A Low Complexity Circuit Architecture for Rapid PN Code Acquisition in UWB Systems Using Iterative Message Passing on Redundant Graphical Models

On Wa Yeung and Keith M. Chugg
Communication Sciences Institute
Department of Electrical Engineering
Viterbi School of Engineering
University of Southern California
Los Angeles, CA 90089
Email: {oyeung, chugg}@usc.edu

Abstract

Rapidly acquiring the code phase of the spreading sequence in an ultra-wideband system is a very difficult problem. In this paper, we present a new iterative algorithm and its hardware architecture in detail. Our algorithm is based on running iterative message passing algorithms on a standard graphical model augmented with multiple redundant models. Simulation results show that our new algorithm operates at lower signal to noise ratio than earlier works using iterative message passing algorithms. We also demonstrate an efficient hardware architecture for implementing the new algorithm. Specifically, the redundant models can be combined together so that memory usage can be reduced substantially. Our prototype achieves a combination of performance and complexity not possible with traditional approaches.

I. Introduction

Pseudo-random or pseudo noise (PN) sequences play an important role in an ultra-wideband (UWB) system. They are periodic sequences with long period in practical systems. In a direct sequence ultra-wideband (DS/UWB) system, the transmitted signal is a train of very narrow pulses with polarities determined by the product of a PN binary sequence and the incoming binary source data sequence. For security reasons, it is often desirable to have PN sequences of very long period, so that to an unintended receiver over a short time interval, the sequences appear to be aperiodic and completely random [1], [2].

For a DS/UWB receiver, the received signal de-spreading is achieved by multiplying the incoming samples by a local replica of the PN sequence. Therefore, the receiver must determine the unknown PN code phase embedded in the transmitted signal by analyzing the data collected from a short (compared to the PN code period) observation window so that it can synchronize the local replica. This is termed PN acquisition and will be the focus of this paper. Once the code phase is acquired, the receiver maintains the PN code synchronization through code tracking.

Traditionally, PN acquisition is achieved by searching explicitly over possible code phases. Reference signals corresponding to different code phases are correlated with the received signal and the one with the largest correlation is selected. Algorithms based on this approach include parallel search, serial search and hybrid search. For long PN sequences, parallel search is too expensive to be practical, serial search is often too slow and hybrid searches, at best, provides only a linear tradeoff between the speed and cost [3], [1].

This work was supported in part by the Army Research Office DAAD19-01-1-0477 and the National Science Foundation CCF-0428940.

For a DS/UWB system, the receiver estimates the arrival time of the pulses (i.e., the frame epoch), samples the incoming noisy signal and performs PN acquisition on these samples. The above process is repeated until acquisition is declared. Letting T_f be the pulse repetition period (frame time) and T_p be the pulse width, there are $\frac{T_f}{T_p}$ possible frame epoch values. As explained in [4], ignoring multipath delay spread exploitation [5], the receiver has to perform up to 100-1000 PN acquisitions to locate the correct frame epoch in low data rate applications. For a UWB system, the PN acquisition has to be completed quickly. If it is too slow, the correct code phase may never be acquired because the frame epoch may change due to timing drift before the receiver finishes evaluating the current frame epoch estimate [2]. In this paper, we focus on fast PN acquisition and frame acquisition is not considered.

Recently, iterative message passing algorithms (iMPAs) similar to the decoding algorithms for low density parity check codes (LDPC) and turbo codes were proposed in [6], [2] for fast PN acquisition in both direct sequence spread spectrum (DS/SS) and DS/UWB systems. Similar approaches have also been proposed in [7], [8]. Our exposition most closely follows that of [2]. These iterative algorithms offer the speed of parallel search and acquisition performance similar to that of serial search at short block lengths. Unlike parallel search, it is practical to implement the proposed algorithm in hardware to acquire PN sequences with long period. There are two drawbacks of the algorithm proposed in [2]. First, the algorithm converges slowly at low SNR. Second, the performance of the algorithm does not scale well with observation length. Specifically, doubling the observation window length lowers the operating SNR of the traditional approaches by 3 dB but only by 1-2 dB for the proposed algorithm.

In this paper, we present a new improved iterative message passing algorithm and its hardware architecture based on the algorithm proposed in [2]. Specifically, we introduce multiple redundant models to mitigate the aforementioned drawbacks discussed in [2]. The new algorithm converges faster and operates at lower SNR without increasing the hardware complexity. We will also demonstrate how to aggregate these multiples models into a single model to reduce memory usage. In our hardware prototype, the spreading sequence is of period $2^{22} - 1$. Rapidly acquiring such a long sequence is impractical by both serial and parallel search, but the logic design based on our architecture can be easily fit into a small field programmable gate array (FPGA).

The remainder of this paper is as follows. In Section II, we introduce the theory of operation. Section III gives a detailed account of our hardware implementations as well as various techniques we used in the optimization. Section IV concludes the paper and gives directions for future work.

II. Theory of Operation

A. Maximal-length Sequences

A maximal-length sequence or m-Sequence is a linear feedback shift register (LFSR) sequence which has the maximum possible period for an r-stage shift register [9], [1]. Mathematically, the sequence can be expressed as

$$x_k = g_1 x_{k-1} \oplus g_2 x_{k-2} \oplus ... \oplus g_r x_{k-r} \tag{1}$$

where $\oplus$ is the modulo-2 addition, $g_0 = g_r = 1$ and $g_k \in \{0, 1\}$ for $1 < k < r$ such that the period of x_k is $2^r - 1$. The generator polynomial is $g(D) = D^r + g_{r-1}D^{r-1} + g_{r-2}D^{r-2} + ... + D^0$ where D is the unit delay operator. Because of their excellent auto-correlation and cross-correlation properties, m-Sequences are widely used as spreading sequences in spread spectrum systems [1].

B. Signal Model

For a DS/UWB system, a standard model for acquisition characterization is: [2], [1]

$$z_k = \sqrt{E_c}(-1)^{x_k} + n_k \tag{2}$$

where z_k, $0 \leq k \leq M-1$, is the noisy sample received by the acquisition module, x_k, $0 \leq k \leq M-1$, is the spreading m-Sequence, E_c is the transmitted energy per pulse and n_k is additive white Gaussian noise (AWGN) with variance $\frac{N_0}{2}$. We also assume that x_k is generated by an r-stage LFSR and $r \ll M \ll 2^r - 1$. This is a much simplified model which assumes no data modulation and does not include the effect of jamming, oversampling, etc, but it is widely used in the literature to benchmark the performance of PN acquisition algorithms.[1]

The goal of the acquisition module is to estimate x_k and the frame epoch simultaneously based on z_k $0 \leq k \leq M-1$. In our design, we obtain the estimate of x_k, denoted by $\hat{x}_k$, by running an iterative message passing algorithm. Because $\hat{x}_k$ has to be consistent with (1), once r consecutive $\hat{x}_k$ are obtained, the rest of the sequence is determined by extrapolating the estimate by (1). As the last step, z_k is correlated with $\hat{x}_k$, $0 \leq k \leq M-1$ to check whether the correlation threshold is reached.

C. Iterative Message Passing Algorithm (iMPA) for Fast PN Acquisition

In traditional PN acquisition approaches, i.e., parallel searches, serial searches and hybrid searches, the received sequence z_k is correlated with up to $2^r - 1$ PN sequences generated by different x_0, x_1, ..., x_{r-1} combinations for the whole observation window and the algorithm chooses the phase corresponding to the highest correlation. The computation complexity is of $O(M \cdot 2^r)$ for all of them. Parallel search is a form of maximum likelihood (ML) decoding of x_k from (2) and serial/hybrid search can be regarded as approximations to ML decoding. Based on this interpretation, we formulate the PN acquisition problem as a decoding problem and apply an iterative message passing algorithm similar to turbo code decoding [10] or LDPC decoding [11], [12].

Since the inception of turbo codes, iterative message passing algorithms have been widely studied. They can be easily derived by constructing the corresponding graphical models for the system and applying a standard set of rules. It is well understood that if the graphical model has no cycles, the algorithm is equivalent to maximum likelihood decoding. Otherwise, the algorithm is heuristic and sub-optimal [13], [14], [15], [16]. However, it is often a good approximation to maximum likelihood decoding and offers near-optimal performance as in the case of turbo code and LDPC decoding.

A detailed discussion of iterative message passing algorithms is beyond the scope of this paper. In the remaining sections, we consider acquiring the m-Sequence with generator polynomial $g(D) = D^{22} + D^1 + D^0$ and only the details relevant to our example are presented. Interested readers can refer to [16], [17], [14], [18] for further details.

Our baseline decoding algorithm is based on a cyclic graphical model (Fig. 1) similar to the polynomial $g(D) = D^{15} + D^1 + D^0$ presented in [2]. The decoding algorithm shows the same performance characteristics as in [2] and suffers the same problems. The slow convergence experienced by the algorithms is attributed to the weak constraints and the flooding activation schedule. The SNR scaling problem is attributed to the existence of regular cycle structures in the graphs in [2]. Qualitatively speaking, this is a "bad" graphical model to apply standard iterative message passing algorithm. The problem is tackled in [2] by inverting the signs of

[1] As shown in [2], this model and the algorithms developed in this paper can be modified to work in sinusoidal carrier systems such as direct sequence spread spectrum systems (DS/SS). In such systems, the model of (2) should be generalized to account for an unknown carrier phase, θ_c. The approach suggested in [2] is to search over a finite set of carrier phase values.

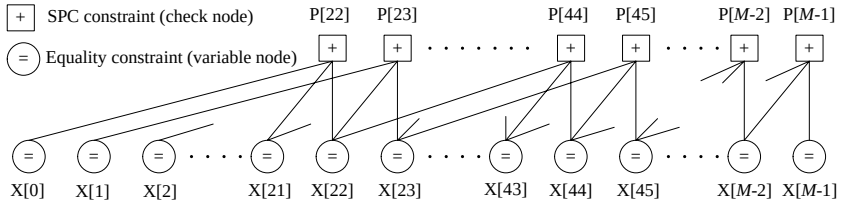


Fig. 1. Tanner graph for $g(D) = D^{22} + D^1 + D^0$.

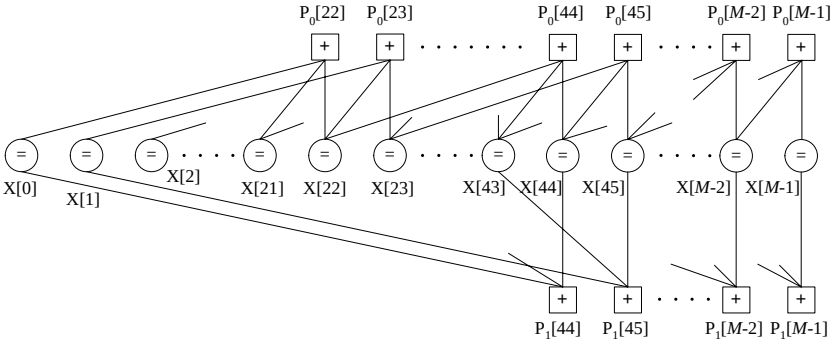


Fig. 2. Forming the 2^{nd} order graphical model using the primary model ($g(D) = D^{22} + D^1 + D^0$) and the 1^{st} order auxiliary model ($g(D) = D^{44} + D^2 + D^0$).

the set of messages corresponding to the least reliable decisions and rerunning the algorithm if acquisition fails. This approach does improve sensitivity, but it still requires many iterations. This motivates us to find a better graphical model on which we can apply the standard iterative message passing algorithm and is more amenable to hardware implementation.

D. Graphical Models with Redundancy

To improve the performance of the iMPA, we introduce a new decoding graph for $g(D) = D^{22} + D^1 + D^0$. It is constructed using multiple graphical models each of which fully captures the PN code structure. In this sense, the model has redundancy. This is equivalent to adding redundant parity checks to the standard parity check matrix. This technique has also been applied in soft decoding of some of the classical codes [19], [20]. Fig. 2 shows the special case of using two models. Each of the subgraphs is based on a different generator polynomial to the same m-Sequence. Mathematically, we introduce different reducible polynomials to generate the same sequence. It is shown in [4] that

$$g(D) = D^{22 \cdot 2^n} + D^{2^n} + D^0,\ n = 0, 1, 2, 3... \tag{3}$$

all generate the same sequence. In this paper, we refer the graphical model based on (3) as the n^{th} order auxiliary model and the one based on $g(D) = D^{22} + D^1 + D^0$ as the primary model. Also, we refer to the model that combines the primary model and the 1^{st}, 2^{nd} ... $(n-1)^{th}$ order auxiliary models as the n^{th} order model. Our decoding graph for an n^{th} order model is formed by constraining the output of primary model and each of the i^{th} order auxiliary models $1 \leq i \leq n$ to be equal. As an example, the graph of the 2^{nd} order model is shown in Fig. 2. The performance improvement by combining multiple models is shown in Fig. 3 and Fig. 4 where the curve for the algorithm in [2] is also included. Even though each individual auxiliary model produces very unreliable decoding decisions, combining them improves the convergence behaviour dramatically. We gain around 1 dB for each additional auxiliary model introduced. Only 10 iterations are required for practical convergence for a 5^{th} order model for $g(D) = D^{22} + D^1 + D^0$.

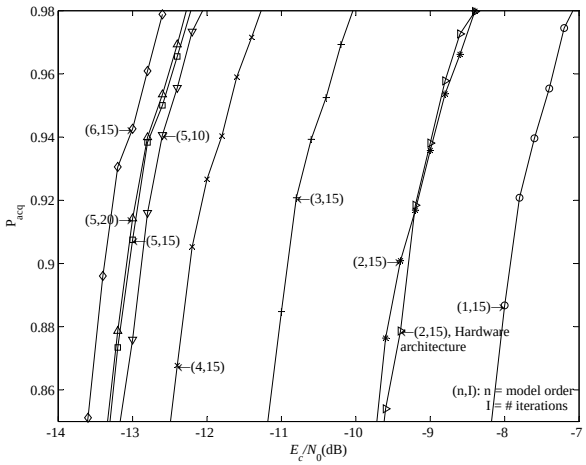


Fig. 3. Acquisition performance vs. E_c/N_0 for using an n^{th} order model on $g(D) = D^{22} + D^1 + D^0$, $M = 1024$. The acquisition performance of our hardware implementation (see Section III) is marked as "(2,15), hardware architecture".

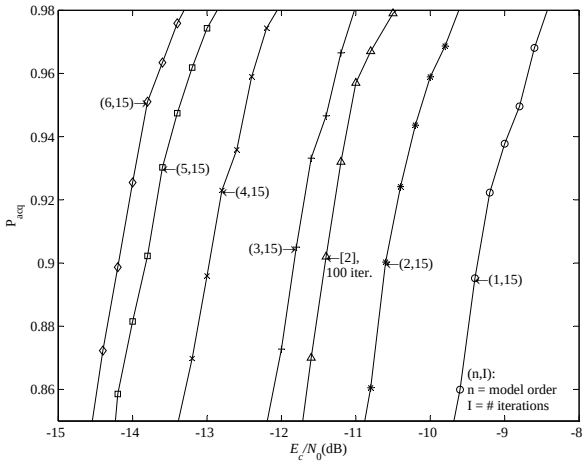


Fig. 4. Acquisition performance vs. E_c/N_0 for using an n^{th} order model on $g(D) = D^{15} + D^1 + D^0$, $M = 1024$. As a reference, the acquisition performance by running the [2] algorithm is marked as "[2], 100 iter.".

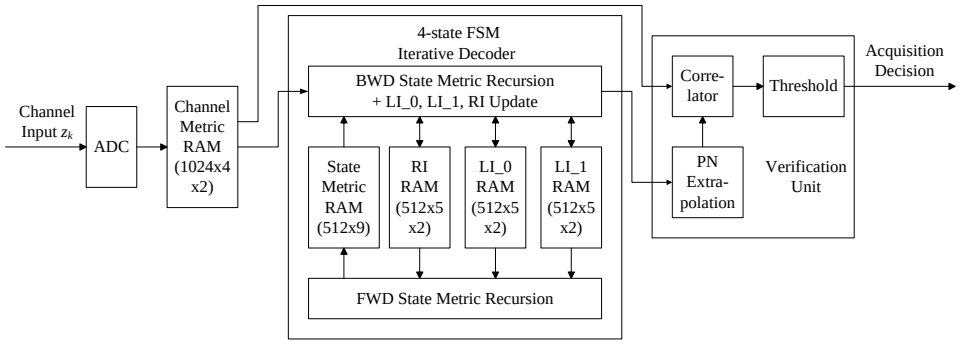


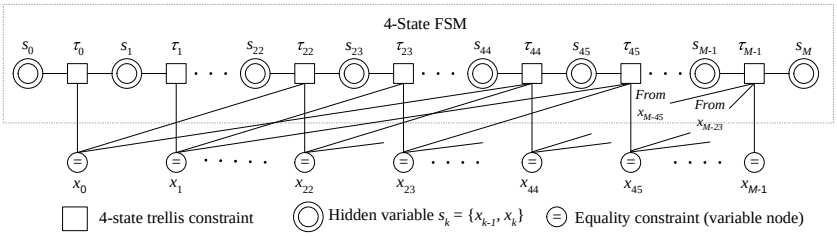
Fig. 5. Block diagram of the acquisition module for $g(D) = D^{22} + D^1 + D^0$.

Our algorithm is summarized as follows. For each block of observation $\{z_k\}$, we run the iterative message passing algorithms based on graphical models such as those in Fig. 2 and Fig. 6 for I iterations. At the end of each iteration, we compare M_{dec} with 0: $\hat{x}_k = 0$ if $\mathrm{M}_{dec} \geq 0$, $\hat{x}_k = 1$ otherwise. We then divide $\{\hat{x}_k\}$ into non-overlapping 22-pulse segments and choose the segment corresponding to the maximum $\sum_{k=22\cdot j}^{22\cdot j+21} |\mathrm{M}_{dec}[k]|$ and set $\hat{x}_k$ equal to the extrapolated value of the chosen segement by (1) for $0 \leq k \leq M-1$. Finally we correlate z_k with $\hat{x}_k$: $c = \sum_{k=0}^{k=M-1} z_k \hat{x}_k$. If c $>$ threshold, we declare acquisition. Note that this last step is required since, in the UWB model considered, an incorrect frame epoch estimate will yield a noise only observation $\{z_k\}$.

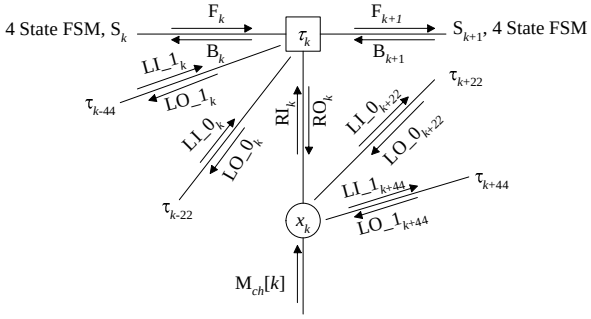
The complexity of both the decoding and correlation operations is of $O(M)$, therefore our algorithm is also of $O(M)$ complexity. There is substantial complexity reduction compared to the traditional approaches. Also, our new algorithm offers better performance with no additional complexity compared to the approach in [2] since we reduce the number of iterations dramatically.

III. Hardware Architecture

In this section, we consider the case of decoding the PN sequence $g(D) = D^{22} + D^1 + D^0$ over an observation window of $M = 1024$ using the 2^{nd} order model architecture. The block diagram of our acquisition module is shown in Fig. 5.



(a) Tanner-Wiberg graph for the 4-state FSM. This is an explicit index diagram.



(b) Detailed view of the messages passed in and out of the 4-state FSM trellis constraint node. For specific update equations, see [4]

Fig. 6. Explicit index diagrams for the 4-state FSM decoder of $g(D) = D^{22} + D^1 + D^0$.

A. 4-State FSM Decoder

As shown in Fig. 5, instead of using a 2-model PN estimator architecture based on Fig. 2, we combine the two models together using a single 4-state FSM based on Fig. 6. The new FSM captures all the information of the original FSMs and lowers the memory requirements from $6M$ messages to approximately $3M$ messages plus state metrics as demonstrated below. Moreover, by using a single FSM, we save routing resources by lowering the bandwidth requirement for the channel metrics ($\mathrm{M}_{ch}[k] = z_k$) memory since it is now accessed only by one FSM soft-in soft-out (SISO) module. Using the 4-state FSM does require more logic in the FSM SISO implementation, but this increase is justified by the the additional savings in memory and routing.[2]

Our 4-state FSM decoder is based on the forward backward algorithm. We define the state as $S_k = \{x_{k-1}, x_k\}$ and the corresponding explicit index diagram (i.e., the Tanner-Wiberg graph) is shown in Fig. 6(a). The messages passed are shown in detail in Fig. 6(b). The update equations are obtained by applying the standard message passing rules [16] on Fig. 6. For complete listing of the state transition table, the implicit index diagram, the detailed decoder structure and the update equations, interested readers can refer to [4].

Simulation results shows that the 4-state FSM decoder implementation improves the performance by 0.2 dB in $\frac{E_c}{N_0}$ as compared to the architecture based on Fig. 2.

We can continue to combine multiple auxiliary models to form a single FSM. For example, we can implement a 3^{rd} order model using a 16-state FSM. However, the exponential growth in state metric memory may outweigh any savings in the message memory for larger n.

B. Forward Backward Algorithm Architecture for Multiple Index Segments

To reduce the internal FSM state metric memory, we divide the observation window into multiple segments and run the forward backward algorithm (FBA) segment by segment. This is a standard approach for implementing turbo decoders [21], [22].

[2]The decoder can also be based on a three FSM model which is discussed in further details in [4].

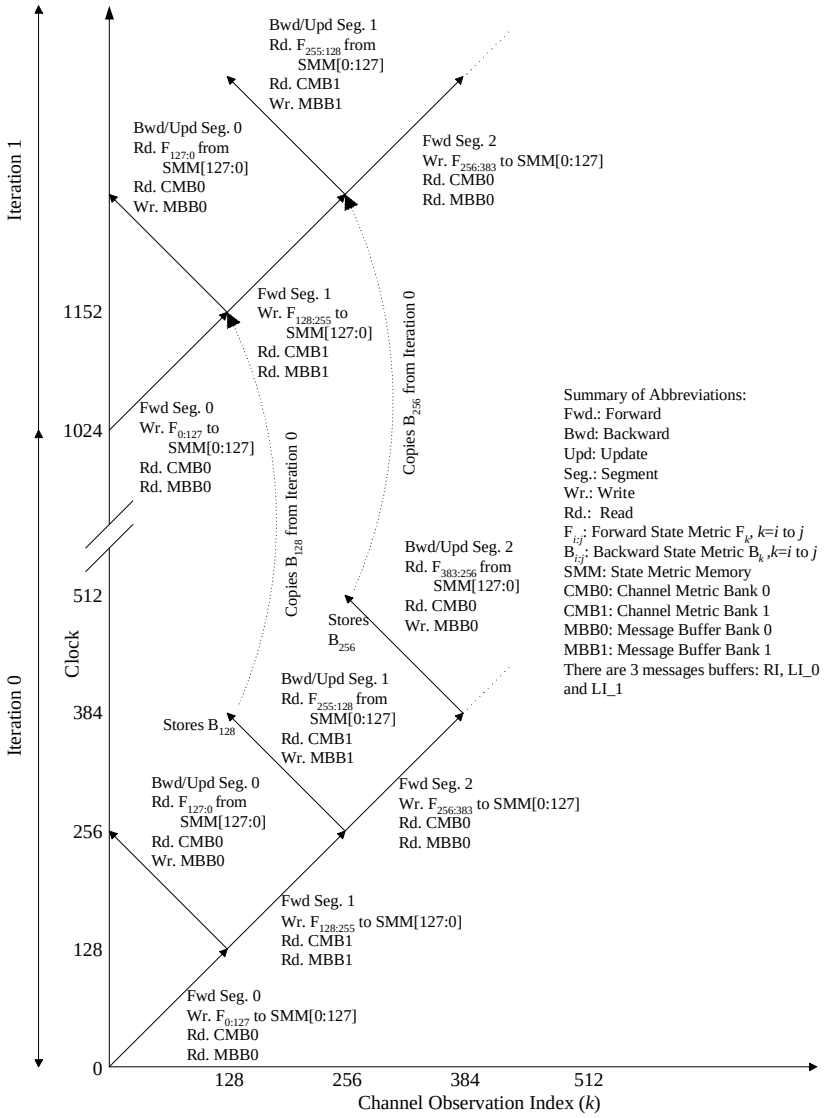


Fig. 7. Processing and memory access pipeline for the 4-state decoder.

In our prototype system, we divide the observation window (1024) into 8 segments. There is one forward unit and one backward unit running 15 iterations. During each iteration, the forward unit updates the state metric sequentially from pulse 0 to 1023. The backward unit computes the state metric in the following order: 127→0, 255→128, ..., 1023→896. Such a sequence of calculations results in one problem: we do not know the backward metric $\mathrm{B}_{128}[i],\ 0 \leq i \leq 3$ when computing $127 \rightarrow 0$, $\mathrm{B}_{256}[i],\ 0 \leq i \leq 3$ when computing $255 \rightarrow 128$, etc. Instead of running the backward unit for an additional "warm-up" period as in [21], [22] which requires an additional backward unit for a full-speed design, we copy the $\mathrm{B}_{128}[i]$ values from the previous iteration. This is feasible because the warm-up period is only required if we are trying to approximate an FBA-SISO in isolation. For an iterative system, starting the backward recursions based on earlier iteration values is equivalent to a change in the activation schedule for the iMPA on the cyclic graph, and as such does not significantly affect the performance. This is a known architecture for implementing iterative decoders with forward-backward based SISO decoders (e.g., see [23], [24]). Once both the forward and backward state metrics become available, $\mathrm{LI\_0}_k$, $\mathrm{LI\_1}_k$, RI_k and $\mathrm{M}_{dec}[k]$ are computed and the FSM state metric memory is released immediately. The processing pipeline is shown in

Fig. 7 which shows the update sequence as well as the corresponding memory access.

C. Bit Width

The bit widths in our system are determined by simulations in two steps. First, we fixed $\mathrm{LI\_0}_k$, $\mathrm{LI\_1}_k$, RI_k to be of 16 bits and determine that 4 bits of ADC output is sufficient. Compared to floating point, there is a performance loss of only 0.2 dB. For each ADC bit width, we have optimized the scale q that sets the ADC dynamic range (ADC_{out} = quantize($q \cdot z_k$)) for performance. For a 4-bit ADC, q_{opt} is found to be 1.65 by simulation. We then determine the bit width of the messages $\mathrm{LI\_0}$, $\mathrm{LI\_1}$ and RI. Simulation shows that they can all be clipped to 5 bits after each (FBA/=) SISO activation with little performance degradation for a 4-bit ADC. For the performance curve of various bit width combinations, the readers can refer to [4, Fig. 12].

For the state metrics, it is shown in [4] that 8 bit is sufficient for 5-bit messages if we normalize $\mathrm{F}_k[i]$ and $\mathrm{B}_k[i]$ $1 \leq i \leq 3$ against $\mathrm{F}_k[0]$ and $\mathrm{B}_k[0]$. Though the normalization approach reduces the memory usage by one-fourth, it impacts the frequency scaling of our circuit by requiring three additional subtractions in the critical path of the forward and backward recursions. Therefore, we do not perform normalization and use 9 bits to represent the state metric instead of 8 bits. This additional 1-bit approach is commonly used in Viterbi decoders and is proven to be correct with two's complement arithmetic [25], [16].

D. Internal Memory Organization

The message memories (LI_0, LI_1 and RI) are divided into two banks, one for the odd FBA segments and the other for the even segments. The channel metric memory is divided into 2 banks each comprising 1024 entries. By carefully scheduling the access sequence as shown in Fig. 7, we can use 2-port memories without contention problems. Readers can refer to [4] for further implementation details. The design can be easily ported to single port memory only architecture by doubling the bus width and time division multiplexing the access.

E. Verification Unit

Our verification unit, shown also in Fig. 5, consists of two parts, a PN sequence extrapolation unit and a correlator unit. The extrapolation unit extends the 22-bit PN estimate it receives to the whole observation window. The correlation unit then correlates this sequence with the channel metric. To improve efficiency, the correlator output is checked every $\frac{M}{4}$ pulses and it must exceed the check point threshold before continuing. If the final correlation value exceeds the final threshold, acquisition is declared. The final threshold is chosen to be $0.65 \cdot q \cdot 1024$, which was found by simulation. This yields good acquisition performance as shown in Fig. 3 and the frequency of false alarm is 0 in 5000 trials when the signal is absent.

F. Hardware Implementation

We implemented the architecture using Verilog HDL. The code is synthesized by Synplicity, then mapped by Xilinx Foundation to a Xilinx Virtex 2 device (XC2v250-6). The number of bits implemented in block RAM is 28160, the number of 4-input LUTs used is 2433 and the number of slices used is 1481. The design can run at 91 MHz. These figures show that memory is the main component of the circuit and justify our decision to trade off logic for memory reduction.

Our baseline design can decode $\frac{Freq_{clk}}{15}$ pulses per second. Assuming a 60 MHz clock, our prototype generates a PN code phase decision every $\frac{15}{60\,MHz} \cdot 1024 = 2.56\ \mu s$. The decode process has to be repeated for each frame epoch estimate until the correct frame epoch is found. Assuming the frame time $T_f = 250$ ns (i.e., pulse rate = 4 Mpulses/s) and pulse

width $T_p = 1.6$ ns, the approximate average acquisition time of our prototype system is $T_{acq} = 2.56\,\mu s \cdot \frac{T_f}{T_p} \cdot 0.5 = 20$ ms with $P_{acq} = 0.95$ at $\frac{E_c}{N_0} = -8.9$ dB. This assumes that half of the frame epoch values are searched on average.

As a comparison, to achieve the same average T_{acq}, hardware based on parallel search and running at the same frequency requires approximately 5.6×10^5 correlators, 5.6×10^5 14-bit comparators and 5.6×10^5 4-bit registers. For serial search, the hardware is trivial if we assume a one addition per clock architecture. However, it takes an average of 5.6×10^3 s (approximately 1.5 hours) to acquire the PN sequence if running at the same frequency.

To further lower T_{acq}, we can instantiate multiple forward and backward units to process multiple data segments in parallel. We expect that the increase in logic will be approximately linear when the speed up factor does not exceed 8 because we already divide the observation window into 8 segments in our iterative decoder and each of them can be run in parallel. For lower speed applications, our design can be further simplified to using single port memory and running the update sequentially. Such a design can save in the number of adders and reduce the routing resources. Therefore we expect the logic gate count will scale linearly for target pulse rate varies from 500 kpulses/s to 32 Mpulses/s.

Our design can also be directly extended to operate at even lower SNR. This requires adding auxiliary model decoders as well as memories for saving the messages from the additional decoders. Since a 6^{th} order model is approximately three times more complex than a 2^{nd} order model, we estimate that the operating E_c/N_0 can be lowered to -13 dB by tripling the gate count or alternatively, increasing the acquisition time by 3 times and tripling the message memory.

IV. Conclusion

In this paper, we present a new hardware architecture for fast PN acquisition in UWB systems based on iterative message passing on a graphical model with redundancies. Our new algorithm improves sensitivity significantly via the introduction of multiple redundant models. Hardware based on the algorithm is economical to implement and can rapidly acquire very long PN sequences. There is no known way to accomplish this with traditional approaches using similar hardware resources.

We examined in detail the design trade-offs in choosing an appropriate architecture for the main component: a forward backward algorithm based decoder. We then demonstrated how to combine multiple redundant models into a single model to reduce memory usage. Finally, we gave a detailed account on our hardware implementation and discuss various implementation techniques. Our design can be fit to a small FPGA while full parallel search is impractical to implement and serial search is five orders of magnitude slower than our design.

Future work will be focused on designing hardware for a more realistic system model which incorporates oversampling, interference and multi-path channel distortions.

References

[1] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications Handbook*. McGraw-Hill, 1994.

[2] K. M. Chugg and M. Zhu, "A new approach to rapid PN code acquisition using interative message passing techniques," *IEEE J. Select. Areas Commun.*, vol. 23, no. 51, June 2005.

[3] A. Polydoros and C. L. Weber, "A unified approach to serial search spread-spectrum code acquisition," *IEEE Trans. Commununication*, vol. vol. 32, pp. 542–560, May 1984.

[4] O. Yeung and K. M. Chugg, "An iterative algorithm and low complexity hardware architecture for fast acquisition of long PN codes in UWB systems," *J. VLSI and Signal Processing (Springer), Special Issue on UWB*, 2005, to appear.

[5] E. A. Homier and R. A. Scholtz, "Rapid acquisition of ultra-wideband signals in the dense multipath channel," in *Digest of Papers, IEEE Conference on Ultra Wideband Systems and Technologies*, May 2002, pp. 105 – 109.

[6] M. Zhu and K. M. Chugg, "Iterative message passing techniques for rapid PN code acquisition," in *Proc. IEEE Military Comm. Conf.*, Boston, MA, October 2003, pp. 434–439.

[7] L. Yang and L. Hanzo, "Differential acquisition of m-Sequences using recursive soft sequential estimation," *IEEE Trans. Wireless Commununication*, vol. 4, no. 1, pp. 128 – 136, Jan 2005.

[8] B. Vigoda, J. Dauwels, N. Gershenfeld, and H. Loeliger, "Low-complexity LFSR synchronization by forward-only message passing," *IEEE Trans. Information Theory*, June 2003, submitted.

[9] S. W. Golomb, *Shift Register Sequences, revised edition*. Aegean Park Press, 1982.

[10] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commununication*, vol. 44, no. 10, pp. 1261–1271, October 1996.

[11] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[12] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEE Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, August 1996.

[13] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University (Sweden), 1996.

[14] S. M. Aji, "Graphical models and iterative decoding," Ph.D. dissertation, California Institute of Technology, 1999.

[15] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[16] K. M. Chugg, A. Anastasopoulos, and X. Chen, *Iterative Detection: Adaptivity, Complexity Reduction, and Applications*. Kluwer Academic Publishers, 2001.

[17] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Information Theory*, vol. IT-27, pp. 533–547, September 1981.

[18] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Trans. Telecommun.*, vol. 9, no. 2, pp. 155–172, March/April 1998.

[19] J. S. Yedidia, J. Chen, and M. Fossorier, "Generating code representations suitable for belief propagation decoding," in *Proc. 40-th Allerton Conference Commun., Control, and Computing*, Monticello, IL., October 2002.

[20] M. Schwartz and A. Vardy, "On the stopping distance and the stopping redundancy of codes," *IEEE Trans. Information Theory*, 2005, submitted.

[21] A. J. Viterbi, "Justification and implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, February 1998.

[22] G. Masera, G. Piccinini, M. R. Roch, and M. Zamboni, "VLSI architectures for turbo codes," *IEEE Trans. VLSI*, vol. 7, no. 3, September 1999.

[23] J. Dielissen and J. Huisken, "State vector reduction for initialization of sliding windows MAP," in *2nd Internation Symposium on Turbo Codes & Related Topics*, Brest, France, 2000, pp. 387 – 390.

[24] A. Abbasfar and K. Yao, "An efficient and practical architecture for high speed turbo decoders," in *IEEE 58th Vehicular Technology Conference*, October 2003, pp. 337 – 341 Vol.1.

[25] A. P. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commununication*, vol. 37, no. 11, November 1989.