Dinamic States Viterbi Decoder Architecture Based on Systolic Array

Ahmad Zaky Ramdani, Trio Adiono
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
ahmadzakyramdani@gmail.com, tadiono@gmail.com

***Abstract*—Viterbi decoder is an error correction method that is widely used in a wide variety of data transmission applications to date. In term of VLSI implementation, architecture optimization is a thing should be highlighted. This paper proposed a new architecture to implementation a viterbi decoder which has a high throughput, minimum area and high flexibility base on projected systolic array.**

Keywords—viterbi decoder; systolic array; high throughput; array processor; VLSI architecture; dinamic states

I. INTRODUCTION

Viterbi decoder has a working principle in which the combination of the received data reanalyzed the possibility of track of code that should be called through the trellis diagram. Since its introduction on 1967, the implementation of viterbi decoder has been being develop and optimized to meet system targeted system requirement. The most classical things of this optimization trade off are speed and area.

As its implementation on communication purposes that majority work on high speed system, throughput became highly priority on implementing viterbi decoder. To obtain high throughput later parallel architecture applied, for sure area become larger. One of architecture had been proposed to obtain high throughput was systolic array. But with a system where the system is paralleled sistolic right into the cells with the ability of computing itself, cause the design will be great and will enlarge as big traceback used. In addition systolic array had been purposed so far employed strong connections system, that cause of huge number of data multiplexing and control.

In this paper, we proposed new systolic architecture that never been implemented on viterbi decoder before. The idea was by combining systolic array with butterfly path metric. Butterfly used to grouping states with vary destination states, and each butterfly adapted as a process element. As states being group has vary destination, add-compare-select process done in each processing element before survivor track send to the next state processing element. The most novel thing in architecture proposed was process elements act as dynamics states handler. It caused connectivity between each process element became more regular and adaptable for another configuration of viterbi decoder. It also reduced number of data multiplexing and control part so need smallest area than another systolic array viterbi decoder. For the verification, in this paper we implement architecture proposed on viterbi decoder R= ½ and k=4. So the architecture will consist of eight states that would be handled by four processing element.

II. EXAMPLE OF A VITERBI DECODER FOR A (4,1/2) CONVOLUTIONAL CODE

Viterbi algorithm is decoder for convolutional code that based on dynamic programming computations. Its data flow can be expressed by a trellis graph. Vertical list of trellis are states, state itself are bits representation of viterbi encoder shift register content.

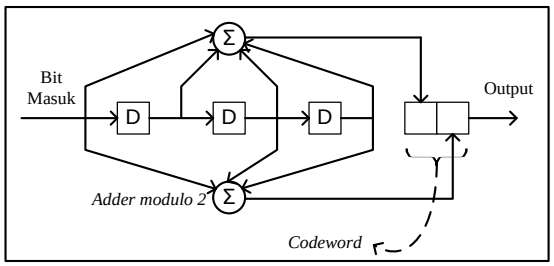

Figure 1 Convolutional code encoder for k=4, R=1/2

Horizontal list are path metric that represented possibility path data receive. Edge weights are given by the branch metric, where the weights are convolutional result from state representation contained shift register. Every sequence data receive being calculated for distance compare to state's weight. Viterbi output path is taken from minimum distance path for a number of iteration called trace back. One of the major steps in the Viterbi algorithm is to update the path metrics according to the following equations (1) (2).

$$pm_{i`,t+1} = \min(pm_{i,t} + bm_{i,i`}, pm_{j,t} + bm_{j,i`}) \quad (1)$$

$$pm_{j`,t+1} = \min(pm_{i,t} + bm_{i,j`}, pm_{j,t} + bm_{j,j`}) \quad (2)$$

Example of trellis diagram for convolutional code with k=4 and R = ½ can see in figure 2. Due to k=4, there will be 8 states from 000 to 111. Later, the diagram is being vertically projected as show in the right part of figure 1 to show connection between states.

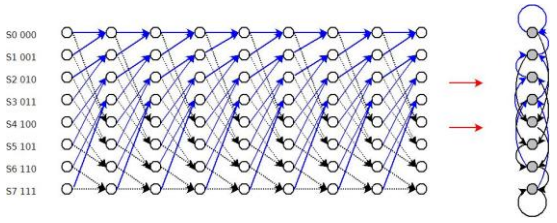

Figure 2 Vertically projection k=4 trellis diagram

To reduce states number, pair of similar destination states are merge using butterfly architecture as show in figure 3. Butterfly trellis diagram can seen in figure 4, where number of states are being updated to only 4 states.

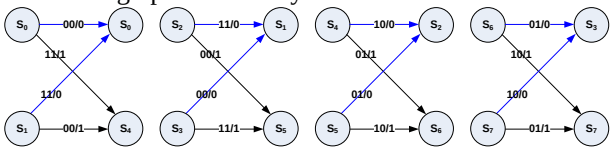

Figure 3 Pair of butterfly states

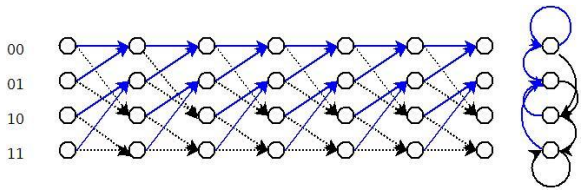

Figure 4 Reduced state trellis diagram and its projection

By using butterfly trellis projection, design datapath actually looks simple. But, irregular connection would cause low flexibility when implemented on different configuration convolutional code. This paper proposed dynamic state trellis to obtain more regular connection. In this dynamic state, each processing element can operate as different state depends on iteration process. For example processing element 1 will act as state 00 at first iteration, but at the second it will be act as state 11. This dynamic cycle is not difficult to adapt in each processing element, because the different of each state is only on the weight use in distance calculation.

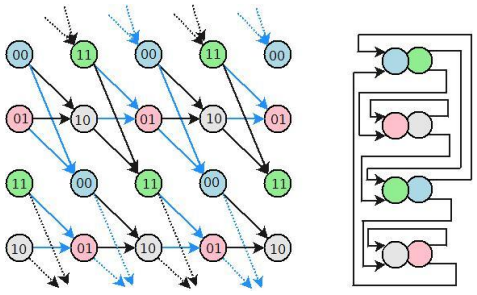

Figure 5 Dinamic state trellis

III. ARCHITECTURE

In this design, processing element (PE) is a component which has the function of trellis and ACS. Trellis function in question is calculating the distance value calculation is obtained if the output given different inputs through Hamming distance calculation. Then each of these calculations becomes a matter of calculation to the next destination state. Each processing element, consist of two states. Each states have two output paths and distances depend on two output possibility ‘0’ or ‘1’ , operation for each bit in the state is done by sub state we called ‘cell’. Both cells in the same state this will lead to a different state each other, according to the state diagram of the circuit change the encoder, but because PE consist of two state with same destination, PE will consist of two pairs of cells that each have same destination state for output ‘0’ and ‘1’. This is because PE is a grouping of two states corresponding butterfly previously described, then for each PE will handle the calculation of two states for each output '0 ' and '1 '.

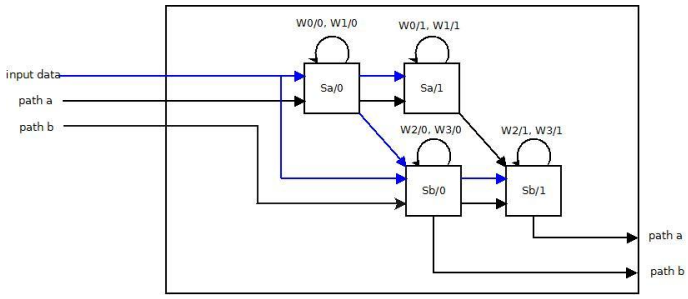

Figure 6 PE structure consist of four cells

The next function wich performed by the PE is add-compare-select(ACS). Main purpose of the ACS block is to choose the best path which will go to a state and throw the other lane. With pair butterfly state being merge on a PE, allows the ACS conducted before the data is sent to the destination state is not after the data is received at the destination state, as being done in many other viterbi decoder implementations. This can increase the efficiency of the circuit because only survivor path being transferred.

The main function of the ACS block is to find the best path to arrive at a single state. As its name, ACS block implies has a working system by adding the measured hamming value from paths arrive to the state, and then compare the total value of the two paths to the state proficiency level (weight), and choose the best path, in that it has the smallest total value of hamming. Which becomes the input of this block are respectively the total distance of the two paths to the state in which the ACS is located, and each path in its path . The width of the data path in accordance with the values used in the trace back for example we used 32 trace back.

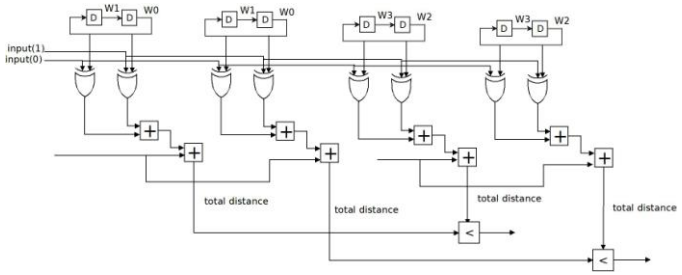

Figure 7 PE datapath

Top-level architecture of the decoder in addition consists of 4 pieces of PE and also some other components such as serial to parallel, traceback counter, and the best route selector . In communication systems in general and the data received serially transmitted due to parallel transmission systems are very susceptible to jitter. Therefore, the data in this block will be accepted as a serial to be converted into parallel ½ as the convolutional code rate use. In this serial to parallel data that has an array of 2-bit input PE will be broadcast for the calculated distance and traced for routes that have passed. For each path will be selected later issued paths with the smallest total distance through the best route selector that looking for track with the smallest distance. Later each best route would be accommodated in parallel and will be issued in series through parallel in serial out register. The track selection is not done in every single iteration, but according issued by the traceback valid counter. Traceback counter will issue any valid value according to the value of the number of iterations used traceback. For example, when the trace back sets to 32 than best path will be choose for every 32 iterations.

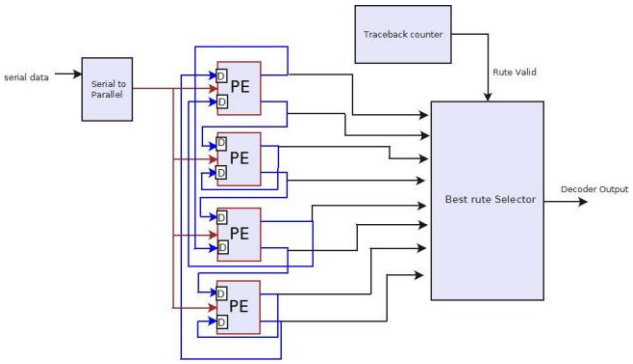

Figure 8 Decoder top level architecture

For another configuration viterbi decoder, some adaptation should be made. If we want to use another convolutional code or another R, architecture can adapted by change cell weight and serial to parallel configuration. For another K, architecture can be adopted by configure number of cell in each processing

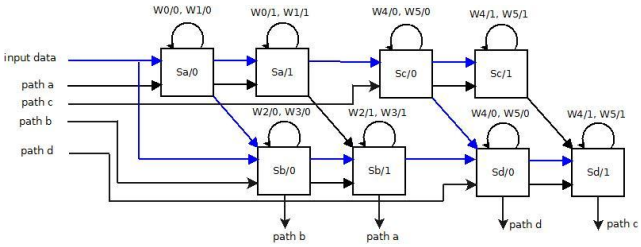

Figure 9 PE adaptation to use in another K

IV. CONCLUTION

This architecture quiet simple and doesn't need any others controlling part. The throughput also high because fully pipeline parallel architecture. It also highly regular so can adapt to another viterbi decoder configuration.

REFERENCES

[1] E. Satorius and T. Truong, "A VLSI Design for a Sistolic Viterbi Decoder," University of Soutern California, 1990

[2] M. Guo, "FPGA Design and Implementation of Systolic Array-Based Viterbi Decoders," Concordia University, 2002

[3] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "Design of a Super-Pipelined Viterbi Decoder," Royal Institute of Technology, 1999.

[4] K. Parhi, "VLSI Digital Signal Processing Systems," Wiley.

[5] M. Biver, H. Kaeslin and C. Tommasini, "In-Place Updating of Path Metrics in Viterbi Decoder" IEEL Journal of Sloid-State Circuits Vol 24. No. 4. August 1989

[6] I. Hidayat, "Design and Analition Viterbi Decoder for DVB Standard," ITB, 2007