Open Cores

Viterbi Transceiver

SandunRathnayake
2/12/2012

This Page is intentionally left blank

Revision History

Rev	Date	Author	Description
0.1	12/02/2012	Sandun R	First draft

Table of Contents

Table of Figures

1. Introduction

1.1 Viterbi TX RX Introduction

The Viterbi Tx Rx is a core that capable of encoding and decoding the data streams in Viterbi scheme.

2. Viterbi Tx Rx

1.2 Overview

Viterbi Tx Rx consist of two major components. They are the Encoder and the Decoder. Encoder is pretty simple and straight forward. Decoder has some complex functionality to recover the original signal from the encoded input to the decoder. The functionality of those modules is described in the following chapters.

1.2.1 Wishbone interface

The design is not wishbone compatible up to now.

1.2.2 Transmit module

One bit stream is input in to this module. Two bit decoded output is given by the module.

1.2.3 Decoder modules

This module contains number of sub modules to perform the operation. They are Branch Metric Calculation modules (BMC), Add Compare Select Modules (ACS), Memory modules (Trellis memory, Display Memory) and Trace Back Modules (TBU).

1.3 Core File Hierarchy

```
trunk

.       bench

.       .       Viterbi_tx_rx_tb.v

.       doc

.       .       Viterbi Tx Rx.pdf

.       rtl

.       .       ACS.v

.       .       bmc

.       .       .       bmc000.v

.       .       .       bmc001.v

.       .       .       bmc010.v

.       .       .       bmc011.v

.       .       .       bmc100.v
```

. . . bmc101.v

. . . bmc110.v

. . . bmc111.v

. . decoder.v

. . encoder.v

. . mem_1x1024.v

. . mem_8x1024.v

. . tbu.v

. . Viterbi_tx_tx.v

1.4 Description of core modules

The top module Viterbi_tx_rx.v contains the instantiations of encoder and the decoder modules. Encoder output is directly connected to the decoder input for this design. It can be separately used according to the requirement.

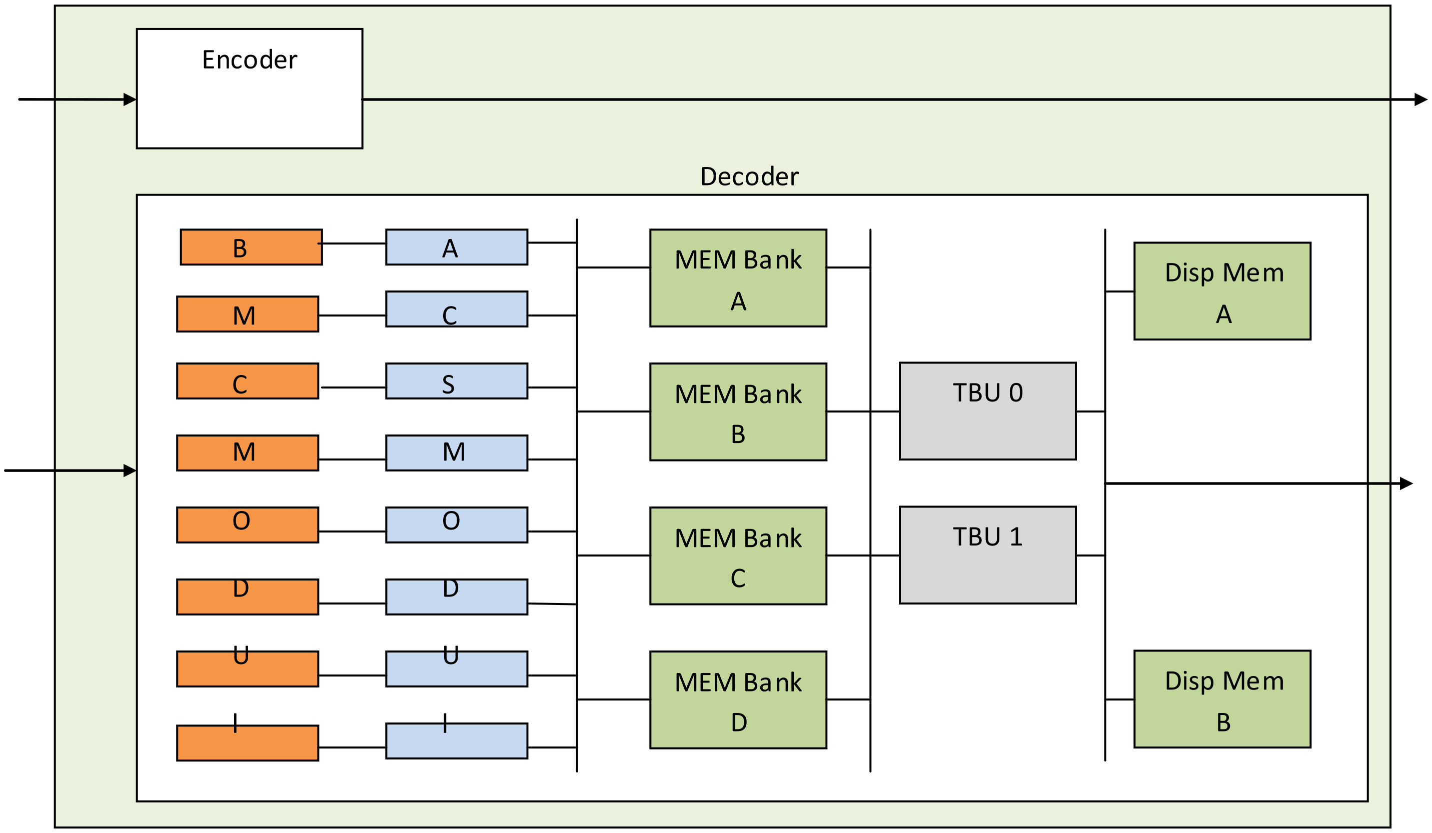


Figure 1 Design Architecture

1.5.1 Description of the Encoder Module

This module receives an input one bit stream. The output stream is two bit. This is a recursive convolution encoder.

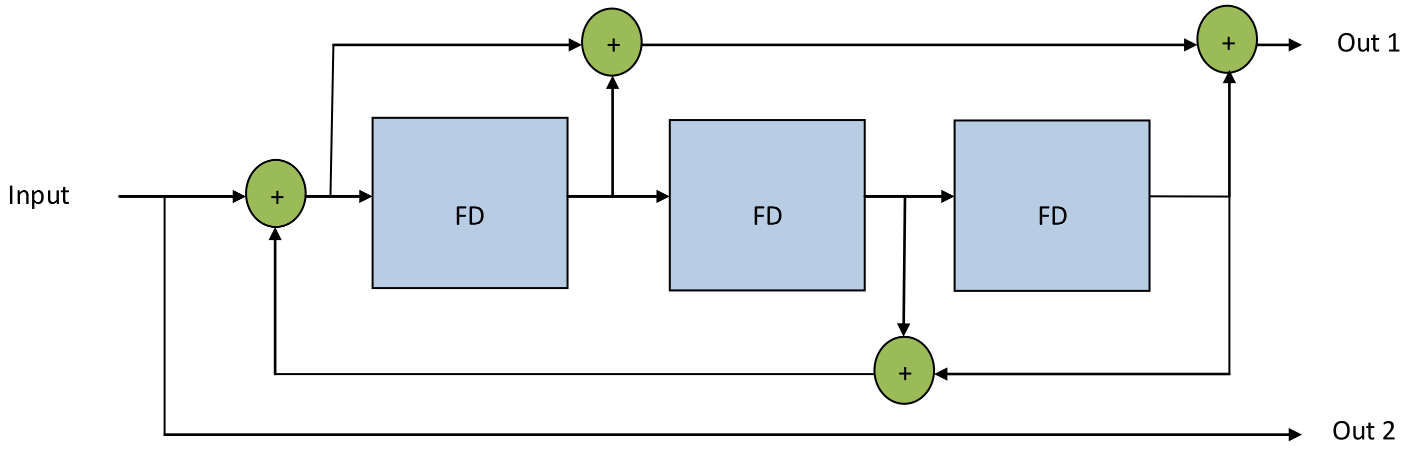


Figure 2 Encoder Operation

The signals **enable** the input and **valid** the output is used for more precisely control the operation to distinguish the idle operation periods of the encoder.

1.5.2 Describe the operation of the BMC module

Branch Metric Calculation (BMC) modules are used to calculate the difference between the desired input to the decoder and received input.

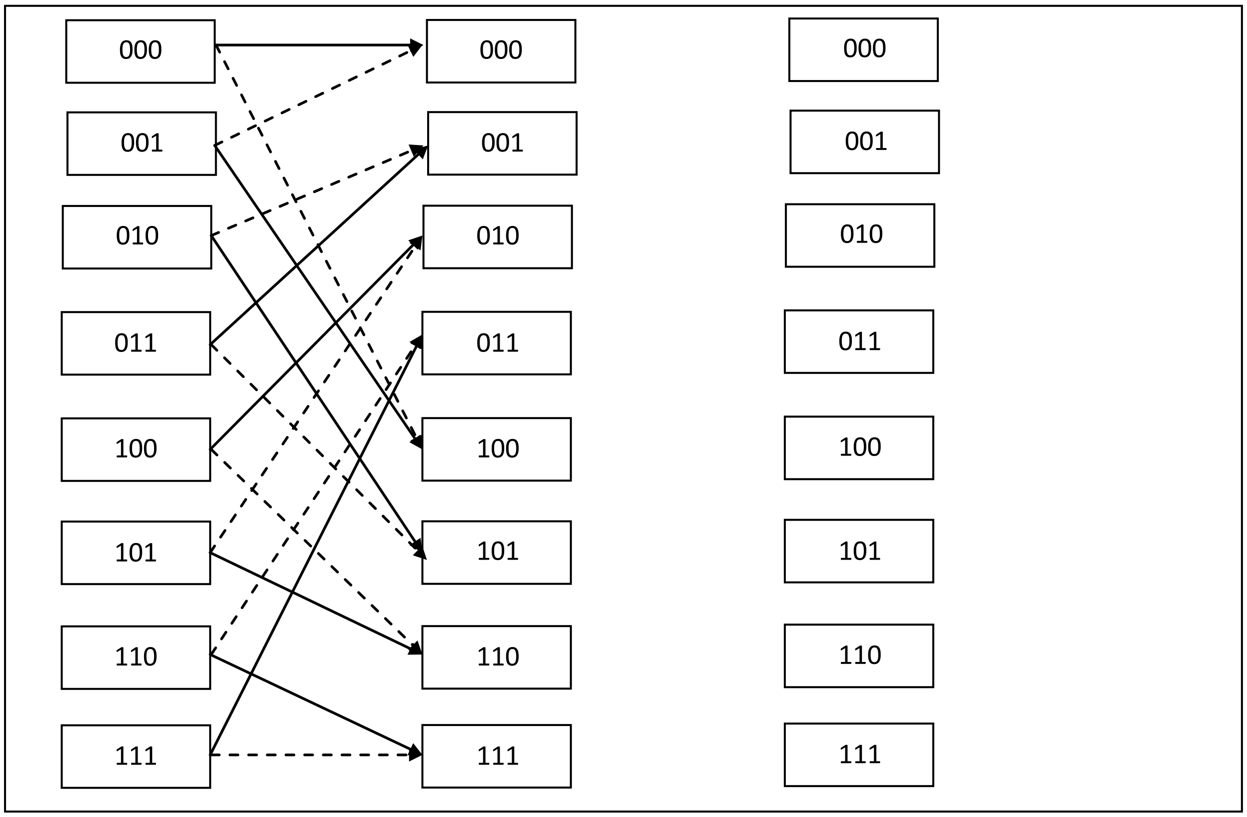


Figure 3 Trellis mesh

The decoder follows the states of the encoder starting the 000 state. This assumes that initial state of the encoder is also 000. The global reset operation coincide the encoder and decoder states.

Each decoder state can be reached from distinct two other states. The BMC computes the difference between the probable inputs from the previous states and send to ACS modules for selection.

ex

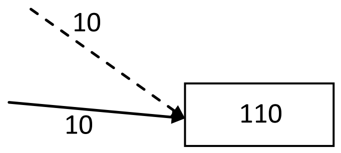


Figure 4 BMC module operation

Current State	Input	Next State	Output
100	1	110	10
101	0	110	01

So the BMC computes the received two bits at this instance to the probable inputs (01,10) and output as a metric.

00	-	No difference
01	-	One field is different
10	-	Totally different

rx_pair is the decoder input to the BMC **path_0_bmc** is the difference from path 0 while **path_1_bmc** is the difference from path 1.

1.5.3 Description of the ACS Module

Add Compare Select (ACS) module

- Add the bmc module output to previous path cost of the arriving state.
- Compare the new path costs of the arriving states.
- Selects the lowest cost predecessor state to build up the path.

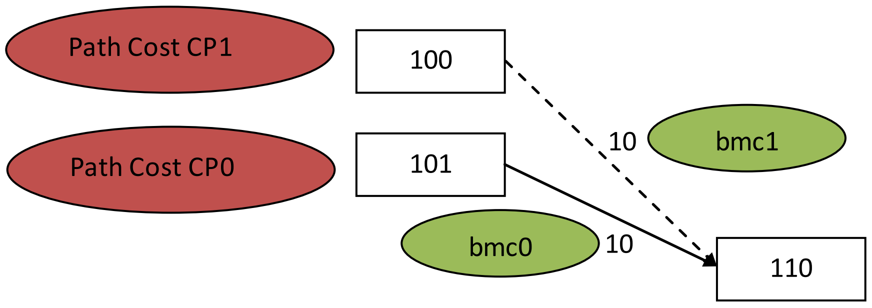


Figure 5 ACS module operation

Next state selects the lowest [**CP(X) + bmc(x)**] path as the arriving path to this state. So each active state consist only 8 active propagating paths.

1.5.4 Description of Memory bank operations

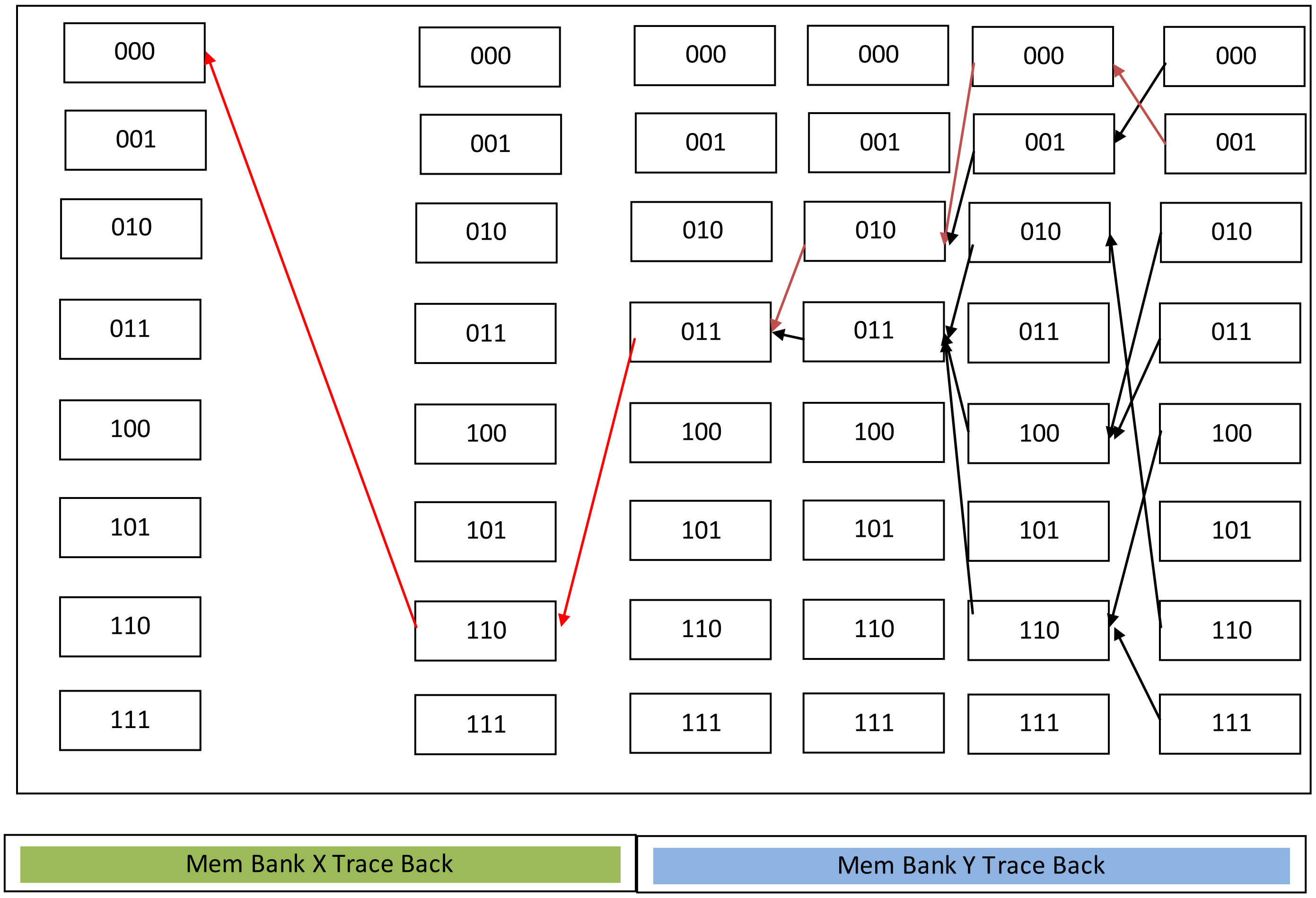
There are 8 ACS modules foe the each state. At each clock edge they select one of the arriving paths (0 or 1) as its sate selection. The memory banks are used to store the current sate selections to build up the trellis.

BANK-A	BANK-B	BANK-C	BANK-D	BANK-A	BANK-B	BANK-C	BANK-D

The four memory banks are written in round robin manner. A single memory bank is 8 bit width for store selections (0 or 1) of the predecessor path. I have used 1024 depth memory. That can be changed according to the precision requirement. Higher memory depth is more likely to be error prone decoder output.

1.5.5 Description of the TBU operation

Trace Back Unit (TBU) is needed to traverse back in the trellis mesh and identify the decoded outputs. There are parallel working two TBU units used in the design.



As we know the next state selects the lowest cost predecessor when the trellis mesh builds. So every instance it has only 8 active paths. When we trace back trellis the final 8 paths get converged at a point. We can get the correct decoder output from that movement.

So we trace back two memory banks. It can start any point of the final state. We assume that after trace back a one memory bank it gets converge the all paths. Then the trace back of the next Memory bank is acceptable.

Then it does not need to trace back all 8 paths at the start. We can start from a random endpoint. It is assured that all end points get converged at an early point in trellis.

The selected decode output is written in to display memory. Separate process output the decoder memory contents.

1.5.6 Overall process description

The below timing graph show the overall operation of the decoder. The arrows indicate that the memory read addressing direction whether forward direction or the reverse direction.

Display memories are 1x1024 bit in size.

Memory Write op	⟶	A	B	C	D	A	B	C	D	A	B	C	D
TBU-0 op	⟵			B	A	D	C	B	A	D	C	B	A
TBU-1 op	⟵				C	B	A	D	C	B	A	D	C
Disp mem-0 wr op	⟵				A		C		A		C		A
Disp mem-1 wr op	⟵					B		D		B		D	
Disp mem-0 rd op	⟶					A		C		A		C	
Disp mem-1 rd op	⟶						B		D		B		D