CHAPTER 5

PROPOSED RECONFIGURABLE RESOURCE SHARING ARCHITECTURE

5.1 INTRODUCTION

Existing reconfigurable architectures allocate many resources without considering the specific application domain during reconfiguration to different application. Functional resources that take large area and long latency can be shared (Yoonjin Kim et al 2005) among the PEs. Therefore, the hardware cost and the delay can be effectively reduced without any performance degradation for some application domains. In this chapter, such resource sharing reconfigurable array architecture template and design space exploration for domain-specific optimization by exploiting efficient implementation of algorithms for WCDMA and OFDM based WiFi technology as discussed in Chapter 4, is discussed.

5.2 RECONFIGURABLE RECEIVER SYSTEM

The block diagram of a reconfigurable receiver architecture that can reconfigure itself to WCDMA or WiFi technology is shown in Figure 5.1. The architecture comprises reusable and reconfigurable functional blocks for implementing different algorithms necessary for WCDMA and WiFi technology. One or more reusable functional blocks can be configured to implement process including multiplication, addition, subtraction and accumulation to support WCDMA and WiFi technology. For example, RAKE receiver algorithms (multiply and accumulate select function) for

WCDMA and FFT (basic butterfly function) for WiFi are implemented in this architecture as shown in Figure 5.2.

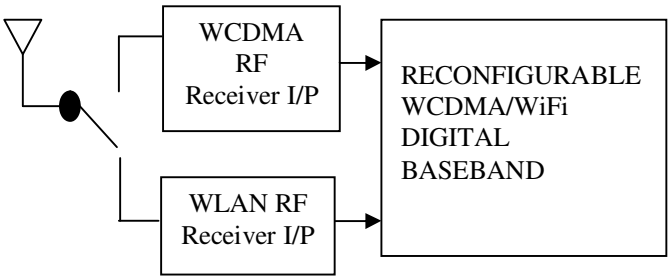

Figure 5.1 Block diagram of reconfigurable receiver system

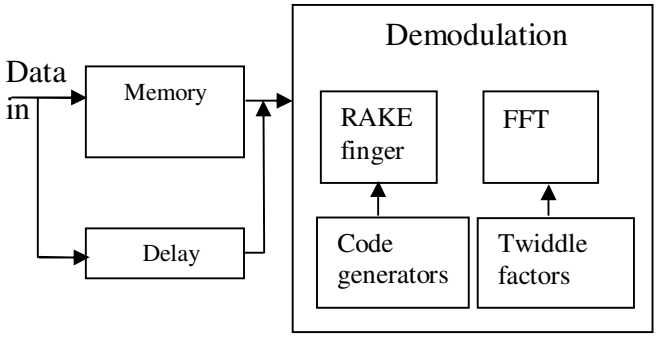

Figure 5.2 Functional blocks of reconfigurable receiver algorithms

5.3 RESOURCE SHARING

Integrating different architectures in a chip is an emerging trend in the SoC era. Building blocks configurable for computing different algorithms used in WCDMA and OFDM based WiFi standards have been explained in Chapter 3. Hardware redundancies always exist between different modules as these modules are combined in a chip. The redundancies increase the required chipset area. To remove hardware redundancies, common resources can be reused among baseband processing tasks that are not active simultaneously. This may lead to silicon area optimization. Certain tasks in the receiver chain of the baseband processing of different standards can be implemented on reconfigurable hardware by sharing the same reconfigurable hardware

resources. Figure 5.3 shows the realization of tasks of WCDMA and OFDM standards on the same reconfigurable hardware.

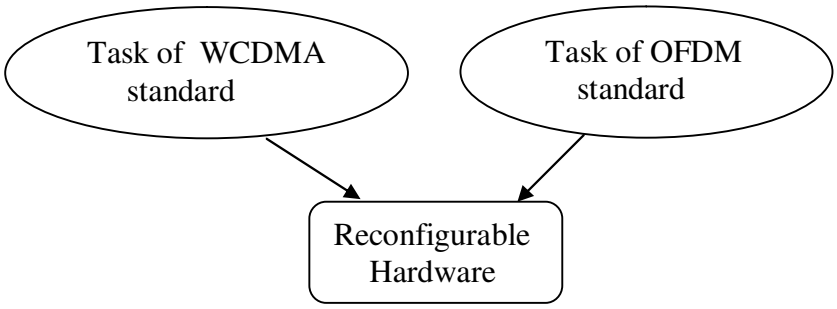

Figure 5.3 Realization of multistandard tasks on the same reconfigurable hardware

The proposed method exploits such a resource sharing method that is applied in WiFi and WCDMA reconfigurable receiver architectures. In order to implement the resource sharing architecture, computationally expensive receiver processing are considered in this thesis. So FFT processor, which is the key kernel of OFDM and RAKE receiver, which is the widely used receiver architecture of WCDMA standard are used to analyze the similarities between them and share the common resources.

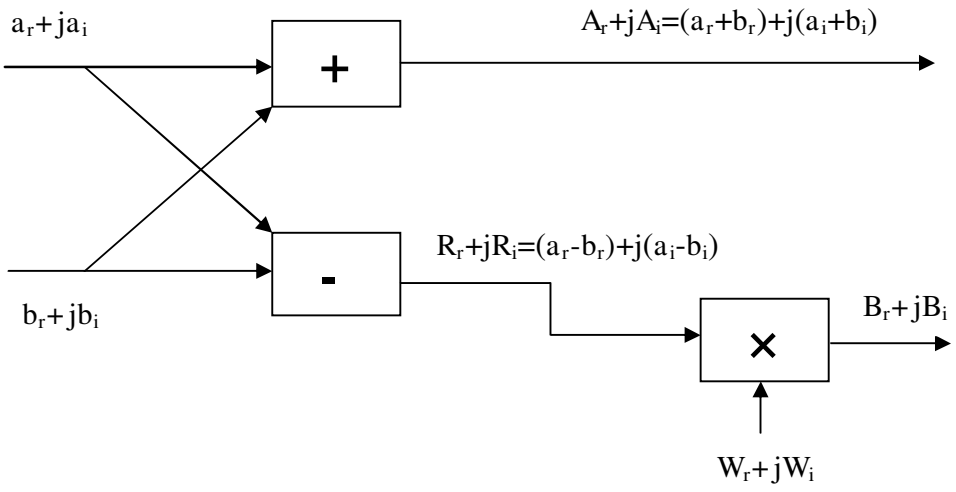

Figure 5.4 Radix-2 DIF FFT butterfly diagram with complex data

An algorithmic strength reduction transformation technique has been discussed in Chapter 4 to implement radix-2 DIF FFT algorithm for WiFi standard. As illustrated in Figure 5.4, an FFT butterfly operation comprises two sets of complex input data (a and b) and one set of complex coefficient data (W), passing a complex addition/subtraction and multiplication, yielding two sets of complex output (A and B). More precisely, the complete FFT butterfly operation calculation process can be broken down into various steps, listed as follows,

1) Read the complex inputs (a_r,a_i) and (b_r,b_i) and access memory locations to fetch the coefficients (W_r,W_i).

2) Perform addition/subtraction of these complex inputs, (a_r+b_r), (a_i+b_i), (a_r-b_r), (a_i-b_i), and store the result in temporary storage locations.

3) So assign $A_r= a_r+b_r$, $A_i= a_i+ b_i$,$R_r= a_r-b_r$ and $R_i= a_i-b_i$

4) Perform multiplication of $R_r\times W_r$, $R_r\times W_i$, $R_i\times W_i$, $R_i\times W_r$ and store the results in temporary storage locations. There are 4 multiplications needed in conventional method.

5) These 4 multiplications are reduced to only 3 applying algorithmic strength reduction technique.

6) So using the above said technique, perform addition/subtraction and multiplication as $(R_r-R_i)\times W_i$, $R_r\times (W_r-W_i)$, and $R_i\times(W_r+ W_i)$.

7) Perform addition of $(R_r-R_i)\times W_i$ and $R_r\times(W_r-W_i)$ to produce B_r.

8) Perform addition of $(R_r-R_i)\times W_i$ and $R_i\times(W_r+ W_i)$ to produce B_i.

9) Write the results into the registers.

The block diagram of PEAS and PEM of PES are shown in Figures 4.5(a) and 4.5(b) in Chapter 4. PEAS and PEM contain the computational resources required to perform the above said steps. PEAS performs addition and subtraction of complex input data in 2-point butterfly structure. PEM performs multiplication with complex twiddle factor coefficient W after subtraction.

Figure 4.3 shows the PE and its resources required for the implementation of RAKE finger in WCDMA. The multiplier-less RAKE receiver algorithm presented in Chapter 4 does not require any multipliers as described in Section 4.1.1.2 of Chapter 4. This algorithm performs the computation as follows.

1) The complex combined spreading and scrambling codes are applied to multiplexer.

2) If C_{sr}=0 and C_{si}=0, it performs addition (R_r+R_i) and subtraction (R_i-R_r)

3) If C_{sr}=0 and C_{si}=1, it performs subtraction (R_r-R_i) and addition (R_i+R_r)

4) If C_{sr}=1 and C_{si}=0, it performs subtraction (R_r-R_i) and subtraction (R_i-R_r)

5) If C_{sr}=1 and C_{si}=1, it performs addition (R_r+R_i) and subtraction (R_i-R_r)

6) The results from the adder and subtractor are applied to another adder and subtractor to perform addition and subtraction with the previous result which has been stored in the register.

7) The output is stored in the register.

8) These steps are repeated over the length of spreading factor.

5.4 SIMILARITY ANALYSIS AND RESOURCE SHARING

The similarities between the FFT-based WiFi and RAKE receiver of WCDMA are summarized below.

Comparing these two operations as discussed above, it is identified that the two adders and subtractors are common for both the architectures. So the PEAS is shared by both receiver architectures. These similarities can be exploited to remove hardware redundancies when two or more hardware-modules are combined in an integrated environment. In this thesis, the design of FFT functions used for OFDM and RAKE fingers used for WCDMA in the same architecture by sharing computational elements of PEAS. So the proposed reconfigurable architecture consists of processing units, their computational elements are shared by both the RAKE receiver operation of WCDMA and FFT operation of OFDM.

5.5 MAPPING APPLICATIONS ON THE PROPOSED ARCHITECTURE

In Chapter 3, the block diagram of the proposed reconfigurable architecture has been given and explained. Figure 5.5 shows the block diagram for mapping application on the proposed architecture. To process the FFT/RAKE finger functions the 64 Bytes of twiddle factor coefficients and combined spreading and scrambling codes are stored in memory block. The received complex data are fed to the array of PES through I/O block. The memory accesses for these operations are coordinated by the control unit. The configuration controller serves as an arbiter for controlling the proposed

architecture and surrounding units to specify operation mode, whether it requires FFT calculation or RAKE receiver function.

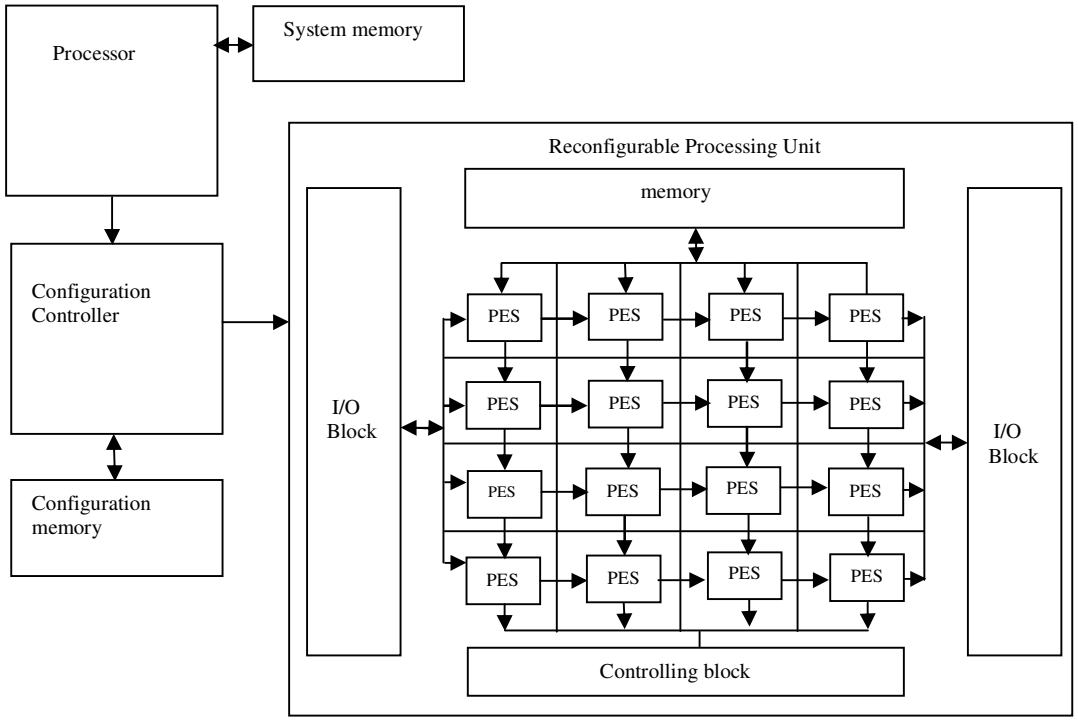

Figure 5.5 Block diagram for application mapping on the proposed architecture

The main contribution in this thesis is an efficient mapping of the application on the proposed reconfigurable architecture. The difficulty of this optimization increases with the amount of data dependencies involved. One major goal in mapping is to minimize the dataflow between computational clusters and achieve high utilization of computational resources. This architecture contains a processor which controls the mapping of application on reconfigurable architecture as shown in Figure 5.5.

5.6 MAPPING FFT ON RECONFIGURABLE ARCHITECTURE

The FFT operates by decomposing an N point time domain signal into N-time domain signals, each contains a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency spectrum. In Figure 5.6 as an example, time domain decomposition of 16 point signal is shown. An N point signal is decomposed into N signals each containing a single point. Each stage uses an interlace decomposition, separating the even and odd numbered samples.

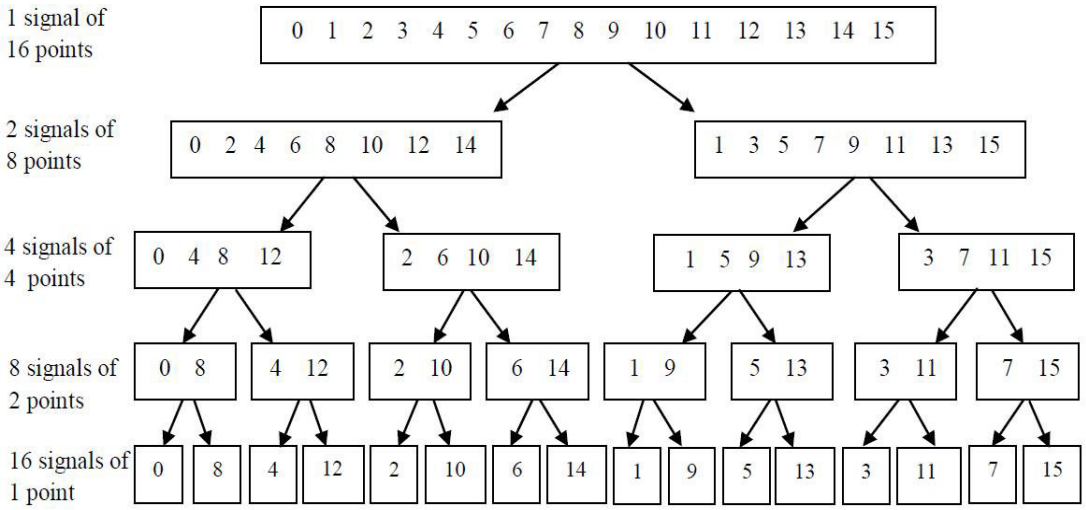

Figure 5.6 Time domain decomposition used in FFT

The butterfly diagram to implement 8-point radix-2 DIF FFT algorithm is shown in Figure 5.7. For 8-point FFT, the computation is accomplished in three stages. The x(0) until x(7) variable is denoted as the input value for FFT computation and X(0) until X(7) is denoted as the output. There are two operations to complete the computation in each stage. The upward arrow will execute addition operation while downward arrow will execute subtraction operation. The subtracted value is multiplied with the

twiddle factor value before being processed into the nest stage. This operation is done concurrently and is known as the butterfly process. For the second stage, there are two butterfly processes with each process getting the variable from the first stage of computation. In the first stage the butterfly process receives eight input variables while in the second stage, each butterfly process receives four input variables.

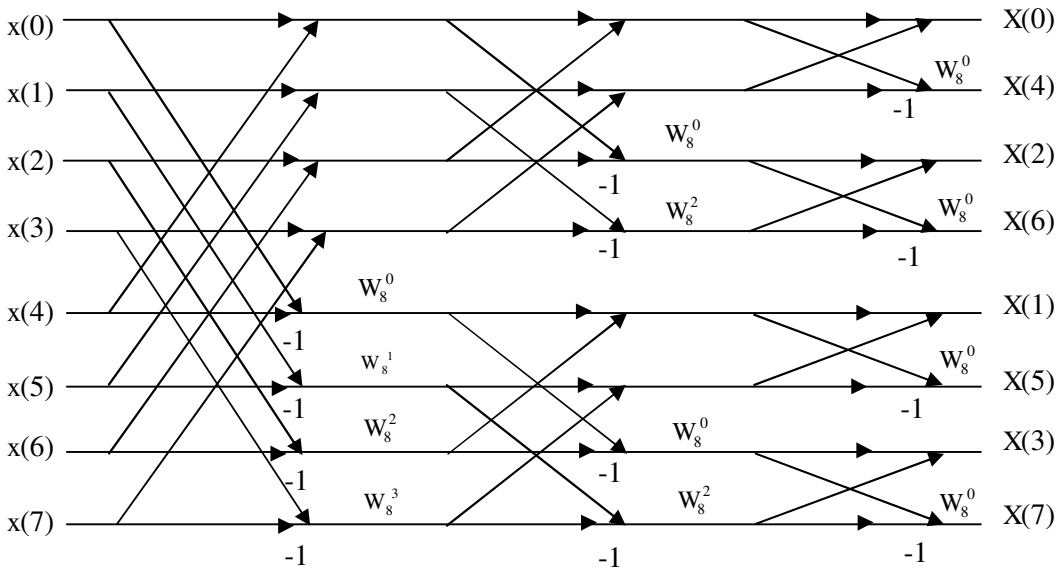

Figure 5.7 8-point DIF FFT butterfly diagram

This process is continued until the third stage. In the third stage, there are four butterfly processes. Each butterfly process is performed concurrently enable quick FFT computation. Mathematically, the butterfly process for each stage can be derived as stated below.

FFT Stage 1

x(0) + x(4) => x'(0)

x(1) + x(5) => x'(1)

x(2) + x(6) => x'(2)

x(3) + x(7) => x'(3)

$[x(0) - x(4)]\ W_8^0 \Rightarrow x'(4)$

$[x(1) - x(5)]\ W_8^1 \Rightarrow x'(5)$

$[x(2) - x(6)]\ W_8^2 \Rightarrow x'(6)$

$[x(3) - x(7)]\ W_8^3 \Rightarrow x'(7)$

FFT Stage 2

$x'(0) + x'(2) \Rightarrow x''(0)$

$x'(1) + x'(3) \Rightarrow x''(1)$

$[x'(0) - x'(2)]\ W_8^0 \Rightarrow x''(2)$

$[x'(1) - x'(3)]\ W_8^2 \Rightarrow x''(3)$

$x'(4) + x'(6) \Rightarrow x''(4)$

$x'(5) + x'(7) \Rightarrow x''(5)$

$[x'(4) - x'(6)]\ W_8^0 \Rightarrow x''(6)$

$[x'(5) - x'(7)]\ W_8^2 \Rightarrow x''(7)$

FFT Stage 3

$x''(0) + x''(1) \Rightarrow X(0)$

$x''(0) - x''(1) \Rightarrow X(4)$

$x''(2) + x''(3) \Rightarrow X(2)$

$x''(2) - x''(3) \Rightarrow X(6)$

$x''(4) + x''(5) \Rightarrow X(1)$

$x''(4) - x''(5) \Rightarrow X(5)$

$x''(6) + x''(7) \Rightarrow X(3)$

$x''(6) - x''(7) \Rightarrow X(7)$

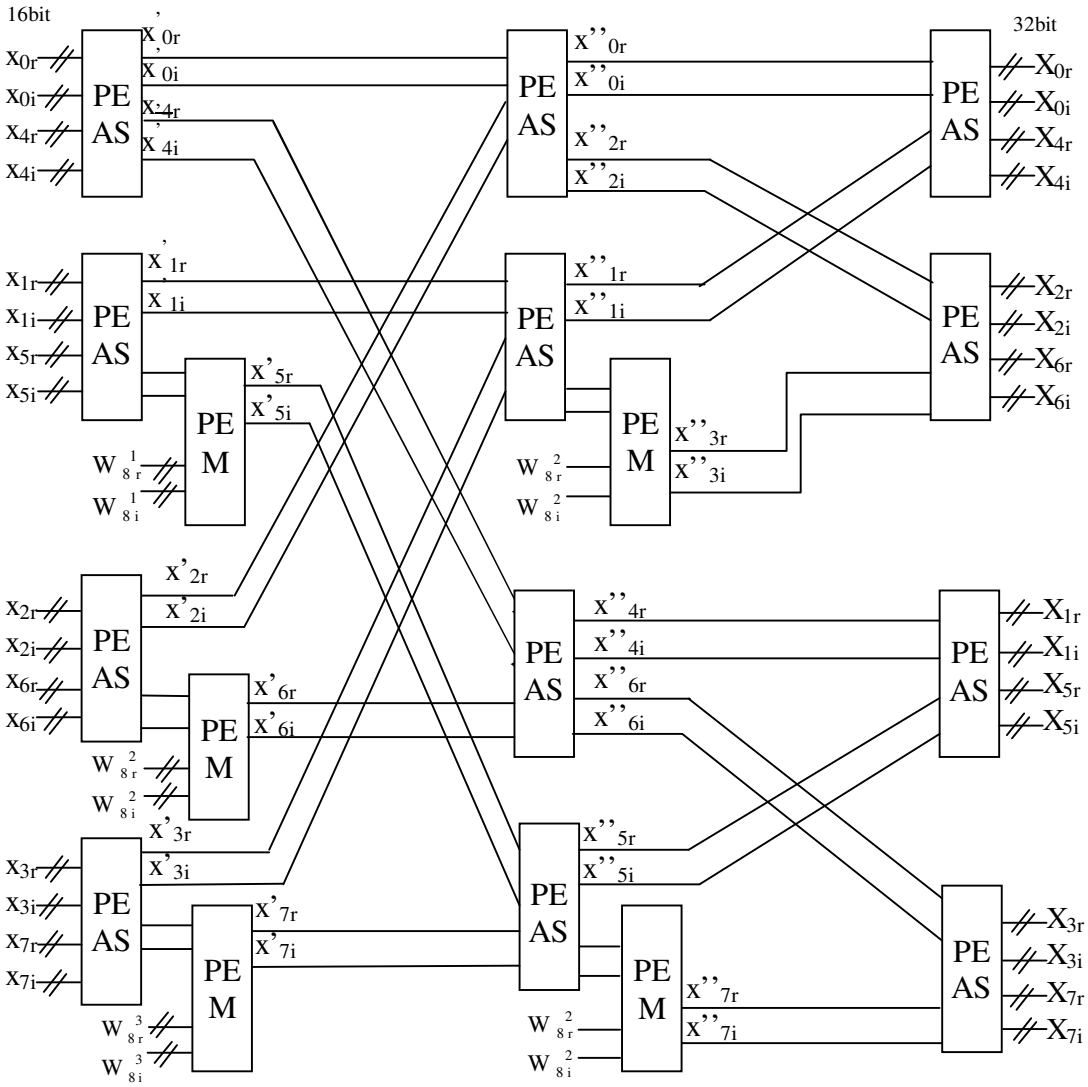

Figure 5.8 DFG of 8-point FFT

All data applied to the computation in each stage are complex numbers. The above said computations are mapped in the proposed architecture as shown in Figure 5.8. The figure shows DFG of complex numbers (real and imaginary parts) between different stages of computation.

A 64-point FFT computes a sequence x(n) of 64 complex valued numbers given another sequence of data X(k) of length 64 according to the Equation (5.1).

$$X(\mathrm{K}) = \sum_{n=0}^{63} x(n)\, W_{64}^{n} \qquad ; \quad K = 0 \; to \; 63 \tag{5.1}$$

For implementing 64 point FFT, it divides FFT into two smaller FFTs of the length 8 (Chin-Teng Lin et al 2006), as shown in Equation (5.2).

$$X(K) = X(8r+s) = \sum_{m=0}^{7} W_8^{mr} \; W_{64}^{ms} \sum_{l=0}^{7} x(8l+m) W_8^{sl} \quad , r = 0 \; to \; 7, s = 0 \; to \; 7 \tag{5.2}$$

The input complex data x(n) are represented by the 2-dimentional array of data x(8l+m). The columns of this array are computed by 8-point FFTs. The results of them are multiplied by the twiddle factors W_{64}^{ms}. And the resulting array of data X(8r+s) is derived by 8-point FFTs of rows of the intermediate result array. The 8-point FFT is implemented by using the algorithm as discussed in Chapter 4, which provides the minimum multiplications. As a result, the 64-point FFT algorithm needs only 120 multiplications.

The proposed 64-point FFT implementation is simulated using ModelSimSE v6.5, coded in VHDL and mapped onto a Virtex-6 FPGA device (xq6vlx130t-rf784) with speed grade (-2) using the tool Xilinx ISE 12.2 and synthesized. Table 5.1 shows the synthesis report of the conventional FFT implementation (FFT with 4 multipliers) and Table 5.2 shows the synthesis report of the proposed FFT implementation. The results of the optimized FFT implementation is compared with the conventional FFT implementation and the merits are discussed in Chapter 6. From this result it is proved that our proposed implementation has significant improvements in terms of area by reducing more number of computational resources.

Table 5.1 Synthesis report of the conventional FFT implementation

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	1,588	80,000	2%
Number used as logic	1,588	80,000	2%
Number of occupied Slices	640	20,000	3%
Number of gates	14,796		
Number of bonded IOBs	128	400	32%

Table 5.2 Synthesis report of the proposed FFT implementation

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	1,282	80,000	1%
Number used as logic	1,281	80,000	1%
Number of occupied Slices	485	20,000	2%
Number of gates	12,440		
Number of bonded IOBs	128	400	32%

5.7 MAPPING RAKE RECEIVER ON RECONFIGURABLE ARCHITECTURE

The RAKE receiver is composed of 4 parallel fingers, each finger is used to despread one multipath component of the received WCDMA signal. In the RAKE finger a complex multiplication in the correlator is simplified to a complex add/subtract structure as explained in Chapter 4. Mapping the RAKE receiver algorithm on the proposed reconfigurable architecture is done using the DFG as shown in Figure 5.9. The computational resources of PEAS are shared between the FFT and RAKE fingers. Since the RAKE receiver exploits multiplier-less algorithm, the PEM processing unit is not used by the correlators of RAKE receiver.

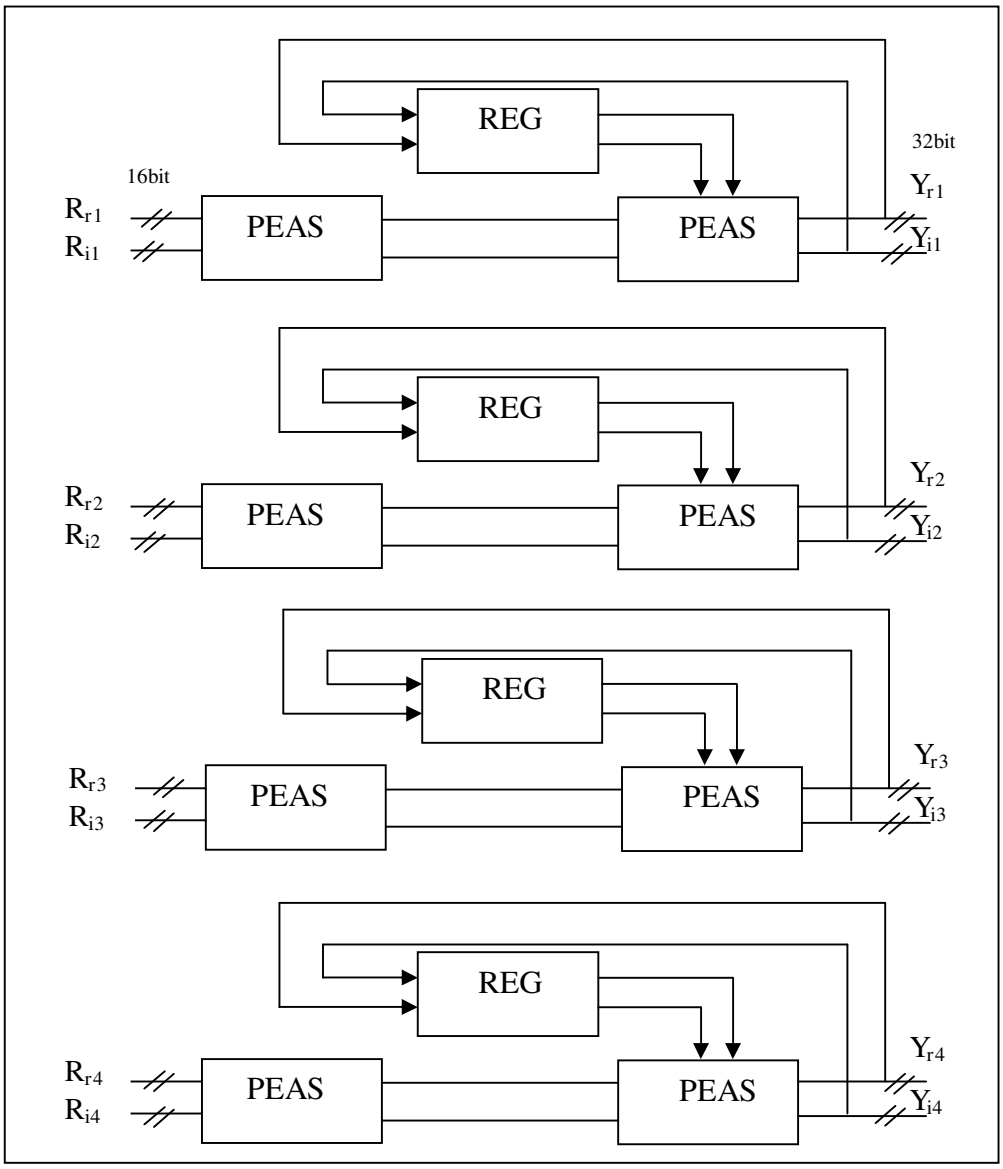

Figure 5.9 RAKE receiver DFG

RAKE receiver is simulated using ModelSimSE v6.5, coded in VHDL and mapped onto a Virtex-6 FPGA device (xq6vlx130t-2rf784) with speed grade (-2) using the tool Xilinx ISE 12.2 and synthesized. Table 5.3 shows the synthesis report of conventional RAKE receiver (RAKE fingers with multipliers) and Table 5.4 shows the synthesis report of proposed RAKE receiver. The results of optimized multiplier-less RAKE receiver implementation is compared with the Conventional RAKE receiver implementation and the merits are discussed in chapter 6. From this result it is

proved that the proposed implementation reduces large number of computational resources and hence the area.

Table 5.3 Synthesis report of conventional RAKE receiver implementation

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	1,426	80,000	1%
Number used as logic	1,423	80,000	1%
Number of occupied Slices	751	20,000	3%
Number of gates	13,898		
Number of bonded IOBs	162	400	40%

Table 5.4 Synthesis report of proposed RAKE receiver implementation

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	128	80,000	1%
Number used as logic	128	80,000	1%
Number of occupied Slices	32	20,000	1%
Number of gates	1328		
Number of bonded IOBs	162	400	40%

5.8 ANALYSIS: PROPOSED DATAPATH VERSUS DATAPATH OF CONVENTIONAL CGRA

This section demonstrates the beneficial features of the proposed datapath of reconfigurable architecture. The proposed datapath of reconfigurable architecture is evaluated with conventional coarse-grained reconfigurable architectures (ADRES and Montium), considering area utilization design metrics. The majority of coarse-grained reconfigurable architectures are cell-based. Each Coarse-Grained Reconfigurable Cell (CGRC) is comprised of discrete computational components, such as one

multiplier and one ALU. A closer look at the CGRC's structure reveals its inherent shortcomings. CGRCs suffer from inefficient utilization of their underlying hardware area, since in every control step only one of their allocated components is utilized. The area utilization gains of the proposed datapath are analyzed for complex multiplication datapath shown in Figure 4.5(b) of Chapter 4. This datapath is mapped on conventional CGRCs of CGRAs and the Table 5.5 illustrates the allocation of resources in case of the conventional CGRA. Each allocation table shows the number of control steps, allocated resources and unutilized area per control step. From the table it is identified that 50% of the computational resources are unutilized in each and every control step. These inherent shortcomings of the CGRCs formed the actual motivation to design an efficient datapath reconfigurable architecture.

Table 5.5 Resource allocation table of conventional CGRA in each CGRC

	CGRC0		CGRC1		CGRC2		
Control step	ALU0	MUL0	ALU1	MUL1	ALU2	MUL2	Non-utilized hardware
1	$X_0 = R_r - R_i$	------	$X_1 = W_r - W_i$	-----	$X_2 = W_r + W_i$	----	$3 \times area_{mul}$
2	------	$Y_0 = X_0 \times W_i$	------	$Y_1 = R_r \times X_1$	------	$Y_2 = R_i \times X_2$	$3 \times area_{ALU}$
3	$B_r = Y_0 + Y_1$	------	$B_i = Y_0 + Y_2$	-------	-------	-------	$3 \times area_{mul} + area_{ALU}$
					Total		$6 \times area_{mul} + 4 \times area_{ALU}$

5.9 MAPPING RESOURCE SHARING ARCHITECTURE

Mapping of the RAKE receiver architecture onto the proposed reconfigurable resource sharing architecture is illustrated by DFG as shown in Figure 5.10. This architecture shares the computational components of FFT

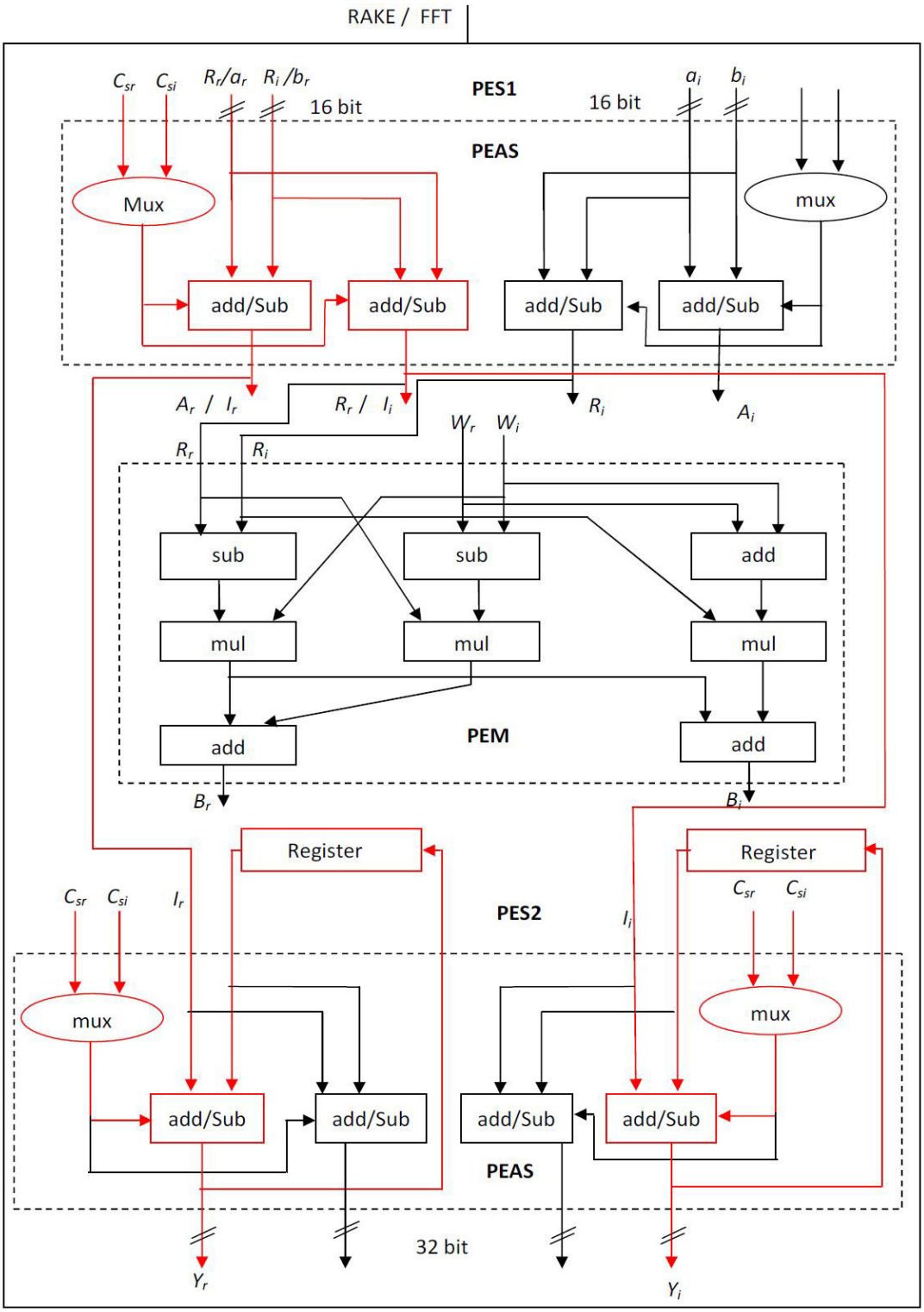

Figure 5.10 Mapping FFT/RAKE receiver on reconfigurable architecture with resource sharing

architecture with RAKE receiver when it is reconfigured to WCDMA receiver from OFDM receiver.

Table 5.6 Synthesis report of reconfigurable architecture without resource sharing

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	3,014	80,000	3%
Number used as logic	3,014	80,000	3%
Number of occupied Slices	385	20,000	2%
Number of gates	28,694		
Number of fully used LUT-FF pairs	1	1,360	1%
Number of unique control sets	1		
Number of slice register sites lost to control set restrictions	7	160,000	1%
Number of bonded IOBs	290	400	72%
IOB Latches	64		

Table 5.7 Synthesis report of reconfigurable resource sharing architecture

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	1,290	80,000	1%
Number used as logic	1,255	80,000	1%
Number of occupied Slices	385	20,000	2%
Number of gates	12,540		
Number of fully used LUT-FF pairs	1	1,360	1%
Number of unique control sets	1		
Number of slice register sites lost to control set restrictions	7	160,000	1%
Number of bonded IOBs	290	400	72%
IOB Latches	64		

From the figure it is observed that red colour blocks are shared by both the receiver architecture and red colour interconnect lines are reconfigured to implement the RAKE receiver algorithm. The reconfiguration is done through the control input RAKE/FFT which is shown in the figure. This Figure shows the resource sharing method for one butterfly operation and one RAKE finger. This process is repeated for 64-point FFT dataflow and RAKE receiver dataflow.

Synthesis report of conventional architecture (reconfigurable architecture without resource sharing method) is shown in Tables 5.6 and Table 5.7 shows the synthesis report of the proposed architecture (reconfigurable architecture with resource sharing method). Implementation results of the proposed reconfigurable resource sharing architecture for WCDMA/OFDM are compared with the conventional architecture, and the merits are discussed in Chapter 6. From the results it is proved that there is a large amount of resources reduced by resource sharing method.

5.10 SUMMARY

In this chapter, resource sharing technique which is used to combine two wireless communication standard's computational operations has been presented. It is accomplished by modifying the PEs of FFT operation to perform RAKE receiver functions. It is found that the proposed reconfigurable architecture allows reduced chip area and it is applicable for multistandard communication systems as this architecture provides flexibility. Data flow architecture for mapping 64-point FFT algorithm and RAKE receiver operation on the proposed reconfigurable architecture has also been discussed in this chapter.