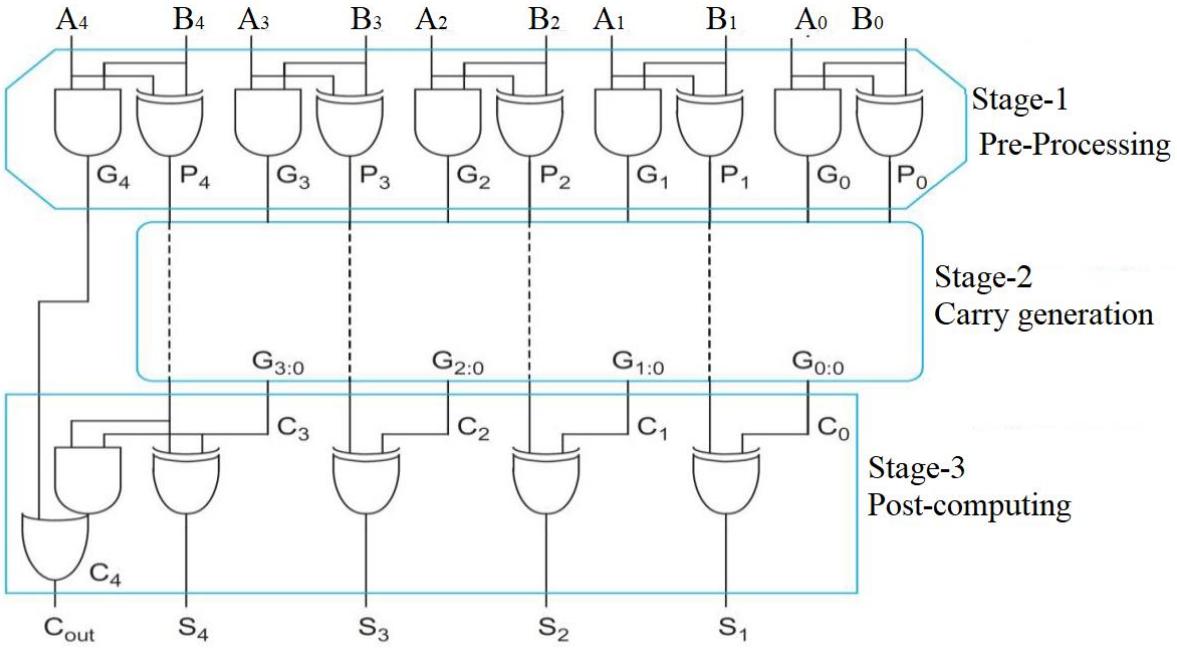


Fig. 2.8 Stages of Parallel Prefix Adder

The Parallel Prefix Adder calculate the prefixes of 2 bits in a group. Then these prefixes are used to compute the prefixes for a group of four bit and going on. To calculate the output carry signal of the specific bit stage these prefixes are used . After the computation, carries are divided into small packages. These are known as processing components. The production of processing component is given by equations (2.12) and (2.13).

$$P_{i:k} = P_{i:j} \cdot P_{j-1:k} \quad (2.12)$$

$$G_{i:k} = G_{i:j} + G_{j-1:k} \cdot P_{i:j} \quad (2.13)$$

In the alternate way, the equations (2.12) and (2.13) can be presented using a symbol "o" symbolized by Brent-Kung. The equation using "o" operation is given below:

$$G_{i:k} : P_{i:k} = (G_{i:j}, P_{i:j}) o (G_{j-1:k}, P_{j-1:k}) \quad (2.14)$$

In the final computation, Sum and final output carry are calculated. It is identical for all parallel prefix adders. The sum and final output carry equations are given in (2.15) and (2.16):

$$S_i = P_i . C_i \quad (2.15)$$

$$C_{i+1} = (P_i . C_0) + G_i \quad (2.16)$$

2.3.1 Brent-Kung adder

The Brent Kung Adder calculate the prefixes of 2 bits in a group. Then these prefixes are used to compute the prefixes for a group of four bit then same calculates for 8 bits groupand going on. These calculated prefixes of Brent Kung adder are used to calculate the output carry signal of the specific bit stage.

After the computation, carries are divided into small packages. These are known as processing components. Now these generated carries are used to calculate the sum bit of that particular stage, for this computation Group Propagate is also used with these carries. Brent Kung adder has total $2\log_2 N - 1$ stages or levels. For example if an adder is designed for 32 bit, then the total number of stages are 9. Brent-Kung adder is a binary tree multi-level topology, hence there are many similarities between traditional multi-level CLA and Brent-Kung adder. For example, the formation of P and G variables are all sent from lower-level network to higher level network in parallel instead of process them in sequence. Figure 2.9 shows the flowchart of 32 bit BKA. In BKA, P and G signals are forwarded to the next stage by using a PG operator. Each node in the fig. 2.9 represents a PG operator. The fanout of a Brent Kung adder is limited to 2.

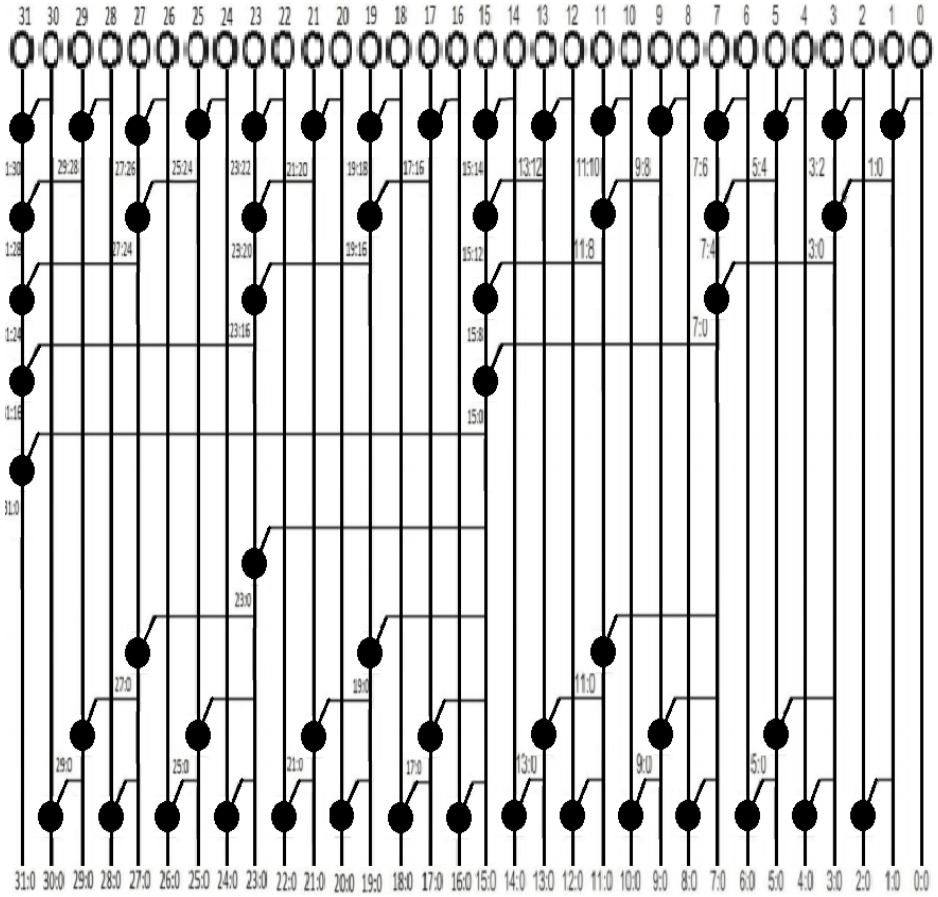


Fig. 2.9 32-bit Brent-Kung Adder [25]

A 32 bit Brent-Kung adder is implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The RTL schematic and synthesized designs are shown in Figs. 2.10 and 2.11respectively. The design uses 51 LUTs. The delay, power and PDP are found to be 8.748ns, 0.248 W and 2.181 nJ respectively.

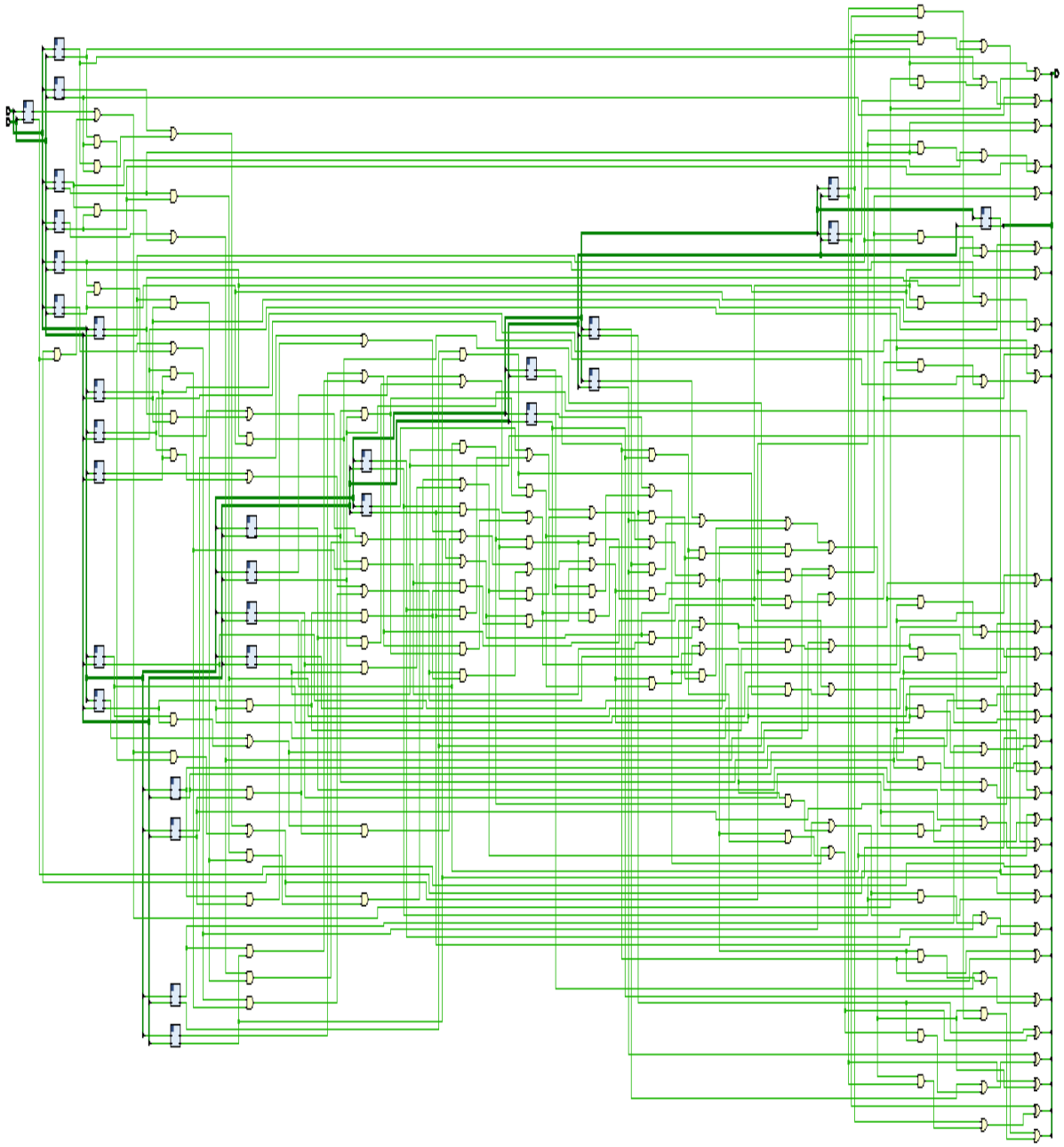


Fig. 2.10 RTL Schematic of Brent-Kung adder

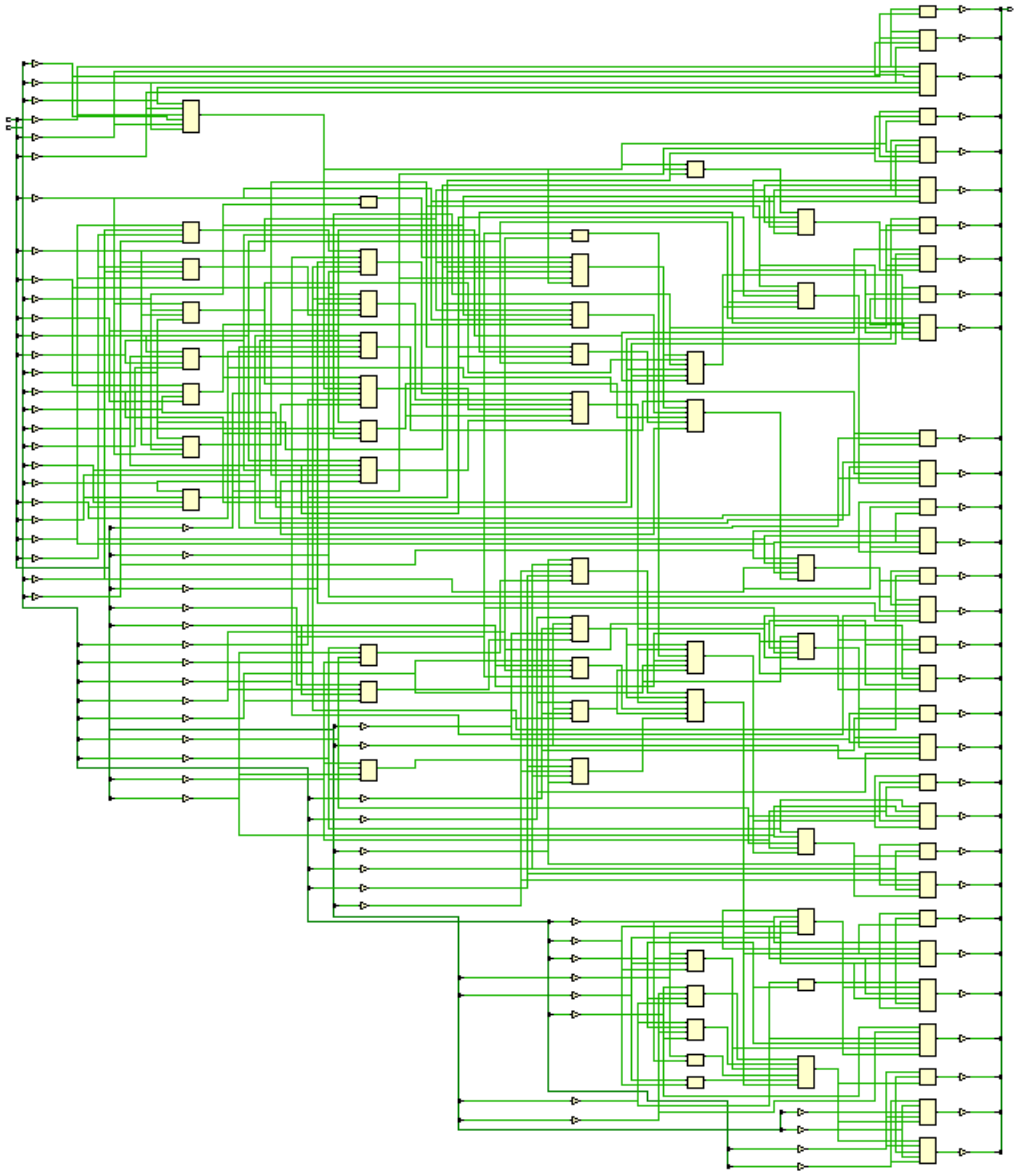


Fig. 2.11 Synthesize design Brent-Kung adder

2.3.2 Kogge Stone Adder

Kogge Stone adder [26] is a logarithmic adder obtained from carry look ahead adder structure. It is one of the fastest parallel prefix adder. This parallel prefix adder has $\log_2 N$ stages, where N is the total number of bits delay though the carry path compared to N for the Ripple Carry Adder.

The fan out of Kogge Stone adder is 2 at each stage. The $\log_2 N$ delay is achieved at higher cost. Kogge Stone adder has more number of PG cells as compare to Brent Kung. That mean means this tree has more PG cells. Hence the number of gates increase to a great amount. The power consumption of Kogge Stone adder is more as compare to other adder designs. The KSA is widely used in high performance applications even at such high costs.

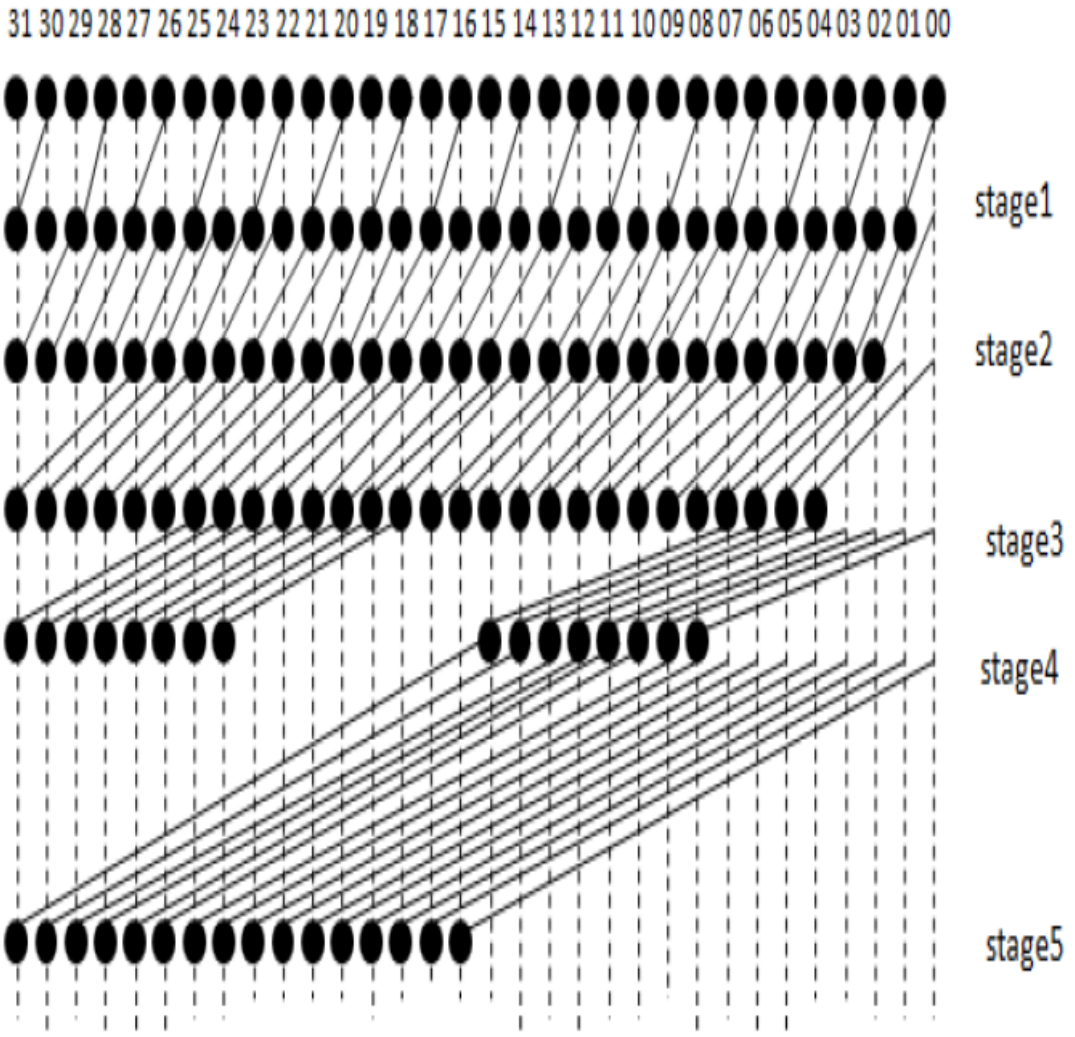


Fig. 2.12 32 bit Kogge Stone Adder [26]

A 32 bit KSA is implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The RTL schematic and synthesized designs are shown in Figs. 2.13 and 2.14 respectively. The design uses 131 LUTs. The delay, power and PDP are found to be 8.736ns, 0.263W and 2.210 nJ respectively.

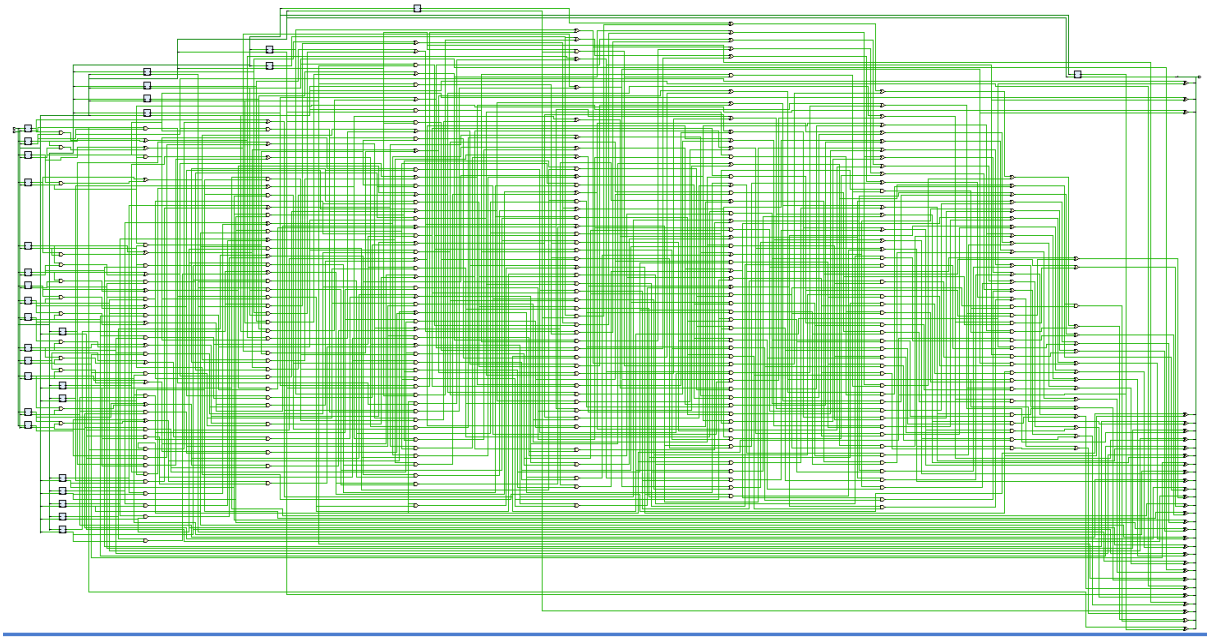


Fig. 2.13 RTL Design of Kogge-Stone Adder

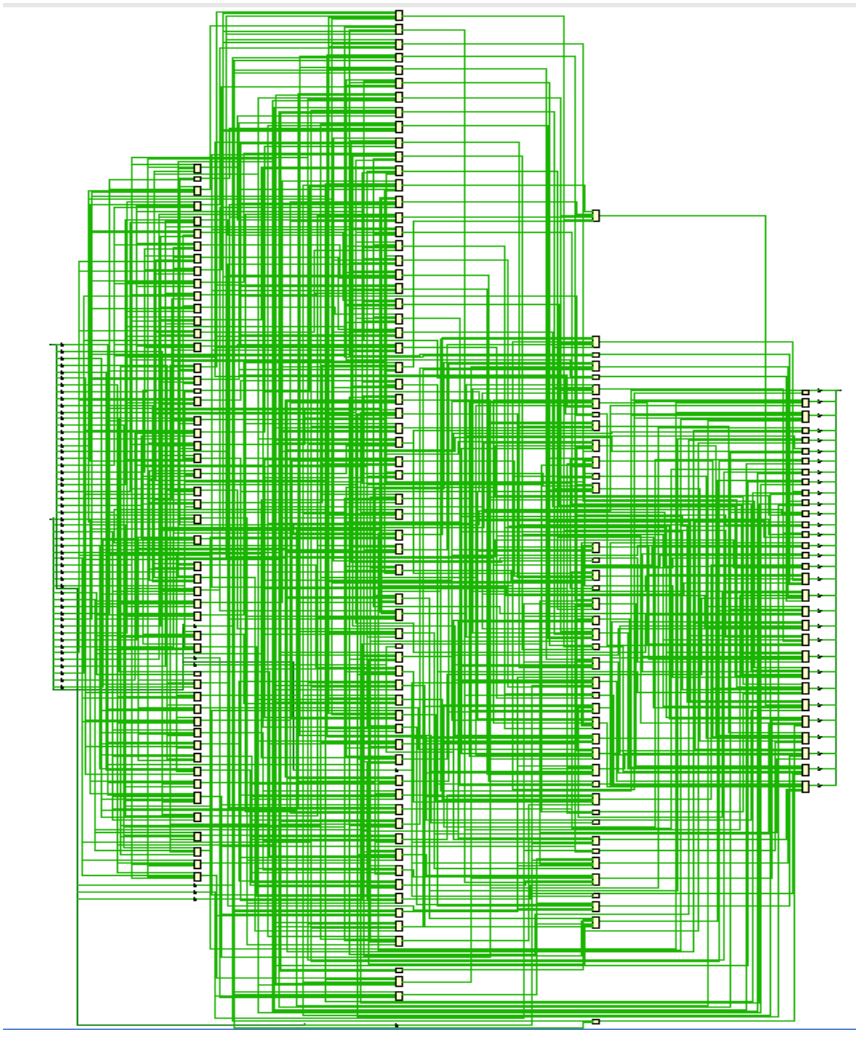


Fig. 2.14 Synthesize design of Kogge-Stone Adder

2.4 Comparison

This section compares 32 bit RCA, CLA, Brent-Kung adder and KSA presented in section 2.1, 2.2, 2.3.1 and 2.3.2 respectively. The results presented in these section are, based on Zynq-7000 xc7z014sclg484-1 FPGA, summarized in Table 2.1 on the basis of area (Look Up Tables (LUT)), maximum delay, power and power delay product.

Table-2.1 Comparison Table of different adder designs

	LUT	POWER(in watt)	DELAY(in ns)	PDP(in nJ)
RCA	30	0.243	13.400	3.256
CLA	68	0.249	8.852	2.204
KSA	131	0.263	8.736	2.210
BKA	51	0.248	8.748	2.181

Following observations are made on the basis of data presented in Table 2.1.

1. The number of LUTs are maximum in KSA and minimum in RCA.
2. The delay is minimum for KSA and there is marginal difference between KSA and BKA in terms of delay.
3. Brent-Kung adder has least power consumption among all designs.

2.5 Summary:

In this chapter, we learned about the different types of adders. Ripple Carry adder design utilizes the lowest chip area when compare to other designs as shown in Table 2.1. Ripple Carry adder has worst case delay in all design. Therefore it is only used in low speed and least area requirement applications. Carry look ahead adder reduces the propagation delay on the cost of LUTs as compare to RCA as shown in Table 2.1. BKA and KSA have many similarities with traditional multi-level CLA. For example, the formation of P and G variables are all sent from lower-level network to higher level network in parallel instead of process them in sequence. KSA is the fastest adder among all designs as shown in Table 2.1. BKA will be using $2\log_2 N - 1$ stages. Hence it takes lesser area if compare with Kogge Stone adder and it has marginal difference in terms of speed.

CHAPTER-3

Traditional Multipliers

There is tremendous demand of computational intensive application for real time data

processing. These applications require sophisticated data acquisition systems and hardware implementing complex arithmetic operations, in particular fast adders and multipliers. Optimizing the design constraints (delay, area and power) of multiplier units can improve the whole system greatly. Though, several options for adders and multipliers are available in literature, there is still scope to improve the performance of these arithmetic blocks. In this chapter some traditional multiplier designs have been discussed and synthesized on Vivado 2019.2 for functional verification of designs.

A multiplier may be realized simply by performing shift and add operations. This method includes calculating partial products, shifting these partial products to the left and then adding them in a organized way. The drawback of this procedure is to determine the partial products, as that includes multiplying a long number (multiplicand) by one digit (of the multiplier) at a time. Such multipliers are serial in nature, use smaller hardware and have slow response. Tree multipliers such as Wallace tree and Dadda multipliers have faster response time.

In this chapter, tree multipliers (Wallace tree and Dadda Multiplier) are implemented and the power, area and delay are computed. Both the multipliers use adders in the final computation therefore the adders discussed in chapter 2 are used and performance is observed. For fairer comparison, 3:2 compressors (full adder) and 2:2 compressors (half adder) in reduction stage of all multipliers.

3.1 **Wallace Tree Multiplier:**

Wallace tree multiplier was first introduced by Chris Wallace in 1964 as an easy and simple way of reducing the partial products by summing them in parallel using the tree structure of Carry Save Adders. Though Wallace tree requires more hardware component than shift and add multipliers, but it produces the results in far less time than shift and add multipliers. A carry save adder can add up to three values simultaneously. The output results of carry save adder is not a single result. Instead, the output results in a set of both a sum and carry bits. The carry-save adder is consists of a group of full adders, each of which adds its three input operands.

In Partial Product Reduction Stage, Wallace Tree uses “As soon as Add” strategy. That mean add as much partial product as possible to reduce the levels. Therefore it can be say that Wallace tree multipliers use a log depth tree structure for the reduction stage. It is faster, but asymmetrical in nature. Wallace tree operate on ease of layout structure for speed.

Wallace tree multipliers are generally avoided in case of low power operations, since

excess of wiring is expected to consume additional power. The Wallace tree multiplier is a high speed multiplier design. Wallace tree reduces the number of partial products arrays to 2 arrays for the computation of final results. In Wallace tree multiplier basically partial products are reduced with help of half adder, full adder and different compressors. Wallace Multiplier is usually used where high speed operation is major requirement

The Wallace tree has three stages, named:

1. Partial Product Generation Stage
2. Partial Product Reduction Stage
3. Final Addition Stage

In the first stage partial products are the multiplication results of multiplicand and multiplier. Here simple "And" gate operation is used in which each bit of multiplicand is multiply by each bit of multiplier. In second stage partial products are reduced to two row arrays by using Wallace tree algorithm. Wallace multiplier use more number of m:n compressors at each level during the reduction stage to accomplish the required two rowed matrix. Here only 3:2 and 2:2 compressors are used. In final addition stage different types of Carry Propagating Adders are used i.e. Ripple Carry Adder, Carry Look Ahead Adder etc.

The reduction procedure of Wallace multiplier is given by the following algorithm [28]:

Step-1 : First generate the partial products.

Step-2 : Move from left to right and check each column. If column height is less than or equal to 2 , then no changes are made and repositioning to the next column. If column height is greater than 2 then reduce by using half adder, full adder or both. When reached to left most column then stop.

Step-3 : Go to the next stage.

Step-4 : Repeat Step-2 and Step-3 until 2 rows are left .

The reduction process of Wallace multiplier is shown in Fig. 3.1.

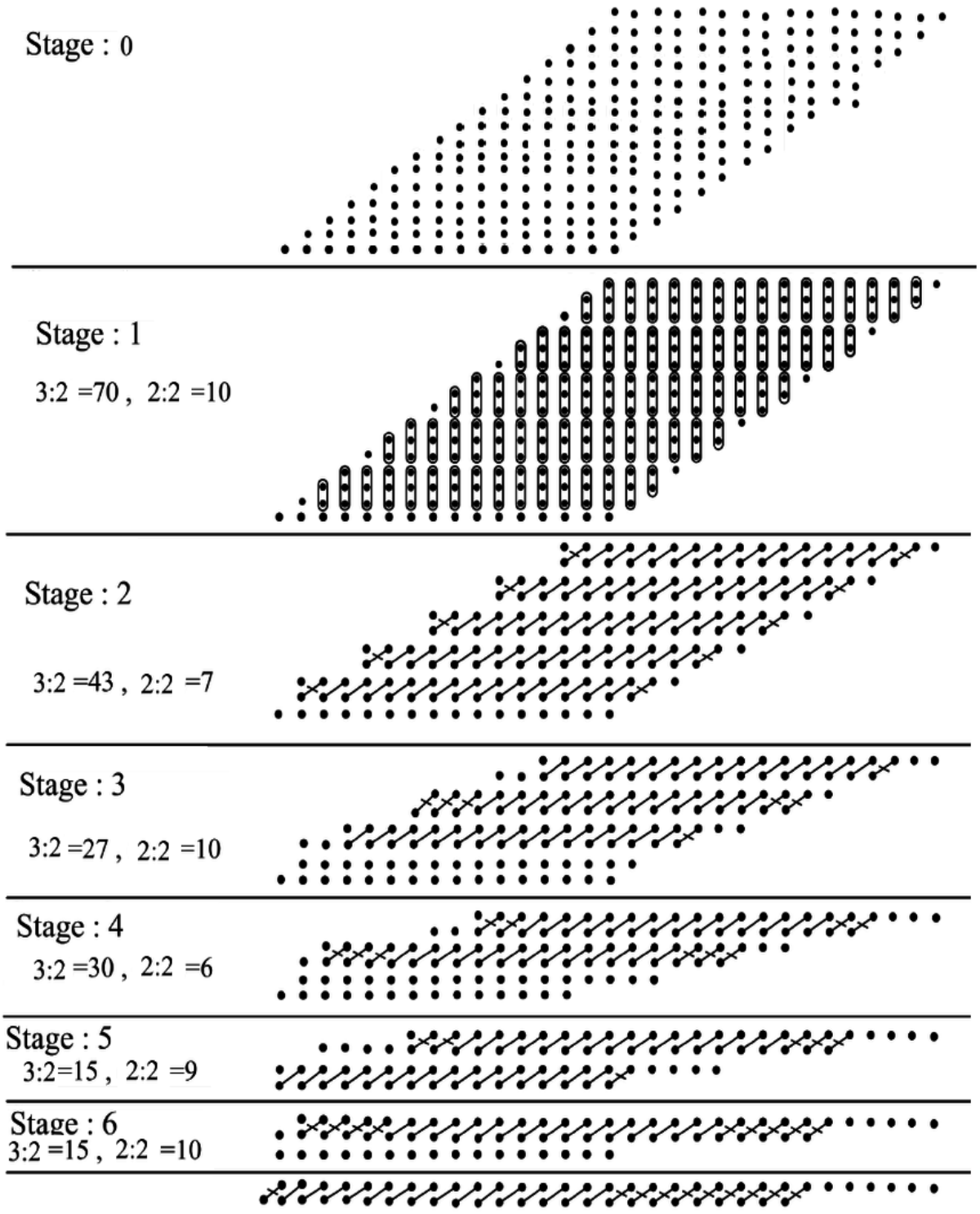


Fig. 3.1 16x16 Wallace tree multiplier reduction process [28]

Here Stage 0 is the partial product generation stage. These dots are the partial products. From Stage 1 to 6, partial products are reduced according to Wallace tree algorithm.

3.1.1 Simulations

In this section four different Wallace tree designs are implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The reduction stage is same in

all designs but for final stage addition, different adders i.e. RCA, CLA, BKA and KSA are investigated.

3.1.1.1 RTL Schematic

The RTL schematic of Wallace tree multiplier is given in Fig. 3.2. This schematic is same for all designs. Here S0 block is the partial product generation stage. Partial product reduction stage is represented by S1 block. S2 block is the final stage addition. In all designs S0 and S1 blocks are same. Only difference is at S2 block due to different adder designs i.e. RCA, CLA etc.

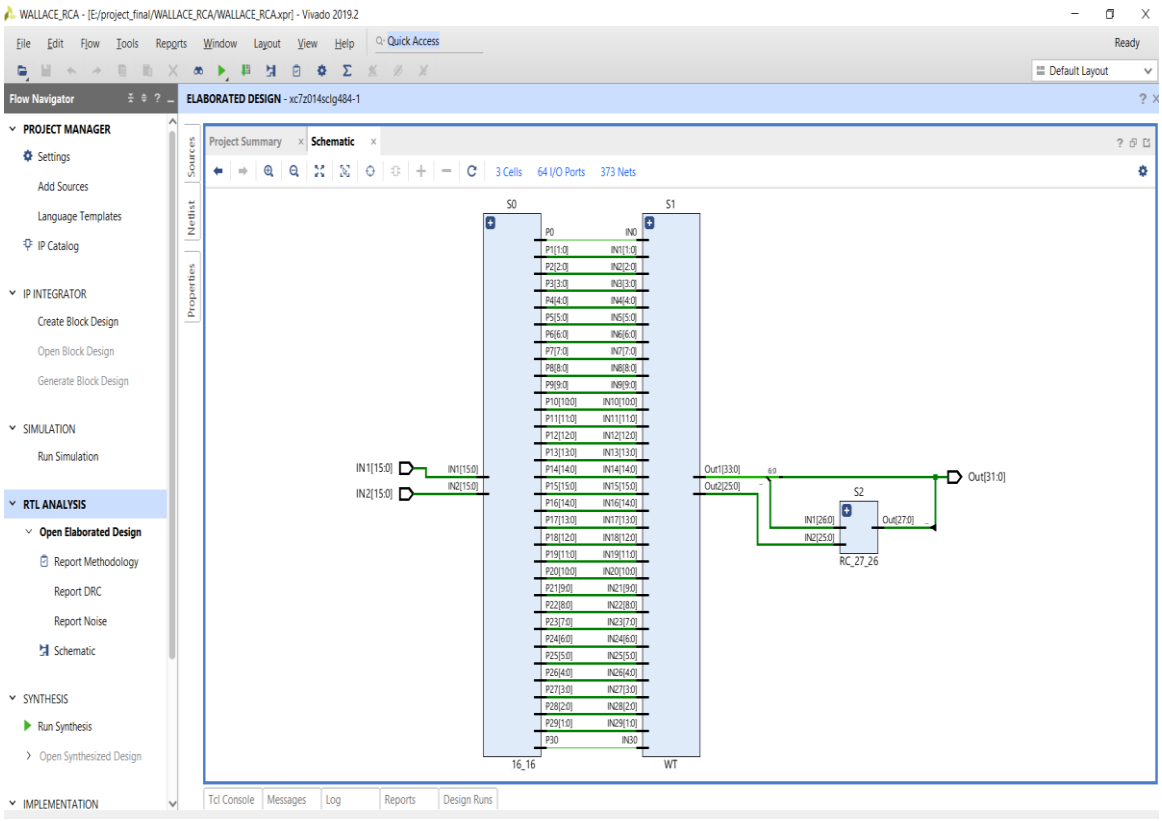


Fig. 3.2 RTL schematic of Wallace tree multiplier

3.1.1.2 Synthesized Design:

In this subsection synthesized design for four implementations are put forward.

Wallace tree with RCA in final stage:

In Fig. 3.3 a synthesized design of Wallace tree with RCA is given. The design uses 394 LUTs and 109 slices. The delay of design is 36.093 ns and the power consumption is 0.28 W. Power delay product (PDP) of the design is found to be 10.11 nJ .

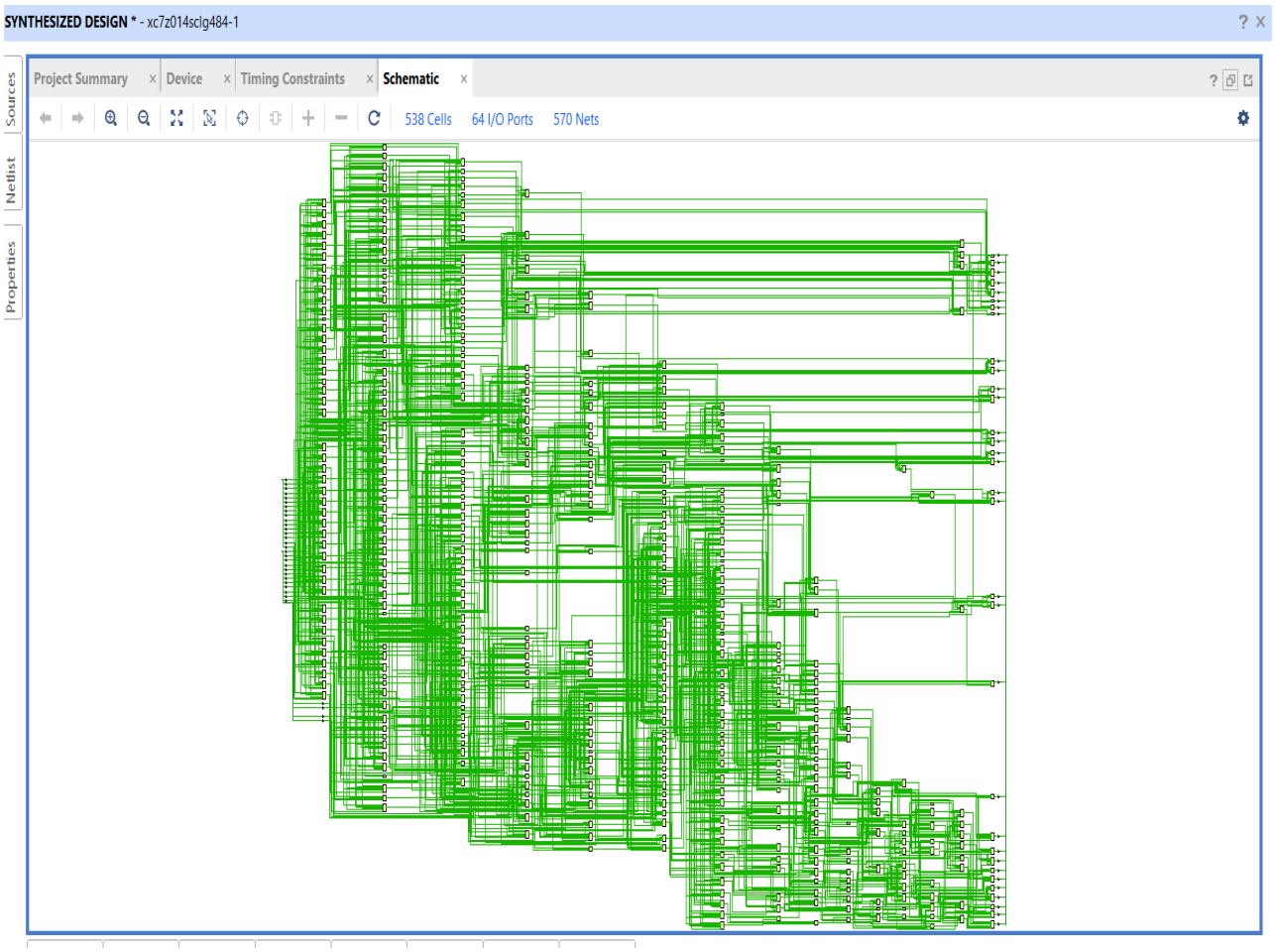


Fig. 3.3 Synthesized design of Wallace tree with RCA

Wallace tree with CLA in final stage:

Synthesized design of Wallace tree with CLA is shown in fig. 3.4. In this design 394 LUTs and 109 slices are used. The delay, power and PDP are found to be 31.442 ns, 0.289 W and 8.835 nJ respectively.

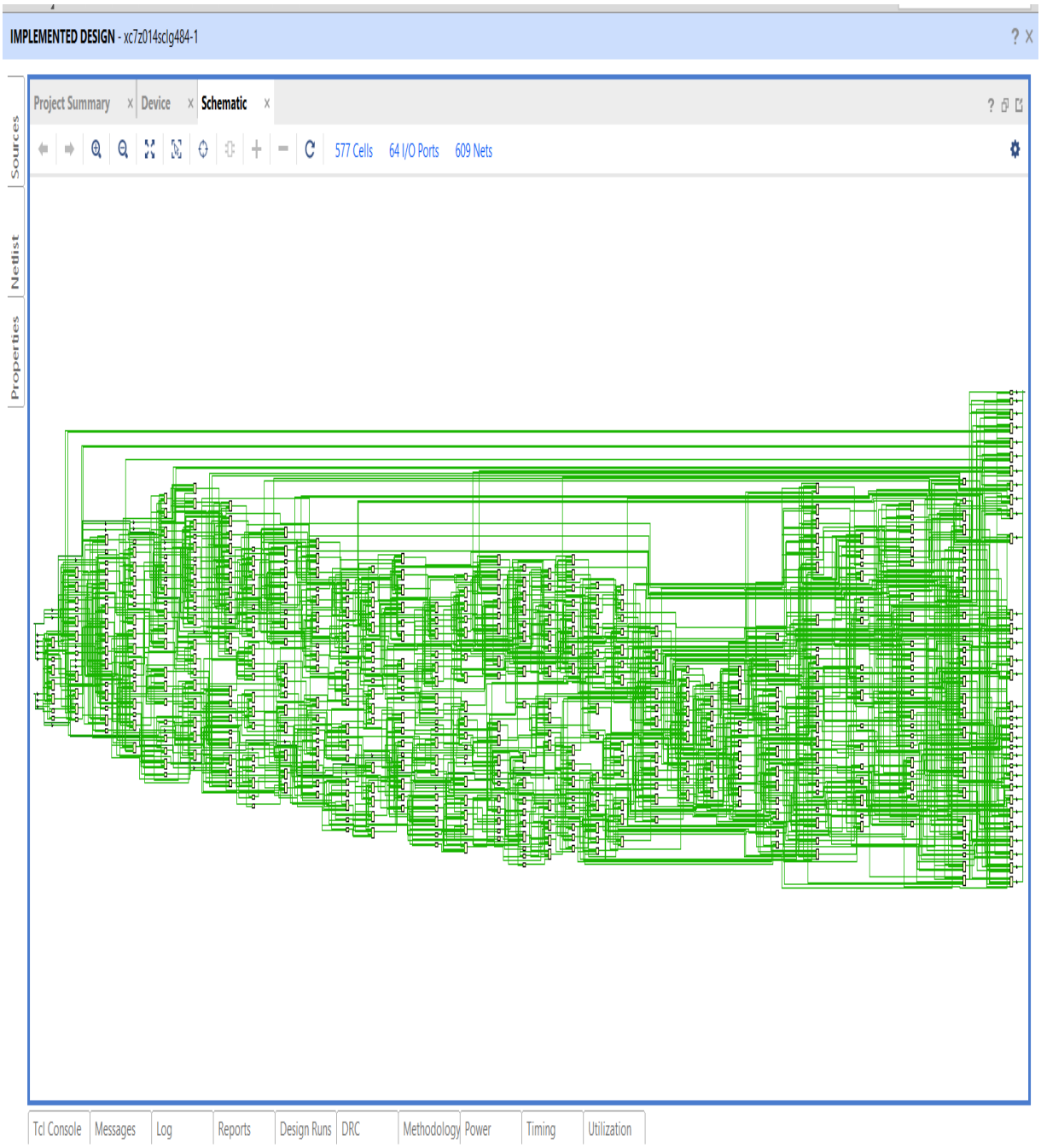


Fig. 3.4 Synthesized design of Wallace tree with CLA

Wallace tree with KSA in final stage:

Figure 3.5 shows synthesized design of Wallace tree with Kogge-Stone Adder. The design uses 511 LUTs and 411 slices. The delay of design is 22.060 ns and the power consumption is 0.293 W. Power delay product (PDP) of the design is found to be 6.463 nJ.

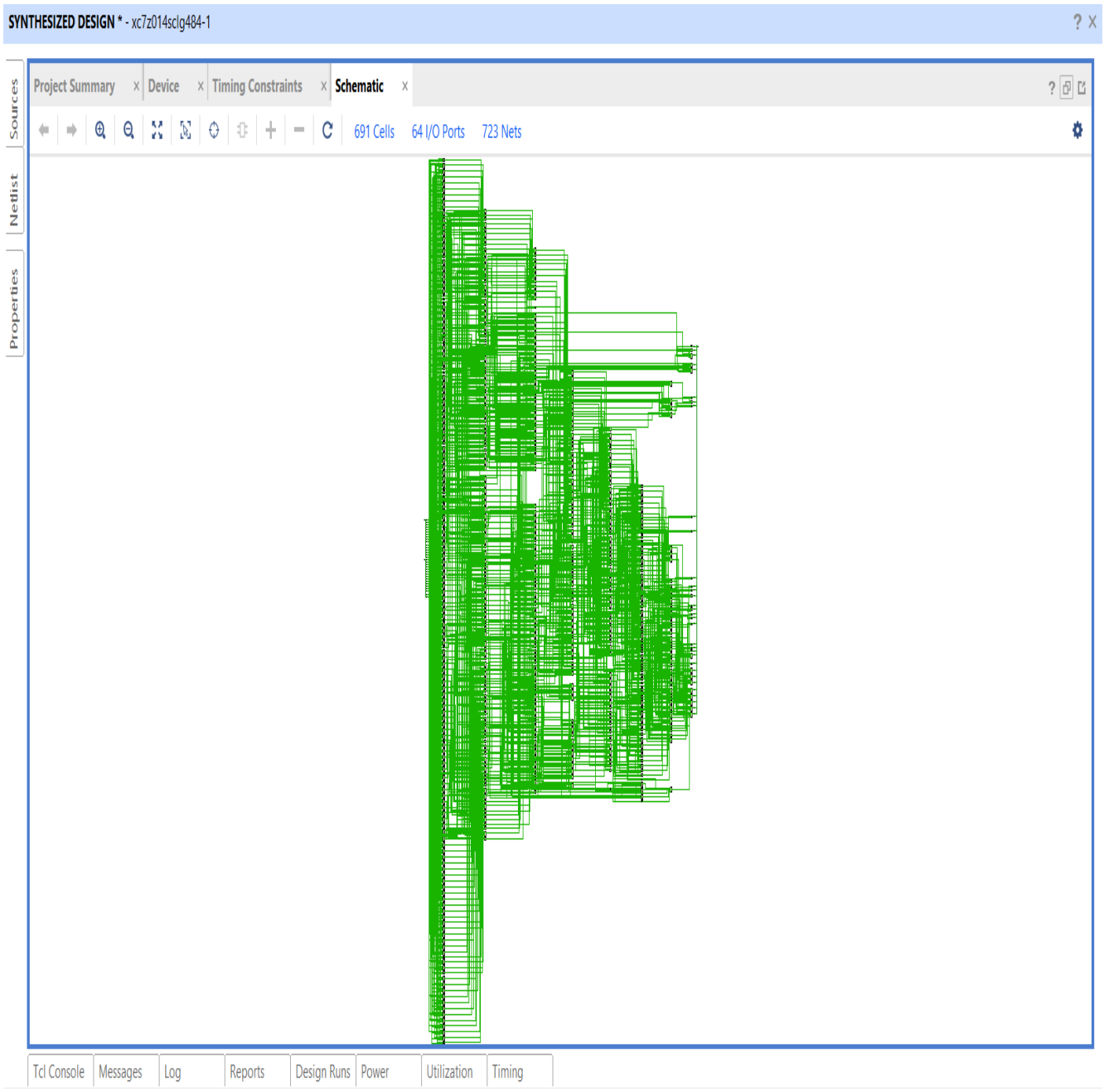


Fig. 3.5 Synthesized design of Wallace tree with Kogge Stone Adder

Wallace tree with BKA in final stage:

Synthesized design of Wallace tree with Brent Kung adder is shown in fig. 3.6. In this design 375 LUTs and 109 slices are used. The delay, power and PDP are found to be 29.577 ns, 0.275 W and 8.133 nJ respectively.

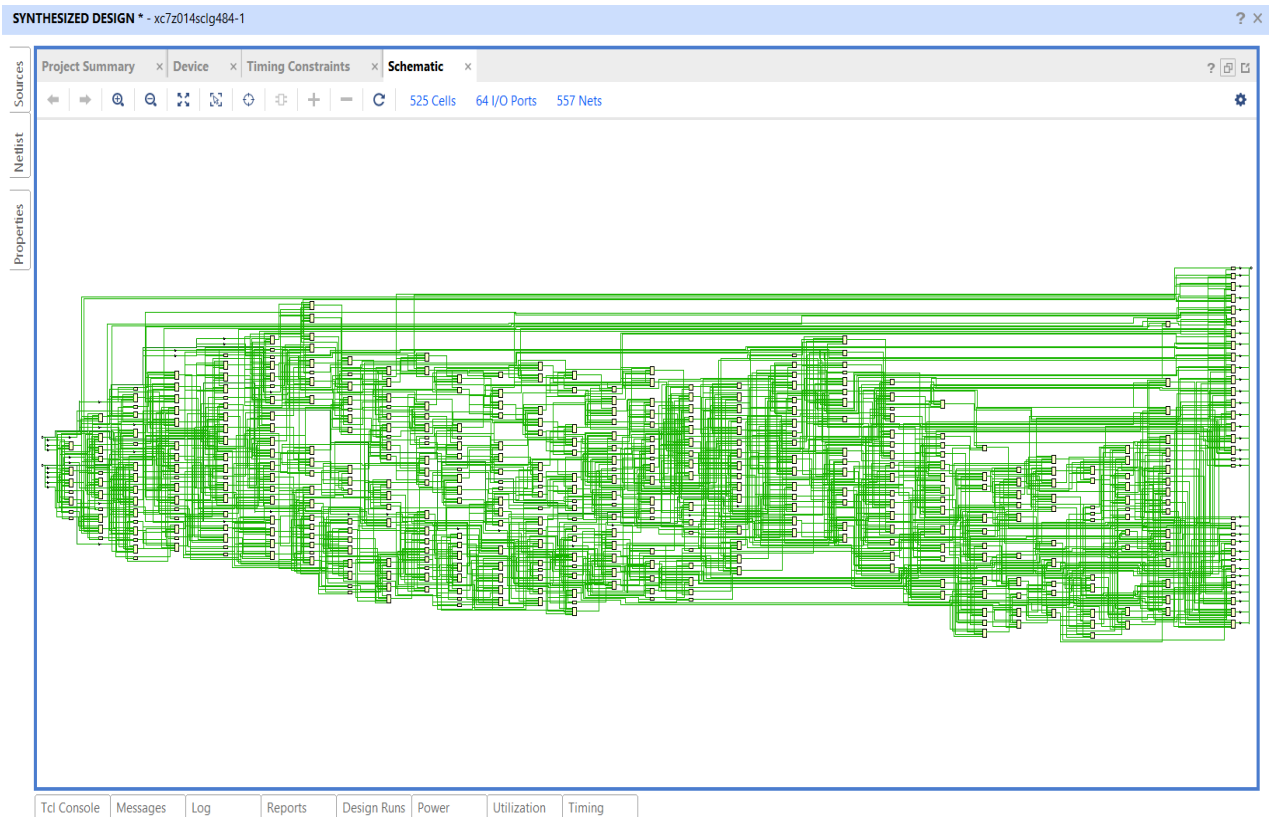


Fig. 3.6 Synthesized design of Wallace tree with Brent Kung adder

All designs are functionally verified using simulation on Vivado 2019.2 and selecting Zynq-7000 xc7z014sclg484-1 FPGA. The designs are compared in terms of area (Look Up Tables (LUT), maximum delay, power and power delay product. The findings are summarized in Tables-3.1.

Table 3.1 Summary of different designs based on Wallace Tree Muliplier

Final Adder	LUTs	Slices	POWER (in Watt)	Delay(in ns)	PDP(nJ)
RCA	394	109	0.28	36.083	10.11
CLA	411	124	0.289	31.442	8.835
KSA	511	141	0.293	22.060	6.463
BKA	375	109	0.275	29.577	8.133

Following are the observations on the basis of Table 3.1.

1. The delay is minimum for design with Kogge-Stone Adder among all designs. However, it uses larger area in terms of LUTs and Slices.
2. The designs with RCA and Brent Kung adder use same number of slices, but the delay is minimum for the later one.
3. The Wallace tree design with Brent Kung adder has least power consumption among all designs.
4. There is only marginal difference between the delay of multiplier design with BKA and the design with KSA.

3.2 Dadda Multiplier:

Dadda Multiplier is one of the well-known column compression multiplier was first

presented by Dadda in 1965 [3]. The functionality of Dadda Multiplier is similar to Wallace Tree Multiplier but Dadda multiplier is marginally faster in nature. Dadda multiplier has less number of components than Wallace tree. In Dadda multiplier, partial products are represented in dots. These partial products are organized in a tree form, which is demonstrated in Fig. 3.7. The sixteen rows are restructured and then partial product reduction is done with the help of parallel m:n compressors. In Fig. 3.7, 3:2 and 2:2 compressors are exploited. A full adder is the realization of a 3:2 compressor which receives 3 inputs and produce 2 outputs. Similarly a half adder is the implementation of 2:2 compressor.

Dadda multipliers use least number of m:n compressors at each level during the reduction stage to accomplish the required two rowed matrix. The reduction procedure of Dadda multiplier is given by the Dadda algorithm [20] :

1. $d_1 = 2$ and $d_{j+1} = [1.5*d_j]$. Here d_j is the maximum height sequence of the matrix for the j^{th} stage. The initial value of j is taken as the maximum such that $d_j < \min(b1, b2)$. Here b1 and b2 are the number of bits of multiplier and multiplicand.
2. Now reduce the height of those columns which is greater than the d_j or which will have more height than d_j as they receive carries from m-n compressors of previous columns.
3. Let j = j-1 and repeat step 2 until the only 2 rows are left.

In final stage partial products are reduced in two-rowed arrays. To get the final multiplier output these two-rowed arrays are added using carry propagation adder [20]. The dot diagram shown in Fig. 3.7 shows this algorithm realization for an 16-bit Dadda multiplier. Here six reduction levels are necessary. In fig. 3.7 dots joint by a Rectangle indicates a full adder. In the same way, two dots combination specifies a half adder.

In first level d_j height is taken as 6, and the columns which have more height than d_j are reduced.. Arrow shows the carry transfer to the next column. Same process is repeated until two rowed matrix left.

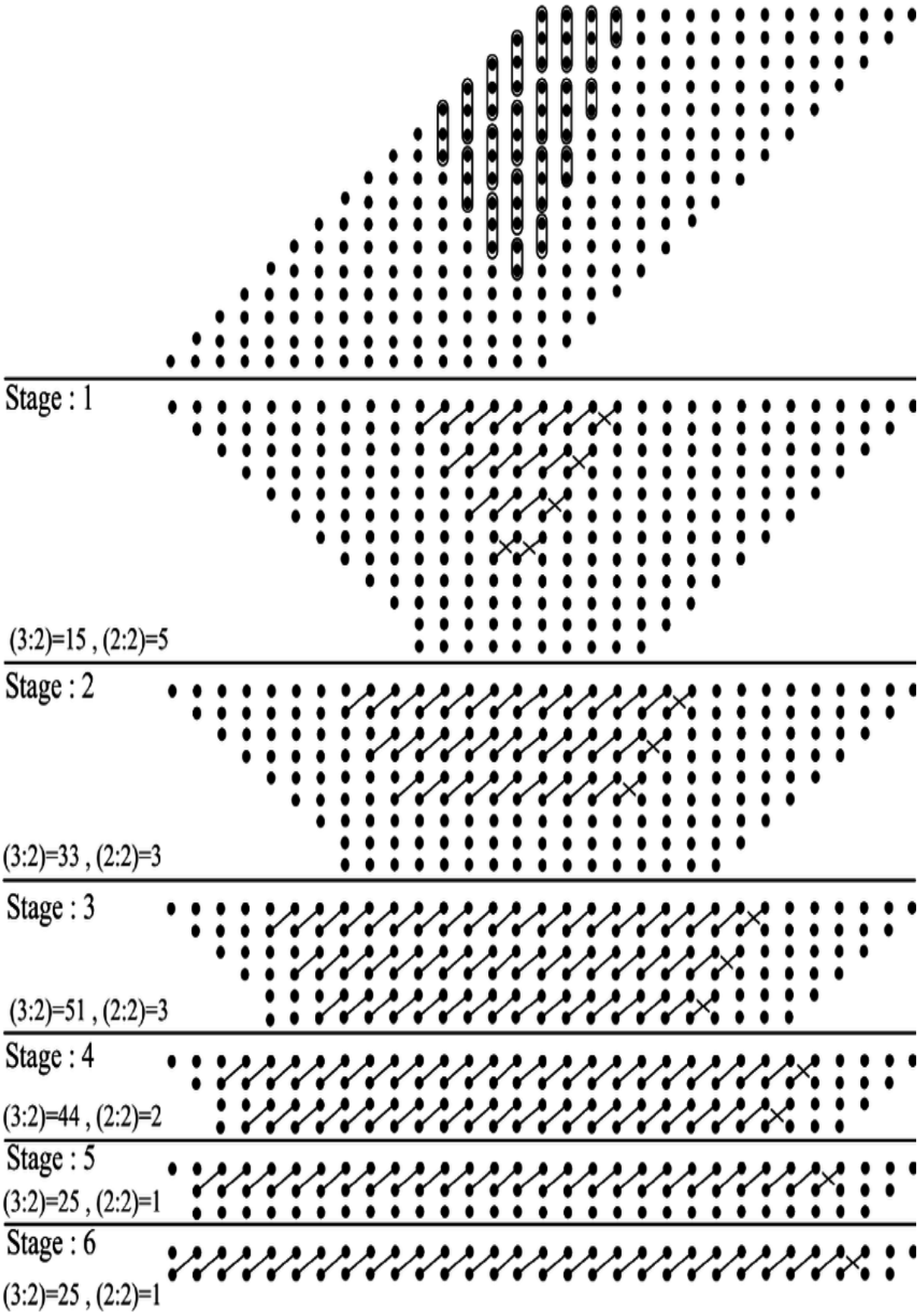


Fig. 3.7 16x16 bit Dadda multiplier design [29]

The reduction process of 16x16 Dadda multiplier is explained below.

Step-1 : First find out the out the j^{th} stage at which we get the maximum height. In case of

16x16 multiplier maximum height that can be achieved is 13 at the stage 6. Hence j=6 is taken as initial stage.

Step-2 : Move from left to right and check each column. If column height is less than or equal to 13, then no changes are made and repositioning to the next column. If column height is greater than 13 then reduce it to 13 by using half adder, full adder or both. When reached to left most column then stop.

Step-3 : Go to the next stage which is decreased by one i.e. $j_{i+1} = j_i - 1$.

Step-4 : Repeat Step-2 and Step-3 until j=1 .

3.2.1 Simulation:

Here four different Dadda multiplier designs are implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The partial product generation and reduction stages are same in all four designs. For final stage addition, different adders i.e. RCA, CLA, BKA and KSA are investigated.

The RTL schematic and synthesized designs are given in following sections..

3.2.1.1 RTL Schematic

The RTL schematic of Dadda multiplier is given in fig. 3.8. The RTL Schematic block is same for all designs. Here S0 block is the partial product generation stage. Partial product reduction stage is represented by S1 block. S2 block is the final stage addition. In all designs S0 and S1 blocks are same. Only difference is at S2 block due to different adder designs i.e. RCA, CLA etc.

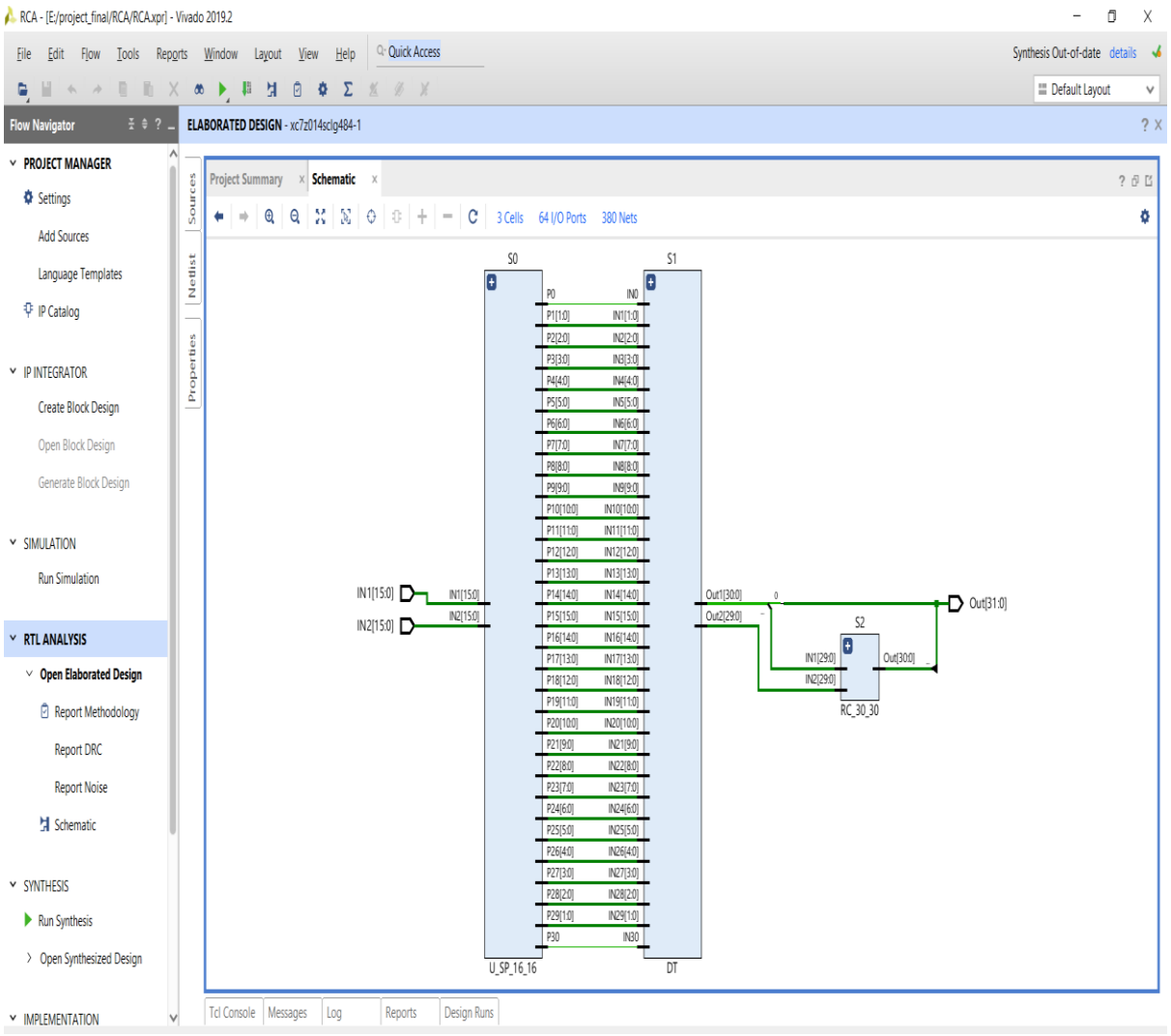


Fig. 3.8 RTL schematic of Dadda multiplier

3.1.1.2 Synthesized Designs:

In this subsection synthesized design of four implementations is given.

Dadda tree with RCA in final stage :

In Fig. 3.9 a synthesized design of Dadda tree with RCA is given. The design uses 347 LUTs and 106 slices. The delay of design is 24.726 ns and the power consumption is 0.277 W. Power delay product (PDP) of the design is found to be 6.85 nJ .

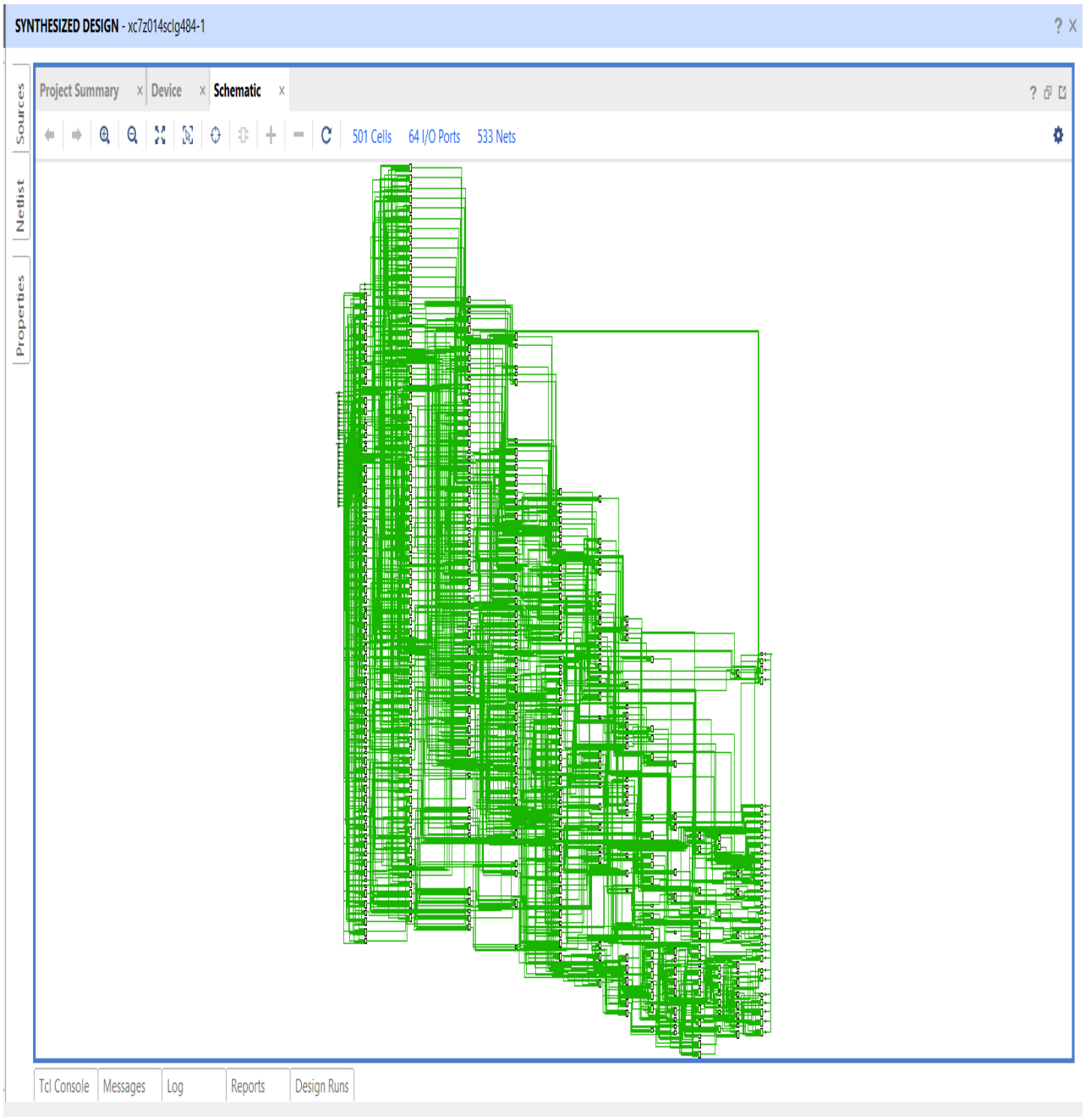


Fig. 3.9 Synthesized design of Dadda multiplier with RCA

Dadda multiplier with CLA in final stage :

Synthesized design of Dadda multiplier with CLA is shown in fig. 3.10. In this design 386 LUTs and 144 slices are used. The delay, power and PDP are found to be 22.176 ns, 0.281 W and 6.231 nJ respectively.

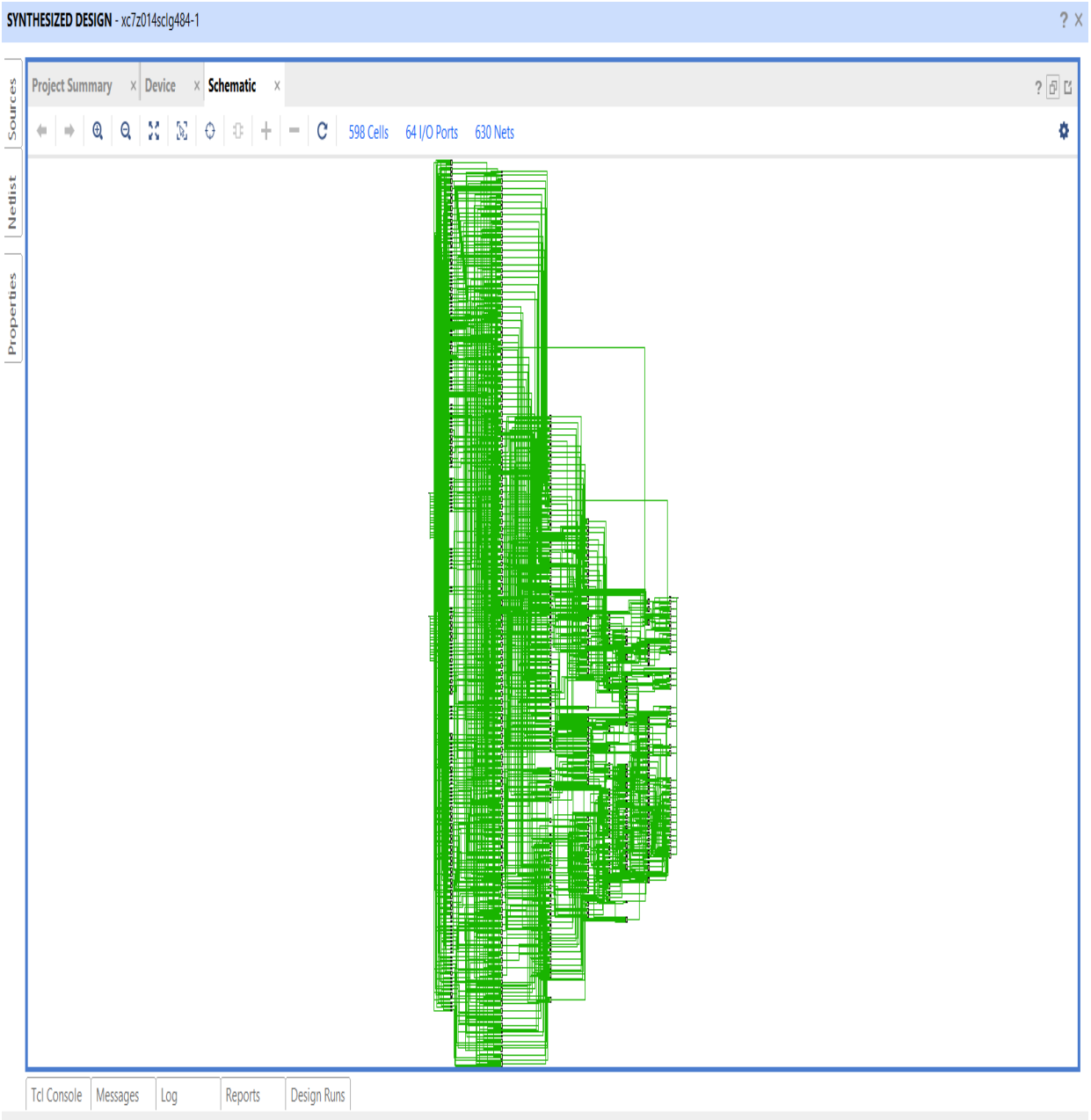


Fig. 3.10 Synthesized design of Dadda multiplier with CLA

Dadda tree with KSA in final stage:

In fig. 3.11 a synthesized design of Dadda multiplier with KSA is given. The design uses 413 LUTs and 121 slices. The delay of design is 20.366 ns and the power consumption is 0.283 W. Power delay product (PDP) of the design is found to be 6.046 nJ .

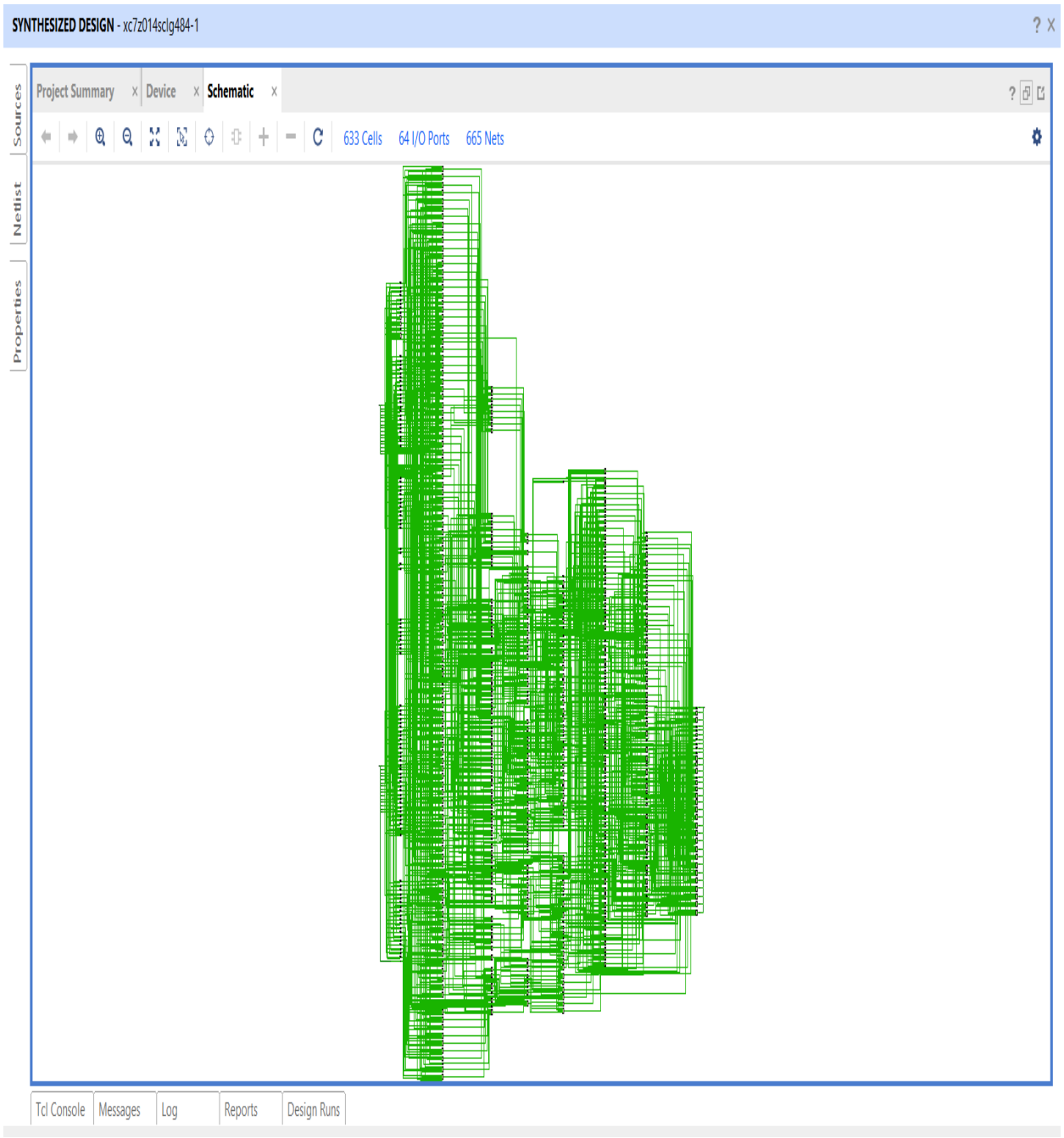


Fig. 3.11 Synthesized design of Dadda multiplier with Kogge Stone adder

Dadda tree with Brent Kung adder in final stage:

Synthesized design of Dadda multiplier with BKA is shown in fig. 3.12. In this design 344 LUTs and 98 slices are used. The delay, power and PDP are found to be 21.098 ns, 0.274 W and 8.133 nJ respectively.

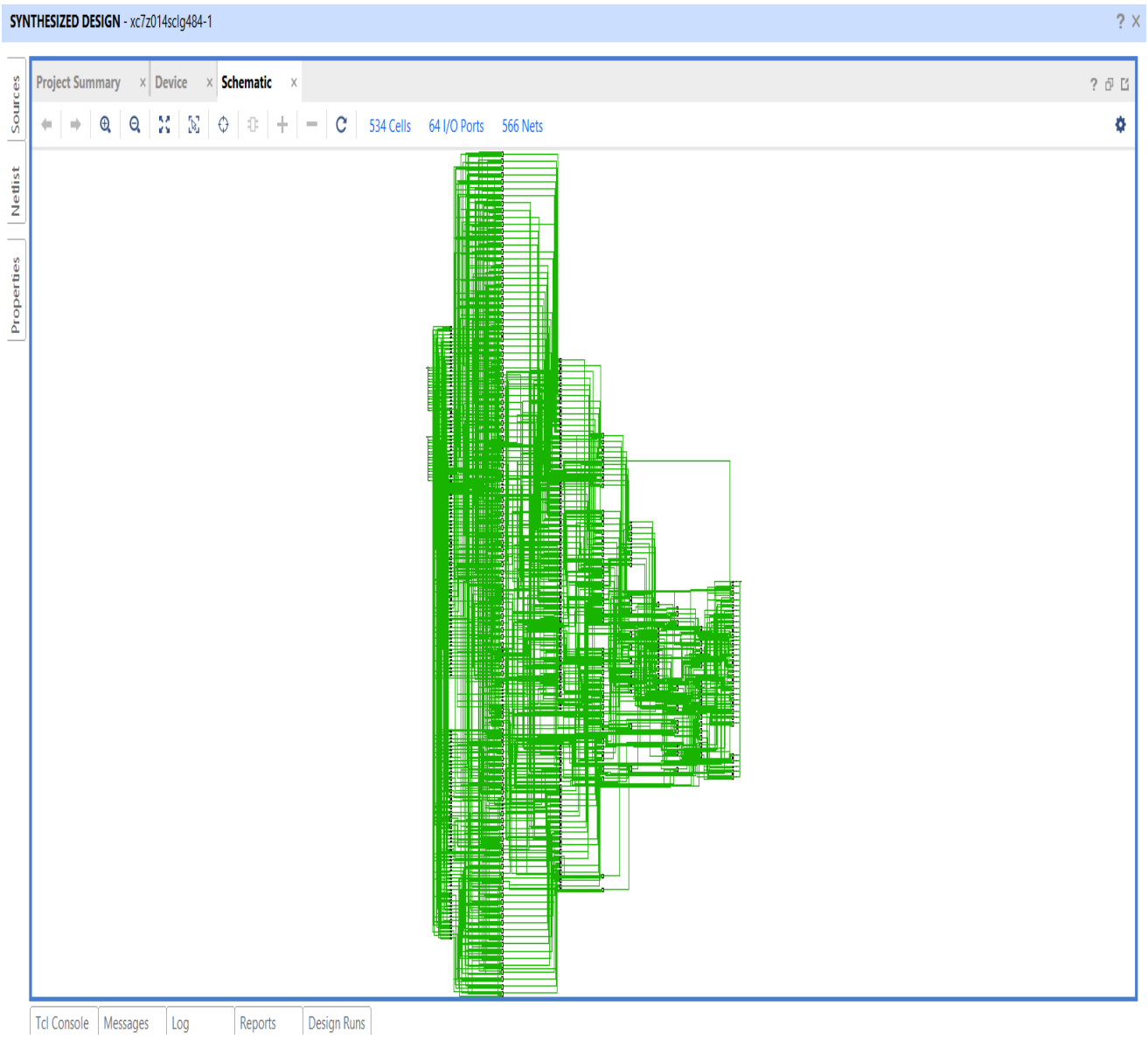


Fig. 3.12 Synthesized design of Wallace tree with Brent Kung adder

All designs are functionally verified using simulation on Vivado 2019.2 and selecting Zynq-7000 xc7z014sclg484-1 FPGA. The designs are compared in terms of area(Look Up Tables (LUT)),maximum delay, power and power delay product. The findings are summarized in Tables 3.2.

Following are the observations made on the basis of Table 3.2.

1. The delay is minimum for design with Kogge-Stone adder among all designs. However, it uses larger area.

3. The designs with RCA and Brent Kung adder have lesser area when compare to other designs, but the delay is minimum for the later one.

4. The Dadda multiplier design with Brent Kung adder has least power consumption among all designs.

5. There is only marginal difference between the delay of multiplier design with BKA and the design with KSA.

Table 3.2 Summary of Dadda Multipliers implementations

Final Stage Adder	LUT	Slice	POWER (in Watt)	Maximum Delay (in ns)	Power-Delay Product (nJ)
Ripple Carry Adder	347	106	0.277	24.726	6.85
Carry Look Ahead Adder	386	144	0.281	22.176	6.231
Kogge Stone Adder	413	121	0.283	20.366	6.046
Brent-Kung Adder	344	98	0.274	21.098	6.054

3.3 Comparison:

All designs are functionally verified using simulation on Vivado 2019.2 and selecting Zynq-7000 xc7z014sclg484-1 FPGA. For fairer comparison, full adders and half adders are used in reduction stage of all designs. For final stage additions we are using one of the adder which are discussed in chapter-2 i.e. RCA , CLA, KSA and BKA.

The results of Tables 3.1 and 3.2 are combined and presented in Table 3.3 to find out a better design. Following observations are made on the basis of data presented in Table 3.3.

1. The number of LUTs and slices increase in Wallace Tree designs than their counterpart employing Dadda multiplier.
2. The delay is minimum for Dadda multiplier design and Kogge-Stone adder among all designs. However, it uses larger area.
3. The Dadda multiplier design with Brent Kung adder in final stage has least power consumption among all designs.
5. There is only marginal difference between the delay of Dadda multiplier design with BKA and the design with KSA.

Table-3.3 Combined Comparison Table

	Final Adder	LUT	Slice	POWER (in Watt)	Delay(in ns)	PDP(nJ)
Wallace Tree Multiplier	**RCA**	394	109	0.28	36.083	10.11
	CLA	411	124	0.293	31.442	8.835
	KSA	511	141	0.281	22.060	6.463
	BKA	375	109	0.276	29.577	8.133

Dadda Multiplier	**RCA**	347	106	0.281	24.726	6.85
	CLA	386	144	0.283	22.176	6.231
	KSA	413	121	0.277	21.366	6.046
	BKA	344	98	0.274	22.098	6.054

Simulation result shows Dadda multiplier designs are better in terms of all design constraints as shown in Table-3.3.

3.4 Summary:

In this chapter, we learned about the reduction process of Wallace tree and Dadda tree multiplier. All the designs are simulated on Vivado 2019.2. Dadda algorithm follows add as less partial product as possible in reduction stage described in section 3.2.The functionality of Dadda Multiplier is similar to Wallace Tree Multiplier but Dadda multiplier is marginally faster in nature and it has less number of components than Wallace tree as depicted in Table 3.3. From the simulation results, it is clear that Dadda multiplier designs take less area and faster when compare with Wallace tree multiplier designs.

CHAPTER 4

Proposed Design

In chapter 2 and 3, different types of adder and multipliers are presented. It was found that Dadda multiplier designs are better in terms of area, power and delay as compare to Wallace tree multiplier. In Wallace tree and Dadda multiplier designs, full adders and half adders were used in chapter 3 for fairer comparison. In literature, Dadda multiplier makes use of different types of compressors like 4:2,5:2 etc. in reduction stage. As per the survey, it is indicated that 4:2 compressors along with full adders and half adders is not investigated in conjunction with Brent-Kung adder.

In this chapter, a Dadda multiplier is implemented using 4:2 compressors along with full adders and half adders in reduction stage and Brent- Kung adder in final stage. This design is then compared with other available design that use same reduction stage but different adder designs for final addition.

4.1 Compressors

Compressors are widely used in multipliers for adding partial product terms to reduce the number of operands. A parallel m:n compressor has m inputs and produce n outputs which compress the m values to n. There are different types of compressors like 3:2, 4:2, 5:2 etc. The 3:2 compressor is widely used as a full adder. In 4:2 compressors there are five inputs in which one input is the carry from previous stage [18]. Three outputs are Sum, Cout and Carry. In multiplier when we add many partial products then to reduce the number of operations it is convenient to use 4:2 compressors instead of 3:2 compressors. In Fig. 4.1, a 4:2 compressor is designed from the combinational circuit which consists of two 3:2 compressors [19] . Here Cout is calculated from X1, X2 and X3 only.

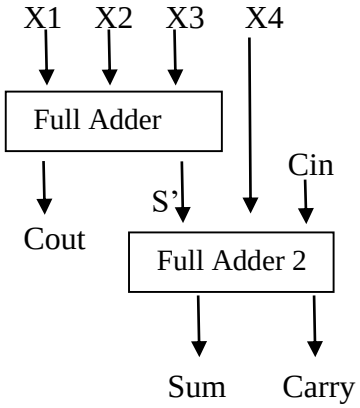


Fig. 4.1 4:2 Compressor

Full Adder 1 takes three inputs X1, X2 and X3 and produced two outputs i.e. S' and Cout [25]. The governing equations of Full Adder 1 are :

$S' = X1 \oplus X2 \oplus X3$ (4.1)

$Cout = X1.X2 + X2.X3 + X3.X1 = (X1 \oplus X2)*X3 + X1.X2$ (4.2)

In Full Adder 2, there are three inputs and two outputs i.e. S', X4, Ci and Sum, Carry. The governing equations are:

$Sum = S' \oplus X4 \oplus Ci = X1 \oplus X2 \oplus X3 \oplus X4 \oplus Ci$ (4.3)

$Carry = (X4 \oplus Ci).S' + X4.Ci = (X4 \oplus Ci).(X1 \oplus X2 \oplus X3) + X4.Ci$ (4.4)

4.2 Dadda Reduction Stage

In Dadda multiplier, partial products are represented in dots. These partial products are organized in a tree form, which is demonstrated in Fig. 4.2 for a 16x16 multiplier. The sixteen rows are restructured and then partial product reduction is done with the help of parallel m:n compressors. In Fig.4.2, 4:2, 3:2 and 2:2 compressors are exploited.

The dot diagram of Fig.4.2 shows this algorithm realization for an 16-bit Dadda multiplier. Here six reduction levels are necessary. Four dots combined by a rectangle indicates that uses 4:2 compressor to compress these dots. The arrow from a dot to 4:2 compressor designates that a carry generated in previous stage is used as an input carry (Cin) into the same 4:2 compressor. Similarly 3 dots joint by a Rectangle indicates a full adder. In the same way, two dots combination specifies a half adder.

In first level d_j height is taken as 6, and the columns which have more height than d_j are reduced.. Arrow shows the carry transfer to the next column. Same process is repeated until two rowed matrix left. The reduction process of 16x16 Dadda multiplier is explained below.

Step-1: First find out the out the j^{th} stage at which we get the maximum height. In case of 16x16 multiplier maximum height that can be achieved is 13 at the stage 6. Hence j=6 is taken as initial stage.

Step-2: Move from left to right and check each column. If column height is less than or equal to 13, then no changes are made and repositioning to the next column. If column height is greater than 13 then reduce it to 13 by using half adder, full adder or both. When reached to left most column then stop.

Step-3: Go to the next stage which is decreased by one i.e. $j_{i+1} = j_i - 1$.

Step-4: Repeat Step-2 and Step-3 until j=1 .

The reduction process of 16x16 Dada multiplier is tabulated in Table 4.1.

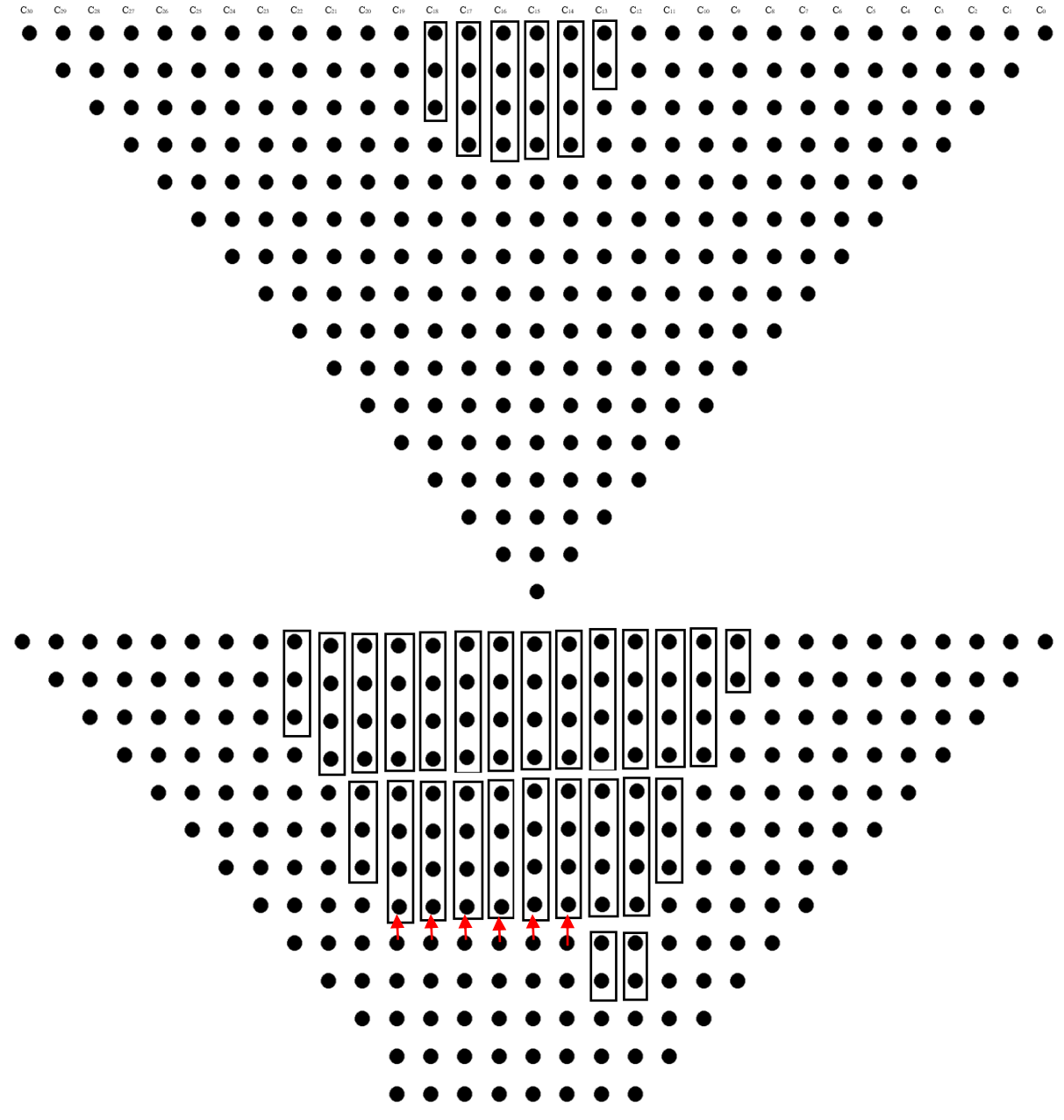
Table-4.1 Number of Stage, Height and Reduce Columns

Level	Stage j	d_j	Reduced Columns (c_i)
1	6	13	$C_{13\text{-}18}$
2	5	9	$C_{9\text{-}22}$
3	4	6	$C_{6\text{-}25}$
4	3	4	$C_{4\text{-}27}$
5	2	3	$C_{3\text{-}28}$
6	1	2	$C_{2\text{-}29}$

In first level, stage j=6 is taken and with every addition in level, decrease the stage j by one. If height of column is less than or equal to d_j ($C_i \leq d_j$), no action required and move to column C_{i+1}. In other cases if

1. $d_j - C_i = 1$, compress it with the help of half adder and move to column C_{i+1}.
2. $d_j - C_i = 2$, compress it with the help of full adder and move to column C_{i+1}.
3. $d_j - C_i > 2$, compress it with the help of 4-2 compressors, half adders and full adders till $C_i \leq d_j$ then move to column C_{i+1}.

In this process place the sum at the bottom of column C_i and place Cout and Carry at the top of column C_{i+1}.


C30 C29 C28 C27 C26 C25 C24 C23 C22 C21 C20 C19 C18 C17 C16 C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 C5 C4 C3 C2 C1 C0

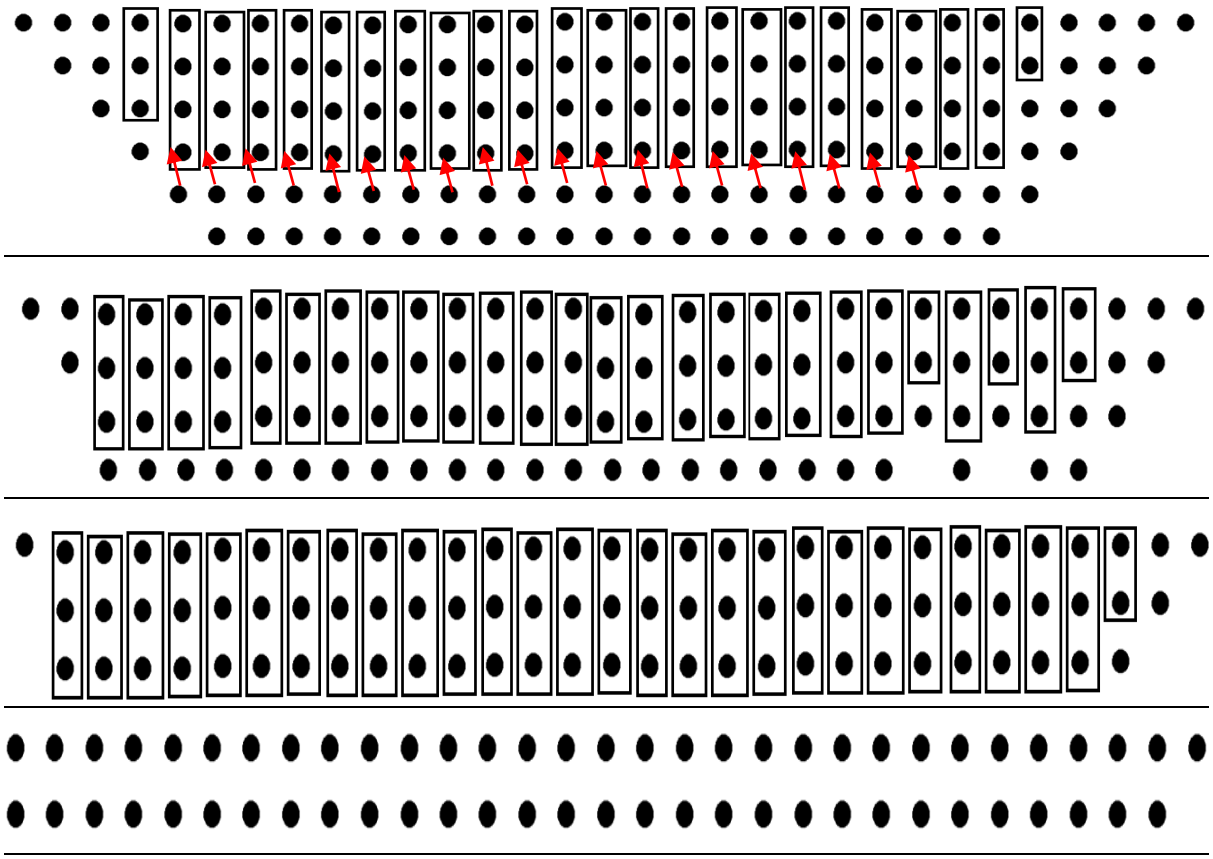


Fig. 4.2 Dot diagram of the 16*16 Dadda Multiplier reduction stage [9]

In the Fig. 4.2, 3 dots combined by a diagonal line indicates that a full adder is used to compress these dots. In the same way, two dots joined by a diagonal line specifies a half adder. In first level d_j height is taken as 6, and the columns which have more height than d_j are reduced. Arrow shows the carry transfer to the next column. Same process is repeated until two rowed matrix left.

4.3 Simulations

In this section four different Dadda designs are implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. The reduction stage is same in all designs but for final stage addition, different adders i.e. RCA, CLA, BKA and KSA are investigated.

Here four different Dadda tree multipliers are designed. The reduction stage is same in all designs but for final stage addition, four different adders i.e. RCA , CLA , BKA and KSA are used. The RTL designs and synthesized designs are given in the following subsections.

4.3.1 Dadda tree with RCA in final stage

In Figs. 4.3 and 4.4 RTL schematic and synthesized design of Dadda tree with RCA are given. The design uses 347 LUTs and 118 slices. The delay of design is 22.880 ns and the power consumption is 0.245 W. Power delay product (PDP) of the design is found to be

5.61 nJ.

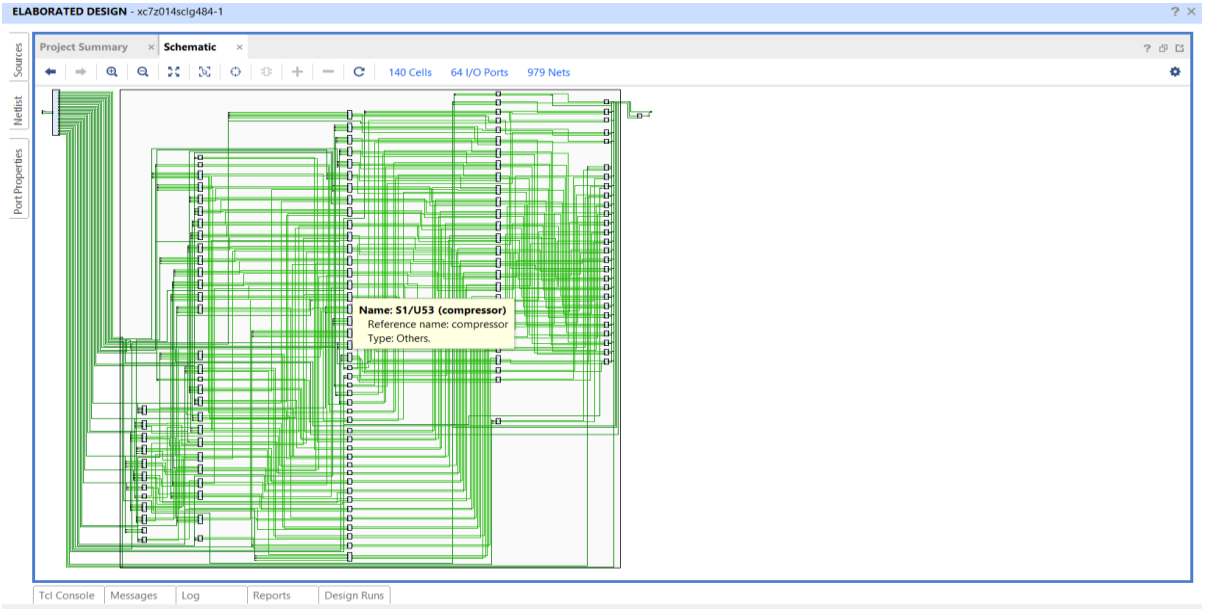


Fig. 4.3 The RTL Schematic of 16x16 Dadda multiplier with RCA

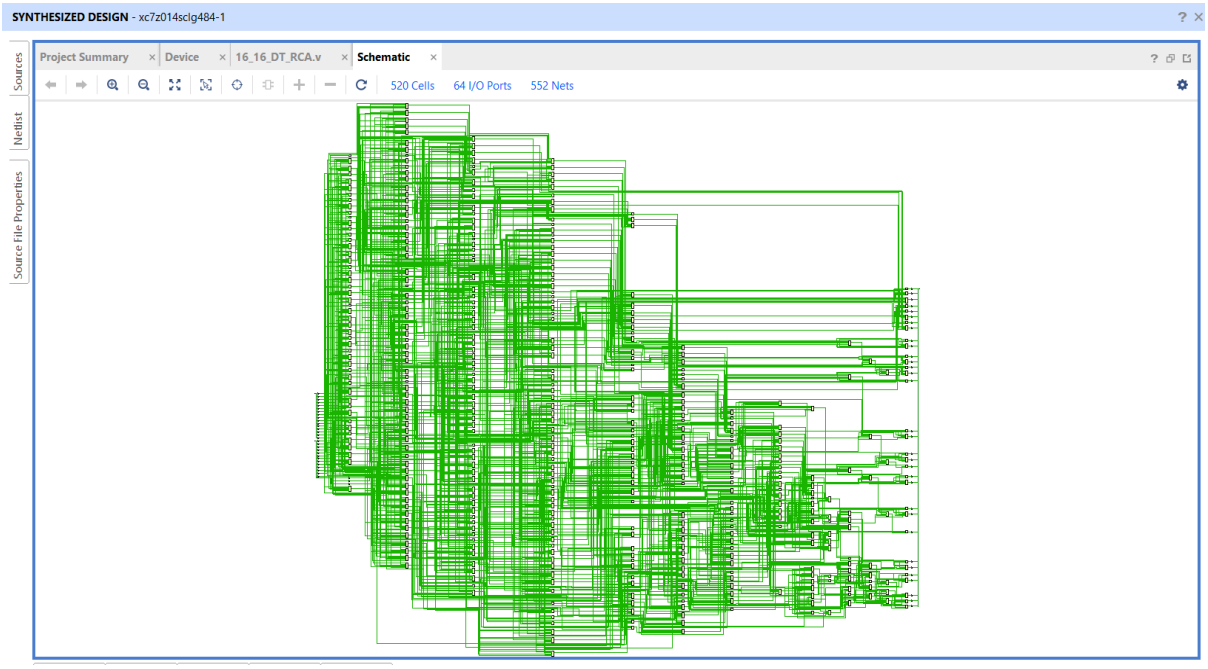


Fig. 4.4 Synthesized design of 16x16 Dadda multiplier with RCA

4.3.2 Dadda tree with CLA in final stage

RTL schematic and synthesized design of Dadda multiplier with CLA are shown in Figs. 4.5 and 4.6. In this design 435 LUTs and 128 slices are used. The delay, power and PDP are found to be 18.415 ns, 0.248 W and 4.566 nJ respectively.

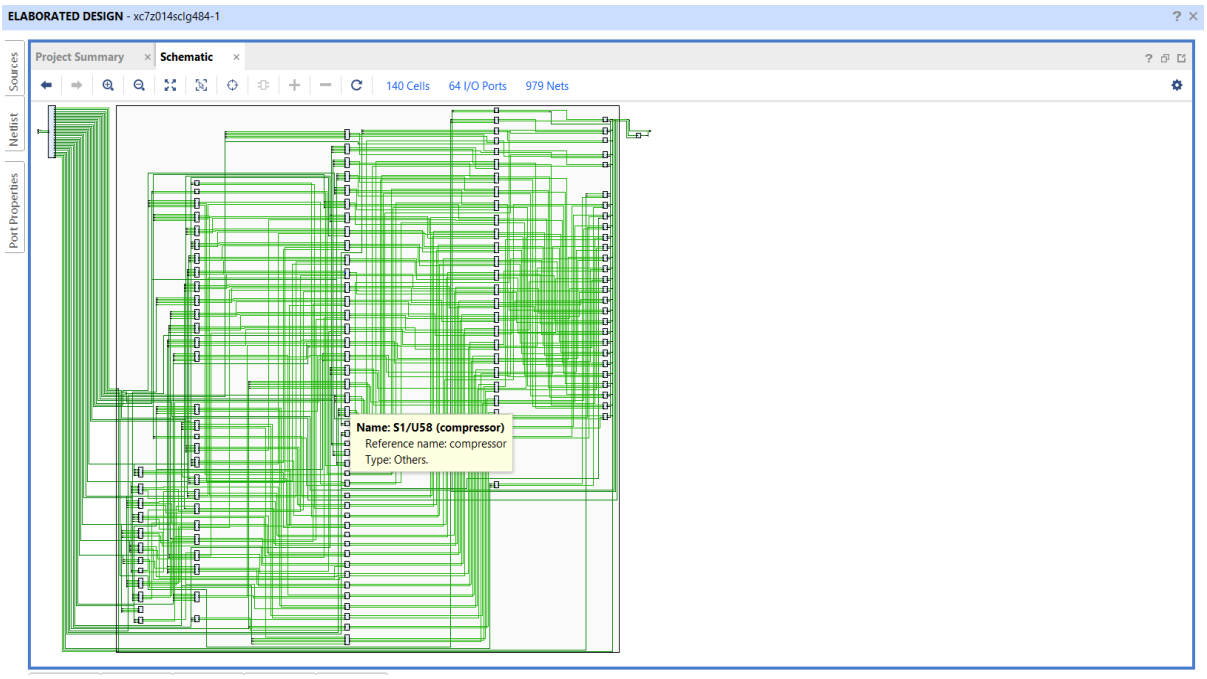


Fig. 4.5 RTL Schematic of 16x16 Dadda multiplier with CLA

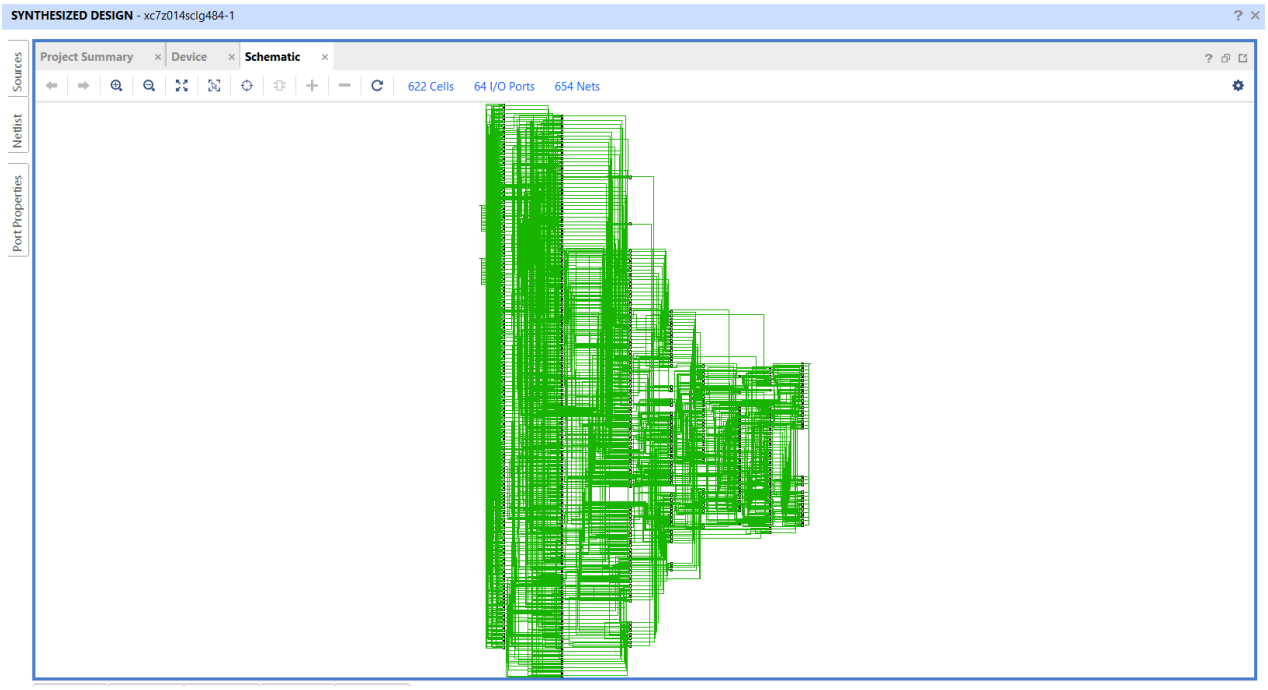


Fig. 4.6 Synthesized Design of 16x16 Dadda multiplier with CLA

4.3.3 Dadda tree with KSA in final stage

In Figs. 4.7 and 4.8 RTL schematic and synthesized design of Dadda multiplier with KSA is given. The design uses 452 LUTs and 131 slices. The delay of design is 17.575 ns and the power consumption is 0.251 W. Power delay product (PDP) of the design is found to be 4.411 nJ .

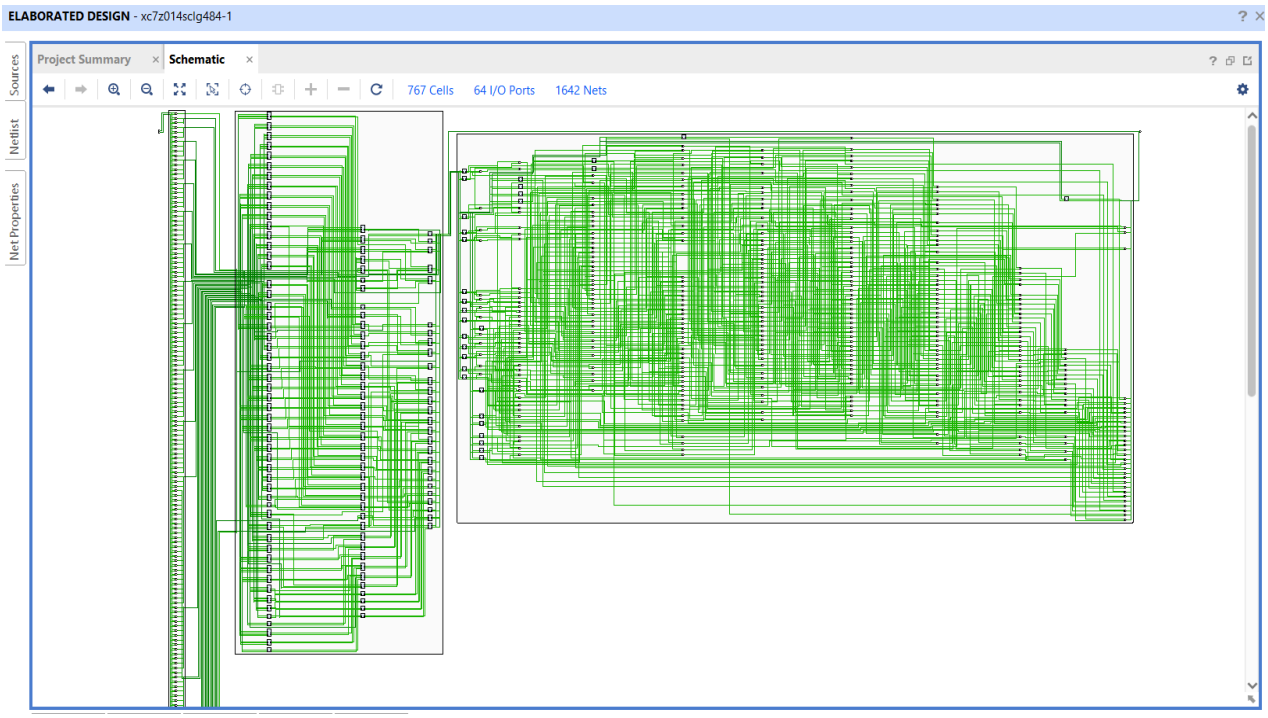


Fig. 4.7 RTL schematic of 16x16 Dadda multiplier with KSA

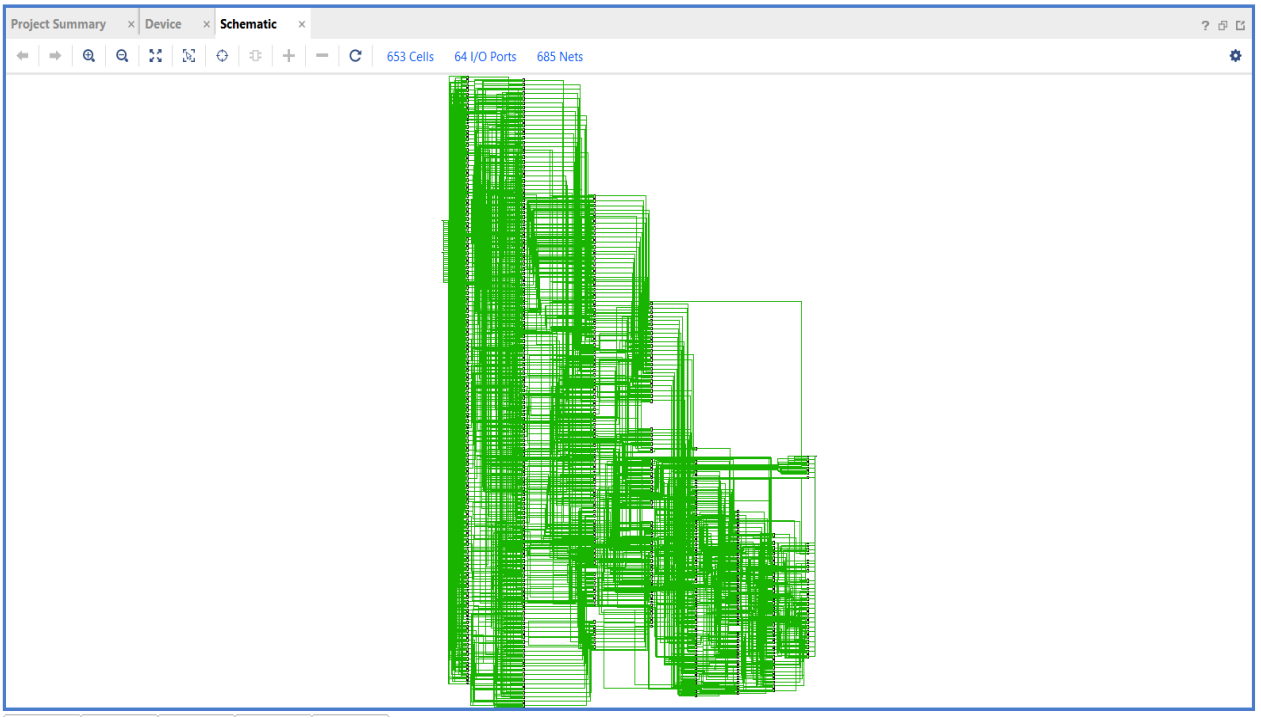


Fig. 4.8 Synthesized design of 16x16 Dadda multiplier with KSA

4.3.4 Dadda tree with BKA in final stage (Proposed Design)

RTL schematic and synthesized design of Dadda multiplier with BKA is shown in Figs. 4.9 and 4.10 respectively. In this design 402 LUTs and 118 slices are used. The delay, power and PDP are found to be 17.928 ns, 0.242 W and 4.338 nJ respectively.

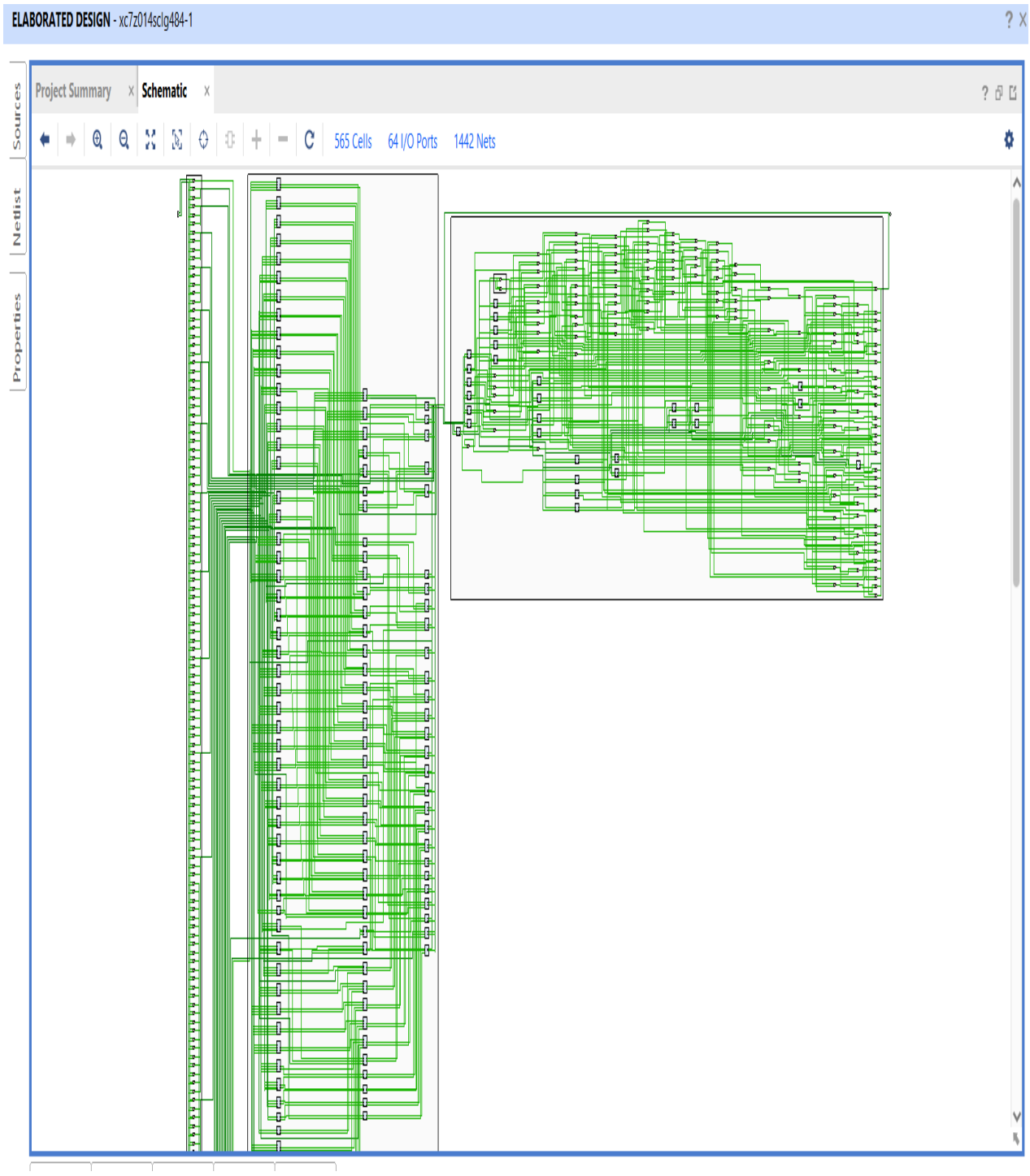


Fig. 4.9 RTL schematic of 16x16 Dadda multiplier with BKA

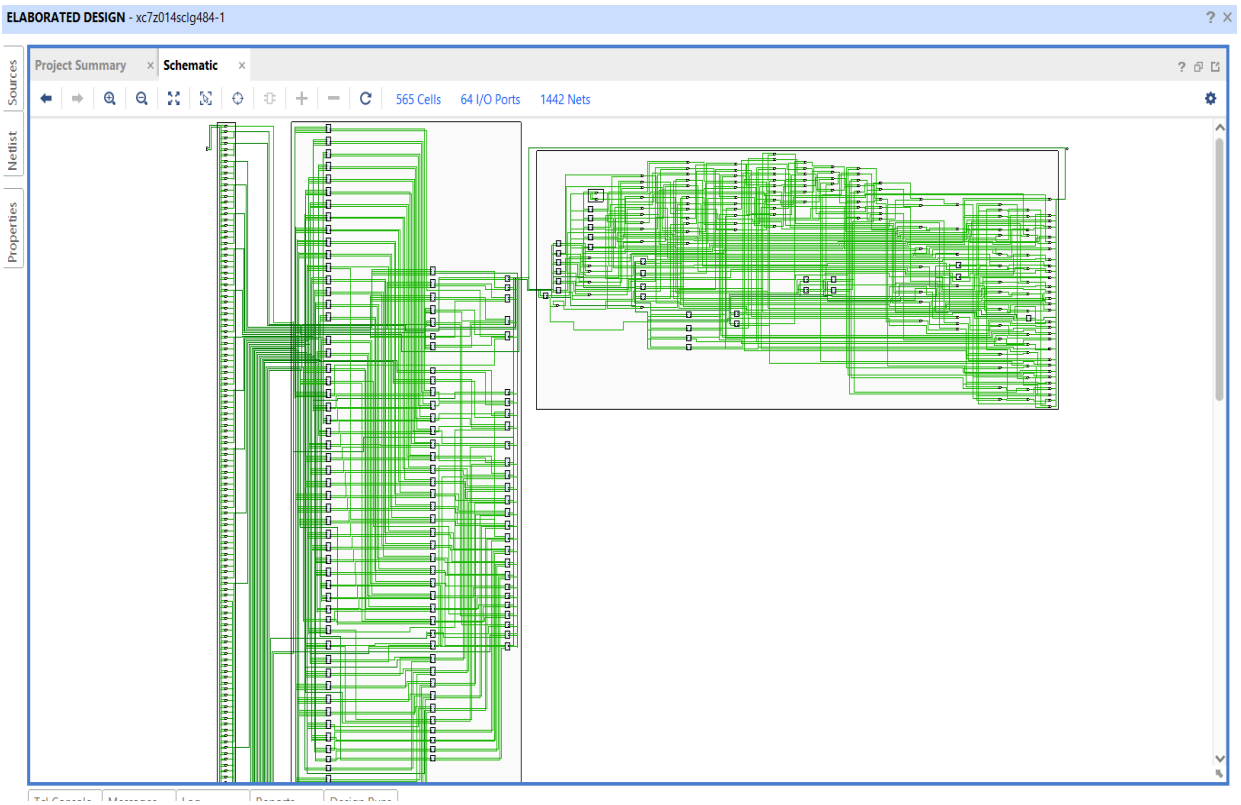


Fig. 4.10 Synthesized design of 16x16 Dadda multiplier with BKA

4.4 Comparison

The proposed design is functionally verified using simulation on VIVADO 2019.2 and selecting Zynq-7000 xc7z014sclg484-1 FPGA. For fairer comparison, Dadda multiplier is also implemented using RCA , CLA and KSA with and without 4:2 compressors. The designs are compared in terms of Look Up Tables (LUT) and Slices used; maximum delay, power and power delay product. The findings are summarized in Table 4.2. Following are the observations:

1. The delay is minimum for design with 4:2 compressors and Kogge Stone adder among all designs. However, it uses larger area.
2. The designs with RCA and Brent Kung adder with compressors use same number of slices, but the delay is minimum for the later one.
3. The proposed design has least power consumption among all designs.
4. There is only marginal difference between the delay of proposed design and the design with 4:2 compressors and Kogge Stone adder.
5. When three faster designs are compared i.e. 4:2 compressors with CLA, Kogge Stone adder and Brent Kung adder (proposed) it is observed that number of LUTs are minimum in proposed design. Therefore proposed design takes less area.

Table 4.2 Dadda Multipliers with 4-2 Compressors

Final Stage Adder	LUT	Slice	POWER (in Watt)	Maximum Delay(in ns)	Power-Delay Product (nJ)
Ripple Carry Adder	382	118	0.245	22.880	5.61
Carry Look Ahead Adder	435	128	0.248	18.415	4.566
Kogge Stone Adder [16]	452	131	0.251	17.575	4.411
Brent-Kung Adder (Proposed)	402	118	0.242	17.928	4.338

4.5 Summary

In this chapter, Dadda tree multiplier implementations with 4:2 compressors along with 3:2 and 2:2 compressors in reduction stage described. A new implementation of Dadda multiplier has been proposed in which 4:2 compressors are used in reduction stage and Brent Kung adder is used in final stage. Simulation results show that the proposed design has low power consumption among the designs employing RCA , CLA and KSA with and without 4:2 compressors. Dadda multiplier design with Kogge Stone adder has the high speed. Dadda multiplier design with RCA has less area than other designs.

CHAPTER-5

Conclusion

In this thesis different adders and multiplier designs are implemented using Verilog in Vivado by selecting Zynq-7000 xc7z014sclg484-1 FPGA. We learned about the different types of adders in chapter-2. We find out that Ripple Carry adder design utilizes the lowest chip area on the cost of speed and KSA is the fastest on the cost of area. Brent Kung adder takes lesser area if compare with Kogge Stone adder and it has marginal difference in terms of speed.

In chapter-3, the reduction process of Wallace tree and Dadda tree multiplier are studied. We find out that the functionality of Dadda Multiplier is similar to Wallace Tree Multiplier but Dadda multiplier is marginally faster in nature and it has less number of components than Wallace tree as depicted in Table 3.3.

A new Dadda multiplier design is proposed in chapter-4. The result shows 4:2 compressors improve the performance of Dadda multiplier and if parallel prefix adder is used in final stage then the designs have better power-delay product. The observation of Tables 3.3 and 4.2 show that the proposed design which is the combination of 4:2 compressors, full adder and half adder in reduction stage and the Brent-Kung adder in final stage has low power dissipation as compare to other designs. It consumes less area (Luts and slices) than the other faster designs consist of CLA and Kogge-Stone adder in final stage with compressors in reduction stage.

The proposed design has slightly more delay as compared with [16]. Therefore if Cost, Area and power is concern than proposed Dadda Multiplier design is a better choice. In future work further compressors like 5:2, 7:2 etc. can be used with or without including 4:2 compressors in reduction stage and different parallel prefix adders in final stage.

References

[1] L. Dadda, "Some schemes for parallel multipliers," in Alta Frequenza, Vol. 34, pp. 349-356, May, 1965.

[2] C. S. Wallace, "A suggestion for a fast multiplier," in IEEE Trans. Electronic Computers, Vol. 13, pp. 14-17, Feb., 1964.

[3] Dod, Shiwani, “Modified Booth Dadda Multiplier Using Carry Look Ahead Adder Design and Implementation,” in International Journal of Computer Science & Engineering Technology (IJCSET), ISSN : 2229-3345, Vol. 7, No. 3, March 2016.

[4] M. Munawar et al., "Low Power and High Speed Dadda Multiplier using Carry Select Adder with Binary to Excess-1 Converter," in 2020 International Conference on Emerging Trends in Smart Technologies (ICETST), Karachi, Pakistan, 2020, pp. 1-4, doi: 10.1109/ICETST49965.2020.9080739.

[5] Palaniappan, Ramanathan, Vanathi, P.T. Agarwal and S. kumar, “High Speed Multiplier Design Using Decomposition Logic,” in Serbian Journal of Electrical Engineering, 2009

[6] P. Anitha and P. Ramanathan, "A new hybrid multiplieusing Dadda and Wallace method", in 2014 International Conference on Electronics and Communication Systems (ICECS), Coimbatore, 2014, pp. 1-4, doi: 10.1109/ECS.2014.6892623.

[7] H. V. R. Aradhya, H. R. Madan, M. S. Suraj, M. T. Mahadikar, R. Muniraj and M. Moiz, "Design and performance comparison of adiabatic 8-bit multipliers," 2016 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), Mangalore, 2016, pp. 141-147, doi: 10.1109/DISCOVER.2016.7806237.

[8] P. Samundiswary and K. Anitha,” Design and Analysis of CMOS Based DADDA Multiplier,” in IJCEM International Journal of Computational Engineering & Management, Vol. 16 Issue 6, November 2013

[9] D. G. Crawley and G. A. J. Amaratunga, "8*8 bit pipelined Dadda multiplier in CMOS," in IEE Proceedings G - Electronic Circuits and Systems, vol. 135, no. 6, pp. 231-240, Dec. 1988, doi: 10.1049/ip-g-1.1988.0033.

[10] P. Gupta, A. Gupta and A. Asati, ” Ultra Low Power MUX Based Compressors for Wallace and Dadda Multipliers in Sub-threshold Regime,” American Journal of Engineering and Applied Sciences Volume 8, Issue 4,Pages 702-716,2015

[11] P.N.V.K. Hasini, T. K. Murthy, "A Novel high-speed transistorized 8x8 Multiplier using 4-2 Compressors," in International Journal of Engineering Research and General Science Volume 3, Issue 2, Part 2, March-April, 2015 ISSN 2091-2730

[12] O. Akbari, M. Kamal, A. A. Kusha and M. Pedram, "Dual-Quality 4:2 Compressors

for Utilizing in Dynamic Accuracy Configurable Multipliers," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1352-1361, April 2017, doi: 10.1109/TVLSI.2016.2643003.

[13] S. Ravi, G. S. Nair, R. Narayan and H. M. Kittur," Low Power and Efficient Dadda Multiplier," in Research Journal of Applied Sciences, Engineering and Technology 9(1): 53-57, 2015

[14] T. Arunachalam and S. Kirubaveni, "Analysis of high speed multipliers," 2013 International Conference on Communication and Signal Processing, Melmaruvathur, 2013, pp. 211-214, doi: 10.1109/iccsp.2013.6577045.

[15] B. Kumar, Potipireddi and A. Asati," Automated HDL generation of two's complement Dadda multiplier with Parallel Prefix Adders," in International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 6, June 2013

[16] M. A. Kumar, A. Sudhakar and J. V. Suman, "Design and Implementation of Compressor based 32-bit Multipliers for MAC Architecture," in International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-9, July 2019.

[17] A. Najafi, B. M. Nezhad and A. Najafi, "Low-power and high-speed 4-2 compressor," 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2013, pp. 66-69.

[18] S. Kumar and M. Kumar, "4-2 Compressor Design with New XOR-XNOR Module," 2014 Fourth International Conference on Advanced Computing & Communication Technologies, Rohtak, 2014, pp. 106-111, doi: 10.1109/ACCT.2014.36.

[19] D. Kumar and M. Kumar, "Modified 4-2 compressor using improved multiplexer for low power applications," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 236-242, doi: 10.1109/ICACCI.2016.7732053.

[20] Swartzlander, "Merged Arithmetic," in IEEE Transactions on Computers, vol. C-29, no. 10, pp. 946-950, Oct. 1980, doi: 10.1109/TC.1980.1675482.

[21] A. Raju, R. Patnaik, R. K. Babu and P. Mahato, "Parallel prefix adders-A comparative study for fastest response," 2016 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, 2016, pp. 1-6, doi: 10.1109/CESYS.2016.7889974.

[22] V. Kanimozhi and G. Shankar , "Design and implementation of Arithmetic Logic Unit (ALU) using modified novel bit adder in QCA," 2015 International Conference on

Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2015, pp. 1-6.
[23] H. Marzouqi, M. Qutayri, K. Salah, D. Schinianakis and T. Stouraitis, "A High-Speed FPGA Implementation of an RSD-Based ECC Processor," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 1, pp. 151-164, Jan. 2016.
[24] U. Penchalaiah and S. K. VG, "Design of High-Speed and Energy-Efficient Parallel Prefix Kogge Stone Adder," 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA), Pondicherry, 2018, pp. 1-7.
[25] Daphni, Samraj, Grace and Kasinadar, “Design and Analysis of 32-bit Parallel Prefix Adders for Low Power VLSI Applications,” in Advances in Science, Technology and Engineering Systems Journal, 2019.
[26] C. H. Chang, J. Gu and M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 10, pp. 1985-1997, Oct. 2004.
[27] M. Nagamatsu, S. Tanaka, J. Mori, K. Hirano, T. Noguchi and K. Hatanaka, "A 15-ns 32*32-b CMOS multiplier with an improved parallel structure," in IEEE Journal of Solid-State Circuits, vol. 25, no. 2, pp. 494-497, April 1990.
[28] G. C. Ram, D. S. Rani, R. Balasaikesava and K. B. Sindhuri, "Design of delay efficient modified 16 bit Wallace multiplier," 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2016, pp. 1887-1891, doi: 10.1109/RTEICT.2016.7808163.
[29] S. S. Sinthura, A. Begum, B. Amala, A. Vimala and V. V. Aparna, "Implemenation and Analysis of Different 32-Bit Multipliers on Aspects of Power, Speed and Area," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, 2018, pp. 312-317, doi: 10.1109/ICOEI.2018.8553859.