Implementation of CORDIC Based RAKE Receiver Architecture

K.S.Chaitanya[1] P.Muralidhar[2] C.B.Rama Rao[3]
Department of Electronics and Communication Engineering
National Institute of Technology
Warangal, India
[1]kodukula.chaitanya@gmail.com, [2]pmurali_nitw@yahoo.co.in, [3]cbrr@nitw.ac.in

Abstract— **RAKE receiver is used in CDMA-based (Code Division Multiple Access) systems and can combine multipath components, which are time-delayed versions of the original signal transmission. Combining is done in order to improve the signal to noise ratio at the receiver. RAKE receiver attempts to collect the time-shifted versions of the original signal by providing a separate correlation receiver for each of the multipath signals. This can be done due to multipath components are practically uncorrelated from another when their relative propagation delay exceeds a chip period. This paper aims to present a system-on-chip (SoC) solution for RAKE receiver using a CORDIC hardware accelerator. The algorithm is implemented on Cyclone II FPGA device chipped on Altera DE2 board. The inbuilt NIOS II soft core processor of the FPGA device acts as the processor for processing applications. The CORDIC algorithm which computes the trigonometric functions is developed as a custom instruction for the NIOS II processor. This hardware accelerator has drastically improved the performance of the algorithm by about 70% when compared with the pure software implementation. This improvement in the performance is achieved at the cost of area. The performance of RAKE receiver is illustrated using bit error rate (BER) calculations. The RAKE receiver performance is examined and compared using maximal ratio and equal-gain combining techniques.**

Keywords-Field programmable gate array (FPGA), system-on-chip (SoC), NIOS II processor, RAKE receiver, CDMA, maximal-ratio combining, equal-gain combining, bit error rate (BER), CORDIC.

I. INTRODUCTION

With the technology advancement in today's society, the ability to communicate with people on the move has evolved remarkably. However, the transmission quality of the signal has deteriorated due to the modernization of the urban cities with skyscrapers and other manmade obstacles. This results in the transmitted signal having to take multiple paths before reaching the intended receiver. Through the multipath transmissions, the signal is severely distorted and attenuated. Methods have to be developed to improve on the signal quality.

Code Division Multiple Access (CDMA) systems use the spread spectrum technology and the RAKE receiver concept to minimize communication errors resulting from multipath effects. In general, the number of multipath signals in the wireless channel is unknown and difficult to predict. The spread spectrum technology aims to spread the information signal over a wider bandwidth to make jamming and interception more difficult. A RAKE receiver allows each arriving multipath signal to be individually demodulated and then combined to produce a stronger and more accurate signal.

In the last few decades, a lot of modern signal processing applications require such a high computational power that only ASICs can fulfill the technical demands. Unfortunately, ASICs are inflexible, costly (development and debugging) and only economical for mass-products. As a consequence, system designers are striving to replace specialized hardware solutions with software based solutions as developments in the field of software radio demonstrate. Due to the fact that even the most commonly used programmable devices, i.e. DSPs, often lack the required processing power, one tries to develop a solution that lays somewhere in between the two extreme programmable signal processing and dedicated hardware. The efforts in this area are summarized with the term reconfigurable computing (FPGAs) [1].

The availability of hard/soft core processors in modern FPGAs allow moving algorithms written for GPP or DSP processors to FPGAs using the core processors. An alternative approach is to move part of the algorithm into hardware (HW) to improve performance. This is a form of HW/SW Co-design, which requires profiling the software to efficiently partition it between HW and SW. This solution could result in a more efficient implementation as a part of the algorithm is accelerated using HW while the flexibility is maintained.

In this paper, the most computational intensive tasks of the algorithm are performed on a dedicated hardware accelerator using CORDIC processing elements. This paper first discusses the theory behind the RAKE receiver and CORDIC algorithm in section II. In section III a description of the implementation is given while section IV displays the results obtained. Finally section V gives the conclusions made from the results obtained.

978-1-4244-4520-2/09/$25.00 ©2009 IEEE

II. Theory

A. RAKE Receiver

Due to reflections from obstacles a radio channel can consist of many copies of originally transmitted signals having different amplitudes, phases, and delays. If the signal components arrive more than duration of one chip apart from each other, a RAKE receiver can be used to resolve and combine them. The RAKE receiver uses a multipath diversity principle. It is like a rake that rakes the energy from the multipath propagated signal components [2].

1) Multipath Channel Model: Multipath can occur in radio channel in various ways such as, reflection and diffraction from buildings, and scattering from trees. An M-ray multipath model [3] is shown in Fig. 1. Each of the M paths has an independent delay, τ , and an independent complex time-variant gain, G.

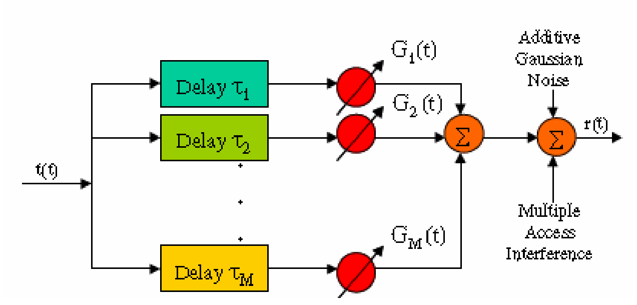


Figure 1. Multipath channel model

2) M-finger RAKE Receiver: A RAKE receiver utilizes multiple correlators to separately detect M strongest multipath components. The outputs of each correlator are weighted to provide better estimate of the transmitted signal than is provided by a single component. Demodulation and bit decisions are then based on the weighted outputs of the M correlators [3]. Each correlator detects a time-shifted version of the original CDMA transmission, and each finger of the RAKE correlates to a portion of the signal, which is delayed by at least one chip in time from the other fingers.

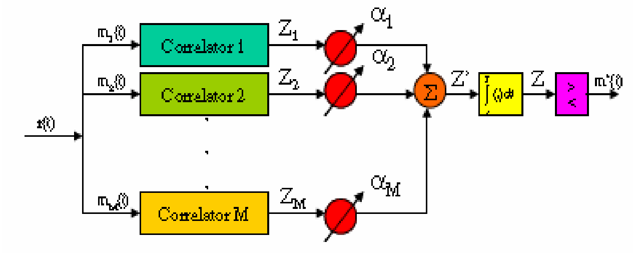


Figure 2. An M-branch RAKE receiver implementation

Assume M correlators are used in a CDMA receiver to capture M strongest multipath components. A weighting network is used to provide a linear combination of the correlator output for bit decision. Correlator 1 is synchronized to the strongest multipath m_1. Multipath component m_2 arrived t_1 later than m_1 but has low correlation with m_1. The M decision statistics are weighted to form an overall decision statistic as shown in Fig. 2. The outputs of the M correlators are denoted as Z_1, Z_2,⋯, and Z_M. They are weighted by α_1, α_2,⋯, and α_M, respectively. The weighting coefficients are based on the power or the SNR (Signal-to-Noise Ratio) from each correlator output. If the power or SNR is small out of a particular correlator, it will be assigned a small weighting factor, α. If maximal-ratio combining is used, following equation 1 can be written for Z' .

$$Z' = \sum_{m=1}^{M} \alpha_m Z_m \tag{1}$$

The weighting coefficients, α_M, are normalized to the output signal power of the correlator in such a way that the coefficients sum to unity, as shown in following equation 2.

$$\alpha_m = \frac{Z_m^2}{\sum_{m=1}^{M} Z_m^2} \tag{2}$$

3) RAKE Receiver Block Diagram: When a signal is received in a matched filter over a multipath channel, the multiple delays appear at the receiver, as depicted in Fig 3. The RAKE receiver uses several baseband correlators to individually process several signal multipath components. The correlator outputs are combined to achieve improved communications reliability and performance [2]. Bit decisions based only a single correlation may produce a large bit error rate as the multipath component processed in that correlator can be corrupted by fading. In a RAKE receiver, if the output from one correlator is corrupted by fading, the others may not be, and the corrupted signal may be discounted through the weighting process.

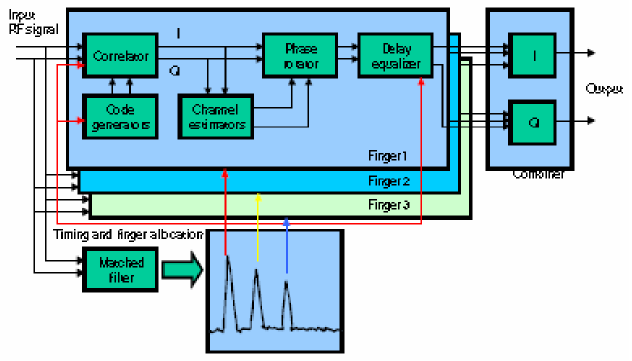


Figure 3. Block diagram of a RAKE receiver

Impulse response measurements of the multipath channel profile are executed through a matched filter to make a successful de-spreading. It reveals multipath channel peaks and gives timing and RAKE finger allocations to different receiver blocks. Later it tracks and monitors these peaks with a measurement rate depending on speeds of mobile station and on propagation environment. The number of available RAKE fingers depends on the channel profile and the chip rate. The higher the chip rate, the more resolvable paths there are, but higher chip rate will cause wider bandwidth. To catch all the energy from the channel more RAKE fingers are needed. A very large number of fingers lead to combining losses and practical implementation problems.

B. *CORDIC Algorithm*

CORDIC stands for **CO**ordinate **R**otation **Di**gital **C**omputer [4]. It calculates the value of trigonometric functions and hyperbolic functions to desired precision. The CORDIC algorithm does not use Calculus based methods such as polynomial or rational function approximation but computes elementary functions using only additions, subtractions, digit shifts, comparisons and stored constants. CORDIC algorithm revolves around the idea of "rotating" the phase of a complex number, by multiplying it by a succession of constant values. However, the "multiplies" can all be powers of 2, so in binary arithmetic they can be done using just shifts and adds; no actual "multiplier" is needed thus it simpler and do not require complex hardware structure as in the case of multiplier. The drawback in CORDIC is that after completion of each iteration, there is a gain which is added to the magnitude of resulting vector which can be removed by multiplying the resulting magnitude with the inverse of the gain. There are two ways in CORDIC algorithm for calculation of trigonometric and other related functions: they are rotation mode and vectoring mode. Both methods initialize the angle accumulator with the desired angle value. The rotation mode, determines the right sequence as the angle accumulator approaches zero while the vectoring mode minimizes the y component of the input vector. CORDIC is generally faster than other approaches when a hardware multiplier is unavailable (e.g. in a microcontroller), or when the number of gates required to implement is to be minimized (e.g. in an FPGA) [5].

The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shifts and adds. The algorithm, credited to Volder [2] is derived from the general rotation transform as shown in equations (3) and (4). The basic equations required to implement CORDIC are:

$$X(i+1) = X(i)\text{Cos}\varphi - Y(i)\text{Sin}\varphi \quad (3)$$

$$Y(i+1) = Y(i)\text{Cos}\varphi + X(i)\text{Sin}\varphi \quad (4)$$

$$X(i+1) = \text{Cos}\varphi\,(X(i) - Y(i)\text{Tan}\varphi) \quad (5)$$

$$Y(i+1) = \text{Cos}\varphi\,(Y(i) + X(i)\text{Tan}\varphi) \quad (6)$$

If the rotation angles are restricted so that $\text{Tan}\varphi = \pm 2e^{-i}$, the multiplication by the tangent term is reduced to simple shift operation. Equations (5) and (6) can now be expressed for each iteration as:

$$X(i+1) = Ki\,[\,X(i)\text{-}Y(i)d_i.\ 2\exp(-i)] \quad (7)$$

$$Y(i+1) = Ki\,[\,Y(i)\text{-}X(i)d_i.\ 2\exp(-i)] \quad (8)$$

$$Z(i+1) = Z(i) - d_i\varphi \quad (9)$$

where $Ki = \text{Cos}\,(\text{Tan}^{-1}\,2.\exp(-i))$
and $d_i = -1$ if $Z(i) < 0$, $+1$ otherwise
which finally provides the following result

$$X_n = A_n\,[X_0\,\text{Cos}Z_0 - Y_0\,\text{Sin}Z_0] \quad (10)$$

$$Y_n = A_n\,[Y_0\,\text{Cos}Z_0 + X_0\,\text{Sin}Z_0] \quad (11)$$

$$Z_n = 0 \quad (12)$$

$$A_n = \Pi_n\,(1 + 2^{-2i})^{1/2} \quad (13)$$

So to reach an expected angle, a series of iterations are required to be performed and in this design the number of iterations are i = 8 and in every iteration the new values of x, y and z depend upon the previous values of the same.

1) Sine and Cosine Calculation: The rotational mode CORDICoperation can simultaneously compute the sine and cosine of the input angle. Setting the y component of the input vector to zero reduces the rotation mode result to:

$$X_n = A_n \cdot X_0\,\text{Cos}Z_0 \quad (14)$$

$$Y_n = A_n \cdot Y_0\,\text{Cos}Z_0 \quad (15)$$

By setting X_0 equals to $1/A_n$, the rotation produces the un-scaled sine and cosine of the angle argument, Z_0. It is worth noting that the hardware complexity of the CORDIC rotator is approximately equivalent to that of a single multiplier with the same word size.

III. Implementation

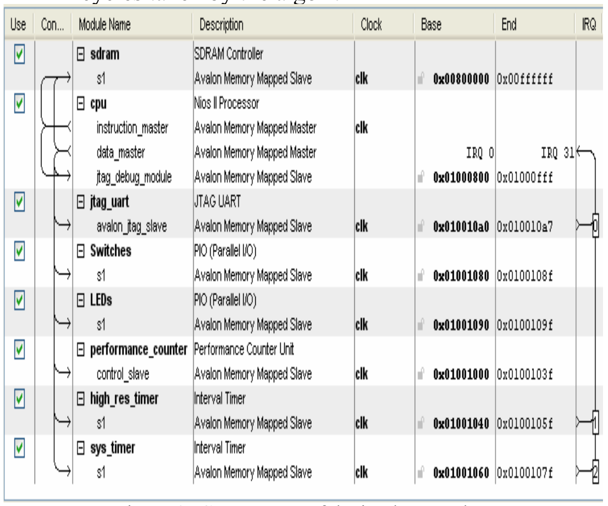
A. *Hardware Platform*

The NIOS II system is built using SOPC builder tool present in the Quartus software and is configured into the device. The device is configured in passive configuration mode- JTAG (Joint test action group) mode. The Quartus II software automatically generates *.sof* files that can be downloaded into FPGA using Byte-Blaster II or USB Blaster Cable for JTAG configuration.

1) Nios II System Design: The NIOS II is Altera's Second Generation Soft-Core 32 bit RISC Microprocessor. NIOS II plus all peripherals written in HDL which can be targeted for all Altera FPGAs is generated by the SOPC builder of Quartus II software and synthesized using Quartus II integrated synthesis. The design of NIOS II system is performed using SOPC builder [7] and the implemented system is shown in Fig. 4. The main components of the system are:

- NIOS II processor
- Avalon Tristate Bridge: To interface all peripherals with NIOS processor, parallel input/output (PIO).
- Parallel Input/Output peripherals
- SDRAM 8MB [9]
- PLL: To provide delayed clock input to SDRAM and Timers
- JTAG UART: To establish communication between PC and the FPGA
- Timer: Issues Interrupts to the processor to obtain timing information

- Performance Counter: Counts the number of clock cycles taken by the algorithm



Use	Con...	Module Name	Description	Clock	Base	End	IRQ
✓		⊟ **sdram**	SDRAM Controller				
		s1	Avalon Memory Mapped Slave	**clk**	**0x00800000**	0x00ffffff	
✓		⊟ **cpu**	Nios II Processor				
		instruction_master	Avalon Memory Mapped Master	**clk**			
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Memory Mapped Slave		**0x01000800**	0x01000fff	
✓		⊟ **jtag_uart**	JTAG UART				
		avalon_jtag_slave	Avalon Memory Mapped Slave	**clk**	**0x010010a0**	0x010010a7	0
✓		⊟ **Switches**	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	**clk**	**0x01001080**	0x0100108f	
✓		⊟ **LEDs**	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	**clk**	**0x01001090**	0x0100109f	
✓		⊟ **performance_counter**	Performance Counter Unit				
		control_slave	Avalon Memory Mapped Slave	**clk**	**0x01001000**	0x0100103f	
✓		⊟ **high_res_timer**	Interval Timer				
		s1	Avalon Memory Mapped Slave	**clk**	**0x01001040**	0x0100105f	1
✓		⊟ **sys_timer**	Interval Timer				
		s1	Avalon Memory Mapped Slave	**clk**	**0x01001060**	0x0100107f	2

Figure 4. Components of the implemented system

2) Custom Instruction: NIOS II processor custom instructions are custom logic blocks adjacent to the ALU in the cpu data path [8]. With custom instructions we can reduce a complex sequence of standard instructions to a single instruction implemented in hardware. The NIOS II CPU configuration wizard provides a facility to add up to 256 custom instructions to the processor. The custom instruction logic connects directly to the NIOS II processor ALU logic as shown in Fig. 5.

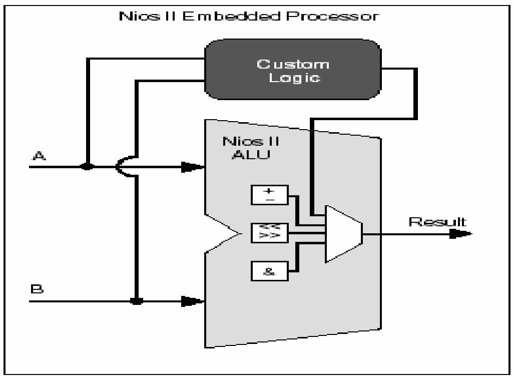


Figure 5. Custom instruction logic in NIOS

The algorithm is implemented in C language and the most computational intensive tasks of the algorithm are performed on a dedicated hardware accelerator using CORDIC processing elements to improve the performance by reducing the number of clock cycles required to implement the algorithm. The CORDIC algorithm is used to compute the sine and cosine values which are required to calculate the in phase and quadrature phase components of the received signals of the RAKE receiver. It is implemented in a pipelined architecture and the Fig. 6 shows one stage of pipelined architecture.

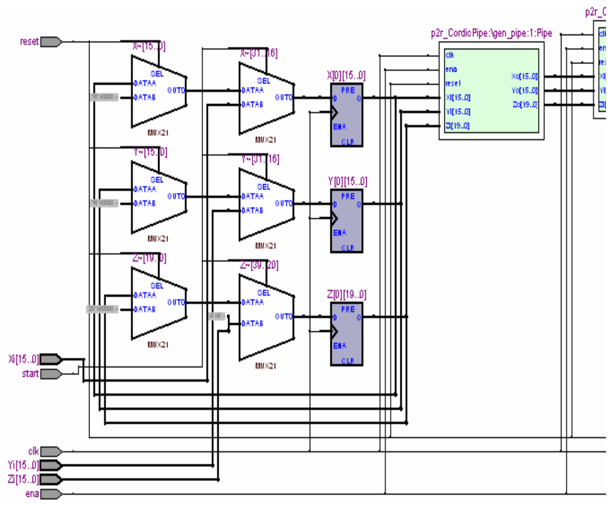


Figure 6. One stage of pipelined CORDIC architecture

The hardware block diagram of the entire NIOS II system is shown in Fig. 7.

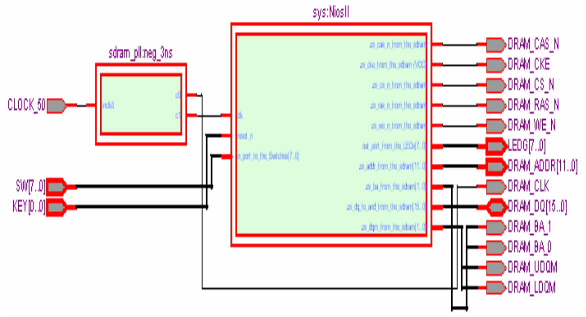


Figure 7. Hardware block diagram of Nios II System

B. *Software Implementation*

The NIOS II Integrated Development Environment (IDE) is used to run the software on top of NIOS II system. The base addresses are specified in the *system.h* header file and we need to include this file in our application code to access different components of the system. The algorithm steps [6] for the software implementation of the rake receiver are given below:

1. Compute the In-phase (I) and Quadrature (Q) phase components of inputs to each RAKE finger.
2. Calculate the Post correlation signal i.e. correlation of the spreading sequence with inputs computed in step 1.
3. Estimate the In-phase (I) and Quadrature (Q) phase components of channel impulse response.
4. Calculate the real and imaginary parts of the RAKE output.
5. The RAKE output is given to a decision device to obtain the received bits.

IV. RESULTS

The RAKE receiver is implemented on Altera FPGA and tested for different input data lengths and different chip rates.

The following results are obtained for 3000 bits of input data length and 64 chips per bit (chip rate). The profiling results regarding number of clock cycles required, time consumed by the algorithm, number of logical elements utilized and number of embedded multipliers utilized by the system with and without CORDIC hardware accelerator are given in Table 1.

Table 1. Performance comparison of the system with and with out CORDIC custom instruction

	With CORDIC Custom Instr	Without Custom Instr	% Impro-vement
Clock Cycles	1.11 E+10	2.80 E+10	60.5
Time (sec)	222.003	561.759	60.5
Logical Elements	5892	3662	37.8
Embedded Multipliers	4	4	0

The profiling results regarding number of clock cycles required, time consumed by the algorithm, number of logical elements utilized and number of embedded multipliers utilized by the system with both CORDIC hardware accelerator and floating point hardware and without both are given in Table 2.

Table 2. Performance comparison of the system with and with out both CORDIC custom instruction and Floating point hardware

	With both accelerators	Without both accelerators	% Impro-vement
Clock Cycles	7.41 E+09	2.80 E+10	73.6
Time (sec)	148.367	561.759	73.6
Logical Elements	11819	3662	69
Embedded Multipliers	11	4	63

The above comparison is graphically represented in Fig. 8.

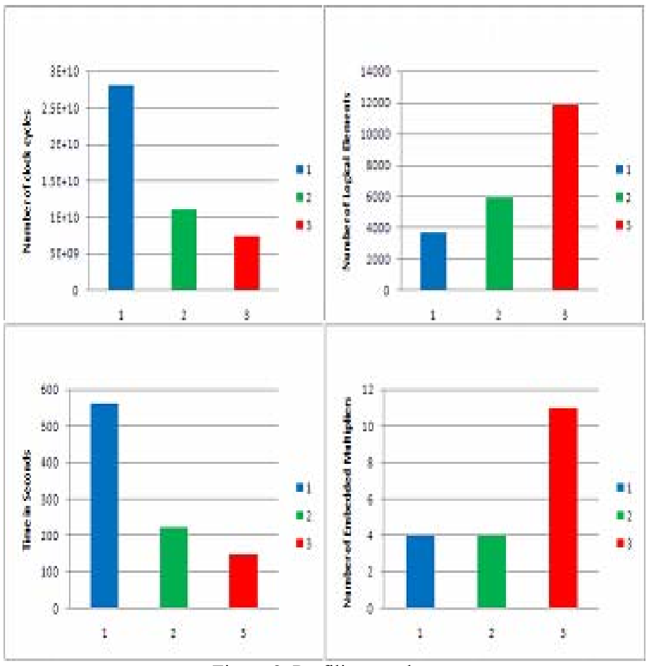


Figure 8. Profiling results
1 (Blue) – Without Custom Instruction
2 (Green) – With CORDIC Custom Instruction
3 (Red) – With CORDIC Custom Instruction and Floating Point Hardware

The performance of the RAKE receiver is illustrated using bit error rate (BER) calculations. These calculations depend on the combining scheme employed in the receiver. The comparison between different combining schemes for RAKE receiver is shown in Fig. 9.

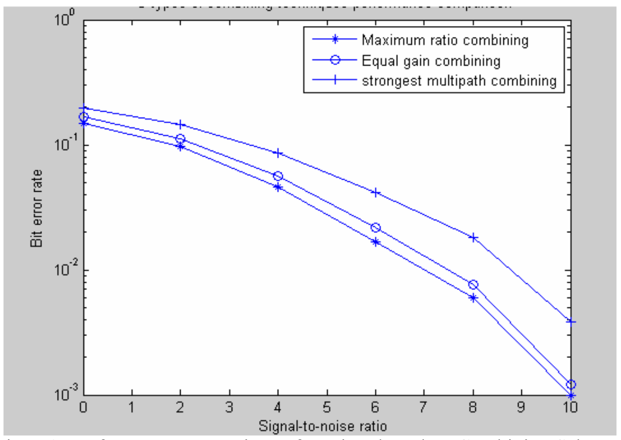


Figure9. Performance comparison of receiver based on Combining Scheme

V. CONCLUSIONS

In this paper, RAKE receiver for CDMA applications is implemented on Cyclone II FPGA using NIOS processor. Two different architectures were proposed to implement the receiver. A comparison between the two architectures shows that using a custom instruction (Hardware accelerator) coupled with the processor in a Co-design configuration reduces the number of cycles required to perform the most computational intensive tasks in the algorithm. We observed that an average of 70% improvement in the number of clock cycles with an average of 60% increase in the number of logic elements. In addition, while using custom instruction we can efficiently utilize the embedded multipliers (60%) provided with the hardware. That is adding custom instruction to the processor architecture improves the performance of the processor without losing the flexibility and with little increase in number of utilized logical blocks.

REFERENCES

[1] Heyne, B. and Goetze, J, "Cordic based algorithms for software defined radio (SDR) baseband processing," workshop on Advances in Radio Science, 2006.

[2] Tero Ojanperä, Ramjee Prasad, Wideband CDMA for Third Generation Mobile Communications, Norwood MA, USA, Artect House Inc., 1998, 439 pp.

[3] Simon Haykin, Michael Moher: Modern Wireless Communications, Prentice Hall 2005, pp. 258–338.

[4] Ray Andraka, "*A survey of CORDIC algorithms for FPGA based computers", ACM 0-89791-978-5/98/01, 1998.*

[5] Vikas Kumar, *FPGA Implementation of DFT Using CORDIC Algorithm*, Masters Thesis, Thapar University, India, June 2008.

[6] G. J. R. Povey, P. M. Grant, and R. D. Pringle, "A decision-directed spread-spectrum RAKE receiver for fast-fading mobile channels," *IEEE Trans. Veh. Technol.*, vol. 45, pp. 491–502, Aug. 1996.

[7] "*Nios II Tutorial*", Altera Corporation, San Jose C.A.

[8] "*Nios Custom Instructions Tutorial*", Altera Corporation, San Jose C.A., 2002.

[9] "*Nios II Hardware Development Tutorial*", Altera Corporation, San JoseC.A.