# Error Correction Technique for Multiple Errors in Parallel FIR Filters

Amritha P.S<sup>1</sup>, Ashwath Rao<sup>2</sup>

<sup>1</sup>ECE dept/ Sahyadri College of Engineering and Management, India <sup>2</sup>Associate Professor ECE dept/ Sahyadri College of Engineering and Management, India

**Abstract:** Filters are broadly utilized as a part of communication and signal processing systems. At times, reliability becomes critical for such systems and fault tolerant usages are required. Over a long time, numerous procedures have been proposed that makes full use of filters structure and properties to accomplish fault tolerance. In certain complex systems, it is commonly found that filters work in parallel. For instance, by applying the same filter to various information signals. In this brief, idea is summed up to demonstrate that protection of parallel filters can be done by utilizing error correction codes (ECCs) by detecting and correcting multiple errors. This new scheme permits more effective assurance if the number of parallel filters is large. Proposed structure is coded using Verilog and synthesized in Xilinx 10.1 and cadence encounter is done. **Keywords:** Error Correction Codes (ECCs), Filters, and multiple bits.

# I. Introduction

Electronic circuits are progressively present in medical, automotive, space applications etc where reliability of the circuit is critical. In order to protect circuits from errors, fault tolerant implementation need to be done. With the improvement in VLSI technology, the ongoing need of FIR filters with less hardware usage giving error free results and less latency has turned out to be more important. Various methods can be utilized to shield a circuit from errors. One among the general techniques which can be utilized to correct the error is triple modular redundancy (TMR) [1]. The TMR technique adds redundancy by tripling the data and utilizes voting logic to correct errors. Because of the triplication in the technique, it increases area and power of the circuit, something that may not be accepted in a few applications.

If the circuit which has to be protected has structural or algorithmic properties, other techniques can be used in protecting circuits from errors. One case is signal processing circuits for which Sum of Square (SOS) check method have been proposed [2]. This is an error detection technique. It cannot detect all the faults. A few systems have been proposed to protect filters from errors. Digital filters utilized as a part of the circuit are finite-impulse response (FIR) filters. For instance, reduced precision replicas were utilized in FIR filters to reduce the expense of implementing modular redundancy [3]. In [4], to recognize errors, relation between the inputs provided to a FIR filter and the memory elements of a FIR filter are utilized. To protect filters the utilization of residue number [5] and arithmetic codes [6] has likewise been proposed. Different implementation structure of FIR filter is used to correct the errors by using one redundant module has been proposed [7]. So far mentioned techniques are used in the protection of single filters.

It is quite common in complex systems to find many filters work in parallel. Several modern communication systems [8] and filter banks [9] has this situation. For these systems, protection of filters will be at higher level by treating parallel filters as a block that needs to be protected. This thought was explored in [10], where two parallel filters with same impulse response fed with different input sequences were considered. It was shown that with only one redundant copy, single error could be corrected. In [11], detection and correction of single fault in multiple modules of parallel FIR filter is done. So far considered schemes are used in detecting and correcting single errors.

In this brief, a simple technique to protect parallel FIR filters from multiple errors is exhibited. As mentioned in [11], same response given to parallel filters with distinctive input sequence is considered. The new scheme depends on the utilization of error correction codes (ECCs) such that each of the output from filter is similar to a bit in ECC codeword. This is a generalization of the plan exhibited in [11] and empowers more effective executions when the number of parallel filters is large. This scheme can likewise be utilized to provide multiple error detection and correction in parallel filter output.

## II. Parallel Filters Having Same Impulse Response

FIR filter response is given by following equation

$$Y[n] = \sum_{l=0}^{\infty} X[n-l] \cdot h[l]$$
(1)

Where X[n] is the input, Y[n] is the output, and h[1] is the impulse response of the filter. At the point when the impulse response h[1] is nonzero, just for a limited number of samples, then the filter is called as FIR filter.

н

v₁[n]



Fig.1. same impulse response given to the parallel filters

Here, a set of 'u' parallel filters with different input signals and same response are taken. These parallel FIR filters are shown in fig 1. Communication systems which use several channels have these kinds of filters in it. An intriguing property for these parallel filters is that the addition of any combination of the output  $Y_i[n]$  can likewise be obtained by sum of the relating inputs  $X_i[n]$  and by multiplying the subsequent signal with the same response h[1]. For instance,

$$Y_1[n] + Y_2[n] = \sum_{l=0}^{\infty} X_1[n-l] + X_2[n-l] \cdot h[l]$$
(2)

This observation will be utilized in order to develop the proposed error correction scheme.

#### **Proposed System** III.

The new procedure depends on the utilization of the ECCs. A basic ECC considers a block of 'u' information bits and produces a block of 'n' bits by including n-u parity check bits. The parity check bits are XOR combination of the 'u' information bits. By correctly designing those combinations it is possible to identify and correct the errors. As a case, let us consider u = 4 and n = 7

For this case, three parity check bits are P1, P2, and P3 that is obtained by the XOR combination of information bits  $d_1$ ,  $d_2$ ,  $d_3$ ,  $d_4$  as follows:

$P_1[n] = d_1[n] \bigoplus d_2[n] \bigoplus d_3[n]$	(3)
$\mathbb{P}_2[n] = d_1[n] \bigoplus d_2[n] \bigoplus d_4[n]$	(4)
$P_{\mathfrak{z}}[n] = d_1[n] \bigoplus d_{\mathfrak{z}}[n] \bigoplus d_4[n]$	(5)

The information and parity check bits are accumulated and can be recouped later regardless of the possibility that there are multiple errors. This is done by calculating the parity check filters and comparing the outcomes with the information stored. For example, errors in d<sub>1</sub> will cause errors on the three parity checks; errors on d<sub>2</sub> will cause errors in P<sub>1</sub> and P<sub>2</sub>; errors on d<sub>3</sub> will cause errors in P<sub>1</sub> and P<sub>3</sub>; and finally errors on d<sub>4</sub> will cause errors in P2 and P3. Therefore, errors in the information bits can be identified and the errors can be corrected.



Fig.2. Proposed structure for four parallel FIR filters with redundant modules used for multiple error correction

The inputs given to four filters are  $X_1, X_2, X_3$  and  $X_4$  in which each input has four coefficients which is of 16 bits each. For example,  $X_1$  has four coefficients i.e.  $X_1[0], X_1[1], X_1[2], X_1[3]$  in which each coefficient is of 16 bits. It is similar for the rest of the inputs.

The check filters Zj is obtained for the parallel filters by applying ECC scheme. Check filters is obtained by XOR combination of the inputs. Because of the combination of inputs, each of the check filter having four coefficients will be of 18 bits each. Thus each filter output will be having seven coefficients which are of 36 bits each.

In case of four filter outputs  $Y_1$ ,  $Y_2$ ,  $Y_3$ , and  $Y_4$ , the parity check filters are given below

$$Z_1[n] = \sum_{l=0}^{\infty} (X_1[n-l] + X_2[n-l] + X_3[n-l] \cdot h[l]$$
(6)

$$Z_{2}[n] = \sum_{l=0}^{\infty} (X_{1}[n-l] + X_{2}[n-l] + X_{4}[n-l] \cdot h[l]$$
(7)

$$Z_{\mathfrak{z}}[n] = \sum_{l=0}^{\infty} (X_{\mathfrak{z}}[n-l] + X_{\mathfrak{z}}[n-l] + X_{4}[n-l] \cdot h[l]$$
(8)

Error detection method is performed to distinguish if faults has occured during the transmission of data starting with one point to other. In this system it can distinguish any number of faults in a module.

error detection is done by checking if  

$$Z_1[n] = Y_1[n] + Y_2[n] + Y_3[n]$$
(9)

$$Z_2[n] = Y_1[n] + Y_2[n] + Y_4[n]$$
(10)

$$Z_{3}[n] = Y_{1}[n] + Y_{3}[n] + Y_{4}[n]$$
(11)

For instance, faults in the filter output  $Y_1$  will create errors on the check filter output  $Z_1$ ,  $Z_2$ , and  $Z_3$ . In the similar manner errors in  $Y_2$  will create errors in the output of check filter  $Z_1$  and  $Z_2$ ; errors in  $Y_3$  will cause errors in  $Z_1$  and  $Z_3$ ; errors in  $Y_4$  will cause errors in  $Z_2$  and  $Z_3$ .

Overall system is shown in fig 2. It is observed that correction of multiple errors can be done by only three redundant filters. In filters, error correction is done by recreating the outputs which has multiple errors using rest of the information and parity check outputs.

For example, when errors on  $Y_1$  is identified, it can be corrected by  $Y_{C1}[n] = Z_1[n] - Y_2[n] - Y_3[n]$ (12)

When errors on  $Y_2$  is identified, it can be corrected by  $Y_{C2}[n] = Z_2[n] - Y_1[n] - Y_4[n]$ (13)

When errors on Y<sub>3</sub> is identified, it can be corrected by

$$Y_{C3}[n] = Z_3[n] - Y_1[n] - Y_4[n]$$
 (14)

When errors on Y<sub>4</sub> is identified, it can be corrected by

$$Y_{C4}[n] = Z_2[n] - Y_1[n] - Y_2[n]$$
 (15)

This technique can be used for any number of parallel filters. The methodology is more appealing when the filters 'u' is large in number. Because of this it slightly affects the complexity of the circuit.

The advantages are bigger for this second configuration. In that case, the relative number of included check filters (n-u)/n are smaller.

For example, when u = 11, only four redundant filters are required to correct multiple errors in a module. When u = 11, n = 15, the inputs given to eleven filters are  $X_{1,} X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}$  in which each input has four coefficients which is of 16 bits each. For example,  $X_1$  has four coefficients i.e.  $X_1[0]$ ,  $X_1[1], X_1[2], X_1[3]$  in which each coefficient is of 16 bits. It is similar for the rest of the inputs.

The check filters Zj is obtained for the parallel filters by applying ECC scheme. Check filters is obtained by XOR combination of the inputs. Because of the combination of inputs, each of the check filter having four coefficients will be of 19 bits each. Thus each filter output will be having seven coefficients which are of 38 bits each.

# **IV.** Simulation Results

## 4.1 Results of four parallel filters

The proposed structure output of the four parallel FIR filter is analyzed. It is coded using Verilog and synthesized on Xilinx ISE 10.1 and cadence encounter is obtained. The proposed structure synthesized on Xilinx ISE 10.1 is shown in the fig. 3 and fig 4. Active low indicates there is error in the corresponding signal. Error detection is shown in fig. 5, error correction is shown in fig. 6, and physical design of the proposed scheme is shown in fig 7 respectively.

Current Simulation Time: 1000 ns		900 ns 905 ns	910 ns	915 ns	920 ns	925 ns	930 ns	935 ns	s 940 ns	945 ns	950 ns	955 ns	960 ns	965 ns	970 ns	975 ns	980 ns	985 ns	990 ns	995 nd 000 ns
🗖 🚮 hh0[17:0]	1										18'h0FFFF									
🗖 😽 hh1[17:0]	1																			
🖬 😽 hh2[17:0]	1										18'h0FFFF									
🗉 😽 hh3[17:0]	1										18'h0FFFF									
🖬 😽 x10[17:0]	1										18'h0FFFF									
🗖 😽 x11(17:0)	1										18'h0FFFF									
🗖 🚮 x12(17:0)	1										18'h0FFFF									
🖬 🚮 x13(17:0)	1										18'h0FFFF									
😐 💸 x20[17:0]	1										18'h0FFFF									
🖬 🚮 x21[17:0]	1																			
🖬 🚮 x22[17:0]	1										18'hOFFFF									
🗖 🚮 x23[17:0]	1										18'h0FFFF									
🖬 🚮 x30[17:0]	1										18'h0FFFF									
🖬 🚮 x31[17:0]	1										18'h0FFFF									
🗉 🚮 x32[17:0]	1																			
🖬 🚮 x33[17:0]	1										18'hOFFFF									
🗖 🚮 x40[17:0]	1										18'h0FFFF									
🗖 🚮 x41[17:0]	1										18'h0FFFF									
🖬 🚮 x42[17:0]	1										18'h0FFFF									
🖽 🔜 x43(17:0)	1										1850FFFF									

Fig 3. Inputs given to the filters having same impulse response

Current Simulation Time: 1000 ns		800 ns005 ns 910 ns 915 ns 920 ns 925 ns 930 ns 935 ns 940 ns 945 ns 950 ns 955 ns 960 ns 965 ns 970 ns 975 ns 980 ns 985 ns 990 ns 995 nd 000 ns
🖬 🚮 yc1 0[35:0]	3	36h0FFE0001
🗉 🚮 yc 20[35:0]	3	36h1FFFC0002
🗉 🚮 yc30[35:0]	3	36 h2FFF40003
🗉 🚮 yc40[35:0]	3	36 h3FFF80004
🗉 😽 yc50[35:0]	3	36 h3FFF40003
🖬 😽 yc60[35:0]	3	36h1FFFC0002
🖬 🚮 yc70[35:0]	3	36h0FFFE0001
🖬 🚮 yc11[35:0]	3	36h0FFFE0001
🖪 😽 yc 21 [35:0]	3	36h1FFFC0002
🗉 😽 yc31 (35:0)	3	36h2FFFA0003
🗄 😽 yc41 (35:0)	3	36%3FF680004
🗄 😽 yc51 (35:0)	3	36h/2FF40003
🖽 😽 yc61 (35:0)	3	36h1FFFC0002
🖽 😽 yc71 [35:0]	3	36h0FFFE0001
🖬 😽 yc12[35:0]	3	36h0FFFE0001
🖬 💏 yc 22[35:0]	3	36h1FFFC0002
🖬 🚮 yc32[35:0]	3	36h2FFFA0003
🖬 😽 yc42[35:0]	3	36%3FF580004
🖬 😽 yc52[35:0]	3	36h2FFFA0003
🗄 😽 yc62[35:0]	3	36h1FFFC0002
🖬 🚮 yc72[35:0]	3	38h0FFFE0001
🖬 😽 yc13[35:0]	3	38h0FFFE0001
🖬 🚮 yc23[35:0]	3	38h1FFFC0002
🖬 💏 yc33[35:0]	3	36h2FFFA0003
🖬 😽 yc 43[35:0]	3	36%3FFF80004
■ Styc53[35:0]	3	361/2FF#40003
• 🕅 yce3[35:0]	J	361111102
• of vc73(35:0)	3	36'h0FFFE0001

Fig 4. Error free output obtained for the input provided

			227	27/4 06
Current Simulation Time: 1000 ns		170 ns 180 ns 190 ns 201	ns 210 ns 220 ns	230 ns 240 ns 250 ns 260 ns 270 ns 280 ns 290 ns 300 ns 310 ns 320 ns 330 ns 340 ns 350 ns 3
all err10	0			
🎉 errt 1	0			
err12	0			
🎉 🛛 errt 3	0			
all errt 4	0			
🏹 err1 5	0			
orr16	0			
🏹 err20	0			
all err21	0			
öll err22	0			
all err23	0			
öll err24	0			
all err25	0			
👌 err26	0			
an 30	0			
👌 err31	0			
åll en 32	0			
SII err33	0			
all err34	0			
SII err35	0			
200026	0			

**Fig 5.** Errors in  $Y_1$  causing errors in  $Z_1, Z_2, Z_3$ 







Fig 7. Physical design of four parallel FIR filters

# 4.2 Results of 11 parallel filters



Fig 8. Inputs given to the filters having same impulse response

🖶 🔂 ve10(37:0)			
- Griffer of or rol	3		
🖬 🚮 yc20[37:0]			
🖬 🚮 yc30[37:0]	3		
wc40137:01	3		
<b>B B</b> (100 <b>E</b> (100	-		
• • • • • • • • • • • • • • • • • • •	ə		
a 🛐 ye60(37:0)	3		
🖬 😽 yc70[37:0]			
🖬 🚮 yc11 (37:0)	3		38h00FFFE0001
Mur 21 (37:01	3		
	0		
a Station of	ə		38 RU2FFFAUUU3
a 💸 yc41 [37:0]	3		
🛚 🚮 yc 51 (37:0)			
🛚 🛃 yc61 [37:0]	3		
vc71[37:0]	3		
W und 2027-01	-		
0 yczz[3/:0]	3		38M01FFFC0002
yc32[37:0]	3		
🛚 😽 yc42[37:0]	3		
Vc52[37:0]	3		
wr62137-01	3		
	0		
yer2[37:0]	3		
vc13[37:0]	3		36100FFFE0001
l 🚮 yc23[37:0]	3		38h01FFFC0002
vc33[37:0]	3		
M vc43137:01	3		
			3010311130009
- wc53[37.0]	3		38mu2FFFA0003
<b>54</b> yc63[37:0]	3		38h01FFFC0002
🛃 yc73[37:0]	3		
😽 yc14[37:0]	3		
M vc24/37:01	3		
Muc24107-01	3		
0 9 9 5 4 [37 .U]	- J		
yc44[37:0]	3		38'h03FFF80004
😽 yc54[37:0]	3		
😽 yc64(37:0)	3		38h01FFFC0002
M vc74[37:0]	3		
Mun 16(27-0)	2		
St versisrioj	э		
i <mark>⊚1</mark> yc25[37:0]	3		38h01FFFC0002
l 😽 yc35(37:0)	3		
😽 yc45[37:0]	3		
vc55(37:0)	3		
We8507:01	2		
a yeosisrioj	3		
M Act 2[31:0]	3		38/h00FFFE0001
🛿 🔂 yc16[37:0]	3		
🛚 😽 yc26[37:0]	3		
l 🚮 ye36[37:0]	3		
Mur 46(37:0)	3		
	0		
S 4000[37.0]	3		
l 🚮 yc66(37:0)	3		
🔊 vc76[37:0]	3		
😹 yc17(37:0)			
	3		
vc27/37:01	3 3		amorrecoma amorrecom amorrecom amorrecom amorrecom
vc27[37:0]	3 3		3eh00FFFC002 3eh00FFE0001 3eh00FFE0001 3eh00FFE0002
vc27[37:0]	3 3 3		##10775500     #5007     #5007     #5007     #5007     #5007     #5007     #5007     #5007     #5007     #5007     #5007     #5007     #5007
yc27[37:0]	3 3 3 3		BenofFFC.0003     SeholfFFC001     SeholfFFC001     SeholfFFC001     SeholfFFC002     SeholfFFC002     SeholfFFC003     SeholfFFC003     SeholfFFC003     SeholfFFC004     SeholfFFC004     SeholfFFC004
8 yc27[37:0] 8 yc37[37:0] 8 yc47[37:0] 8 yc67[37:0]	3 3 3 3 3		2 #10/77-000 ShoupFF000 ShouFFE000 ShouFFE000 ShouFFE000 ShouFFE000 ShouFFE000 ShouFFE000
vc27[37:0]     vc37[37:0]     vc47[37:0]     vc57[37:0]     vc57[37:0]	3 3 3 3 3		3 HIGHTFLOOD 3 HIGHTFLOOD
vc27[37:0]     vc37[37:0]     vc47[37:0]     vc47[37:0]     vc67[37:0]     vc67[37:0]     vc67[37:0]     vc77[37:0]	3 3 3 3 3 3		a HINFY CNUE SHNGFY ECOD 3 HINFY ECOD
vc27[37:0]     vc37[37:0]     vc47[37:0]     vc47[37:0]     vc67[37:0]     vc67[37:0]     vc67[37:0]     vc77[37:0]     vc18[37:0]	3 3 3 3 3 3 3		3H00FFE0001         3H00FFE0001           3H00FFE0001         3H00FFE0001           3H00FFE0002         3H00FFE0003           3H00FFE0003         3H00FFE0003           3H00FFE0001         3H00FFE0001
yc27(37:0)     yc47(37:0)     v(yc37(37:0)     v(yc47(37:0)     v(yc47(37:0)     v(c67(37:0)     v(c77(37:0)     v(c77(37:0)     v(c18(37:0)     v(c18(37:0)     v(c28(37:0)	3 3 3 3 3 3 3 3		at NUTF COUL           binupf F COUL           shouf F COUL
<ul> <li>yc27[37:0]</li> <li>yc37[37:0]</li> <li>yc47[37:0]</li> <li>yc47[37:0]</li> <li>yc67[37:0]</li> <li>yc67[37:0]</li> <li>yc67[37:0]</li> <li>yc77[37:0]</li> <li>yc18[37:0]</li> <li>yc28[37:0]</li> <li>yc28[37:0]</li> <li>yc28[37:0]</li> </ul>	3 3 3 3 3 3 3 3		amount         Should Free Good           should Free Good         Should Free Good
0         yc27[37:0]           0         yc37[37:0]           0         yc47[37:0]           0         yc47[37:0]           0         yc67[37:0]           0         yc67[37:0]           0         yc67[37:0]           0         yc77[37:0]           0         yc77[37:0]           0         yc18[37:0]           0         yc28[37:0]           0         yc28[37:0]           0         yc28[37:0]           0         yc28[37:0]	3 3 3 3 3 3 3 3		3H 10177-5000           SHN0FF E0001           3H00FF E0001           SHN0FF E0001           SHN0FF E0002           SHN0FF E0003
0         yc27[37:0]           0         yc37[37:0]           0         yc47[37:0]           0         yc57[37:0]           0         yc57[37:0]           0         yc57[37:0]           0         yc57[37:0]           0         yc77[37:0]           0         yc28[37:0]           0         yc28[37:0]           0         yc48[37:0]	3 3 3 3 3 3 3 3		amount         Should Free Cool           Should Free Cool         Should Free Cool
•         yc27[37:0]           •         yc47[37:0]           •         yc48[37:0]           •         yc48[37:0]           •         yc48[37:0]           •         yc48[37:0]	3 3 3 3 3 3 3 3		2810/775/0003 2810/0755/0003 2810/0775/0003 2810/0755/0003 2810/0775/0003 2810/0755/000 2810/0755/000 281000 2810/0755/000 281000 281000 281000 281000 281000 281000 281000 281000 2810 281
•         yc27[37:0]           •         yc37[37:0]           •         yc47[37:0]           •         yc57[37:0]           •         yc67[37:0]           •         yc77[37:0]           •         yc73[37:0]           •         yc28[37:0]           •         yc48[37:0]           •         yc48[37:0]           •         yc48[37:0]	3 3 3 3 3 3 3 3		amoutre.com         amoutre.com           shindar k.com         amoutre.com
•         yc27[37:0]           •         yc27[37:0]           •         yc47[37:0]           •         yc57[37:0]           •         yc58[37:0]	3 3 3 3 3 3 3 3		amount         amount           amount
•         yc27[37:0]           •         yc27[37:0]           •         yc57[37:0]           •         yc28[37:0]           •         yc58[37:0]	3 3 3 3 3 3 3 3		amount         amount           behavior         behavior           behavior         behavior      behavior         behavior </td
• yc27(37:0)           • yc27(37:0)           • yc47(37:0)           • yc77(37:0)           • yc77(37:0)           • yc77(37:0)           • yc78(37:0)           • yc88(37:0)           • yc48(37:0)           • yc48(37:0)           • yc68(37:0)           • yc78(37:0)           • yc78(37:0)           • yc78(37:0)           • yc78(37:0)           • yc78(37:0)           • yc78(37:0)	3 3 3 3 3 3 3 3		amount         amount           amount
••••••••••••••••••••••••••••••••••••	3 3 3 3 3 3 3 3		amount         Second Second           Second Second         Second Second           Secon
• ye27[37:0]           • ye27[37:0]           • ye57[37:0]           • ye58[37:0]           • ye28[37:0]           • ye28[37:0]           • ye58[37:0]           • ye78[37:0]           • ye78[37:0]           • ye78[37:0]           • ye78[37:0]           • ye38[37:0]	3 3 3 3 3 3 3 3		amount         amount           amount         amount      amount         am
The second sec	3 3 3 3 3 3 3 3		amount         Semount           Semount         Semount           Semount <t< td=""></t<>
■         y.27137.00           ■         y.27137.01           ■         y.28137.01	3 3 3 3 3 3 3 3		amount         amount           amount
B         y-27 (37:0)           B         y-28 (37:0)	3 3 3 3 3 3 3 3	Image: Section of the sectio	2010/07/2000           3hv00/FF20001           3hv00/FF20003
By (22137 0)           By (42137 0)           By (42337 0) </td <td>3 3 3 3 3 3 3 3</td> <td></td> <td>amount         Should Freedood           Should Freedood         Should Freedood</td>	3 3 3 3 3 3 3 3		amount         Should Freedood           Should Freedood         Should Freedood
Ref (22137 0)           Ref (22137 0)           Ref (2137 0)	3 3		amount         Second           Second         Second      Second         Se
8         y.2737 0           8         y.2737 0           8         y.2737 0           8         y.2737 0           8         y.2717 0           9         y.2217 0           9         y.2317 0 </td <td>3 3 3 3 3 3 3 3</td> <td></td> <td>amount         Sharaki Freedont           Sharaki Freedont         <t< td=""></t<></td>	3 3 3 3 3 3 3 3		amount         Sharaki Freedont           Sharaki Freedont         Sharaki Freedont           Sharaki Freedont <t< td=""></t<>
8         y.2737 0           8         y.2377 0           8         y.2337 0           9         y.2337 0 </td <td>3 3 3 3 3 3 3 3</td> <td>Image: Section of the section of t</td> <td>amount         Shouth # Coold           Shouth # Coold         Shouth # Coold           Shouth # Ecoold         Shouth # Ecoold           Shout #</td>	3 3 3 3 3 3 3 3	Image: Section of the section of t	amount         Shouth # Coold           Shouth # Coold         Shouth # Coold           Shouth # Ecoold         Shouth # Ecoold           Shout #
By (r.2127 o)           By (r.2107 o)	3 3 3 3 3 3 3 3	Image: Section of the section of t	amount         Should FF 2000           Should FF 2000         Should
Byr2173 0     Byr2173 0     Wy2173 0     Syr47373 0     Syr473373 0     Syr4733373 0	3 3 3 3 3 3 3 3	Image: Section of the section of t	AT 1077-000           Shuqir F (2001)           Shuqir F (2002)           Shuqir F (2002)           Shuqir F (2002)           Shuqir F (2003)           Shuqir F (2004)           Shuqir F (2004)           Shuqir F (2003)           Shuqir F (2004)           Shuqir F (2004)           Shuqir F (2003)           Shuqir F (2004)           Shuqir F (2002)           Shuqir F (2003)           Shuqir F (2003) <td< td=""></td<>
8         y.2737 0           8         y.2737 0           8         y.2737 0           8         y.2717 0           8         y.2717 0           8         y.2717 0           9         y.2717 0 </td <td>3 3 3 3 3 3 3 3</td> <td></td> <td>Immon Processor         Immon Processor           Shandar Facoda         Immon Processor           Shanda</td>	3 3 3 3 3 3 3 3		Immon Processor         Immon Processor           Shandar Facoda         Immon Processor           Shanda
B         4         c.2717 0           B         4	3 3 3 3 3 3 3 3	Image: Section of the sectio	AT 1077 - Volu           Shing FF 2003           Shing FF 2003           Shing FF 2003           Shing FF 2003           Shing FF 2004           Shing FF 2003           Shing FF 2003           Shing FF 2004           Shing FF 2003           Shing FF 2003           Shing FF 2004           Shing FF 2003
B         P	3 3 3 3 3 3 3 3		amount         Should FF 2000           Should FF 2000         Should

Fig 9. Error free output obtained for the input provided



Fig 10. Error in  $Y_2[1]$ ,  $Y_2[2]$  and  $Y_2[4]$  causing corresponding errors in  $Z_{1,}Z_2$ ,  $Z_3$ 

Current Simulation

🖬 😽 yc12[37:0]	3	38h00FFFE0001
🗉 🚮 yc22[37:0]	3	38h01FFFC0002
🗉 🚮 yc32[37:0]	3	38h02FFFA0003
🗉 🚮 yc42[37:0]	3	38h03FFF80004
🗉 🚮 yc52[37:0]	3	38h02FFFA0003
🗉 🚮 yc62[37:0]	3	38h01FFFC0002
🖬 🚮 vc72(37:0)	3	38%00EEEE0001

Fig 11. Errors on filter output  $Y_2$  is corrected and corrected output is shown above



Fig 12. Physical design of eleven parallel FIR filters

#### V. Conclusion

A new technique has been introduced to protect parallel FIR filters from multiple errors that are generally found in modern signal processing circuits. The methodology depends on applying ECCs to the parallel filters outcomes to recognize and correct errors. This scheme can be utilized for any number of parallel filters with the same impulse response that process different input signals. This technique provides huge advantage if the number of parallel filters are more.

#### References

- [1] M. Nicholaidis, "Design for soft error mitigation," IEEE Trans. Device Mater. Rel., vol. 5, no. 3, pp. 405-418, Sep. 2005.
- [2] A. Reddy and P. Banarjee "Algorithm-based fault detection for signal processing applications," IEEE Trans Comput,. vol.39, no 10, pp. 1304-1308, Oct. 1990.
- [3] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 4, pp. 336-348, Apr. 2006.
- [4] T. Hitana and A. K. Deb, "Bridging concurrent and non-concurrent error detection in FIR filters," in Proc. Norchip Conf., 2004, pp. 75-78.
- [5] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in Proc. IEEE IOLTS, Jul. 2008. pp. 192-194.
- [6] Z. Gao, W. Yang, X. Chen, M, Zhao, and J. Wang. "Fault missing rate analysis of the arithmetic residue codes based fault-tolerant FIR filter design," in Proc. IEEE IOLTS, Jun. 2012, pp. 130-133.
- [7] P. P. Vaidyanathan. Multirate Systems and Filter Banks. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
- [8] P. Reviriego, C. J. Bleakley, and J. A. Maestro, "Structural DMR: A technique for implementation of soft-error-tolerant FIR filters," IEEE Trans. Circuits Syst., Exp. Briefs, vol. 58, no. 8, pp. 512-516, Aug. 2011.
- [9] A. Sibille, C.Oestges, and A. Zanella, MIMO: From Theory to implementation. San Francisco, CA, USA: Academic Press, 2010.
- [10] P. Reviriego, S. Pontarelli, C. J. Bleakley, and J. A. Maestro, "Area efficient concurrent error detection and correction for parallel filters," IET Electron. Lett., vol. 48, no. 20, pp. 1258-1260.
- [11] Zhen Gao, Pedro Reviriego, et.al.,"Fault tolerant parallel filters based on error correction codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst, vol. 23, no. 2, Feb. 2015