CHAPTER 6
HIGH SPEED DEEP PIPELINED VITERBI
DECODER ARCHITECTURE

6.1 INTRODUCTION

Convolutional codes are widely used in modern high performance digital wireless communication systems such as IEEE 802.11, IEEE 802.16, and Multiband-OFDM ultra wideband (UWB) systems. Because of the highly regular computation and storage operation, VLSI architecture for a Viterbi decoder has been widely deployed for the channel decoder for high-speed wireless systems. As the data rate of the wireless applications becomes higher, the power consumption of the Viterbi decoder, which could account for as much as one-third of the power consumption of the baseband processing (Henning and Chakrabarti 2004), becomes one of the most critical issues for designing a Viterbi decoder.

To meet the high throughput requirement of the modern communication systems, the fully parallel architecture is commonly used for implementing the Viterbi decoder (Lin et al 2005). The classical high throughput implementation of a Viterbi decoder is based on the fully parallel implementation method, where ACS units are assigned to each state to iterate one trellis stage (Simmons 1990). Clearly, such a design is excessively complex and area consuming for large constraint lengths. At the other extreme, a single ACS unit can be used to sequentially update all the survivor sequences. Such a design is almost entirely memory dominated and extremely
slows for practical applications (Henning and Chakrabarti 2004). Many ACSs are running at a high clock frequency and hence they consume a lot of power. At the same time, the SMU also consumes high power consumption because of the large number of memory accesses. In some case, it is more than half of the total power consumption of the decoder (Henning and Chakrabarti 2004). There are two known methods for the implementation of the SMU, namely the register exchange method (RE) and the trace back (TB) method (Feygin and Gulak 1993). In general, RE has the advantage of high speed, low latency, and simple control but it consumes more power than the TB mechanism since it needs to move the data among the memories in every cycle. Therefore, the TB mechanism is commonly used for the implementation of the SMU.

Trellis decoding is pervasive in digital communication systems for error correction and signal detection. The well-known Viterbi algorithm (VA) (Forney 1973), performs a breadth-first exhaustive search to realize maximum-likelihood (ML) trellis decoding. Because of its highly regular/parallel computation and data storage/retrieval operations, VA is well suited for high-throughput VLSI implementations and hence is being widely used in real-world applications. However, the use of exhaustive search makes the Viterbi decoder essentially not power efficient, particularly for applications demanding large trellises.

The real-world application of limited search trellis decoders pales in comparison to that of Viterbi decoder. This is mainly due to their lack of operational regularity and parallelism, which makes high-throughput hardware implementation problematic. Hardware decoder design has been studied for several limited search algorithms including Fano algorithm (Fano 1963), stack algorithm (Jelinek 1969), M-algorithm (Anderson and Mohan 1984) and T-algorithm (Simmons 1990). Since the Fano and stack algorithms
are essentially path-serial, i.e., process only one path at one time, they are not suited for high-throughput applications but they can realize very low power consumption (Singh et al 1999). Most prior work (Lee et al 1991) on these two algorithms focused on the data storage/retrieval structure design for moderate decoding speed-up. Belonging to the family of breadth-first search algorithms that also includes the VA, - and - algorithms perform nonexhaustive breadth-first search, where the number of survivor paths at each decoding depth is typically much less than the total number of trellis states, leading to much less computational complexities. Although - and - algorithms have greater potentials for parallel trellis decoding, all the previous work only targeted on the path-serial implementations of these two algorithms. Simmons (1988), developed sorting-based and nonsorting-based path-serial - and - algorithms decoders.

Chan et al (1996) developed a path serial -algorithm decoder that is similar to the state-serial Viterbi Decoder but has longer critical path. We note that the drawback of irregular data storage/retrieval in - and - algorithms is completely concealed by the path-serial decoding process and thus is not an issue in path-serial decoders. The superior power efficiency of the -algorithm has been recently demonstrated by Henning and Chakrabarti (2004).

To the best of our knowledge, no limited search trellis decoder can achieve the throughput comparable to a state parallel Viterbi decoder. In this paper, we have proposed a parallel processing and pipelining techniques implemented in the Viterbi decoder to improve the speed. And also deep pipelining has been implemented in two sub units of the decoder namely add-compare-select unit and the trace back unit.
6.2 HIGH SPEED TECHNIQUES FOR VITERBI DECODER

6.2.1 Pipelining and Parallel Processing for High Speed and Low Power

The two main advantages of using Pipelining and parallel processing are higher speed and lower power. It has already been specified that pipelining and parallel processing can increase the sample speed. Now consider the use of these techniques for increasing sample speed where power consumption does not need to be reduced (Chandrakasan et al 1992).

Before moving on, two formulas are reviewed: one for computing the propagation delay of CMOS circuits and the other for computing the power consumption. The propagation delay $T_{pd}$ is associated with charging and discharging of the various gate and stray capacitances in the critical path. For CMOS circuits, the propagation delay can be written as:

$$T_{pd} = \frac{C_{charge} V_0}{k(V_0 - V_t)^2}, \quad (6.1)$$

where, $C_{charge}$ denotes the capacitance to be charged/discharged in a single clock cycle, i.e., the capacitance along the critical path, $V_0$ is the supply voltage and $V_t$ is the threshold voltage. Parameter $k$ is a function of technology parameters $\mu$, $\frac{W}{L}$ and $C_{ox}$. The power consumption of a CMOS circuit can be estimated using the following equation,

$$P = C_{total} V_o^2 f \quad (6.2)$$

where, $C_{total}$ denotes the total capacitance of the circuit, $V_o$ is the supply voltage, and $f$ is the clock frequency of the circuit. Note that equations (6.1)
and (6.2) are based on simple approximations and are appropriate only for a 1st-order analysis.

### 6.2.2 Pipelining for High Speed and Low Power

As mentioned earlier, pipelining can be used to reduce the power consumption of a Viterbi decoder. Let

\[
P_{\text{seq}} = C_{\text{total}} V_o^2 f
\]

(6.3)

Represent the power consumed in the original Viterbi decoder. It should be noted that \( f = \frac{1}{T_{\text{seq}}} \), where \( T_{\text{seq}} \) is the clock period of the original sequential Viterbi decoder. Now consider an \( M \)-level pipelined system, where the critical path is reduced to \( \frac{1}{M} \) of its original length and the capacitance to be charged / discharged in a single clock cycle is reduced to \( \frac{C_{\text{charge}}}{M} \). Notice that the total capacitance does not change. If the same clock speed is maintained, i.e., the clock frequency \( f \) is maintained, only a fraction of the original capacitance, \( \frac{C_{\text{charge}}}{M} \) is being charged / discharged in the same amount of time that was previously needed to charge / discharge the capacitance, \( C_{\text{charge}} \) (Figure 6.1). This implies, then, that the supply voltage can be reduced to \( \beta V_o \), where \( \beta \) is a positive constant less than 1. Hence, the power consumption of the pipelined Viterbi decoder will be

\[
P_{\text{pip}} = C_{\text{total}} \beta^2 V_o^2 f = \beta^2 P_{\text{seq}}
\]

(6.4)

Therefore, the power consumption of the pipelined system has been reduced by a factor of \( \beta^2 \) as compared with the original system.
Figure 6.1 Critical path lengths for original and 3-level pipelining systems

The power consumption reduction factor $\beta$ can be determined by examining the relationship between the propagation delay of the original Viterbi decoder and the pipelined Viterbi decoder. The propagation delay of the original Viterbi decoder is given by

$$T_{seq} = \frac{C_{charge} V_o}{k(V_o - V_f)^2}$$  \hspace{1cm} (6.5)$$

The propagation delay of the pipelined Viterbi decoder is given by

$$T_{pip} = \frac{C_{charge} \beta V_o}{M \frac{k}{k(\beta V_o - V_f)^2}}$$  \hspace{1cm} (6.6)$$

It should be noted that the clock period, $T_{clk}$ is usually set equal to the maximum propagation delay, $T_{pd}$ in a circuit. Since the same clock speed is maintained for both Viterbi decoders from equations (6.5) and (6.6) the following quadratic equation can be obtained to solve for $\beta$,

$$M (\beta V_o - V_f)^2 = \beta (V_o - V_f)^2$$  \hspace{1cm} (6.7)$$
Once $\beta$ is obtained, the reduced power consumption of the pipelined Viterbi decoder can be computed using equation (6.4)

### 6.2.3 Parallel Processing for High Speed and Low Power

Parallel processing, like pipelining, can reduce the power consumption of a system by allowing the supply voltage to be reduced. In a $L$-parallel system, the charging capacitance does not usually change while the total capacitance is increased by $L$ times. In order to maintain the same sample rate, the clock period of the $L$-parallel circuit must be increased to $LT_{seq}$, where, $T_{seq}$ is the propagation delay of the sequential circuit given by equation 6.5. This means that $C_{charge}$ is charged in time $LT_{seq}$ rather than in time $T_{seq}$. In other words there is more time to charge the same capacitance (Figure 6.2). This means that the supply voltage can be reduced to $\beta V_o$.

![Figure 6.2 Critical path lengths for sequential and 3-parallel systems](image)

The propagation delay considerations can again be used to compute the supply voltage of the $L$-parallel system. The propagation delay of the original system is given by equation (6.5), while the propagation delay of the $L$-parallel system if given by
\[ LT_{seq} = \frac{C_{charge} \beta V_o}{k \left( \beta V_o - V_t \right)^2} \] (6.8)

from equations (6.5) and (6.8), the following quadratic equation can be obtained to compute \( \beta \):

\[ L(\beta V_o - V_s)^2 = \beta (V_o - V_t)^2 \] (6.9)

The power consumption of the \( L \)-parallel system can be calculated as

\[
P_{par} = \left( L C_{charge} \right) \left( \beta V_o \right)^2 \frac{f}{L} = \beta^2 C_{charge} V_o^2 f = \beta^2 P_{seq} \] (6.10)

where, \( P_{seq} \) is power consumption of the sequential system given by equation (6.3). Therefore, as in the pipelined system, the power consumption of the \( L \)-parallel system has been reduced by a factor of \( \beta^2 \) as compared with the original sequential system.

### 6.2.4 Combining Pipelining and Parallel Processing

The techniques of pipelining and parallel processing can be combined for high speed and low power consumption. The principles remain the same, i.e., pipelining reduces the capacitance to be charged / discharged in one clock period and parallel processing increase the clock period for charged / discharging the original capacitance (Figure 6.3). The propagation delay of the parallel-pipelined Viterbi decoder is obtained as follows:
Figure 6.3  (a) Charging /discharging of entire capacitance in clock period $T$. (b) Charging /discharging of entire capacitance in clock period $3T$ using parallel and 2 stages if pipelining

\begin{equation}
LT_{pd} \frac{C_{charge}/M}{k} \beta V_o = \frac{LC_{charge}V_o}{k(V_o - V_t)^2}
\end{equation}  \hspace{1cm} (6.11)

Using this equation the following quadratic equation is obtained

\begin{equation}
ML (\beta V_o - V_t)^2 = \beta (V_o - V_t)^2
\end{equation}  \hspace{1cm} (6.12)

As an example, consider the case when $L = M = 2$, $V_o = 5 \text{V}$ and $V_t = 0.6 \text{V}$. By using equation (6.12), $\beta$ is found to be approximately 0.4, which means that the power consumption can be reduced and operating speed of the Viterbi decoder increased. It should be noted that the supply voltage cannot be lowered indefinitely by applying more and more level of pipelining and parallelism. There is a lower bound on the supply voltage that is dictated by the process parameters and noise margins.
6.3 PROPOSED THROUGHPUT INCREASING METHODOLOGIES

The decoder core consists of two basic architectures: a fully parallel implementation which gives fast data throughput at the expense of silicon area and a serial implementation which occupies a small area but requires a more number of clock cycles per decoding. In this work, a Viterbi decoder of rate of 1/6 and trace back depth of $L$ greater than and equal to $5k$ is designed for various architectures like non-pipelined, parallel with pipelined and deep pipelined. The main bottle nicks of Viterbi decoder are ACS unit and TB unit. In this design the implementation of the ACS and trace back unit is important to improve the throughput of the decoder.

To obtain high throughput implementation, the maximum inherent parallelism needs to be extracted. The parallel architecture performs the computation simultaneously. It may also try to decode multiple outputs at a single clock which leads to high silicon area so it’s better to go for other methods. The parallel processing and pipelining method combines both architectures which suit better for higher $k$ value and the complexity of this architecture also high when compared to deep-pipelined decoder.

![Figure 6.4 High Speed Viterbi decoder Architecture using parallel processing and pipelining](image)
Figure 6.5 Parallel and Pipelined execution of Viterbi decoder

Generally, a high throughput rate is achieved if the circuit has a very short critical path. The critical path of a synchronous circuit is that path between two buffers that has the largest delay and hence determines the maximum achievable clock frequency of the circuit. This is done by applying the pipelining concept to the blocks of the decoder shown in Figure 6.5 and also the corresponding architecture of Viterbi decoder shown in Figure 6.4. The pipeline register block consists of a set of positive edge triggered D-type flip flops. The total number of flip flops required is equal to the number of states multiplied by the number of state metrics bits, i.e. $8 \times 6 = 48$.

Figure 6.6 Deep pipelined design of ACS (Add-Compare-Select) unit
The throughput can be still improved by using deep pipelining method. In this method, the pipelining concept is applied within ACS and TB units up to the trace back depth i.e. \((L \geq 5^*_k)\). The ACS unit performs add-compare-select operations when it is deep pipelined. It performs add as one part and select, compare operations as other part shown in Figures 6.6 and 6.7 show the deep pipelined design of ACS unit. Likewise the trace back unit is also performed as storing metrics and trace back which is shown in Figures 6.8 and 6.9. Compared to other methods, deep pipelining method shows a tremendous reduction in memory usage and the throughput of the chip also increases considerably. But in the non-pipelined system the silicon area required is high that results in high power consumption. This is considerably reduced in deep pipelining method.

Figure 6.7 Deep pipelined execution of ACS (Add-Compare-Select) unit

Figure 6.8 Deep pipelined design of Trace Back (TB) unit
The advantages of using deep-pipelined architecture are: low Area, high Throughput, and high Speed. The speed of the decoder can be improved by applying pipelining approach to ACS and TBU. The whole decoder works on pipelining principle in which the sub blocks also perform pipelining operation up to the trace back depth. To extract the inherent parallelism, the branch metric and ACS unit are operated in parallel in which output is obtained at a single pulse.

Deep Pipelining is performed between ACS and TBU where multiple instructions are overlapped in execution. Deep Pipelining does not decrease the time for individual instruction execution. Instead, it increases instruction throughput and it is determined by how often it exits the pipeline.

In order to compare the throughput variation, all of these architectures are designed with both single and dual RAM decoder and their performance characteristics are tabulated.

In Figure 6.10 shows the single RAM decoder performs operation in conventional manner i.e. simply storing the values of the state metrics and outputs the error free data during trace back. In Figure 6.11 shows the dual RAM mode, the BMU and ACS values are computed for the first message.
stream and stored in RAM1 during the first cycle. Meanwhile, the BMU and ACS values for second message stream are computed and stored in RAM2. At the same time values stored in RAM1 are used by TBU for trace back operation. Then the values of RAM2 are given to TBU when next stream of values enters the memory for storage. The connection between dual RAM and TBU is performed in alternative manner. Adding delay elements so that TBU can be used efficiently should do synchronization between two paths. From the results it can be proved that single RAM decoder operates at a speed lesser than dual RAM model.

Figure 6.10 Trace back with single RAM

Figure 6.11 Trace back with dual RAM
6.4 PERFORMANCE CHARACTERISTICS OF THE DECODER

The performance characteristics of the decoder can be valued from the synthesized report obtained when the decoder is implemented in FPGA (Proakis and Masoud Salehi 2003). The values for the entire factors list below are tabulated in Table 6.3. The performance of the decoder depends upon the following factors:

6.4.1 Bit Error Rate

The fidelity of information transmission in the presence of noise is conveniently measured in terms of error rate or probability of error that is the probability that the receiver output differs from the transmitted. The bit-error-rate (BER) monitors the error rate on the transmission channel.

Decoded data from the Viterbi decoder is re-encoded using the Convolutional encoder and compared with a delayed version of the data input to the decoder. An error is indicated if the delayed and encoded data differ. The two sets of symbols can differ only if there is an error on the channel or if the Viterbi decoder has decoded incorrectly. The probability of the decoder incorrectly decoding is significantly smaller than the probability of a channel bit error, therefore the BER output gives a good estimate of the errors on the channel.

\[
\text{BER} = \frac{\text{Total Errors}}{\text{Total number of bits}} \quad (6.13)
\]

6.4.2 Throughput

The throughput of the design indicates that the number of bits released per second from that device. The calculation of the throughput of the decoder is given below:
Data rate = clock rate  

(6.14)

For e.g. If the clock is 150 MHz, the symbol rate = 150 Mbps.

6.5 RESULTS AND DISCUSSION

Comparisons in terms of area (gates) and speed (throughput in Mbps) have been obtained from actual FPGA implementations. These are shown in Tables 6.1, 6.2 and 6.3. A fixed constraint length \((K = 4)\) Viterbi decoder was implemented using three different methods with single RAM and dual RAM. Among these three methods parallel processing with deep-pipelined method gives the maximum throughput rate. In this Viterbi decoders were implemented using 2 levels pipelining with state parallel processing. Compared to all other methods, only 35% of resources utilized to implement deep-pipelined viterbi decoder. The deep-pipelined architecture implementation achieved a throughput rate of 145.07 Mbps, which is far off with the non-pipelined method.

Table 6.1 Throughput rate comparison with existing methods

<table>
<thead>
<tr>
<th>Viterbi decoder</th>
<th>Throughput (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-parallel fixed (K = 7) Chadha and Cavallaro (2001)</td>
<td>32</td>
</tr>
<tr>
<td>Area-efficient fixed (K = 7) Chadha and Cavallaro (2001)</td>
<td>13.5</td>
</tr>
<tr>
<td>State-parallel ((K = 3–7)) Chadha and Cavallaro (2001)</td>
<td>19.7</td>
</tr>
<tr>
<td>Area-efficient fixed (K = 10) Chadha and Cavallaro (2001)</td>
<td>1.594</td>
</tr>
<tr>
<td>Area-efficient ((K = 7–10)) Chadha and Cavallaro (2001)</td>
<td>12.625–1.578</td>
</tr>
<tr>
<td>Proposed Parallel and Deep pipelined (single RAM) (k=4)</td>
<td>129.25</td>
</tr>
<tr>
<td>Proposed Parallel and Deep pipelined (dual RAM) (k=4)</td>
<td>145.7</td>
</tr>
</tbody>
</table>
The only previous work, that is directly comparable to our work is the one reported in Chadha and Cavallaro (2001) based on a state-parallel implementation for constraints 3 to 7 only. FPGA implementations would yield much more improved overall performance.

In the FPGA prototype, the path metric RAM’s are mapped onto Virtex distributed memory, while Virtex built in block dual-port RAM’s are used for survivor memory. One port is used to receive the survivor data from the ACS module and the other accommodates the trace back operation. This leads to very simple and regular trace back architecture.

It is important to set a maximum period constraint on the core clock input. It may be possible to improve slightly on these values by trying different seed values for the place and route software or by setting the pack factor to give a less dense layout. If necessary, performance can be increased by selecting a faster speed grade. It can be inferred that by using deep pipelining method better results can be achieved than other methods.

From the bar chart Figure 6.12 and Table 6.2 it can be identified that gates count, slices, LUT’s required by the deep pipeline method on an average is 63-65% lesser, than that of other two methods.
Table 6.2  Device utilization statistics using various methods of VD implementation

<table>
<thead>
<tr>
<th>Device utilization Statistics</th>
<th>Device Type</th>
<th>Non pipelining</th>
<th>Parallel processing and pipelining</th>
<th>Deep pipelining</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of slices</td>
<td>Virtex 4</td>
<td>1096</td>
<td>1059</td>
<td>2363</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>1088</td>
<td>1049</td>
<td>2361</td>
</tr>
<tr>
<td></td>
<td>Virtex 2</td>
<td>1076</td>
<td>1055</td>
<td>2367</td>
</tr>
<tr>
<td>No. of 4 Input LUT’s</td>
<td>Virtex 4</td>
<td>1119</td>
<td>1241</td>
<td>1925</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>1108</td>
<td>1242</td>
<td>1941</td>
</tr>
<tr>
<td></td>
<td>Virtex 2</td>
<td>1122</td>
<td>1232</td>
<td>1925</td>
</tr>
<tr>
<td>Gates Count</td>
<td>Virtex 4</td>
<td>17,283</td>
<td>18,511</td>
<td>37,372</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>17,067</td>
<td>18,514</td>
<td>37,442</td>
</tr>
<tr>
<td></td>
<td>Virtex 2</td>
<td>17,269</td>
<td>18,254</td>
<td>37,420</td>
</tr>
</tbody>
</table>

Figure 6.12 Bar chart comparison of gates count using different methods
Table 6.3 Performance statistics of various methods

<table>
<thead>
<tr>
<th>Performance Statistics</th>
<th>Device Type</th>
<th>Non pipelining</th>
<th>Parallel processing and pipelining</th>
<th>Deep pipelining</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Single RAM</td>
<td>Dual RAM</td>
<td>Single RAM</td>
</tr>
<tr>
<td>Characteristics Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Frequency</td>
<td>Virtex 4</td>
<td>91.465</td>
<td>111.867</td>
<td>129.251</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>65.157</td>
<td>82.563</td>
<td>98.941</td>
</tr>
<tr>
<td></td>
<td>Virtex 2</td>
<td>61.827</td>
<td>73.135</td>
<td>93.484</td>
</tr>
<tr>
<td>Throughput (Mbps)</td>
<td>Virtex 4</td>
<td>91.465</td>
<td>111.867</td>
<td>129.251</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>65.157</td>
<td>82.563</td>
<td>98.941</td>
</tr>
<tr>
<td></td>
<td>Virtex 2</td>
<td>61.827</td>
<td>73.135</td>
<td>93.484</td>
</tr>
<tr>
<td>Delay (ns)</td>
<td>Virtex 4</td>
<td>7.834</td>
<td>6.903</td>
<td>7.737</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>10.509</td>
<td>8.806</td>
<td>10.107</td>
</tr>
<tr>
<td>Bit Error Rate (BER)</td>
<td>Virtex 4</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Virtex 2</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Power (mw)</td>
<td>Virtex 4</td>
<td>452.08</td>
<td>452.08</td>
<td>280.97</td>
</tr>
<tr>
<td></td>
<td>Virtex 2p</td>
<td>450</td>
<td>460.03</td>
<td>420</td>
</tr>
<tr>
<td></td>
<td>Virtex 2</td>
<td>351.30</td>
<td>351.30</td>
<td>420.80</td>
</tr>
</tbody>
</table>

Figure 6.13 Bar chart comparison of throughput using different methods
From the bar chart Figure 6.13 and Table 6.3 it can be identified that the throughput of the deep pipeline method on an average increases by 30-41% greater than the other two methods. From Figure 6.14 the Virtex 4 device has high performance than other devices like Virtex 2,2p.

![Comparison of different Virtex devices](image)

**Figure 6.14  Comparison chart of different Virtex devices**

### 6.6 SUMMARY

Broadband access raises new demands for channel coding. Besides lower decoding power and decoding complexity, high speed decoding performance is highly desired. This chapter proposed a high speed and area efficient viterbi decoder based on a full parallel state and deep pipelined structure. This high-speed deep-pipelined structure reduces the delay of the critical path and speed up the throughput. This chapter presents techniques at the algorithm and VLSI architecture levels to realize parallel state and deep-pipelined decoder VLSI implementation. A Viterbi decoder of constraint length 4 and code rate 1/6 is designed and simulated with VHDL and further implemented in Xilinx Virtex device using trace-back based architecture. With the use of the 0.22µm processes, the high-speed operations exceeding 145 Mbps are achieved, while comparing throughputs of the previous works we have achieved maximum.
The device utilization of the decoder includes number of slices used, number of LUT used, gates count used for the design. From the tabulated results of the design it can be inferred that the deep pipelining method utilized only 35% of resources when compared to all the other methods and 41% throughput greater than the other two methods. High performance base band design is very important and challenging for wireless communications IC chipset development. With the comparable decoding rate and incomparable flexibility to the Viterbi decoder implemented on FPGAs, this kind of decoder may have many applications, such as the channel decoder in mobile communication system and the error-correcting scheme in software radio system. This Viterbi decoder can provide high performance and high-speed MODEM solutions for the latest numerous short range wireless communication standards.