

Design and Implementation of an Efficient FFT Processor using Modified Booth Multiplier

¹A. Manimaran, ²Aby K. Thomas and ³S.K. Sudheer
 ¹Hindusthan University, Chennai, India
 ²Department of Electronics and Communication Engineering, Hindustan University, Chennai, India
 ³Department of Opto Electronics, University of Kerala, Kerala, India

Key words: CSLA, modified booth, slices, LUT (look up tables), delays, PPG

Corresponding Author: A. Manimaran *Hindusthan University, Chennai, India*

Page No.: 134-139 Volume: 10, Issue 5, 2017 ISSN: 1997-5422 International Journal of Systems Signal Control and Engineering Application Copy Right: Medwell Publications

INTRODUCTION

Xiaojin *et al.* OFDM is an efficient wireless standard that falls under the category of multi carrier modulation technique where sub carriers are orthogonal to each other. The orthogonality condition helps to eliminate cross talk between the sub channels. It simplifies the design of transmitter and receiver unlike conventional FDM.

Liu and Kosakowski (2014) The block diagram of OFDM is shown in Fig. 1. The cyclic encoding is done initially on the input data then the serial input data is converted to parallel data by using serial to parallel convertor and then send to signal mapper which is executed by performing modulation techniques (Su *et al.*, 2015; Jeon *et al.*, 2011) such as 32-QAM,64-QAM and 128-QAM. The interleaver block reduces the error bit to the encoder and also increase resistance to fading. To spread the errors out in the bit stream that is presented to the error correction decoder (Jiang *et al.*, 2007). Now, the interleaved symbol is

Abstract: The FFT plays an important role in OFDM regarding its performance. An efficient multiplier is needed to perform butterfly operation in FFT. This multiplier is related to area, power and time. The performance of OFDM depends upon the multiplier. The proposed multiplier is developed by using Verilog HDL and implemented by using Model Sim 6.3c for stimulation and Xilinx 12.4 for synthesis. The proposed multiplier reduce the number of slices, LTU (look up tables) and hence area and delay got reduced.



Fig. 1: OFDM modem

modulated into subcarrier by Inverse Fast Fourier Transform (IFFT) followed by cyclic pefix. The cyclic prefix serves two purposes as a guard interval, it eliminates the inter symbol interference from the previous symbol and also input the synchronization (Yang and Lee 2014). The original output is extracted for the receiver by demodulation process which is the reverse of all the above process.

Chan and Choy (2010) and More and Panat (2014) the advantages of using OFDM instead of single carrier is that it has the ability to cope with severe channel condition (Kim and Choi, 2008) such as attenuation of high frequency in long copper wire, interference narrow band, fading due to multi path without equalization filters. Yang and Lee (2014) OFDM is used in certain application like digital audio broadcasting (Chan and Choy, 2010), digital video broadcasting, IEEE 802.11 and so on.

MATERIALS AND METHODS

Problem identification: The power and speed of the Orthogonal Frequency Division Multiplier (OFDM) modem are dominated by IFFT/FFT Li et al. (2007) Yu et al. (2011). In specific, the performance of the Fast Fourier Transform (FFT) is degraded if the multiplier used in the FFT consumes more power, area and delay. Many works have been contributed such as pipelined FFT (Su et al., 2014), parallel FFT (Jeon et al., 2012), parallel-pipelined FFT (Huang and Chen, 2012) and serial FFT (Chang and Parhi, 2003). The entire works on FFT suggest an architectural level development that will improve the performance FFT. But a trivial solution is given to improve the performance of the FFT based on computational components (i.e., multipliers, adders and Partial Product Generator (PPG) architecture. This paper contributes to the improvement of FFT by developing an efficient multiplier to compute the butterfly operation, thereby improving the overall performance of OFDM.

Proposed multiplier: The proposed multiplier drive out the complexity of multipliers and adders present in the FFT which plays a critical component in OFDM that consumes more area and power. The layout the proposed efficient multiplier is shown in Fig. 2.

This multiplier comprises of Carry Select Adder (CSLA), Partial Product Generator (PPG) and enhanced logic of modified Booth logic with Bit Parallel Multiplication (BPM) concept. Let us consider M is the number of multiplier bit and N is the multiplicand bit then (MXN) produces M partial product and (M+N) product.

The proposed 16 bit multiplier produces 16 partial products which can be reduced to 8 by using booth encoder logic there by reducing area and power. The



Fig. 2: Proposed multiplier

Table 1 : Table for booth logic

Х	X-1		Action
0	0		No action, >>1
0	1		(U+Y)&>>1
1	0		(U-Y)&>>1
1	1		>>1
<u>U</u>	V	X	X-1
U	V	Х	X-1
0000	0000	0100	0
0000	0000	0010	0
0000	0000	0001	0
0000 0110	0000 0000	0001 0001	0 0
0000 0110 0011	0000 0000 0000	0001 0001 0000	0 0 1
0000 0110 0011 1101	0000 0000 0000 0000	0001 0001 0000 0000	0 0 1 1

enhanced logic of carry select adder will overcome the problem of propagation delay and place and route. Thus, by utilizing various logics such as CSLA, Booth, BPM, we have developed an efficient multiplier interms of area, power and delay.

Booth logic with BPM: The logic behind the modified booth is to reduce the partial product. Consider the basic table for the booth implementation as shown in Table 1.

Consider that X is the input to the multiplier. U is the initial value. Y is the value which is taken from the BPM module that is the twiddle factor value. Each twiddle factor is designed individual by the basic of BPM (bit parallel multiplication) concept. The output of BPM undergoes further multiplication on booth logic which reduces the complexity of booth multiplication. The reduction in area is achieved by reducing the partial product using booth logic and in addition the carry select adder is used to reduce the propagation delay Let us consider X = 0100, Y = 1010 (Table 2).

Partial product generator: Here, the booth logic is utilized to generate the reduced partial product. Figure 3



 $-a_0 \ a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ a_6 \ a_7 \ a_8 \ a_9 \ a_{10} \ a_{11} \ a_{12} \ a_{13} \ a_{14} \ a_{15} \times \ b_0 \ b_1 \ b_2 \ b_3 \ b_4 \ b_5 \ b_6 \ b_7$

Fig. 3: PPG for 16-bit multiplier



Fig. 4: PPG reduced by booth multiplier



Fig. 5: FFT processor

Table 3:	Performar	nce evaluati	ion							
Twiddle	Slices		LUTs		Delays (n sec)					
factor										
FFT	Existing	Proposed	Existing	Proposed	Existing	Proposed				
0.707	22	21	43	41	19.162	19.140				
0.707inv	24	19	45	37	19.682	18.377				
0.9239	27	15	51	29	20.792	17.969				
FFT	917	89	1662	1528	23.295	23.280				

shows the partial product generator for the proposed 16-bit multiplier. A total of 16 partial products is present in the diagram as an output of multiplication without booth logic. Further a total of 8 partial products are produce by using combined logic of booth and BPM. Thus, the 16 partial products are reduced to 8 as show in Fig. 4 and Table 3.

Carry select adder: The final stage of the partial product is added by using carry select adder which can compute the addition of two binary numbers at high speed. The novel logic behind the CSLA adder is that reduce the propagation delay. The addition process carried by using adders such as ripple carry adder, CLA, CSA produces 'n' propagation delay for 'n' bits. By using CSLA, (vn-1) propagation delay occurs for 'n' bits.

Implementation of proposed multiplier in FFT: The Fast Fourier Transform (FFT) architecture comprises of input and output buffer, FFT unit, butterfly unit, control unit and twiddle factor. The proposed efficient multiplier is a multiplier which is implemented in butterfly unit as shown in Fig. 5.

The butterfly unit computes the twiddle factor multiplication, where the twiddle factor is stored and retrieved from the buffer whenever it is required. The FFT unit is made up of butterfly unit where the proposed efficient multiplier is implemented the combination of booth logic with BPM module, CSLA adder will increase the speed of FFT operation. Here the input and output buffer is used to store the input bits and computed output bits, respectively.

The control unit which plays a vital role in FFT operation that controls the whole FFT processor. Clk_in, enable, FFT, start, ready, D_valid are the various signals used in the control unit. The input buffer is loaded when the start signal is high in the same manner the computed output is stored in the output buffer while the D_valid is high.

RESULTS AND DISCUSSION

Performance evaluation: The proposed multiplier and the fast fourier transform(FFT) core is developed using verilog HDL, implemented in software for simulation Model Sim 6.3C and for synthesis Xilinx 12.4. In this performance evaluation we analyzed the slices, LUTs (Look Up tables) and delay for various twiddle factor values and the whole FFT processor (Fig. 6 and 7).

1 🖬 🖶 🖉 🛛 🐒 🕯	08221	1	¢	0.3	44	+ I 🖬	100 p	- 1 E		0 0		45	±٠	1 2	ad 1 🗗	3+1 %	0.0				
Messag	es			1																	
/top_sixteen/dk	-No Data-																			کا دی ا	
/top_sixteen/rst																					
/top_soteen/data		13	14	15					_										_		
/top_sixteen/data		13	14	15					_										_		
/top_sixteen/data	∫No Data-	0		120	-8	-16	0	-28	4	-12	12	-48	-6	514	4	-20	-2	-10	32	0	
/top_sixteen/data,	iNo Data-	0		120	-8	þ	-16	12	-12	4	-28	32	+10	-2	-20	4	514	-6	-48	10	
/top_sixteen/data,	rNo Data-	9	10	111	12	13	14	15	_										_		
🍫 /top_sixteen/data,	jNo Data-	9	10	11	12	13	14	15	_									- -	_		
/top_sixteen/data	rNo Data-	5	ó	7	3	9	10	<u> </u>	12	13	14	15							_		
🧇 /top_sixteen/data,	JNo Data-	5	6	7	8	9	10	a	12	13	14	15							_		
/top_sixteen/data,	JNo Data-	1	2	3	4	15	6	7	3	9	10	11	12	13	14	15			_		
🤣 /top_sixteen/data,	iNo Data-	1	2	3	4	15	6	7	3	9	10	11	12	113	[14	15			_		
🥠 /top_sixteen/rin_a		10	12	14	16	18	20	22	23	24	25	26	27	28	29	30			_		
/top_sixteen/rin_a		18	20	22	23	24	25	26	27	28	29	30							_		
🍫 /top_sxteen/lin_a		10	12	14	16	118	120	22	23	24	28	26	27	23	29	30					
/top_sixteen/in_ai		18	20	22	23	24	25	26	27	23	29	30									
👉 /tap_sixteen/rin_s		-8							-7	-6	-5	1 4	+3	1-2	-1	0			_		
🤣 /top_sixteen/rin_s		-8	-8	-8	-7	-6	-5	4	-3	2	-1	0							_		
🍫 /top_sixteen/lin_si		-8							-7	6	5	-4	-3	-2	-1	0					
🍫 /top_sixteen/in_si		-8	-8	-8	-7	1-6	-5	-4	-3	2	-1	0									
/top_sixteen/rin_m		23	32	36	39	42	45	48	50	52	54	56	57	58	59	60					
🌖 /top_sixteen/rin_m	1 -No Data-	-12	8	0	0				_										_		
/top_sixteen/rin_m		-15	511	-6	10				_										_		
/top_sixteen/tin_m		-15	511	6	10				_										_		
/top_sixteen/in_m		73	32	36	39	42	45	48	50	52	54	56	57	58	\$9	60			_		
/top_sixteen/in_m		0	3	12	10				-										_		
🌛 /top_soteen/lin_m		5	11	15	0				_										_		
🌖 /top_sixteen/in_m		-6	10	15	10				_										_		
/tap_sixteen/fin_m	uNo Data-	-8							-7	16	15	-4	1-3	1-2	-1	0			_		
🧄 /top_sixteen/lin_m	ul_0 -No Data-	3							7	8	5	4	3	12	1	0			_		
/top_sixteen/rin_m	00 -No Data-	-8	18	-8	-7	-6	5	4	-				-3	12	61	0			-		
/top_sixteen/rin_m		-12	-12	-12	18		-6						1-2		0						
/top_sixteen/rin_m	02 -No Data-	-8	-8	18	07	-6	1-5	4					13	2	-1	0					
A /top_sixteen/in_m	03 No Data-	0	0	0	10		0						10		0						
			alman a		rdimmer			nition of	alarea a		diameter (rd house		all and a		dama	dumm			THE OWNER
10		Shield I																ins i			
Cursor	3.4	ins ins																			
	► <	21																	_		_

Fig. 6: The simulation output of FFT processor

n +□∂×	0	Design Overview		ton sivi	een Project	Status (03/15/	2016 - 11	-35-39)			
View: 💿 🏭 Implementation 💿 🌆 Simulation		Summary	Project File:	check vise	p	arser Froms		No.	Ferrore		
Hierarchy *	0	Module Level Utilization	Modula Namer	tro syleen	-	Implementation State:			Placed and Routed		
mux_16 (mux_16.v)	0	Timing Constraints	Target Device:	yr3s200-5eo208							
Vor top_sixteen (top_sixteen.v)	Ø	Pinout Report Clock Report	Product Version:	ISF 12.4		• Warnings	23	131	Naroinos	(12 new)	
fd2 - four_d (four_d.v)		Static Timing	Design Goal:	Balanced		• Routing F	esults:	ALS	ionals O	ompletely Routed	
fd3 - four_d (four_d.v)	\$B	Errors and Warnings	Design Strategy:	Xilinx Default (unlocked)		Timing Constraints: Final Timing Score:			Constrain	ts Met	
B V fd4 - four_d (four_d.v) B V fd5 - four_d (four_d.v)	A	Synthesis Messages	Environment:	System Settings					0 (Tming Report)		
 v roor_o (roor_o.v) v sal - sign_add (sign_add.v) v sal - sign_add (sign_add.v) 		Map Messages Place and Route Messages									
sa3 - sign_add (sign_add.v)		Timing Messages	a color and the states	De	vice Utilizati	on Summary		lane a			Ŀ
sə4 - sign_add (sign_add.v) 🖕		Bitgen Messages	Logic Utilization		Used	Available		Utilization	N	ote(s)	
No Processes Running		Detailed Reports	Number of size Hip Hops Number of 4 input UUTs Number of occupied Sizes Number of Elizer containing only related logic			123	3,840	-	11%		_
		Synthesis Report				0.40	3,040	-	2776		
Derion Summary/Reports		Map Report *			-	0.40	0.40		100%		
Design Utilities		Design Properties	Number of Sizes contain	ing only related logic		0	849	-	0%		_
User Constraints		Enable Message Filtering	Total Number of 4 insuit 11	Te		528	3.840	-	3046		
View RTL Schematic		Optional Design Summary Contents Show Clock Report Show Failing Constraints Show Warnings	Number used as logic Number used as a route-thru			447	0/0 10				
View Technology Schematic					1	1			-		
Generate Port Synthesis Si			Number used as Shift re	visters	80	80			-		_
😑 🔥 Implement Design		- Show Errors	Number of bonded IOBs	of bonded ICBs		34 141	2	24%			
Comparison Translate			Number of BLFGMUXs		1	1	1 8		12%		
E Place & Route			Average Fanout of Non-Cl	ock Nets		.05		1			
Start 🎕 Design 🚺 Files 🚺 Ubraries	Σ	Design Summary (Implemented)		new_mulu.v							
sle											++ 🗆 é
otal time: 2 secs rocess "Generate Post-Place & F	lout	e Static Timing" completed succes	efully								

Fig. 7: The output of FFT processor which shows reduction of area

Int. J. Syst. Signal Control Eng. Appl., 10 (5): 134-139, 2017



Fig. 8: The output of FFT processor which shows the reduction in delay



Fig. 9: The output of FFT processor which shows the reduction in power

Slices refers to the statement condition, if slice reduces the number of wires used in the coding also get reduced. LUTs is the gate count. By comparing the existing system without modified multiplier and the proposed multiplier, the slices and LUTs get reduced compare to the design by Su *et al.* (2014). So that, the overall propagation delay and the power consumption are also reduced (Fig. 8 and 9).

CONCLUSION

The Fast Fourier Transform with efficient multiplier design is proposed in this paper. The proposed multiplier improves the performance of FFT interms of area, power and speed. The proposed multiplier reduces the delay and area which in turn improves the overall performance of OFDM as OFDM depends upon the FFT processor. Hence the proposed FFT processor outperforms the conventional FFT architecture. The proposed FFT processor is developed by using Model Sim-6.3C and Xilinx-12.4 interms of verilog HDL.

RECOMMENDATION

In future the multiplier proposed in this study can be implemented in various architecture.

REFERENCES

Chan, P.W. and C.S. Choy, 2010. Performance evaluation of OFDM de-modulator with various multiplier architectures for UWB system. Proceedings of the 2010 IEEE Asia Pacific Conference on Circuits and Systems, December 6-9, 2010, IEEE, Kuala Lumpur, Malaysia, pp: 418-421.

- Chang, Y.N. and K.K. Parhi, 2003. An efficient pipelined FFT architecture. IEEE. Trans. Circuits Syst. II: Analog Digital Signal Process., 50: 322-325.
- Huang, S.J. and S.G. Chen, 2012. A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems. IEEE. Trans. Circuits Syst. I: Regul. Pap., 59: 1752-1765.
- Jeon, D., M. Seok, C. Chakrabarti, D. Blaauw and D. Sylvester, 2011. A super-pipelined energy efficient subthreshold 240 MS/s FFT core in 65 nm CMOS. IEEE. J. Solid-State Circuits, 47: 23-34.
- Jiang, M., B. Yang, R. Huang, T.Y. Zhang and Y.Y. Wang, 2007. Multiplierless fast Fourier transform architecture. Electron. Lett., 43: 191-192.
- Kim, D. and H.W. Choi, 2008. Advanced constant multiplier for multipath pipelined FFT processor. Electron. Lett., 44: 518-520.
- Li, X., Z. Lai and J. Cui, 2007. A low power and small area FFT processor for OFDM demodulator. IEEE. Trans. Consum. Electron., 53: 274-277.

- Liu, X. and M. Kosakowski, 2014. Max-log-MAP soft demapper with logarithmic complexity for \$M \$-PAM signals. IEEE. Signal Process. Lett., 22: 50-53.
- More, T.V. and A.R. Panat, 2014. FPGA implementation of FFT processor using vedic algorithm. Proceedings of the 2013 IEEE International Conference on Computational Intelligence and Computing Research, December 26-28, 2013, IEEE, Enathi, India, pp: 1-5.
- Su, X., L. Xi, X. Tang, Z. Zhang, S. Bai, W. Zhang and X. Zhang, 2014. A multistage CPE scheme based on crossed constellation transformation for M-QAM. IEEE. Photonics Technol. Lett., 27: 77-80.
- Yang, S.W. and J.Y. Lee, 2014. Constant twiddle factor multiplier sharing in multipath delay feedback parallel pipelined FFT processors. Electron. Lett., 50: 1050-1052.
- Yu, C., M.H. Yen, P.A. Hsiung and S.J. Chen, 2011. A low-power 64-point pipeline FFT/IFFT processor for OFDM applications. IEEE. Trans. Consum. Electron., 57: 40-40.