FPGA Implementation of Rapid PN Code Acquisition Using Iterative Message Passing Algorithms

Wei Wang , Zhihua Wang
Harbin Engineering University

INTRODUCTION

Spread-spectrum (SS) techniques are widely used in many military communication systems because they have many advantages, such as ranging capabilities, antijam protection, low probability of detection or interception, and multiple access capability. There are many forms of SS, such as a direct sequence spread spectrum (DS/SS), frequency hopping (FH), hybrid DS-FH, and ultrawideband (UWB) [1], [2], [3]. In all cases, pseudorandom or pseudonoise (PN) sequences play important roles. They are periodic sequences with a long period in practical systems. The long PN sequences are desirable to reduce the probability of detection by an unintended receiver over a short time interval. But it is also a challenge for a receiver to detect and acquire the signal [4].

Most detection algorithms are based on the correlation between the received PN code and its local replicas that are shifted until the right alignment is achieved [5]. The traditional approaches of PN code acquisition are serial search [6], parallel search [7], and hybrid search [8]. Serial search is simple but slow to acquire; parallel search is fast to acquire but complex. Hybrid search provides a linear scale tradeoff between these two extremes.

To reach rapid PN code acquisition at low complexity, a new approach based on iterative message passing algorithms (iMPAs) has been proposed in [1], [9], [10], [11]. It is generalized from the well-known turbo-decoding algorithm [12]. Turbocodes are a class of forward error correction codes. The iterative decoding algorithm is defined as a technique employing a soft-output decoding algorithm that is iterated several times to improve the error performance. However, the acquisition performance of iMPAs degrades at low signal-to-noise ratios (SNRs). A soft information improvement using multiple samples in one chip is proposed in [13]. In iterative acquisition, soft information is a metric about the channel output, and it is the initial input of the iterative process. But the timing error affects the improvement performance. Maximum-likelihood (ML) estimation is introduced to mitigate the timing error.

Although iMPA has been studied intensively in recent years, only a few papers focus on the hardware implementation of iMPA. In [4] Yeung and Chugg present low-complexity hardware architecture for fast acquisition of long PN codes in a UWB system. The implementation of the message passing detection unit and decoder architecture based on iMPA is introduced in [5].

This article addresses the design of field-programmable gate array (FPGA) implementation of PN sequences acquisition algorithms using iMPA with soft information improvement. The principle diagram of acquisition algorithms is introduced first. Then the architecture of FPGA implementation is described, and every module is implemented in detail. Finally, the performance of FPGA implementation is given.

Authors' address: W. Wang, Z. Wang, Harbin Engineering University, Automation, Room 409, Building 31, No. 145 Nantong Street, Nangang District, Harbin, Heilongjiang 150001, China, E-mail: (chinaww2006@yahoo.com.cn). This work is supported by the New Century Excellent Talents Support Program (NCET-11-0827), Fundamental Research Funds for the Central Universities (HEUCFX-41308), Heilongjiang Postdoctoral Special Fund (LBH-TZ0410), and Innovation of Science and Technology Talents in Harbin (2013RFXXJ016).
Manuscript received December 10, 2012, revised July 30, 2013, and October 9, 2013, ready for publication November 20, 2013.
DOI. No. 10.1109/MAES.2014.120228.
Review handled by C. Fernandez-Prades.
0885/8985/14/ $26.00

THEORY OF OPERATION

The PN sequences acquisition algorithm based on iMPA is shown in Figure 1. The basic principle is that iMPA module uses soft information provided by the received signal to estimate the current state of the PN sequence. Then, an initial state vector with a higher belief is sent to the linear feedback shift register so as to generate a credible local PN sequence directly. Finally, the decision unit is used to decide whether the right phase of the PN sequence has been found. The whole system mainly contains five blocks: digital downconverter, ML estimator, arithmetic mean unit, iMPA module, and the PN code generator and decision module. The core of the system consists of the ML estimator, arithmetic mean unit, and iMPA module, which are introduced below in detail.

THE SIGNAL MODEL

For a binary phase shift keying (BPSK)–modulated DS/SS system, the model of received signal can be written as

$$r = s(t,\tau) + n(t) \tag{1}$$

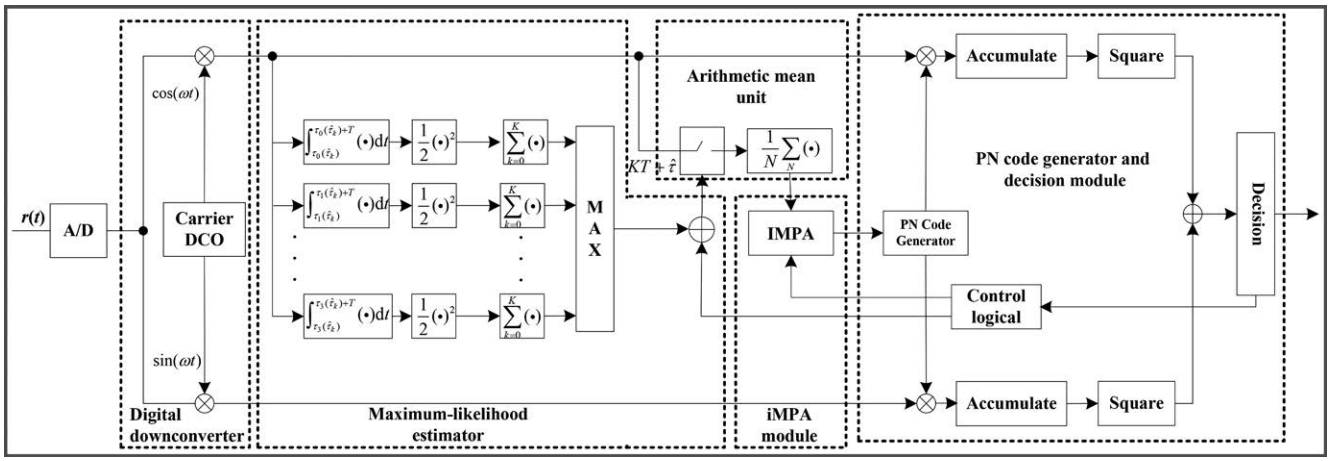

Figure 1.
The principle diagram of PN sequences iterative acquisition algorithms.

where $s(t, \tau)$ is the transmitted signal and $n(t)$ is additive white Gaussian noise with power-spectral density $\sigma 2$. The transmitted signal can be represented as

$$s(t,\tau) = \sum_{k=-\infty}^{\infty} \sqrt{E_c}(-1)^{x_k}\ p(t - kT_c - \tau)\cos(\omega t + \varphi) \tag{2}$$

where E_c is the transmitted energy per chip, x_k is the m sequence with index k, ω is the carrier frequency, φ is the carrier phase, $p(\cdot)$ represents m-sequence pulse shaping, T_c is the chip interval, and τ is the time delay.

For simplification, we suppose ω to be constant and φ to be zero. So after analog-to-digital (A/D) sampling at the chip rate and digital downconversion, the base band signal can be represented as

$$z(kT_c) = \frac{1}{2}\sqrt{E_c}(-1)^{x_k} + w_k \tag{3}$$

where w_k is noise sample. As a result, an observation vector $z = [z_0, z_1, \ldots, z_{M-1}]$ can be obtained after M times sampling $r << M << 2^r - 1$, where r is the stage of the m sequence.

THE PRINCIPLE OF IMPA

The iMPA is similar to low-density parity check code and turbocode decoding. A detailed discussion of iMPA is given in [1]. Here, we just give an example of rapid acquisition based on iMPA. The generating polynomial of the m sequence is $g(D) = 1 + D + D^{15}$. Each check node enforces the constraint $x_k \oplus x_{k-1} \oplus x_{k-15} = 0$ for the appropriate value of k. The number of valid local configurations is $x_k, x_{k-1}, x_{k-15} \in \{(0,0,0),(0,1,1),(1,0,1),(1,1,0)\}$. The factor graph with one check node is shown in Figure 2.

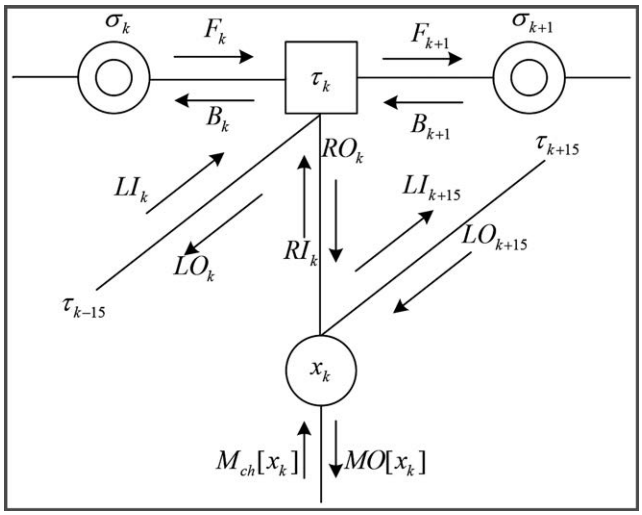

Figure 2.
The m-sequence factor graph of one node.

In Figure 2, F_k and B_k are defined as the forward and backward message variables, respectively. LI_k, RI_k, LO_k, and RO_k are defined as the $\tau_{k\text{-}15}\_\tau_k$ and $x_k\_\tau_k$ input and $\tau_k\_\tau_{k\text{-}15}$ and $\tau_k\_x_k$ output update messages. Mch_k is the initial message, and MO_k is the update message. The logarithmic likelihood ratio of observation vector z_k is defined as initial message

$$Mch_k = \log\frac{P(z_k|x_k=1)}{P(z_k|x_k=0)} = \frac{2\sqrt{E_c}}{\sigma^2}z_k \tag{4}$$

$P(z_k \mid x_k)$ is presented as the likelihood function of received signal, and

$$P(z_k|x_k) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(z_k-(-1)^{x_k})^2}{2\sigma^2}\right) \tag{5}$$

considering each variable node and check node (x_k, τ_j). The updated message from the check node to the variable node is defined as $\Delta\eta_{j,k}$. The updated message from the variable node to the check node is defined as $\Delta u_{k,j}$. The process of iMPA is shown as follows:

(1) Initialization: Assign initial values to $\Delta u_{k,j}$ and $\Delta\eta_{j,k}$:

$$\Delta u_{k,j} \leftarrow Mch_k, \Delta\eta_{j,k} \leftarrow 0,\ 0 \le k, j \le M-1 \tag{6}$$

(2) Check node update

$$\Delta u_{k,j} = Mch_k + \sum_{\substack{\forall n:\tau_n \to x_k \\ n\neq j}} \Delta\eta_{n,k},\ 0 \le k, j \le M-1 \tag{7}$$

where the subscript of sum $\forall n : \tau_n \to x_k$ represents the set of all check nodes that connect with variable node x_k except τ_j.

(3) Variable node update

$$\Delta\eta_{j,k} = \prod_{\substack{\forall n:x_n \to \tau_j \\ n\neq k}} \sin\Delta\mu_{n,j} \cdot \min_{\substack{\forall n:x_n \to \tau_j \\ n\neq k}} \left|\Delta\mu_{n,j}\right|, \quad 0 \le k, j \le M-1 \tag{8}$$

where the subscript of product $\forall n : x_n \to \tau_j$ represents the set of all variable nodes that connect with check node τ_j except x_k. Implementing steps 2 and 3 for all nodes once represents an iterative procedure. To obtain enough stability, many iterative procedures are needed.

(4) Calculate the metric of soft information:

$$\Delta so_k = Mch_k + \sum_{\forall n:\tau_n \to x_k} \Delta\eta_{n,k},\ 0 \le k \le M-1 \tag{9}$$

(5) Decision:

$$x_k = \begin{cases} 1, \Delta so_k < 0 \\ 0, \Delta so_k > 0 \end{cases}, 0 \le k \le M-1 \tag{10}$$

Finally, a credible local PN sequence is generated based on the decision above.

SOFT INITIAL INFORMATION IMPROVEMENT

The iMPA can acquire the PN code rapidly, which is discussed in [1]. But the acquisition performance degrades at low SNRs. In [13], a new approach named the soft information improvement iterative (SIII) method has been proposed to solve the problem at low SNRs. Here, we use the arithmetic mean unit to improve the quality of the soft initial information with multiple samples per chip. Suppose there are N samples in one chip. By using an arithmetic mean unit, a new observation can be obtained from (3):

$$z_k = \frac{1}{N}\sum_{n=0}^{N-1} r(nT)\cos\omega nT = s_k(\tau,\omega) + n_k,\ 0 \le k \le M-1 \tag{11}$$

where $\tau = lT (0 \le \tau < T_c,\ 0 \le l < N)$ represents the normalized timing delay. T is the chip interval. Consider the worst situation: if two neighboring chips have opposite polarity (i.e., $x_k x_{k+1} = -1$), then we can find a proper approximation to represent $s_k(\tau, \omega)$ as follows:

$$\frac{s_k(\tau,\omega)}{\sqrt{E_c}(-1)^{x_k}} \approx -\frac{1}{2} + \frac{1}{N} - \frac{1}{2}\frac{\sin(2\omega NT - \pi/N)}{2\omega} + \frac{\sin(2\omega lT - \pi/N)}{2\omega} \tag{12}$$

and the noise part is:

$$n_k = \frac{1}{N}\sum_{n=0}^{N-1} n(nT)\cos\omega nT \tag{13}$$

with a zero mean and the variance $\sigma^2/2N$. So the power of noise decreases, and the SNR increases. The acquisition performance is improved.

However, the improved information capacity decreases when a timing error exists. As an extreme, no useful information can be obtained from the channel observation. So we should eliminate the timing error before using the arithmetic mean unit to ensure the quality of the improved information.

ML ESTIMATION FOR TIMING ERROR MITIGATION

The ML estimation can obtain the optimal approximation with sufficient data, which is an effective approach to measure the timing delay. Consider the received signal models represented in equations (1) and (2). Because the polarity of

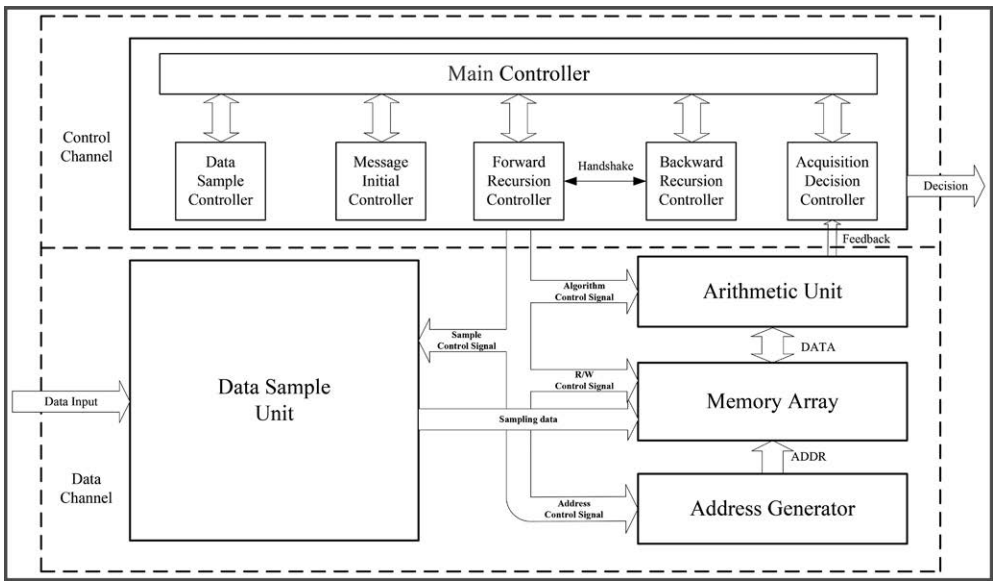

Figure 3.
The block diagram of FPGA implementation based on the iterative acquisition algorithm.

the m sequence has uniform probability, the average likelihood ratio of received signal can be obtained as

$$\bar{\Lambda}_L(\tau)=\sum_{k=0}^{K-1}\ln\left[\cos h\left(ch_k(\tau)\right)\right] \tag{14}$$

where $ch_k(\tau)=\dfrac{2\sqrt{E_c}}{\sigma^2}\int_{T_c} r(t)\cdot s(t,\tau)\,dt.$

Using approximation as

$$\ln\cos h\, x\approx\begin{cases}\dfrac{1}{2}x^2, & \left(|x|\ll 1\right), \text{ for low SNR}\\ |x|, & \left(|x|\gg 1\right), \text{ for high SNR}\end{cases} \tag{15}$$

we can obtain the ML estimation of timing delay:

$$\hat{\tau}=\arg\max_{\tau}\bar{\Lambda}_L(\tau)=\arg\max_{\tau}\left(\sum_{k=0}^{K-1}\frac{1}{2}\left(ch_k(\tau)\right)^2\right) \tag{16}$$

According to (16), an ML estimation scheme can be formed, where $\hat{\tau}$ is the final destination and K is the data record length. In this scheme, τ is divided into $[\tau_1, \tau_2, \tau_3, \tau_4]$, and then each $\bar{\Lambda}_L(\tau_i)$ is calculated. Finally, we choose τ_i which results in a maximum value of $\bar{\Lambda}_L(\tau)$ as the estimation result, namely, $\hat{\tau}$. This estimation result can be used in the next estimation $(\tau=\hat{\tau})$ to further reduce the uncertainty of the timing delay. After several iterative processes, a precise estimation can be obtained.

FPGA IMPLEMENTATION

In this section, we present the FPGA implementation based on iMPA. Every module is described in detail.

IMPLEMENTATION ARCHITECTURE

Based on the principle diagram of the PN code acquisition algorithm in Figure 1, the acquisition algorithm contains two types of process. The first process is made up of the digital downconverter, ML estimation, arithmetic mean, and correlation. They process data independently. The other process is control logic, whose primary function is passing and updating messages. Following the FPGA design methodologies, the implementation is divided into two channels. The first channel is the data channel containing algorithms about data processing, and the other channel is the control channel. Both channels use an algorithm state machine (ASM).

The FPGA implementation architecture of iterative acquisition algorithms is shown in Figure 3. The control channel and the data channel constitute the whole acquisition algorithm. In the control channel, there are five subcontrollers: the data sample controller, message initial controller, forward recursion controller, backward recursion controller, and acquisition controller. Under the control of main controller, the data sample controller samples data via A/D sampling and obtains the state of sampling. The message initial controller initializes the information of iterative acquisition. The forward/backward recursion controllers cooperate to update and store the iterative message. Finally, the decision is given by the acquisition decision controller based on the results of the iterative process.

In the data channel, there are four modules: the data sample unit, arithmetic unit, memory array, and address generator. To generate the initial information of iteration, the data sample unit contains the digital downconverter module, ML estimation module, and arithmetic mean module. The arithmetic unit accomplishes the iterative process. The memory array stores the iterative messages, and the address generator generates the address of memory array following the address control signal. In the system, the basic bit width of the A/D sample is 4 bits, and the bit width of the forward/backward state variable is 8 bits.

CONTROL CHANNEL

The control channel is divided into one main controller and five subcontrollers. ASM is used to implement every module.

The main state machine is the core of control channel, which controls every subcontroller to work orderly. The main state machine is made up of five states: idle state S_IDLE, collect state S_COLLECT, message initial state S_INITIAL, itera-

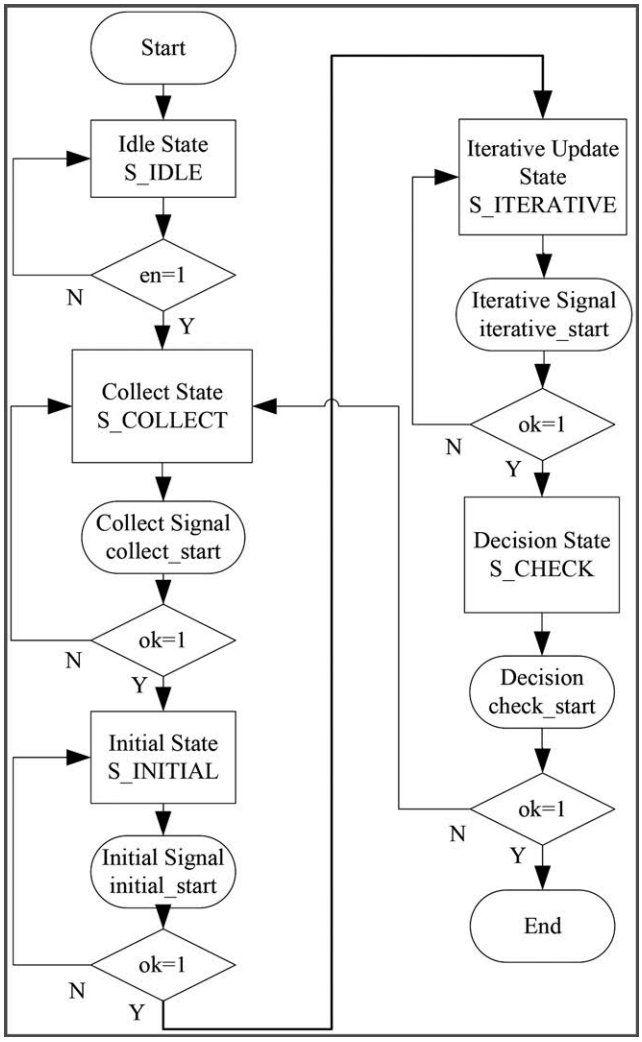

Figure 4.
The diagram of the main state machine.

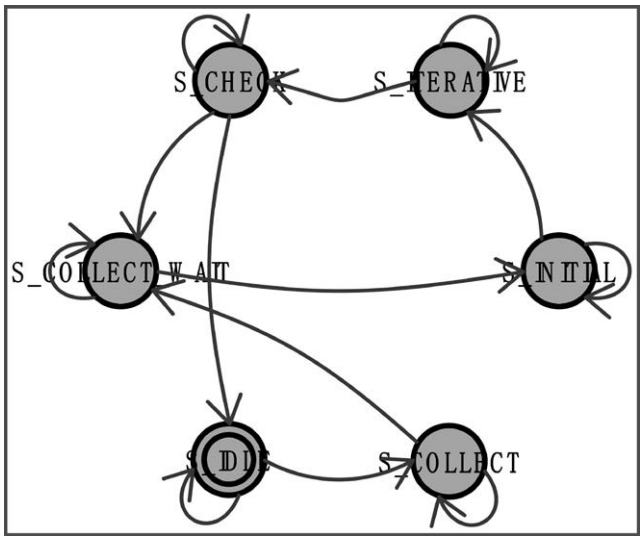

Figure 5.
The FPGA implementation of the main state machine.

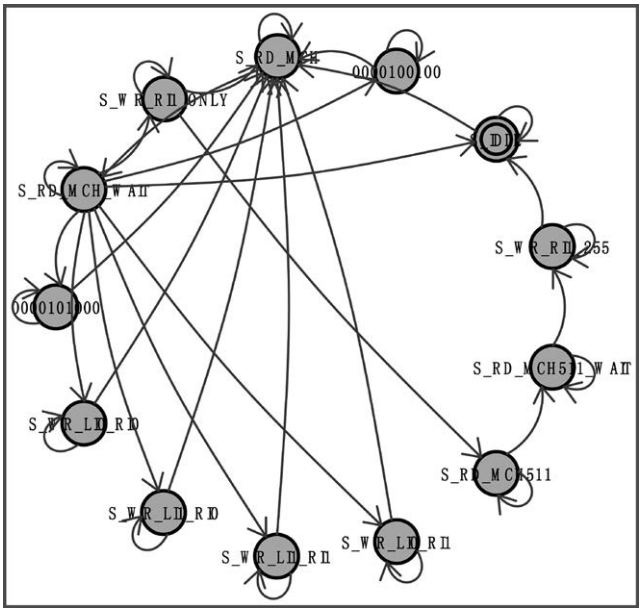

Figure 6.
The implementation of the signal initial FSM.

tive updating state S_ITERATIVE, and check state S_CHECK. Figure 4 shows the diagram of main state machine.

When the reset signal appears, the initial signal is sent to the data sample unit. The initial state is changed to the collect state when the enable signal happens. In the collect state, the data sample controller is controlled by the collect signal. The accomplished signal appears when the data are sampled sufficiently. Then, the collect state is changed to the message initial state. The message initial state starts the initial controller to initialize the initial message of iteration. Until the initial process is finished, the state is changed to the iterative update state, which is made up of forward recursion and backward recursion. When the iterative algorithm is finished, the decision state checks whether the acquisition condition is sufficient. If the condition exceeds acquisition threshold, the acquisition is successful. Otherwise, a new acquisition process begins.

The ASM of the main state machine is implemented in Verilog hardware description language (HDL) and synthesized using Synplify Pro. Quasi-one-hot coding is used to improve the speed of the finite state machine (FSM). To adjust the timing between every pair of states, the signal collect wait state is added. The FPGA implementation of the main state machine is shown in Figure 5.

The message initial state machine reads data from the data sample unit and initializes the initial information of iterative processing. Initial state S_IDLE, read collect state S_RD_Mch, and write state S_WR_LI_RI are in this module. The function of the message initial state machine is just reading and writing data. Although it is simple in logic, it consumes much memory. To reduce the memory consuming in this module, every message variable uses independent memory space. However, the complexity increases, so we add some external state to make the timing simple. Figure 6 shows the FPGA implementation of the message initial state machine.

The forward/backward iterative state machine is the core of an iMPA. It is also the most complex part of this algorithm. We analyze the two state machines at the same time because they work in parallel. The forward/backward iterative algorithm is implemented in five steps, shown in Figure 7. First, the state machine accomplishes the forward recursion algorithm of segment 0. In this step, RI and LI messages are read from the message memory of segment 0. Then, minimum sum operation is accomplished to obtain the forward metric state F of the next segment and store it in state metric memory (SMM), denoted by writing $F_{0:127}$ to $SMM_{0:127}$.

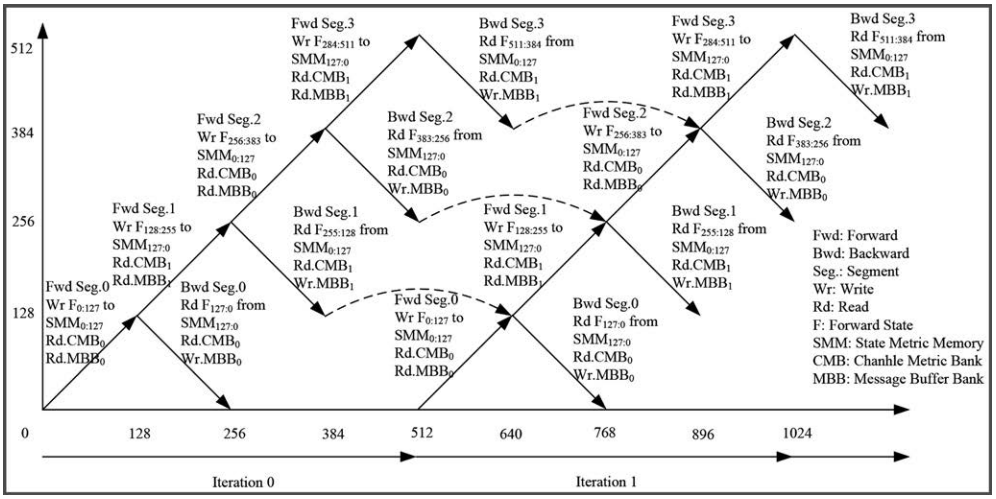

Figure 7.
The read/write schedule of the message variable in the forward/backward iterative process.

Second, the state machine implements the backward recursion of segment 0 and the forward recursion of segment 1. In this step, the new message variables RI and LI are updated through the reading message variables RI and LI from backward recursion and reading forward SMM. Because the forward recursion writes SMM and reads RI and LI simultaneously, the conflict of read/write should be solved. Handshake protocol is a good approach to solve this problem. RI_BUFFER and LI_BUFFER are set in the forward recursion unit, which stores the copy of RI and LI. So we can write RI and LI and read RI_BUFFER and LI_BUFFER at the same time, denoted by reading $F_{127:0}$ from $SMM_{127:0}$ and writing $F_{128:255}$ to $SMM_{127:0}$.

Third, the state machine implements the backward recursion of segment 1 and the forward recursion of segment 2. The details are similar to the second step and denoted by reading $F_{255:128}$ from $SMM_{0:127}$ and writing $F_{383:256}$ to $SMM_{0:127}$.

Fourth, the state machine implements the backward recursion of segment 2 and the forward recursion of segment 3. The details are similar to the second step and denoted by reading $F_{383:256}$ from $SMM_{127:0}$ and writing $F_{384:511}$ to $SMM_{127:0}$.

Fifth, the backward recursion of segment 3 and the forward recursion of the first iteration are accomplished. So the forward recursion of segment 0 in the next iteration begins. The details are similar to the second step and denoted by reading $F_{511:384}$ from $SMM_{0:127}$ and writing $F_{0:127}$ to $SMM_{0:127}$. The data of the first segment that has been updated in the first recursion is updated again by the second iteration, and then the iteration is accomplished.

The next step is similar to the above five steps until the final backward recursion of the final iteration is accomplished. In this process, there is no forward recursion, because no recursion is needed. So it just read $F_{511:384}$ from $SMM_{0:127}$.

Based on the above analysis, the FPGA implementation of forward recursion and backward recursion is given in Figures 8 and 9. Between them, handshake protocol is used.

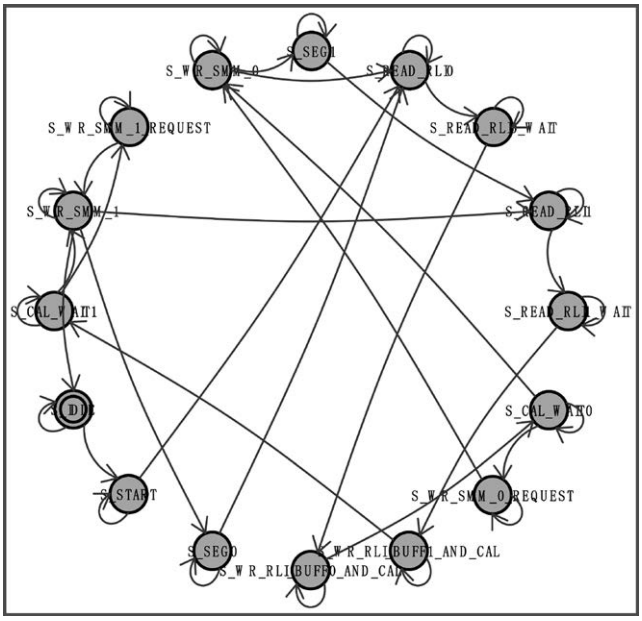

Figure 8.
The FPGA implementation of the forward iterative state machine.

DATA CHANNEL

The data channel is almost regular logic, because it just repeats the same operation for the data arithmetic. Selecting the data channel and transferring the data between arithmetic logic and inner register is the main job of the data channel. In the PN iterative acquisition circuit, there are four modules: the data sample unit, addition compare select (ACS) arithmetic logic, and memory array and address generator.

The data sample unit plays important role in data channel. Its design is showed in Figure 10. The data sample and the iterative algorithm implement synchronically by using a ping-pong operation. After A/D sampling at the chip rate and digital downconversion, the base band signal is put into

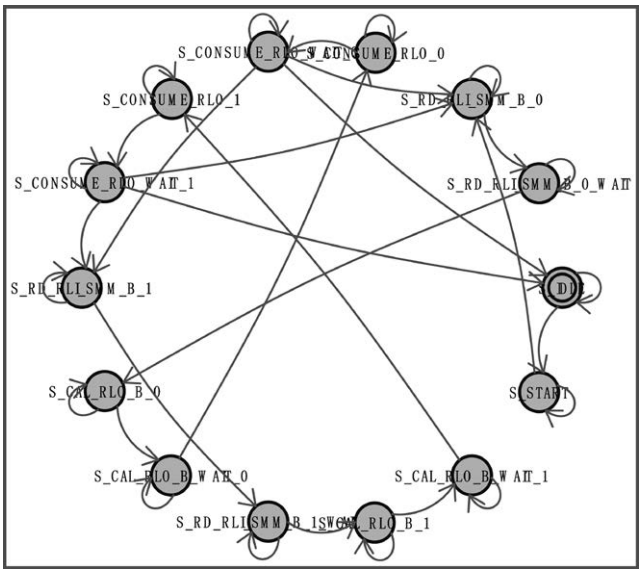

Figure 9.
The FPGA implementation of the backward iterative state machine.

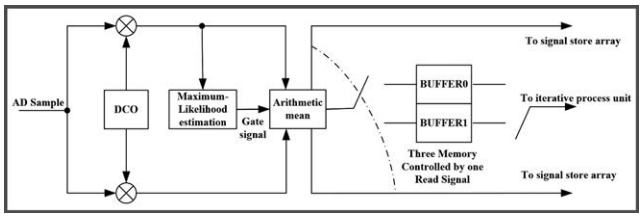

Figure 10.
The diagram of the data sample unit.

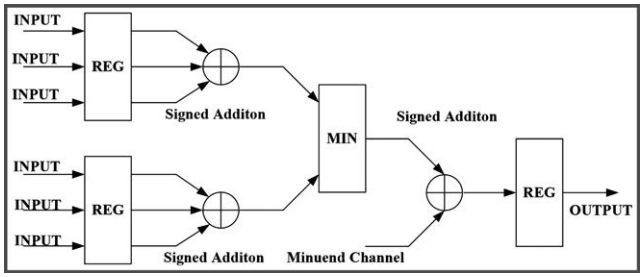

Figure 12.
The diagram of ACS.

ML estimation. Then, the arithmetic mean module starts, when the start signal is coming from ML estimation. Finally, the mean of multiple samples is calculated and stored in buffers. Because the ping-pong operation has two buffers, the data are stored in two buffers by turns. When one is full, the other begins to store. The schematic of the FPGA implementation is shown in Figure 11.

The ACS algorithm logic is similar to the arithmetic logic unit used in a central processing unit (CPU). It accomplishes basic algorithms such as addition, compare, and select. Because ACS works multiple times, it affects the clock frequency of system. The standard operation of ACS is

$$F[x_k] = \min\left(RI[x_k] + LI[x_{k-15}], F[x_{k-1}] + RI[x_k]\right) - \min\left(0, F[x_{k-1}] + LI[x_k]\right)$$

$$LO[x_k] = \min\left(B[x_k] + RI[x_k], F[x_{k-1}]\right) - \min\left(F[x_{k-1}] + B[x_k] + RI[x_k], 0\right)$$

$$RO[x_k] = \min\left(B[x_k] + LI[x_{k-15}], F[x_{k-1}] + B[x_k]\right) - \min\left(0, F[x_{k-1}] + LI[x_{k-15}]\right)$$

$$B[x_{k-1}] = \min\left(B[x_k] + RI[x_k], LI[x_{k-15}]\right) - \min\left(B[x_k] + LI[x_{k-15}] + RI[x_k], 0\right) \quad (17)$$

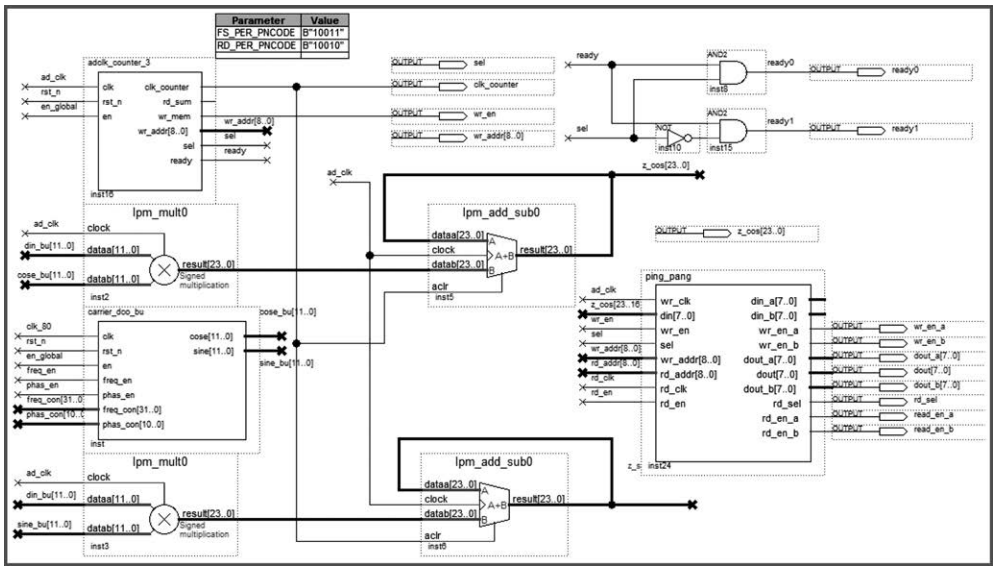

Figure 11.
The schematic of the FPGA implementation about the data sample unit.

Figure 12 shows the diagram of ACS. First, it reads the input message variable. Then it sums and compares the data. Finally, the ACS module updates the messages based on the compared results. The FPGA implementation circuit of ACS is shown in Figure 13. It includes the update circuit of message variable LO and LI.

The memory array is similar to random access memory (RAM) of a computer, while the address generator is similar to the address decoder of a CPU. The memory array is used to store the initial variable, intermediate message

Table 1.

Parameter of Every Memory in the Message Memory Array				
Name of Memory	**Bit Width**	**Bit Depth**	**Count**	**Type**
Channel metric memory Mch	4	512	2	Dual-port RAM
SMM	9	128	1	Dual-port RAM
Message various memory RI	5	256	2	Dual-port RAM
Message various memory LI	5	256	2	Dual-port RAM
Message various memory RI_BUFFER	5	128	2	Dual-port RAM
Message various memory LI_BUFFER	5	128	2	Dual-port RAM
Output message memory Mdec	0	15	5	Shift register

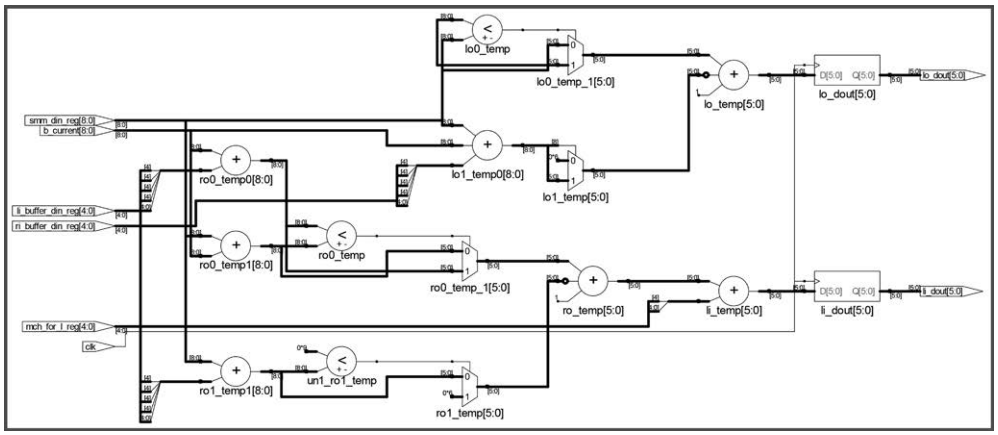

Figure 13.
The implementation circuit of ACS.

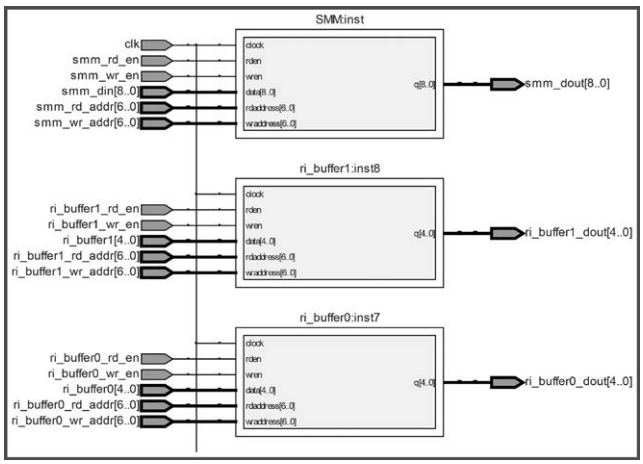

Figure 14.
The circuit of the message memory array.

variable, and output message variable. The detail is shown in Table 1.

To reduce the complexity of SMM, the data module with 512 data units is divided into four segments. Every segment has 128 data units. The iMPA is implemented segment by segment. In the forward recursion algorithm, the forward unit update state variable F is from 0 to 511, and in the backward recursion algorithm, the state variable B is updated as the order of 127 to 0, 255 to 128, 383 to 256, and 511 to 383. Based on above analysis, in the message updating process, the variable B should not be stored. Therefore, the memory depth of state variable F is 128, denoted by SMM. The message variable memory RI, LI, RI_BUFFER, and LI_BUFFER are used to store middle variable. Sampling data of two channels are stored in Mch. Mdec stores the message results.

Dual-port RAM based on the M4K module is used in FPGA implementation. The design result is shown in Figure 14. In the circuit, we can see that every memory is independent and controlled by the read/write signal from the control logic and the address from the address generator. Both the control logic and the address generator are synchronized.

RECOVERY AND DECISION

The recovery and decision module is a part of acquisition decision controller. The recovery and decision module is used to control the process of acquisition and decision. The flow chart is shown in Figure 15. It starts to read Mdec from memory by turns when the enable signal is coming from the main controller.

Then, the vectors in five special positions are calculated. The local PN sequence is generated by the PN sequence recovery module based on the vectors and correlated with the data in channel memory. Finally, the mean of the correlation results in two channels is compared with the threshold to decide whether the acquisition was successful. If the corre-

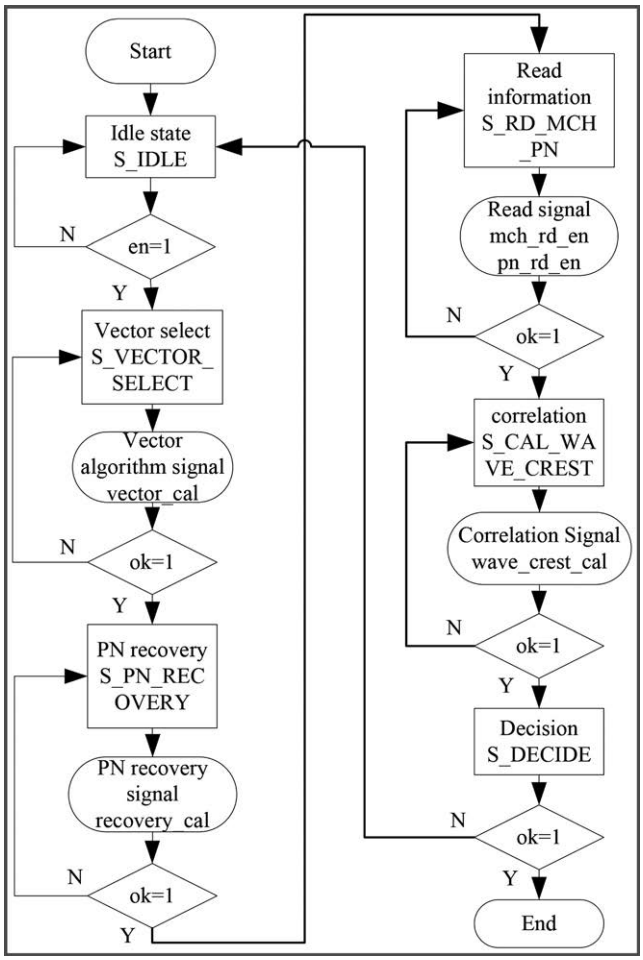

Figure 15.
The flow chart of the acquisition and decision module.

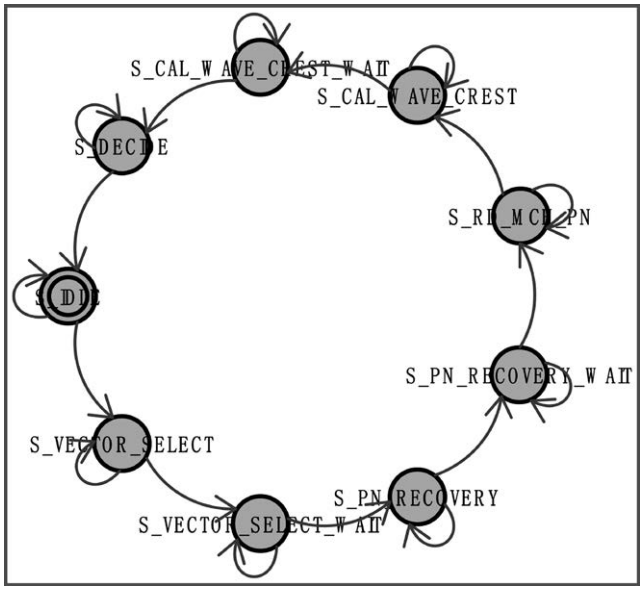

Figure 16.
The FPGA implementation of the acquisition and decision module.

lation results exceed the acquisition threshold, the acquisition was successful. Otherwise, a new observation vector is formed and processed until the acquisition is successful. The FPGA implementation is shown in Figure 16.

The PN sequence recovery module recovers the local PN sequence based on the PN vectors and its position. A forward/backward PN sequence generate module is shown in Figure 17. The PN sequence recovery module is made up of two m-sequence generators, two sequence memory spaces, and a sequence synthesizer. The two m-sequence generators are mirror images of each other. This means that if there is an m-sequence polynomial $g(D)$, then its mirror polynomial is $g(R)(D) = D^T \cdot g(1/D)$. The m-sequence generator generates two-direction m sequences, which will be stored in memory. Then, the sequence synthesizer generates a sequence with the two m sequences based on the position of where the PN vectors are in the sequence.

The threshold decision module correlates the local PN sequence generated by the PN sequence recovery module and the channel messages stored in the local. The diagram of the threshold decision module is shown in Figure 18. It shows that I and Q channel data correlate with the register in two different channels. Then, the sum of the square is accomplished in two channels and compared to the threshold to decide whether the acquisition was successful. The FPGA implementation circuit of the threshold decision module is shown in Figure 19.

EXPERIMENT AND VERIFICATION

In this section, the acquisition probability with three kinds of acquisition methods is given. They are the SIII method, the

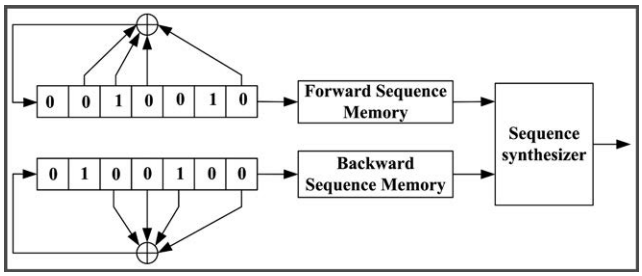

Figure 17.
The diagram of the forward/backward PN sequence generate module.

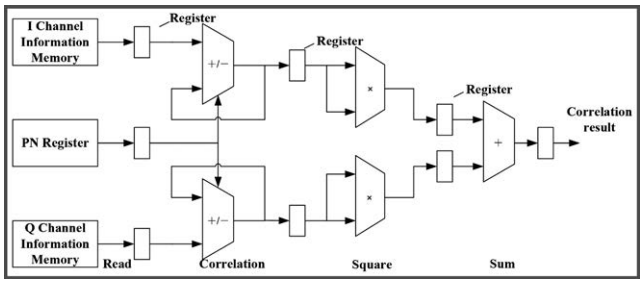

Figure 18.
The diagram of the threshold decision module.

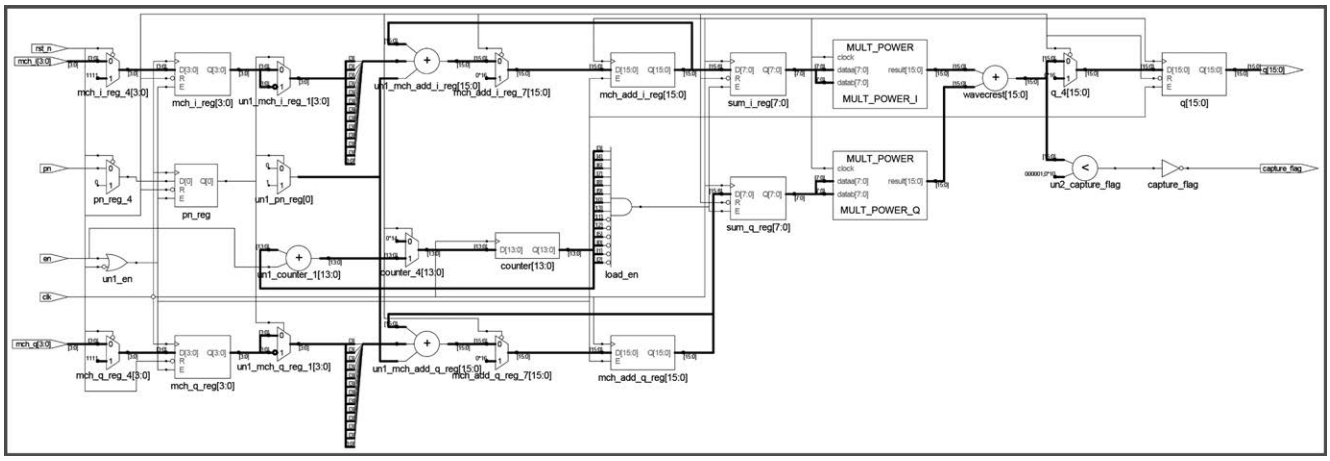

Figure 19.
The FPGA implementation circuit of the threshold decision module.

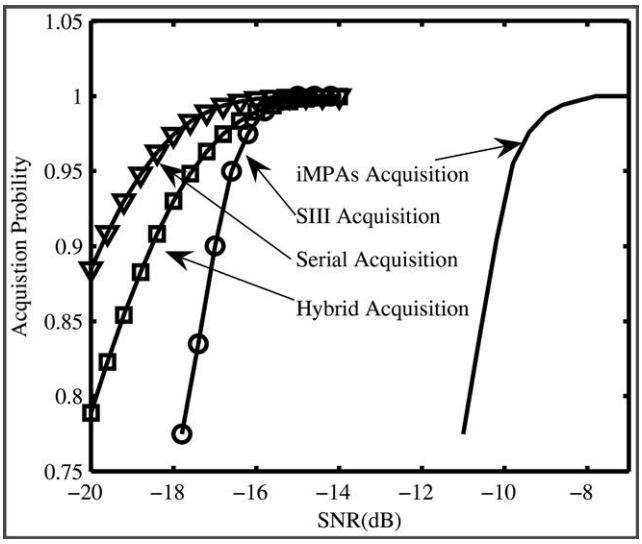

Figure 20.
The acquisition probability comparison.

iMPA acquisition method, and the conventional acquisition method, which includes serial acquisition and hybrid acquisition (here, a structure with eight branches is adopted). Assume the chip interval $1/T_c$ = 1 MHz, the carrier frequency $\omega/2\pi$ = 5 MHz, and the sampling frequency is 20 MHz. Set the length of observation M = 512, the data record length of ML estimator K = 256, and the iteration time as 20. From Figure 20, we can observe that the SIII method can reach a rapid acquisition with a lower SNR than the iMPAs at a given acquisition probability. Compared to iMPAs, the SIII method provides approximately 7 dB of SNR improvement. When the SNR is higher than −16 dB, the acquisition probability of the SIII method is close to conventional acquisition methods.

The architecture is implemented by Verilog HDL. The code is synthesized by Synplify Pro and then mapped by the Altera Foundation to an Altera Stratix 1 device (EP1S80B956C6). The total logic element is 32586, the total memory bit is 686592, and the used digital signal processing block 9-bit elements are 10.

To validate the performance of the acquisition circuit designed above, we designed a test bench. In the test bench, a 15 order PN sequence is used. Here, we design a SS signal generator to produce the PN sequence and carrier wave. The modulate mode is BPSK. It also can simulate the delay of the SS signal and noise. The test bench is implemented in ModelSim, which can combine the signal generator, PN sequence acquisition circuit, and test file. Through configuring the signal generator, the whole test is divided into two parts. In the first part, the SNR of the input signal is −10 dB; in the other part, the SNR is −15 dB.

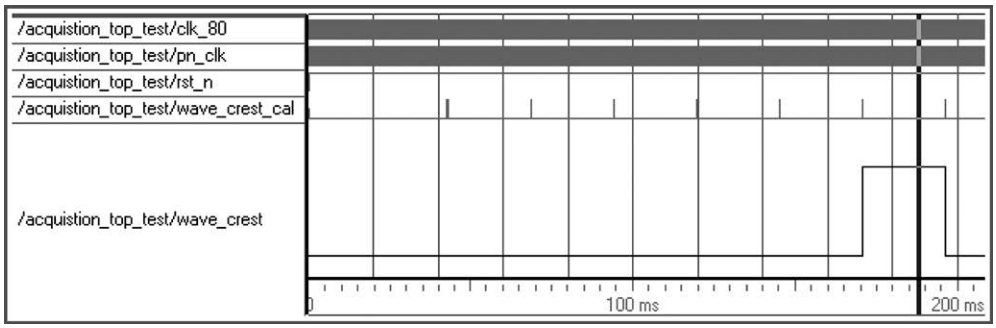

Figure 21.
The correlation peak value under the SNR of –10 dB.

The correlation peak value under the SNR of −10 dB is shown in Figure 21. We can see that under this SNR, only

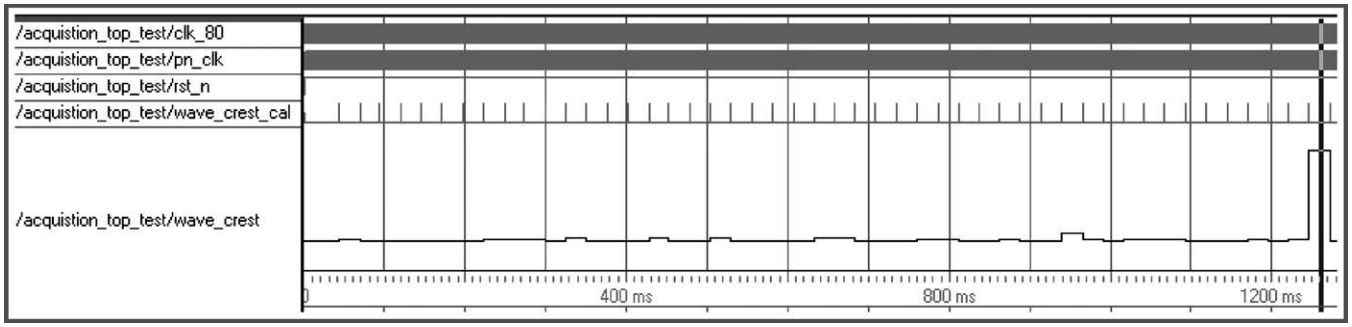

Figure 22.
The correlation peak value under the SNR of –15 dB.

six modules are consumed to produce the correlation peak value. The corresponding acquisition time is T_c = 6×512×50 μs = 0.154 s. Figure 22 shows the correlation peak value under the SNR of –15 dB. The counts of the consuming module equal 48, and the corresponding acquisition time is T_c = 48×512×50 μs = 1.23 s. At the same condition, the average acquisition time of a parallel search with 100 pathways is $T_c = \frac{1}{100}\times\frac{1}{2}\times\left(2^{15}-1\right)\times 50\ \mu s = 268.42$ s. We can see that the acquisition speed of the PN sequence iterative acquisition algorithm is higher than the traditional method, especially for a long PN sequence. So, in the SS system with a long PN sequence, iMPA is a good choice.

CONCLUSION

In this article, we present new hardware architecture for rapid PN code acquisition using an iMPA. Our new algorithm improves the quality of the initial soft information by using multiple samples per chip. The architecture of FPGA implementation about the iMPA is proposed. It is divided into the control channel and the data channel. The control channel mainly controls the timing of the system, and the data channel takes charge to sample and process data. The two channels cooperate to accomplish the acquisition. The implementation of every module is described in detail.

A test bench is designed to test the performance of the implementation circuit based on iMPA. Through analyzing the acquisition time under SNRs of –10 and –15 dB and comparing this with the traditional method, we conclude that the acquisition speed of the PN sequence iterative acquisition algorithm is higher than that of the traditional method. So our FPGA implementation is successful and is a good choice for an SS system. ◈

REFERENCES

[1] Chugg, K. M., and Zhu, M. A new approach to rapid PN code acquisition using iterative message passing techniques. *IEEE Journal on Selected Areas in Communications*, Vol. 23 (June 2005), 51.

[2] Seunghwan, W., and Hanzo, L. Iterative spreading-sequence acquisition in the multiple receive antenna aided DS-UWB down-link. In *Proceedings of the IEEE 68th Vehicular Technology Conference*, Sept. 2008, 1–5.

[3] Seunghwan, W., and Hanzo, L. Two-stage code acquisition employing search space reduction and iterative detection in the DS-UWB down-link. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2008, 136–141.

[4] Yeung, O. W., and Chugg, K. M. An iterative algorithm and low complexity hardware architecture for fast acquisition of long PN codes in UWB systems. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, Vol. 43, 1 (Apr. 2006), 25–42.

[5] Rovini, M., Principle, F., Fanucci, L., and Luise, M. Implementation of message-passing algorithms for the acquisition of spreading codes. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, NV, Sept. 2008, 1–5.

[6] Polydoros, A., and Weber, C. L. A unified approach to serial search spread-spectrum code acquisition. *IEEE Transactions on Communications*, Vol. 32, 5 (May 1984), 542–560.

[7] Chawla, K. K., and Sarwate, D. V. Parallel acquisition of PN sequences in DS/SS systems. *IEEE Transactions on Communications*, Vol. 42, 5 (1994), 2155–2164.

[8] Simon, M. K., et al. *Spread Spectrum Communications Handbook.* New York: McGraw-Hill, 1994.

[9] Zhu, M., and Chugg, K. M. Iterative message passing techniques for rapid code acquisition. *Proceedings of the IEEE Military Communications Conference*, Vol. 1 (2003), 434–439.

[10] Yang, L., and Hanzo, L. Iterative soft sequential estimation assisted acquisition of m-sequence. *IEE Electronics Letters*, Vol. 38, 24 (2002), 1550–1551.

[11] Gallager, R. G. Acquisition of I-sequences using recursive soft sequential estimation. *IEEE Transactions on Communications*, Vol. 52, 2 (Feb. 2004), 199–204.

[12] Berrou, C., Glavieux, A., and Thitmajshima, P. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proceedings of the IEEE International Conference on Communications*, Geneva, Switzerland, May 1993, 1064–1070.

[13] Wang, W., Zong, N., Tang, J., and Lambotharan, S. Soft information improvement for PN sequence iterative acquisition. In *Proceedings of the International Conference on Computational Intelligence and Security*, 2010, 533–536.