HARDWARE-EFFICIENT IMPLEMENTATION OF THE SOVA FOR SOQPSK-TG

Ehsan Hosseini, Gino Rea
Department of Electrical Engineering & Computer Science
University of Kansas
Lawrence, KS 66045
ehsan@ku.edu

Faculty Advisor:
Erik Perrins

ABSTRACT

In this paper, we present a hardware-efficient architecture of a demodulator for shaped offset quadrature phase shift keying, telemetry group version (SOQPSK-TG). The demodulation is done using the soft-output Viterbi algorithm (SOVA), which is implemented by the two-step traceback method. In this method, two traceback operations are employed to find the maximum-likelihood (ML) path and the competing path. Using the proposed architecture, the tracebacks are done at the same time as the demodulator is generating output bits and their reliabilities. This method has been shown to require less storage than the well-known register-exchange method. Finally, we present the performance results from the FPGA implementation.

INTRODUCTION

The modern telecommunication systems benefit from high performance soft-input soft-output Forward Error Correction (FEC) coding schemes such as iterative codes and serially concatenated codes. These modern FEC algorithms can provide a near channel capacity error correction performance without the need of data retransmission. Consequently, the demodulator in those systems ought to be modified such that it provides the soft values required by the decoder. It may also use the estimations from the decoding algorithm in an iterative manner.

This work mainly focuses on the hardware implementation of a shaped offset quadrature phase shift keying, telemetry group version (SOQPSK-TG) demodulator. The demodulator is desired to be implemented in an FEC system. SOQPSK-TG signal is a highly bandwidth efficient and constant envelope that can be viewed as a continuous phase modulation (CPM) signal. From the CPM point of view, SOQPSK-TG is a partial response version of SOQPSK, which was adopted in the IRIG 106-04 standard for aeronautical telemetry [1]. The CPM description of SOQPSK also enables us to employ maximum likelihood sequence detectors implemented using the Soft Output Viterbi Algorithm (SOVA) [2], to estimate the sequence of transmitted bits and their reliabilities. Therefore, the core operation of proposed SOQPSK demodulator will be SOVA.

Several works have been presented on the hardware implementation of SOVA and its basic operation, the Viterbi algorithm. The original version of SOVA [3] suffers from high complexity due to the need to

update of log likelihood ratio (LLR) values for every state in the trellis as the maximum likelihood path is unknown. Two different architectures are employed known as register-exchange and traceback method for SOVA detection. The first architecture suffers from substantial amount of required hardware and the latter one imposes a large delay. A simplification to the SOVA results in a low complexity decoder presented in [4], named as Two-Traceback method. The main drawback of the Two-Traceback method is the traceback operation which needs to be performed for every new received symbol. Obviously, the latency of traceback reduces the overall throughput of decoder.

In this work, we extend the systolic Viterbi decoder presented in [5] to the Two-Traceback SOVA implementation, which results in a faster decoding time. The systolic Viterbi algorithm allows the traceback operation, updating and storing of the information sequence to be done simultaneously during a single clock cycle. Furthermore, two systolic arrays are required to perform the traceback on both maximum likelihood path and the competing path, which will be described in details. The target platform for our design is a Field Programmable Gate Array (FPGA) which provides needed speed and flexibility. Finally, the implementation results are presented and compared to register-exchange method, which confirms significant saving in terms of required hardware.

The rest of this paper is organized as follows. First, the SOVA and its application to SOQPSK are presented. The hardware architecture for the Two-Step SOVA is detailed in the next section. Then, the implementation results for the proposed architecture are summarized. Finally, our conclusions are drawn.

SOVA FOR SOQPSK DETECTION

In this section, we briefly present the soft-output viterbi algorithm for binary codes and then its application to SOQPSK is touched on. We suppose the reader is familiar with basics of Viterbi algorithm and CPM signals.

A. Soft Output Viterbi Algorithm

The SOVA is a modified version of Viterbi algorithm that produces reliability values for every decoded information bit. Consequently, the SOVA can be implemented by complementing the VA such that it generates the reliability values corresponding to the maximum likelihood path based on the metric values at each state. As in VA, the computations in SOVA are performed on a trellis with a number of states depending on the memory length of the underlying convolution. However, the SOVA assigns a soft value to every state which is updated by the probability of making a wrong decision at that state and if the bits on the merging paths are different. The details of the original SOVA can be found in [2]. Here, we briefly describe the update equations for the soft information (reliabilities).

Assuming the Viterbi algorithm at a state s_k at time step k, the difference between the incoming branches to that state is denoted as Δ. Additionally, two paths are merged at state s_k, one corresponds to the winner path or path-1, and the other corresponds to the looser path or path-2. A sequence of bits $\hat{\mathbf{u}}(s_k)$ and reliabilities $\hat{\mathbf{L}}(s_k)$ is also assigned to each of the two paths. The reliabilities are updated by comparing the input bits on the looser path with the ones on the winner path. If the bits at time step j are different, i.e. $\hat{u}_j^1 \neq \hat{u}_j^2$ the reliabilities are updates by

$$\hat{L}_j(s_k) = \min\{\Delta, \hat{L}_j^1\} \tag{1}$$

Next, consider the case that the bits on both paths are the same for some time step j. The corresponding

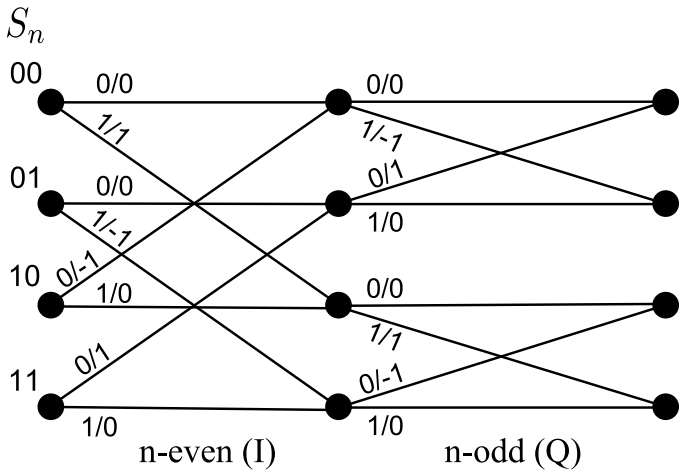

Figure 1: Four state time-varying trellis. The labels on the branches indicate the input bit/output symbol based on the precoder of 3.

reliability is updated using,

$$\hat{L}_j(s_k) = \min\{\Delta + \hat{L}_j^2, \hat{L}_j^1\} \tag{2}$$

In order to reduce the complexity of Viterbi algorithm, the computations are done on a finite length moving window called as traceback. Therefore, the reliability values and the estimated bits are generated having a latency equal to the length of traceback.

B. Iterative Detection of SOQPSK

The SOQPSK signal can be modeled as a precoder connected to a standard CPM modulator [6]. The precoder converts the binary data to ternary in order to impose OQPSK-like characteristics on the CPM signal. The precoder is recursive and can be described by a four-state time varying trellis as shown in Figure 1. The precoder's function can be formulated as,

$$\alpha_n \triangleq (-1)^{n+1}(2c_{i-1} - 1)(c_i - c_{i-2}), \quad c_i \in \{0, 1\}, \quad \alpha_i \in \{-1, 0, 1\}. \tag{3}$$

Therefore, the precoder's output is a function of current symbol c_i and three state variables: n-even/n-odd, c_{n-1} and c_{n-2}. Here, the n-even/n-odd has been removed from state variable and considered as the time index. This leaves us as a four-state time varying trellis depicted in Figure 1. Therefore, the state variable $c_{n-1}c_{n-2}$ can be uniquely represented as the CPM phase state.

Describing the SOQPSK modulation on the state diagram leads us to the adoption of VA for its detection. Since the trellis is time varying, the VA works on different trellises for even and odd symbol times. In order to implement the VA, we need to derive the branch metric increment for the SOQPSK signal. The branch metric increment for partial response CPM is

$$Z(n, [\tilde{a}_n, \tilde{S}_n]) \triangleq \text{Re}\left[e^{-j\tilde{\theta}_{n-1}} \int_{nT}^{(n+1)T} r(t) e^{-j2\pi h \alpha_n q(t-nT)} dt\right] \tag{4}$$

where $q(t)$ is the partial-response phase pulse such as the one defined by SOQPSK-TG. The branch metric increment is a function of the hypothesized branch vector $[\tilde{a}_n, \tilde{S}_n]$, which corresponds to a unique hypothesized ternary symbol $\tilde{\alpha}_n$ and the CPM phase state $\tilde{\theta}_{n-1}$.

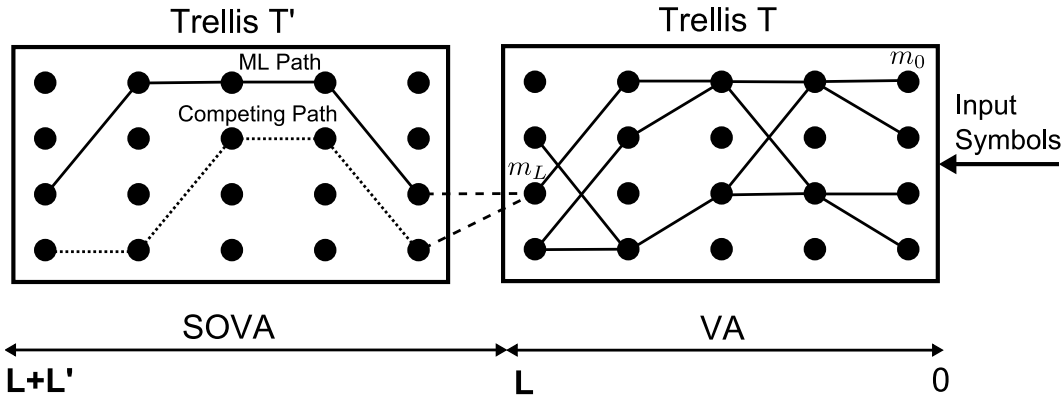

Figure 2: Two-Step SOVA

The computation of branch metric increment in (4) is performed using matched filters (MF). Since this work mainly deals with the hardware implementation of the SOVA itself, the details on the MFs are not discussed. Interested readers may refer to [7] for a complete discussion.

TWO-STEP SOVA DECODER

The complexity of the SOVA mainly comes from the storage and update of the bits and reliabilities over all the survivor paths, which is equal to the number of states N_s. Originally, the SOVA was described in [3] neglecting equation (2). The performance of SOVA seems not to suffer from ignoring the second case and further simplifies the storage requirements as the decoder only needs to store the reliabilities over the maximum likelihood path. However, the decoder mainly benefits from this fact only after it knows the maximum likelihood path.

The motivation for the Two-Step SOVA, which was presented in [4], is to perform the update operation after the final survivor path is completely (or almost completely) established. In this scheme, two Viterbi decoding operations are done on two trellises. Let us recall T as the first trellis on which decoding operation is done, with a traceback length of L and N_s states. The first Viterbi operation is carried out on T and after L time steps, it estimates the maximum likelihood path. This path has an starting state (node) as m_0 and terminates at state (node) m_L on trellis. If L is sufficiently large, m_L does not depend on the future symbols. Hence, the first Viterbi algorithm determines the states over which the maximum likelihood path is formed.

Once the maximum likelihood path is known, it is trivial to perform the SOVA on a second trellis to update the reliabilities according to equation (1). The second trellis T' is basically the same as T, however, the received symbols are delayed such that the maximum likelihood path is determined by the first trellis. In this fashion, the binary decisions and difference between branch metrics (Δ) at time step $j = 0$ are delayed L cycles. The delayed values are used in the second trellis to update the bits and reliabilities on the most likely path. Since the first trellis' task is only finding the global survivor states, the original Viterbi algorithm is employed. The Two-Step SOVA method is depicted in Figure 2.

The authors in [4] has presented an architecture for the Two-Step Traceback method. The decoder consists of two parts corresponding to Two Trellises. In the first part, the VA algorithm is performed on the received symbols consisting of updating the cumulative metric, comparing and selecting the most likely path. Then it performs a traceback on the winner path to find its starting state at L time steps

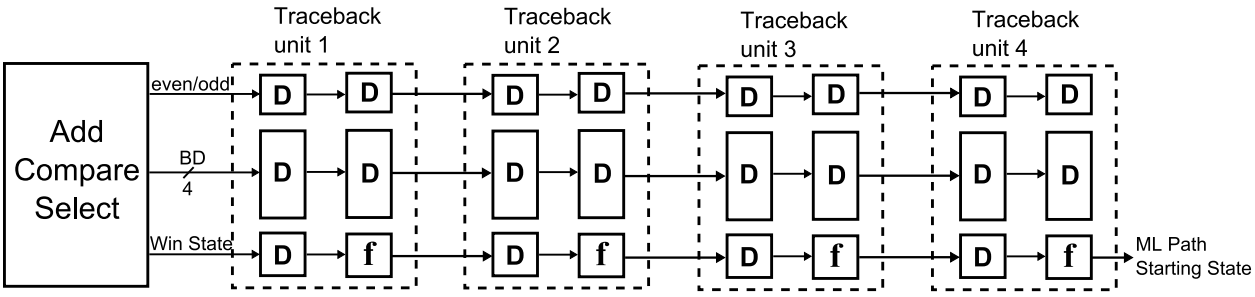

Figure 3: The systolic traceback architecture for finding the input bits on the survivor path.

before the current time. In the second part, SOVA is performed on a delayed version of the input symbols. Additionally, two traceback logics are employed, one for the ML path and the other one for the competing path. Both of these tracebacks start from state m_L, already determined by the first part. The second SOVA computes the difference between metrics for state m_L based on the delayed symbols and sends them to the input of a shift register. The traceback logics continuously access the shift register and update its contents based on the binary decisions at different levels for every time step. The last value of the shift register is considered as the final reliability of a symbol and is appeared on the output port of the decoder. Although the proposed decoder reduces the complexity of classical SOVA decoders, it suffers from a relatively low throughput. The main bottleneck is the implementation of the traceback logics. In that architecture, a traceback of length L has to be done for outputting one symbol, which requires L clock cycles.

In order to keep high-throughput, the traceback needs to be completed within one clock cycle. This solution requires access to all path metrics and decisions over the traceback length. This method was implemented in [8] using orthogonal memory access. Another solution is using a systolic architecture in which the binary decisions and metrics are propagated through shift registers eliminating complex memory accesses. The latter method was presented in [5] for the original Viterbi algorithm. In this work, we extend this method to SOVA and the time-varying Trellis.

The first part of the Viterbi algorithm is updating cumulative metric for each state, finding the winner branches and the global survivor. This operation is done in a module named as the Add, Compare and Select (ACS) unit. The outputs of ACS, winner branch for each state and the winner state, are sent to the first traceback module to find the starting state of the ML path.

In the systolic architecture, the traceback is done continuously on the results of the ACS unit in a pipelined fashion, instead of doing the entire traceback on a single value. Therefore, the decoder is able to operate at the same rate as the incoming symbols. The systolic traceback architecture is shown in Figure 3. The traceback module consists of four units corresponding to a traceback length of four. The task of each unit is to traceback the trellis one time step back based on the branch decisions (BD) and the even/odd indicator. The core of each unit is a traceback function **f** that determines the starting state on the trellis according to ending state and BDs. For example, if the ending state is (01), the BDs are $\{0,1,1,0\}$ and the trellis is in the even time, the result of this function is state (11), which can be verified from the trellis diagram in Figure 1. The function **f** is implemented using a look-up table having 7 bits as the input: 4 bits for BD, 2 bits for state and 1 bit for even/odd indicator.In the proposed architecture, the traceback algorithm is done a pipelined manner in which the starting state of a path is found $2L$ clock cycles after its ending state is found. Therefore, the traceback module can be seen as traceback function with a latency of $2L$ clock cycles, which is able to process at the same time as the Viterbi algorithm computes the survivor

state and branches. However, this latency is twice as the expected latency of L (equal to traceback length) because two levels of registers are used in each unit for BD and even/odd signals. The extra register can be explained as follows. Suppose a traceback unit performs a traceback from state m_j to m_{j-1} in one clock cycle based on the branch decisions at time j, BD_j. At the same time BD_j is shifted by one. The next traceback unit performs the operation on m_{j-1} and the received BD. However, the received BD corresponds to time j. Therefore, an extra delay (register) is introduced for BD in each unit to align the states and branch decisions in time.

The traceback module of Figure 3 delivers the starting state of the ML path and also the winner branches, aligned in time with the starting state, at its output. These values can be used to find the input bits on the ML. The only task remained is to find the input bits on the competing path and comparing them with the ML path bits in order to update the reliabilities. The computation of reliabilities requires the difference between branch metrics for the winner state. This subtraction is carried out inside the ACS module for all states. The results, denoted as Δ_i, are delayed $2L$ clock cycles to be synchronized with the BD and the starting state of ML path. The starting state is then used to choose one of Δ_i values which will be later used in the update of reliabilities. These rather straightforward tasks are not shown in Figure 3 for the sake of simplicity.

In order to find the input bits on the competing path, it is impossible to use the same method as used for the ML path, i.e. finding the starting state of the path and deriving the input bits. This is due to the fact that the competing path is not continuous as the ML path, i.e. it might be totally different from one clock cycle to the next clock cycle. Therefore, the input bits over the competing path must be computed within the traceback operation. The traceback architecture for the competing path is illustrated in Figure 4. The architecture for the competing path traceback is essentially the same as ML path with some modifications as described in the followings. The competing path diverges from the ML path on the winner state. Therefore, the first traceback is the traceback of the winner state via the looser branch. Hence, the BD values ought to be inverted for the first traceback unit as shown in the figure. The rest of the traceback is performed through the winner branches at each time step. Another modification is the addition of the input bit to the traceback function **f**, i.e. the module **f** generates the input bit which had caused the transition from state m_{j-1} to m_j. The generated input bits are basically the bit over the competing path which are shifted using the structure of Figure 4. The competing path traceback module will also have a latency of $2L$ clock cycles.

Based on the above discussion, the two traceback modules generate the input bits over the ML path $\hat{u}_j^1, \hat{u}_{j-1}^1, \ldots, \hat{u}_{j-L}^1$ and the competing path $\hat{u}_j^2, \hat{u}_{j-1}^2, \ldots, \hat{u}_{j-L}^2$. These bits update the contents of reliability registers, $\hat{L}_j, \hat{L}_{j-1}, \ldots, \hat{L}_{j-L}$, which requires L numerical and binary comparisons. The circuit of such an update logic is depicted in Figure 5. Finally, the value of the last register $\hat{L}_{j-L}$ is signed according to $\hat{u}_{j-L}^1$ and is sent to the output of SOVA as the soft decoded value.

FPGA IMPLEMENTATION

We use the two-step SOVA architecture for the FPGA implementation of the SOQPSK-TG demodulator. The demodulator implementation includes three matched filters (MF) corresponding to each of three symbols in SOQPSK (-1, 0 and 1) and a SOVA module. The SOVA module consists of an Add/Compare/Select (ACS), the ML path traceback unit (MLTB) and the competing path traceback unit (CTB) as shown in Figure 6.

In our hardware implementation, it is assumed that the received signal is sampled sixteen times the

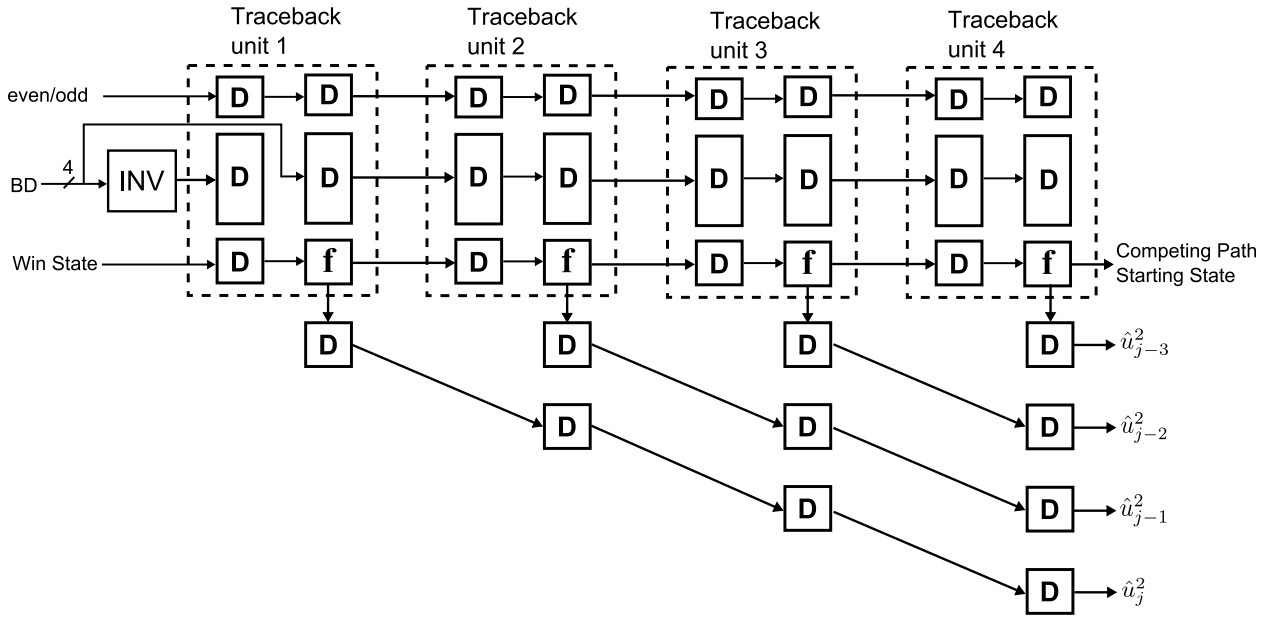

Figure 4: The systolic traceback architecture for finding the input bits on the competing path.

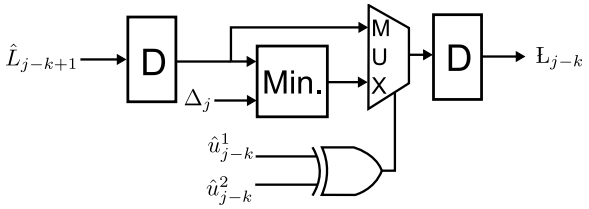

Figure 5: The update circuit of reliabilities based on equation (1).

symbol rate. The sampled values are correlated with the precoded MF coefficients and down sampled by sixteen. Therefore, the MFs output three complex signals corresponding to ternary symbols or a total of six values. Inside the ACS, the cumulative metric (CM) is updated for each of the four states based on the values from MFs and the state of the trellis. The CM values are compared on the branches of the trellis and the winner branch and the metric are determined for each state. The state with the maximum metric is chosen as the global survivor and is sent out for the traceback operation. Additionally, the difference between metric for each state Δ_i is delivered to the output for the update of reliabilities. The MLTB unit performs the traceback using the systolic architecture to find the starting state of the ML path. It also determines the input bits over the ML path. In our implementation, a traceback length of $L = 16$ is considered. Therefor, the MLTB generates the input bits and the survivor state with a latency of 32 clock cycles. However, one value is generated at every clock cycle due to the pipelined architecture. The CTB performs two major tasks: Firstly, it tracebacks the competing path upon which the input bits are determined. Secondly, it updates the reliabilities based on the circuit of Figure 5. A traceback length of 16 is also considered for CTB, resulting in an additional 32 clock cycles of latency to the SOVA.

Eight bits of precision is used for representing the received signal, MF outputs, prior reliabilities and the output reliabilities. The eight-bit representation consists of one sign bit, three integer bits and four fractional bits. However, some of the internal signals are represented using a higher number of bits due to

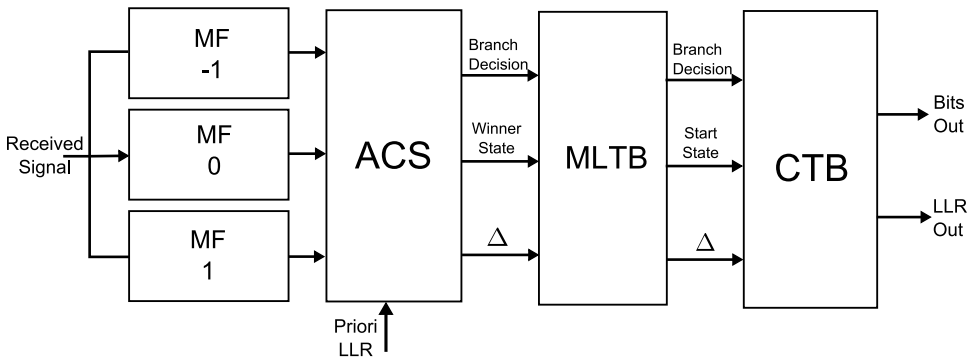

Figure 6: The block diagram of SOQPSK-TG demodulator.

Method	Clock Freq.	Slices	Registers	Utilization Ratio
Two-Step	143 MHz	478	935	2.7%
Register exchange	152 MHz	1391	1768	8%

Table 1: The implementation results for SOQPSK-TG demodulator.

overflow concerns. For example, the cumulative metric, inside ACS, utilizes 17 bits.

The proposed SOQPSK-TG demodulator is written in VHDL and verified using Modelsim. In order to observe the quantization effects on the demodulation, a bit-precise MATLAB model has also been generated and were simulated with the same input data as VHDL. Figure 7 shows the performance of the VHDL implementation compared to MATLAB simulations. The Bit error rate (BER) plots show that the performance of the VHDL implementation is almost identical to the MATLAB quantized model. Moreover, a floating point MATLAB BER plot is provided and compared to VHDL results. At BER 10^{-4}, the performance of VHDL simulation is about 0.25 dB worse than the floating point MATLAB simulations due to the fixed-point effects.

The VHDL design is implemented on Xilinx Virtex 5 110xt FPGA using the Xilinx ISE tool. According to the implementation results, the design is capable of running at a maximum clock frequency of 143 Mhz. The design's critical path is located within ACS unit since the operations of add, compare and select has to be completed within one clock cycle for every incoming symbol. The main advantage of the proposed architecture is its relatively low amount of resources consumed on the FPGA chip. The implemented SOQPSK-TG demodulator occupies only 478 slices of the FPGA, which is equal to 2.7% of the available resources. The large saving on the resources can be revealed when the design is compared to other methods. Hence, another SOQPSK demodulator was implemented using the register exchange method for SOVA. The latter demodulator has a utilization ratio of 8%, which is nearly three times the proposed two-step demodulator. The detailed information on hardware aspects is provided in Table 1 for both architectures.

CONCLUSIONS

In this paper, we described a low area FPGA implementation of an SOQPS-TG demodulator. The proposed design is based on a systolic architecture for the Two-Step SOVA, which is able to deliver one decoded value every clock cycle after an initial delay. Therefore, the bottleneck of the Two-Step SOVA, which was the traceback, is eliminated. The functionality of the demodulator was verified by generating a

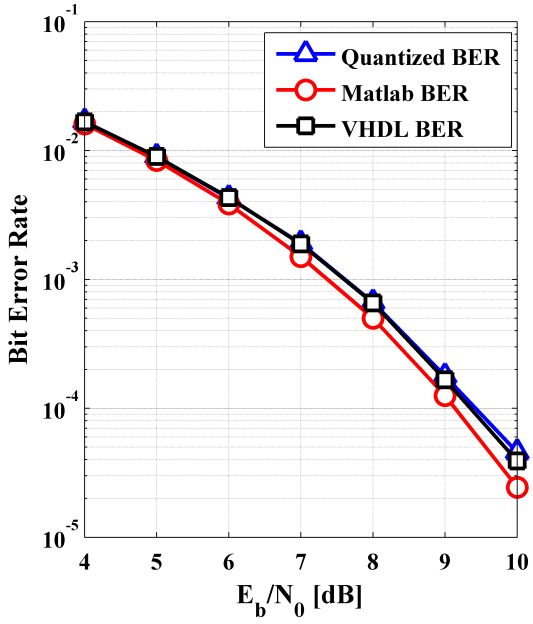

Figure 7: The BER performance of the implemented SOQPSK demodulator according to VHDL and MATLAB simulations.

BER curve and comparing it to quantized and floating point MATLAB simulations. The implementation results also show a large amout of saving in terms of FPGA resources compared to ordinary methods, i.e. register exchange. Finally, the circuit can be used as a demodulator connected to FEC decoders such as low-density parity check (LDPC) or convolutional decdoders leading to a bandwidth-efficient and reliable receiver for telemetry applications.

ACKNOWLEDGEMENTS

The authors would like to thank the Test Resource Management Center (TRMC) Test and Evaluation/Science and Technology (T&E /S&T) Program for their support. This work was funded by the T&E/S&T Program through the U.S. Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI), contract number W900KK-09- C-0018 for High-Rate High-Speed Forward Error Correction Architectures for Aeronautical Telemetry (HFEC).

REFERENCES

[1] Range Commanders Council Telemetry Group, Range Commanders Council, White Sands Missile Range, New Mexico, *IRIG Standard 106-00: Telemetry Standards*, [online] ed. (Available: http://www.ntia.doc.gov/osmhome/106.pdf).

[2] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between sova and max-log-map decodings," *IEEE Communications Letters*, vol. 2, pp. 137–139, 1998.

[3] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," in *Proc. of IEEE Globecom '89, Dallas, Texas*, pp. 1680–1686, November 1989.

[4] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A low complexity soft-output viterbi decoder architecture," in *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, vol. 2, pp. 737 –740 vol.2, 23-26 1993.

[5] T. K. Truong, M.-T. Shih, I. S. Reed, and E. H. Satorius, "A vlsi design for a trace-back viterbi decoder," *IEEE Transactions on Communications*, vol. 40, pp. 616–624, March 1992.

[6] E. Perrins and M. Rice, "Reduced-complexity approach to iterative detection of coded soqpsk," *IEEE Transactions on Communications*, vol. 55, pp. 1354–1362, July 2007.

[7] J. B. Anderson, T. Aulin, and C.-E. Sundberg, *Digital Phase Modulation*. Plenum Press, 1986.

[8] D. Garret and M. Stan, "A 2.5 mb/s, 23 mw sova traceback chip for turbo decoding applications," in *IEEE Int. Symp. Circuits and Systems*, pp. 61–64, May 2001.