Specification of Viterbi HDL Code Generator

PART I

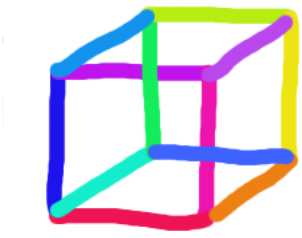
Introduction

Mikael Johnson

Electronic Engineering

Department of Tsinghua University

27th May 2004



Abstract

Viterbi algorithm is the most likelihood decode algorithm of convolution code. Viterbi decoder means the VLSI implementation of Viterbi algorithm. In the area of communication, convolution code is very popular, so how to improve the performance and reduce the power and area of the decoder is important. In the other hand, different protocols use different convolution code and varied applications have different requirement for throughput, area and power. So design of reusable Viterbi decoder is important, too. In present project, a reusable Viterbi decoder was carried out. This decoder adopted the Process Element (PE) technique, which made it easy to adjust the throughput of the decoder by increasing or decreasing the number of PE. By the method of Same Address Write Back (SAWB), we reduced the number of registers to half in contrast with the method of ping-pong.

This decoder supported punctured convolution code and was data-driven, which means the circuit was able to work under different data rate and avoid those invalid operations. The core parameters, such as the generation words of convolution code, the number of PE, the depth of TBU and maybe the radix of buttfly, are all configurable. Some programs have been developed in Perl to generate the Verilog code from these parameters automatically. A radix-2 four-buttfly 64-state punctured (4, 1, 6) Viterbi decoder has been generated and has been successfully simulated and synthesized under some CMOS process.

Contents

1 Basic Ideal **4**

1.1 Process Element 4

1.2 Coordinate of State 4

1.3 Same Address Write Back (S.A.W.B) 5

1.4 Division of States 7

1 Basic Ideal

1.1 Process Element

PE, Process Element, was one of the most popular architecture in digital signal process. The PE architecture of Viterbi decoder has been introduced in these papers[1, 2]. Fig1 is the basic structure of PE in Viterbi decoder. which consists of PMU, ACSU, BMU and

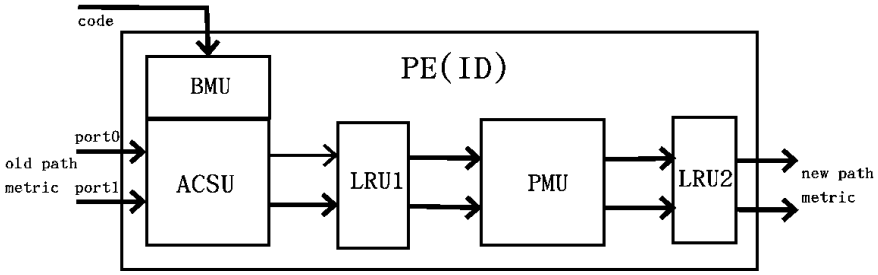


Figure 1: Structure of PE

LRU. By assembling different numbers of PE, we can get the state-serial, part parallel or full parallel structrue of Viterbi decoder. And because the PMU is scattered into each PE, this structure is more area efficient than that structure of only one PE.

1.2 Coordinate of State

In order to make two or more PE work together, we should send the appropriate state to the appropriate PE at the appropriate time. On this purpose, we can regard the binary presentation of each state as a coordinate and give special "meanings" to each bit. Look at such a binary presentation of state[2]

$$S = [X_u X_{u-1} \cdots X_1, Y_w Y_{w-1} \cdots Y_1, Z_v Z_{v-1} \cdots Z_1], \quad \begin{array}{l} X_i, Y_i, Z_i \in \{0,1\} \\ u, w, v \in N, u+w+v = m \end{array} \tag{1}$$

In eq.1 m is the depth of convolution coding. So, we define such a rule named "The Rule of PE" to explain the "meanings":

The Rule of PE When $[X_u X_{u-1} \cdots X_1, Y_w Y_{w-1} \cdots Y_1, Z_v Z_{v-1} \cdots Z_1]$ is the binary presentation of state S, S should be sent to the port $[Z_v Z_{v-1} \ldots Z_1]$ of PE $[Y_w Y_{w-1} \ldots Y_1]$

at time $[X_u X_{u-1} \dots X_1]$.

Let $Slice = [X_u X_{u-1} \dots X_1], Identity = [Y_w Y_{w-1} \dots Y_1], Path = [Z_v Z_{v-1} \dots Z_1]$, so *Slice*, *Identity* and *Path* indicate the time of calculating path metric, the ID of PE executing this calculation and the input port of PE respectively. u, v and w are three important parameters in the Rule of PE, because they determine the total slices for calculating(2^u),the radix of buttfly(2^v) and the total number of PE(2^w) respectively. Based on eq1 and the Rule of PE, we can use binary presentation of state as the coordinate of it.

1.3 Same Address Write Back (S.A.W.B)

Usually there are two ways to design PMU: one is ping-pong method and the other is S.A.W.B. The later means write the new path metric back to where we read the old path metric from. It cost less storage unit than the method of ping-pong, which uses two same ram or register file to storage the new path metric and the old path metric respectively. By the method of S.A.W.B we reduce the storage units, but because it changes the storage address of path metric, we need some address transformation unit and more routing network. So being aware of the state flow graph in PE is very important when we adopt S.A.W.B method. Fig2 demonstrates the state flow in PE.

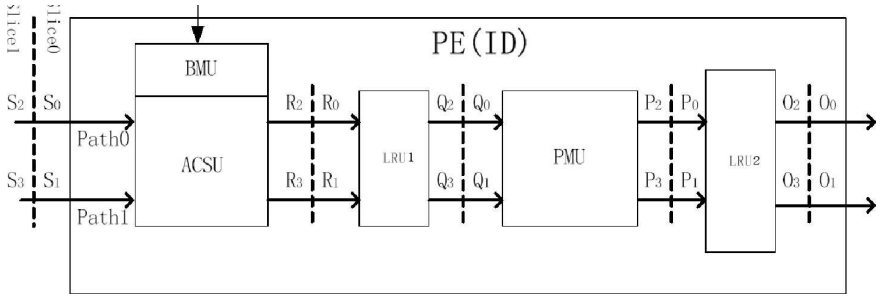


Figure 2: State Flow

In fig2 Path0 and Path1 are two in-ports of Radix-2 buttfly; ID is the number of PE; Slice0 and Slice1 are the taps. S_i, R_i, Q_i, P_i and O_i indicate the states at each in-port or

out-port of submoudles of PE respectively, $i \in \{0, 1, \ldots, 2^m - 1\}$. Code is the covolution code through the noise channel.

Firstly we define some operations:

$$\begin{aligned}
S_i &= [X_u X_{u-1} \ldots X_1, Y_w Y_{w-1} \ldots Y_1, Z_v Z_{v-1} \ldots Z_1] \\
Shuffle^v(S_i) &= [Z_v Z_{v-1} \ldots Z_1, X_u X_{u-1} \ldots X_1, Y_w Y_{w-1} \ldots Y_1] \\
\Gamma^v_{u+w}(S_i) &= [Shuffle^v(X_u X_{u-1} \ldots X_1, , Y_w Y_{w-1} \ldots Y_1), Z_v Z_{v-1} \ldots Z_1] \\
&= [Y_v Y_{v-1} \ldots Y_1 X_u X_{u-1} \ldots X_{v+1}, X_v X_{v-1} \ldots X_1 Y_w Y_{w-1} \ldots Y_{v+1}, Z_v Z_{v-1} \ldots Z_1]
\end{aligned}$$

Because of the ACS operation in ACSU, R_i is not the same of S_i and there exists a transformation between S_i and R_i as following:

$$\begin{aligned}
R_i &= Shuffle^v(S_i) \\
&= [Z_v Z_{v-1} \ldots Z_1, X_u X_{u-1} \ldots X_1, Y_w Y_{w-1} \ldots Y_1] \qquad (2)
\end{aligned}$$

Eq2 is determined by the ACS operation in ACSU. From it we can see the least w bits of R_i are always equal to $[Y_w Y_{w-1} \ldots Y_1]$, the ID of PE.

Let's look at state O_i and leave Q_i and P_i for later. O_i is transformed from R_i through LRU1, PMU and LRU2. The least w bits of O_i should be $[Y_w Y_{w-1} \ldots Y_1]$, supposing that LRU1, PMU and LRU2 only reschedule the sequence of state flow and not change the set of states. Because the out-ports of PE are connected to certain in-ports of another PE, O_i should be certain S_i at certain in-port of certain PE with the same slice. That indicates the most u bits of O_i should be $[X_u X_{u-1} \ldots X_1]$, the slice of S_i. Any more, O_i is rearranged from R_i and so the set of O_i and the set of R_i should be same. All above, we appoint such value to O_i:

$$\begin{aligned}
O_i &= [X_u X_{u-1} \ldots X_1, Z_v Z_{v-1} \ldots Z_1, Y_w Y_{w-1} \ldots Y_1] \qquad (3) \\
&= [Shuffle^u(Z_v Z_{v-1} \ldots Z_1, X_u X_{u-1} \ldots X_1), Y_w Y_{w-1} \ldots Y_1] \\
&= \Gamma^u_{v+u}(Z_v Z_{v-1} \ldots Z_1, X_u X_{u-1} \ldots X_1, Y_w Y_{w-1} \ldots Y_1)
\end{aligned}$$

Eq3 meets all the requirements described above. Maybe there exist some other appointment of Q_i, but eq3 maybe the simplest.

Further more if $< S >_j$ denotes the address of state S at cycle j (cycle is the time during which we calculate the path metric of all states), then $< R_i >_{j+1} = < O_i >_j$ is directly derived from the define of S.A.W.B. And based on eq2, eq3 we can write out the address transformation caused by S.A.W.B :

$$\begin{aligned} < S >_{j+1} &= \text{S.A.W.B}(< S >_j) \\ &= < \Gamma^u_{v+u}(S) >_j \end{aligned} \tag{4}$$

Finally, Q_i and P_i are determined by R_i and O_i respectively. If there are no LRU1 or LRU2, Q_i and P_i will be equal to R_i and O_i respectively. Because LRU1 or LRU2 is not a storage unit, it rerouting the in-ports into out-ports, how to choice the value of Q_i and P_i is determined by what structure of the PMU. We will determined the value of Q_i and P_i in the next section.

1.4 Division of States

In PMU we will use register file for path-metric storage. If we use only one register file for read and write, we will need a register file with two read and two write ports for radix-2 buttfly. But this kind of register file is very complicated and not area-efficient, so we decide to divide the set of all states calculated by each PE into different subsets. If we use Radix-2^v buttfly, we will divide these states into 2^v subsets and storage them the same number of register files.

On this purpose, we should distinguish between the Q_i with the same slice and between the P_i with the same slice, too. In another words, if we use function Θ to distinguish between different Q_i and between different P_i, it should satisfied this:

Class Rule: $\forall\ Q_j, Q_k \in \{Q_i | Q_i$ with the same slice $\}$ and $Q_j \neq Q_k$, we have $\Theta(Q_j) \neq \Theta(Q_k)$; $\forall\ P_j, P_k \in \{P_i | P_i$ with the same slice $\}$ and $P_j \neq P_k$, we have $\Theta(P_j) \neq \Theta(P_k)$, too.

In order to make problem simple, we provide that u is multiple of v. It is not difficult to be satisfied for $v = 1$. With provision of this, we divide the binary presentation of state

into $\frac{u}{v}$ groups each having v bits. Then we sum these $\frac{u}{v}$ groups with mod-2^v-add. From the sum we could divide set {Q} or {P} into 2^v subsets. It is not hard to be validated. So LRU1 reroutes R_i into Q_i and LRU2 reroutes P_i into O_i with the right order which is determined by the Class Rule.

References

[1] Montse Boo, Francisco Arguello, Javier D.Bruguera, Ramon Doallo, and Emilio L.Zapata. High-Performance VLSI Architecture for the Viterbi Algorithm. *IEEE Trans. Communications*, 45(2), February 1997.

[2] F.Arguello, J.D.Bruguera, R.Doallo, and E.L.Zapata. Parallel Architecture for Fast Transforms with Trigonometric Kernel. *IEEE Trans. Parallel and Distributed Systems*, 5(10), October 1994.