# Viterbi Decoder HW Description

# Overview

This Viterbi Decoder adopts a unified and mature HW architecture, which is parameter- configurable and supports the convolutional decoding used in LTE, NB-IOT and GSM/GPRS/EDGE. The sliding window technique with forward trace back algorithm was implemented to reduce the Path Metric buffer. The configurable trellis based on the unified HW architecture can support different constraint lengths from 4 to 7, and the code rate from 1/2 to 1/6.

# Architecture

 Figure  Architecture Diagram

The viterbi\_core contains 64 BMU, 64 ACS, one PM normalize block, one traceback block and the control FSM in the top. The BMU is branch metric unit for branch metric calculation which is indexed by state. The ACS is Add-Compare-Select unit to generate the surviving path and the path metric of current time Tn. It’s also indexed by state. The surviving path bits of 64 states are combined to 64-bit width data and be stored to 64x64 PM buffer. The pm\_normalize block can generate the max PM and its state index. It also do path metric normalizing. This block contains 64 PM registers to hold the PM values for next time ACS processing( as PM of time Tn-1), which are feed back to the corresponding ACS block, according to the trellis structure. The traceback block reads the surviving path bits from PM buffer and extracts the correct decoding bits. The traceback process starts from max PM state, or the state 0 at the end of tail-bit mode.

# Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Direction** | **Width** | **Description** |
| clk\_i | in | 1 | Clock input |
| rst\_an\_i | in | 1 | Asyn reset, low active |
| rst\_sync\_i | in | 1 | Sync reset, high active |
| frame\_start\_i | in | 1 | a pulse to trigger one frame decoding |
| register\_num\_i | in | 2 | 0: 64-state, 1: 32-state, 2: 16-state, 3: 8-state |
| valid\_polynomials\_i | in | 3 | 0: R=1/2, Polynomials 1&2 are valid  1: R=1/3, Polynomials 1,2&3 are valid  2: R=1/4, Polynomials 1,2,3&4 are valid  3: R=1/5, Polynomials 1,2,3,4&5 are valid  4: R=1/6, Polynomials 1,2,3,4,5&6 are valid |
| tail\_biting\_en\_i | in | 8 | 0: tail-bit 1: tail-biting |
| polynomial1\_i | in | 8 | Convolutional encoding polynomial 1 |
| Polynomial2\_i | in | 8 | Convolutional encoding polynomial 2 |
| Polynomial3\_i | in | 8 | Convolutional encoding polynomial 3 |
| Polynomial4\_i | in | 8 | Convolutional encoding polynomial 4 |
| Polynomial5\_i | in | 8 | Convolutional encoding polynomial 5 |
| Polynomial6\_i | in | 8 | Convolutional encoding polynomial 6 |
| infobit\_length\_i | in | 12 | 0: 0 bit  1: 1 bit  ....  0x3ff: 1023 bit |
| decoding\_length\_i | in | 13 | Valid when Tail-biting==1 and decodingLength>=InfoBitLength  1: 1 bit  0x3ff: 1023 bit |
| frame\_done\_o | out | 1 | A pulse after one frame decoding complete |
| busy\_o | out | 1 | Is high when viterbi\_core is decoding |
| src\_start\_addr\_i | in | 12 | Source data start address |
| dst\_start\_addr\_i | in | 12 | output data start address |
| src\_rd\_o | out | 1 | Read enable to input buffer |
| src\_addr\_o | out | 12 | Read address to input buffer |
| src\_rdata\_i | in | 24 | Read data from input buffer |
| dst\_wr\_o | out | 1 | write enable to output buffer |
| dst\_addr\_o | out | 12 | write address to output buffer |
| dst\_wdata\_o | out | 8 | write data to output buffer |
| tb\_wr\_o | out | 1 | write enable to PM 64x64 buffer |
| tb\_rd\_o | out | 12 | Read enable to PM 64x64 buffer |
| tb\_addr\_o | out | 8 | addr to PM 64x64 buffer |
| tb\_wdata\_o | out | 12 | Write data to PM 64x64 buffer |
| tb\_rdata\_o | out | 8 | read data from PM 64x64 buffer |

# Sub Blocks

## 4.1 Functions of control logic

The control logic is not a dedicate verilog module, just code segments in viterbi\_core. The functions of decoding control logic:

* Input buffer’s read enable and address generation.
* Output buffer’s write enable and address generation.
* Decoding flow control.
* Trigger traceback once the PM buffer full or decoding to end.

## 4.2 Ctrl FSM

The decoding process can be split into two stages, the first stage is fetching the soft-bits from input buffer and injecting them to the trellis processing chain, which includes BMU, ACS and PM\_Normalize. The second stage is traceback, which is behind the first stage. The second stage start once the surviving path buffer is full or at the end of decoding. Based on the different decoding length, the FSM have 3 states for processing: RUN\_HALF\_TB, RUN\_HULL\_TB and FLUSH\_ALL. Both two stages are included in each processing state. But the number for fetching soft-bits and the traceback length are different among these processing states.



**Figure 2 CTRL FSM Diagram**

* **Two counters for the FSM**

**trellis\_idx\_r**: is just for counting how many data left for decoding. When frame\_start\_i is asserted, it is initialized to infobit\_length\_i when tail bit mode, or decoding\_length\_i when tail biting mode.

**fetch\_src\_cnt\_r**: is for counting how many data need be fetched from input buffer in this processing state. It’s a decreased counter. When it equals to 0, the first stage is finished and the second stage starts to run.

* **State Description:**

**IDLE:** idle state, the default state after reset. When decoding\_start is asserted, it jumps to CHECH\_REMIN state. After decoding is done, the FSM goes back to this state.

**CHECH\_REMIN:** this state is for checking how many data left, depending the left data number, it jumps to 3 different processing states.

1. To RUN\_FULL\_TB: at the beginning of decoding, if the left data number is more than 64.
2. To RUN\_HALF\_TB: after the decoding start, if the left data number is bigger than 32.
3. To FLUSH\_ALL: at the beginning of decoding, if the left data number is NOT more than 64, Or at the end of decoding, the PM buffer need be flushed to empty.

**RUN\_FULL\_TB:** In this state, the initialize value of **fetch\_src\_cnt\_r** is 64. The trace back length is 64. When the segment\_done is asserted, it jumps back to CHECH\_REMIN, and the trellis\_idx\_r is reduced by 64.

**RUN\_HALF\_TB:** In this state, the initialize value of **fetch\_src\_cnt\_r** is 32. The trace back length is 64. When the segment\_done is asserted, it jumps back to CHECH\_REMIN, and the trellis\_idx\_r is reduced by 32.

**FLUSH\_ALL:** In this state, the initialize value of **fetch\_src\_cnt\_r** is trellis\_idx\_r. if at the beginning of decoding, The trace back length is trellis\_idx\_r. otherwise the traceback length is trellis\_idx\_r+32. When the segment\_done is asserted, it jumps back to IDLE, and the trellis\_idx\_r is counted to 0. The frame\_done is generated during FSM changes from FLUSH\_ALL to IDLE.

* **Key signals:**

**segment\_done:** a pluse after one segment is done. It’s an output signals of traceback module.

**decoding\_start**: it’s asserted after the codeword generation in BMU is done, and bmu is ready to accept the soft\_bit data.

## 4.3 BMU

The BMU is used to calculate the Branch Metric (BM). It has 2 functions: One is for the codeword generation, another is BM calculation. After the frame\_start asserted, the BMU will take 2 cycles to generate the codeword based on its state index value. Then the bmu\_ready\_o is asserted to trigger the FSM start. The control logic will fetch the soft\_bits data from input buffer, and feed to to BMU. BMU takes 1 cycle to generate the BM data.

## 4.4 ACS

The ACS is Add-Compare-Select module. It takes 1 cycles to calculate the PM value. Its functions:

1. Add: generate tow path metric value

pm\_low: the PM of path:State k>>1 🡪 State k, the path bit is 0.

pm\_high: the PM of path: State ( k+ states\_num) >> 1 🡪 State k, the path bit is 1.

2. Compare: compare pm\_low and pm\_high.

3. Select: selecting the max value of pm\_low/high, and restoring the surviving path bit.

**Unified ACS:**

For 8/16/32/64 states, the trellis are different. To use the unified ACS module, each ACS has one previous PM input (prev\_low\_i) for pm\_low generation, and 4 previous PM inputs (prev\_high1/2/3/4\_i )for pm\_high generation.

For ACS of state k ( 0~63 ):

*prev\_low\_i: PM value of State k>>1.*

*prev\_high1\_i: PM value of State (k>>1+4).*

*prev\_high2\_i: PM value of State (k>>1+8).*

*prev\_high3\_i: PM value of State (k>>1+16).*

*prev\_high4\_i: PM value of State (k>>1+32).*

In the ACS, the selection logic for prev\_high:



## 4.5 PM\_Normalize

This block is used to select the max PM value and its corresponding state index. And then all PMs from ACS will subtract the max PM value, as normalization.

## 4.6 traceback

This block is for trace back. It needs the trace back start state index (start\_state\_index\_i), the trace back length(tb\_len\_i) and the trace back start address(tb\_start\_addr\_i).

**start\_state\_index\_i**: In tail biting mode, it always comes from the output of pm\_normalize. In tail bit mode, it comes from the output of pm\_normalize, except the end of decoding, it starts from state 0.

**tb\_len\_i**: it is calculated by ctrl FSM.

**tb\_start\_addr\_i**: it’s the last write address of surviving path buffer (also called tb buffer).

**The traceback process:**

1. Read data from tb buffer from tb\_start\_addr
2. Select the surviving path bit( *get\_bit\_s* ) from read data. It’s selected by a 64 mux, the selecting index is the state\_index\_r. The initial value of state\_index\_r is state\_state\_index\_i.
3. Update the state\_index\_r, and read the next data from addr-1. Repeat from 2 until the trace back length is reduced to 0.

The state index update as this:

*if(get\_bit\_s)*

*state\_index\_r <=(state\_index\_r>>1) + left\_shif\_num\_r;*

*else*

*state\_index\_r <=(state\_index\_r>>1);*

the left\_shif\_num\_r is from the state number:

*when 64 : left\_shif\_num\_r = 6'b100000;*

*when 32: left\_shif\_num\_r = 6'b010000;*

*when 16: left\_shif\_num\_r = 6'b001000;*

*when 8: left\_shif\_num\_r = 6'b000100;*

**The output stream:**

According the sliding window algorithm, each time only the half of tb buffer depth will generate the output bits. But at the decoding end, the tb buffer needs be flushed to empty, and every entry of this buffer will generate one output bit. To make it simple, this block uses to left- shifter to generate the output steams.

half\_tb\_bits\_r is 32 bits left-shifter: It’s used when the output length is 32.

full\_tb\_bits\_r is 64 bits left-shifter. It’s used only at the end of decoding decoding.

# Integration Requirement

## 5.1 Timing

First stage: 3 cycles for each soft-bits data.

Second stage: 2 cycles for each track back step.

Example. If decoding length is 96.

First sliding window : 3x64+2x64

Second sliding window: 3x32+2x64

Totally: (3x64+2x64) +( 3x32+2x64 ) = 544 cycles

## 5.2 register/bus interface

This is only a Viterbi core, which is not including the register/bus interface. During decoding, all parameter inputs should be locked, or it will cause errors.

For ping-pong operation, the src\_start\_addr\_i and dst\_start\_addr\_i should be configured depending on the different ping-pong buffers.