



Fraunhofer Institut
Integrierte Schaltungen

***CorePool* FHG_VITERBI**

Databook

- subject to change without notice -

Version: v1.5

Date: 06.08.02

Document History

Table 1: Document History

Version	Date	Responsible	Description
V 1.0	14.04.98	Leyh	Databook created
V 1.1	12.05.98	Leyh	Update
V 1.2	10.07.98	Leyh	Corrections
V 1.3	03.11.98	Leyh/Shn	Update
V 1.4	03.05.99	Leyh	Update
V 1.5	27.07.01	Leyh	Update

Contact

CorePool
Fraunhofer Institute Integrated Circuits

Am Wolfsmantel 33
91058 Erlangen
Germany

Phone: +49 (0) 9131 776 777
Fax: +49 (0) 9131 776 499
Email: info@corepool.com
Internet: <http://www.corepool.com>

Table of Contents

Document History	2
Contact	2
CorePool Library FHG_VITERBI	7
References	7
FHG_VITERBI_PAR	9
Purpose	9
Features	9
Design Kit	9
Requirements	9
Block Diagram	10
General Information	10
Signal Description	10
Parameter Description	11
Interfaces	12
VHDL Usage through Component Instantiation	13
Waveforms and Timing Tables	14
Application Note	15
FHG_VITERBI_SER	17
Purpose	17
Features	17
Design Kit	17
Requirements	17
Block Diagram	18
General Information	19
Signal Description	19
Parameter Description	21
Interfaces	23
Clocking of the FHG_VITERBI_SER	23
VHDL Usage through Component Instantiation	24
Waveforms and Timing Tables	26
Application Note	30

FHG_VITERBI_PIPE	33
Purpose	33
Features	33
Design Kit	33
Requirements	33
Block Diagram	34
General Information	35
Signal Description	36
Parameter Description	38
Interfaces	40
Clocking of the FHG_VITERBI_PIPE	40
VHDL Usage through Component Instantiation	41
Waveforms and Timing Tables	43
Application Note	47
FHG_VITERBI_SP	49
Purpose	49
Features	49
Design Kit	49
Requirements	49
Block Diagram	50
General Information	51
Signal Description	52
Parameter Description	54
Interfaces	56
Clocking of the FHG_VITERBI_SP	56
VHDL Usage through Component Instantiation	57
Waveforms and Timing Tables	59
Application Note	63

List of Tables and Figures

Document History	2
FHG_VITERBI_PAR	9
Block Diagram	10
Signal Description	10
Parameter Description	11
Waveforms of external data and control signals.	14
Timing table of internal data memory read and write accesses.	14
Application example	15
FHG_VITERBI_SER	17
Block Diagram	18
Signal Description	19
Parameter Description	21
Initialization	26
Timing Table of Initialization Process	26
Soft-Input Synchronization	27
Timing table of Soft-Input Synchronization	27
Waveform of external write and read access of path metric memory	28
Timing table of an external access of path metric memory	28
Waveform of external read and write acces of external survivor memory	29
Timing table of an external access of survivor memory	29
Application example	30
FHG_VITERBI_PIPE	33
Block Diagram	34
Signal Description	36
Parameter Description	38
Initialization	43
Timing Table of Initialization Process	43
Soft-Input Synchronization	44

Timing table of Soft-Input Synchronization	44
Waveform of external write and read access for path metric memories	45
Timing table of an external data read access	45
Waveform of external read and write acces of external survivor memory	46
Timing table of an external access of survivor memory	46
Application example	47
FHG_VITERBI_SP	49
Block Diagram	50
Signal Description	52
Parameter Description	54
Initialization	59
Timing Table of Initialization Process	59
Soft-Input Synchronization	60
Timing table of Soft-Input Synchronization	60
Waveform of external write and read access of path metric memory	61
Timing table of an external access of path metric memory	61
Waveform of external read and write acces of survivor memory	62
Timing table of an external access of survivor memory	62
Application example	63

1. CorePool Library FHG_VITERBI

The CorePool library FHG_VITERBI comprises four different parameterizable and synthesizable Viterbi Decoder architectures for convolutional code rates $R_c = 1/n$.

Each architecture is suitable for a certain range of symbol rates. For all Viterbi Decoder architectures the tunable parameters are the generator polynomials, the convolutional code rate, the wordlength of the soft-input words, and the survivor length.

The FHG_VITERBI_PAR is a fully parallel Viterbi Decoder architecture allowing high data throughput (page 9).

The FHG_VITERBI_SER (page 17) and the FHG_VITERBI_PIPE (page 33) are serial and serial-pipelined Viterbi decoders suitable for low symbol rates. They are both characterized by small chip areas.

In the FHG_VITERBI_SP (page 49) the number of parallel Add-Compare-Select (ACS) calculators is scalable with an additional parameter. Thus, this architecture can be tailored to the desired symbol rate at an optimized chip area.

References

For more detailed information about the FHG_VITERBI refer to standard literature about the Viterbi Algorithm and Viterbi decoders (e. g. Viterbi & Omura: Principles of Digital Communication and Coding, McGraw-Hill, 1985).

2. FHG_VITERBI_PAR

Purpose

The FHG_VITERBI_PAR is a fully parameterized and synthesizable rate-1/n Viterbi decoder for high data rates. Its parameter set (constraint length, code rate, generator polynomials, number of soft bits, survivor length) enables the designer to implement Viterbi decoders for any rate-1/n convolutional code and thus provides versatility for the design of corresponding applications.

Features

- Fully Parametrized Viterbi Decoder for Rate-1/n Convolutional Codes
- Fully Synthesizable
- Constraint Length, Generator Polynomials, and Code Rate parameterized (optional inversion of code bits)
- Survivor Length of Viterbi Decoder parameterized
- Wordlength of Soft-Input parameterized
- Soft-Input in Offset-Binary and Two's Complement Format
- Up to 40 MSymbols/sec Decoding Rate (Constraint Length $K=7$, Code Rate $R_c=1/2$, $0.8\mu\text{m}$ Technology)
- Registered Inputs and Outputs

Design Kit

- Technology Independent Implementation as Synopsys Design Ware Components
- VHDL/Verilog Source Code Simulation Models
- VHDL/Verilog Compliance Test Suite
- Auxiliary Simulation Models for User Testbenches
- Synthesis and Testsynthesis Scripts
- Example Design and Testchip for $K=7$, $R_c=1/2$ convolutional code available
- Design Support, Netlist Synthesis Service, and Consulting available

Requirements

Simulation

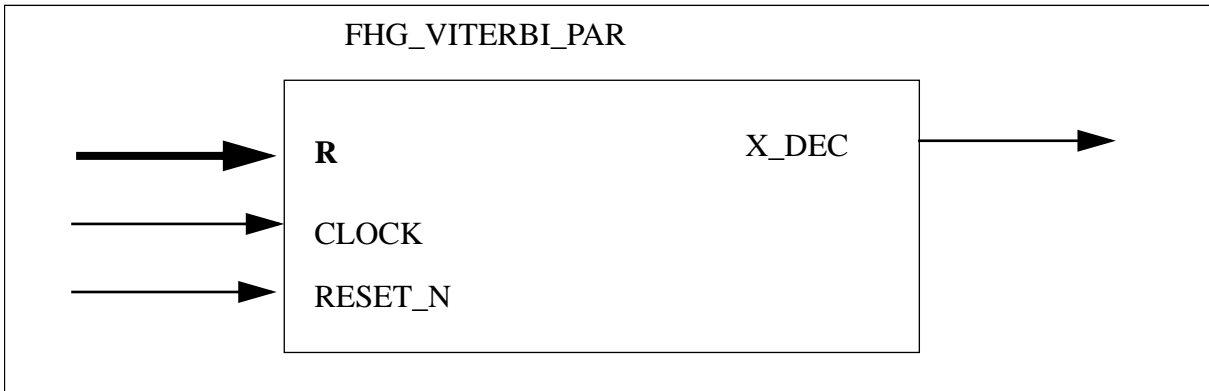
- VHDL IEEE-1076 Simulator
- Verilog IEEE-1364 Simulator

Synthesis

- Synopsys Design Compiler

Block Diagram

Figure 1: Block Diagram



General Information

It is assumed that the soft-quantized bits for all code bits are applied parallelly to the input of the Viterbi decoder (as is the case e.g. for QPSK). If this is not the case, a serial-to-parallel conversion of the received data is required.

Signal Description

Table 2: Signal Description

Pin	Direction	Bitwidth	Description
CLOCK	IN	1	Symbol Clock
RESET_N	IN	1	Low active asynchronous reset.
R	IN	CODE_RATE *SOFT_BITS	Soft Input: CODE_RATE code bits ($R_c=1/\text{CODE_RATE}$) each quantized with SOFT_BITS bits in Offset Binary Format
X_DEC	OUT	1	Decoded Bits

The input port CLOCK is the symbol clock. The code symbols are applied parallelly to the input of the Viterbi Decoder with the corresponding code symbol rate.

The input port RESET_N is a low active asynchronous reset for the FHG_VITERBI_PAR. An asynchronous reset should be performed at the beginning of the decoding process.

The input port R consists of the soft-input data. Each of the CODE_RATE code bits is quantized with SOFT_BITS bit in Offset-Binary Format. The input code bits are numbered from code bit CODE_RATE-1 down to code bit 0 each starting with the MSB and ending with the LSB.

The output port X_DEC of the FHG_VITERBI_PAR provides the decoded bit stream calculated by the Viterbi decoder. Every clock cycle one bit is decoded.

Parameter Description

Table 3: Parameter Description

Name	Type	Default Value	Value Range	Description
CONSTRAINT_LENGTH	Integer	none		Constraint Length
CODE_RATE	Integer	none		Coderate $R_c = 1/\text{CODE_RATE}$
SOFT_BITS	Integer	none		Number of Soft-Input Bits per Code Bit
SURVIVOR_LENGTH	Integer	none		Survivor Length
GEN_POLY	Integer	none		Generator Polynomials $\text{GEN_POLY} = (g_{\text{CODE_RATE}-1}, \dots, g_0)_2$ $g_i, i=0, \dots, \text{CODE_RATE}-1$
INVERT	Integer	none		Optional Inversion of Code Bits $\text{INVERT} = (\text{inv}_{\text{CODE_RATE}-1}, \dots, \text{inv}_0)_2$ $\text{inv}_i, i=0, \dots, \text{CODE_RATE}-1$

The parameter CONSTRAINT_LENGTH is the constraint length of the convolutional code. Its value can be any convenient length.

The parameter CODE_RATE is the number of code bits generated for each information bit by the convolutional encoder resulting in a code rate $R_c=1/\text{CODE_RATE}$.

The parameter SOFT_BITS is the number of soft-quantized bits for each code bit. Usually, one (hard decision) to three bits (soft decision) are used for quantization dependent on the desired bit-error rate (BER) which should be achieved under the condition of a given SNR per bit of the communication channel.

The parameter SURVIVOR_LENGTH characterizes the survivor length used by the Viterbi decoder. As a rule of thumb, the survivor length should be chosen four to six times the constraint length ($\text{SURVIVOR_LENGTH} = (4..6) \cdot \text{CONSTRAINT_LENGTH}$). The larger the survivor length is chosen, the lower is the resulting BER.

The generator polynomials of the convolutional code are given by GEN_POLY. The generators for each of the CODE_RATE code bits given in binary form (CONSTRAINT_LENGTH bits per generator) are put in one bit vector. Thereby, the generator polynomials are numbered from code bit CODE_RATE-1 down to code bit 0. GEN_POLY is the resulting bit vector of length CODE_RATE*CONSTRAINT_LENGTH converted to integer.

The parameter INVERT allows an optional inversion of the code bits generated by the convolutional encoder. If an inversion of one of the CODE_RATE code bits is desired, the corresponding bit in INVERT has to be set to 1, otherwise 0 has to be chosen. The bits are numbered from code bit CODE_RATE-1 down to code bit 0. INVERT is the resulting bit vector of length CODE_RATE converted to integer.

In the section "Application Note" (p. 15) an example is given how the parameter set has to be chosen for a special application.

Interfaces

The control interface of the FHG_VITERBI_PAR consists of the systemclock CLOCK and the low active asynchronous reset input RESET_N. The data interface consists of the soft-quantized data input R and the decoded bit stream X_DEC.

VHDL Usage through Component Instantiation

```
library IEEE, FHG_VITERBIDW;
use IEEE.std_logic_1164.all;
use FHG_VITERBIDW.FHG_VITERBI.all;

entity MY_VITERBI is
  port ( CLOCK      : in std_logic;
        RESET_N    : in std_logic;
        R           : in std_logic_vector(5 downto 0);
        X_DEC      : out std_logic
        );
end MY_VITERBI;

architecture MY_VITERBI_RTL of MY_VITERBI is

  constant GEN_POLY          : integer := 253;
  constant INVERT            : integer := 2;
  constant CONSTRAINT_LENGTH : integer := 4;
  constant CODE_RATE        : integer := 2;
  constant SOFT_BITS        : integer := 3;
  constant SURVIVOR_LENGTH  : integer := 20;

begin

  MY_VITERBI_CORE : FHG_VITERBI_PAR
    generic map ( GEN_POLY          => GEN_POLY,
                  INVERT            => INVERT,
                  CONSTRAINT_LENGTH => CONSTRAINT_LENGTH,
                  CODE_RATE         => CODE_RATE,
                  SOFT_BITS         => SOFT_BITS,
                  SURVIVOR_LENGTH   => SURVIVOR_LENGTH
                )
    port map ( CLOCK      => CLOCK,
              RESET_N    => RESET_N,
              R           => R,
              X_DEC      => X_DEC
            );

end MY_VITERBI_RTL;
```

Waveforms and Timing Tables

Figure 2: Waveforms of external data and control signals.

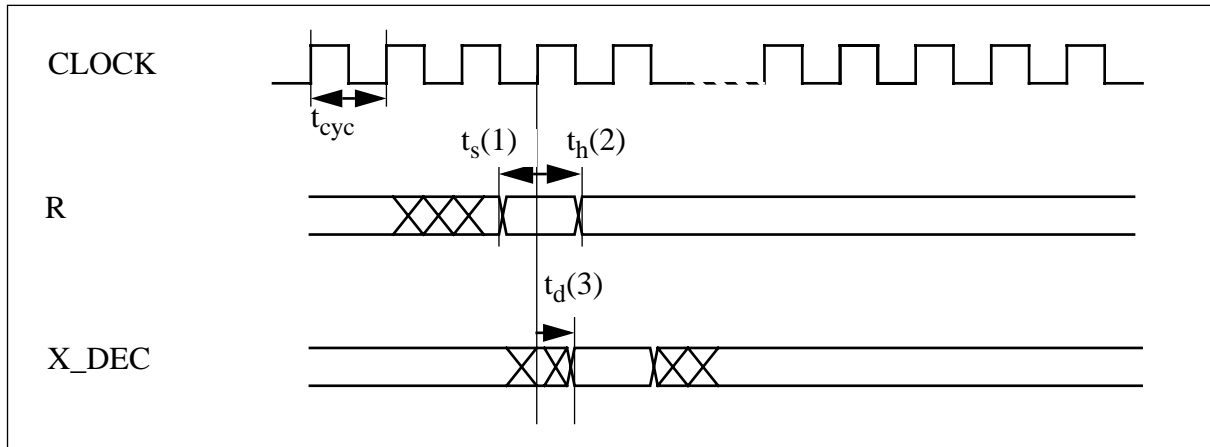


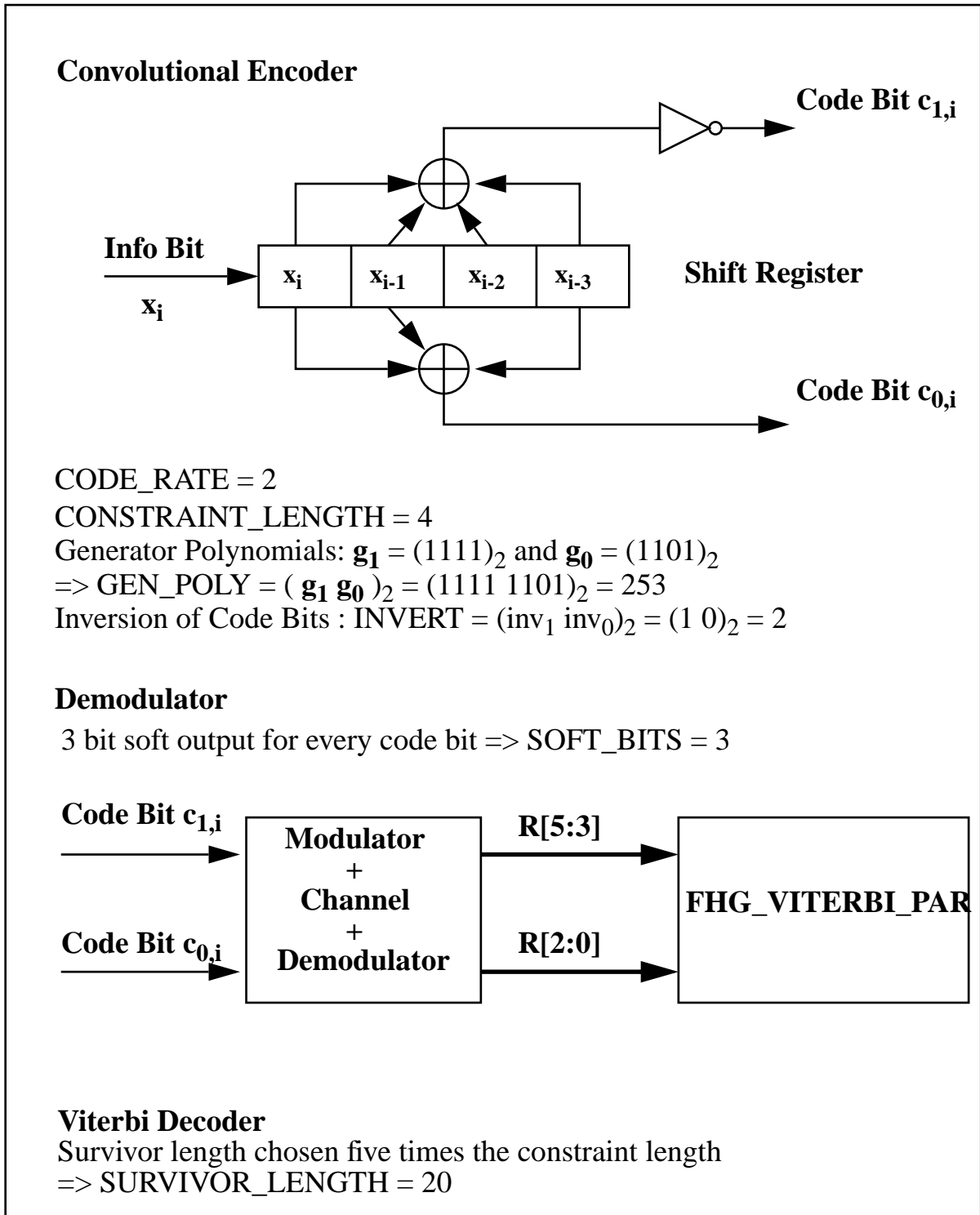
Table 4: Timing table of internal data memory read and write accesses.

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{rise}-\text{CLOCK}_{rise})$	Clock cycle				ns
$t_s(1)$	$t_s(\text{R}-\text{CLOCK}_{rise})$	Setup time for R to rising clock edge				ns
$t_h(2)$	$t_h(\text{CLOCK}_{rise}-\text{R})$	Hold time for R from rising clock edge				ns
$t_d(3)$	$t_d(\text{CLOCK}_{rise}-\text{X_DEC}_{valid})$	Delay time for X_DEC from rising clock edge				ns

Application Note

The following example shows the parameters of the FHG_VITERBI_PAR for a special application. This parameter set is also used in section "VHDL Usage through Component Instantiation", p. 13.

Figure 3: Application example



3. FHG_VITERBI_SER

Purpose

The FHG_VITERBI_SER is a fully parameterized and synthesizable rate-1/n Viterbi decoder for low data rates. Its parameter set (constraint length, code rate, generator polynomials, number of soft bits, survivor length) enables the designer to implement Viterbi decoders for any rate-1/n convolutional code and thus provides versatility for the design of corresponding applications.

Features

- Fully Parametrized Viterbi Decoder for Rate-1/n Convolutional Codes
- Fully Synthesizable
- Constraint Length, Generator Polynomials, and Code Rate parameterized (optional inversion of code bits)
- Survivor Length of Viterbi Decoder parameterized
- Wordlength of Soft-Input parameterized
- Soft Input in Offset-Binary and Two's Complement Format
- Up to 400 kilosymbols/sec decoding rate (Constraint Length $K=7$, Code Rate $R_c=1/2$, 0.8 μ Technology)
- Registered Inputs and Outputs
- External RAMs for Path Metric Memory and Survivor Memory Unit required

Design Kit

- Technology Independent Implementation as Synopsys Design Ware Components
- VHDL/Verilog Source Code Simulation Models
- VHDL/Verilog Compliance Test Suite
- Auxiliary Simulation Models for User Testbenches
- Synthesis and Testsynthesis Scripts
- Design Support, Netlist Synthesis Service, and Consulting available

Requirements

Simulation

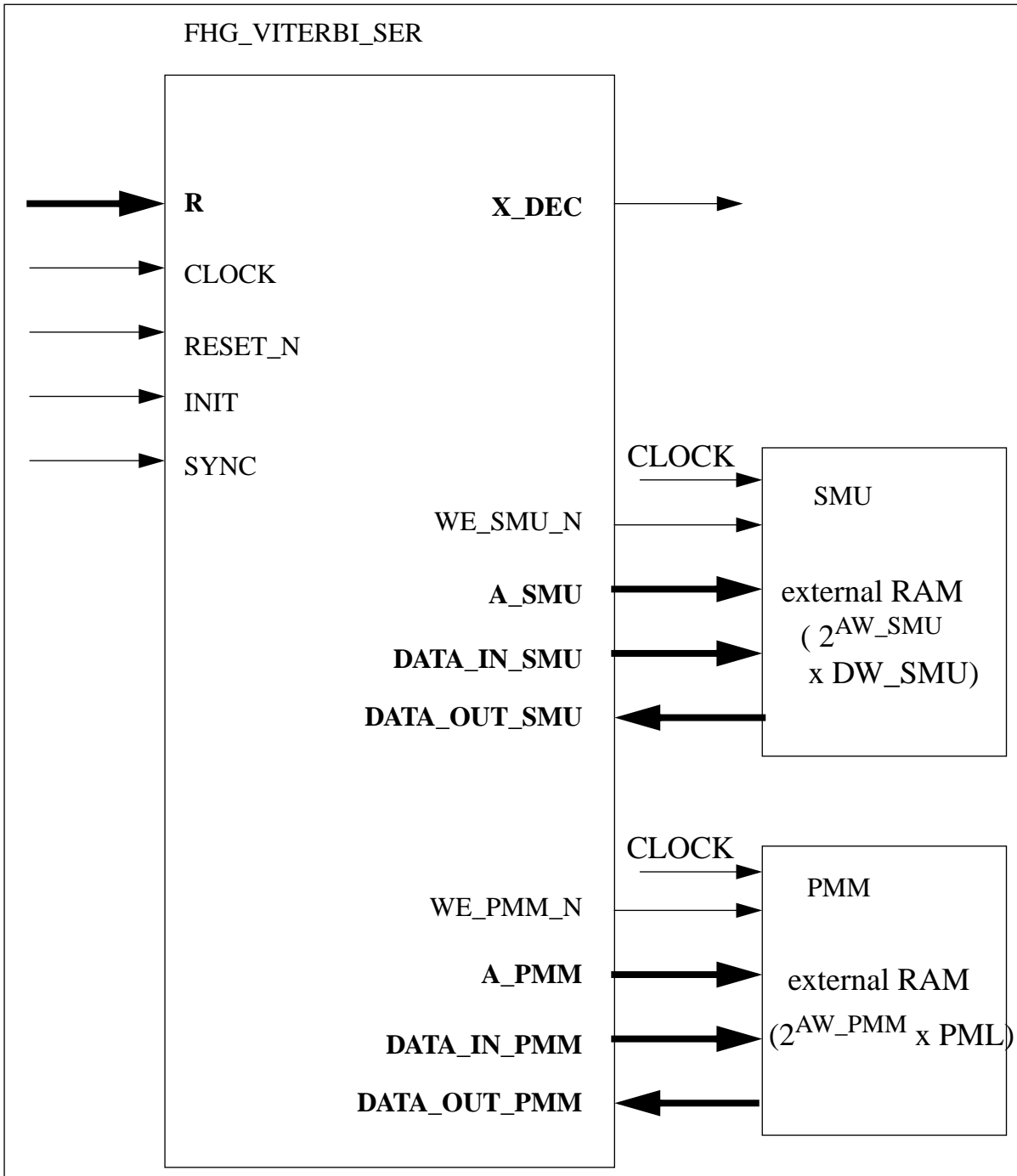
- VHDL IEEE-1076 Simulator
- Verilog IEEE-1364 Simulator

Synthesis

- Synopsys Design Compiler

Block Diagram

Figure 4: Block Diagram



General Information

The FHG_VITERBI_SER and its environment is a fully synchronous design. Therefore, all blocks of the design (FHG_VITERBI_SER, SMU, PMM) have to be provided with the same CLOCK signal. It is assumed that the the soft-quantized bits for all code bits are applied parallelly to the input of the Viterbi decoder (as is the case e.g. for QPSK). If this is not the case, a serial-to-parallel conversion of the received data is required.

The memories for the path metrics (Path Metric Memory PMM) and the survivor paths (Survivor Memory Unit SMU) have to be implemented as external RAMs with corresponding memory sizes and datawidths. PMM and SMU have to be synchronous Single-Port-RAMs with a clock input, a low active write enable input, and an address input, i. e. if the write enable signals are low, the input data are stored with the next rising edge of the clock. Formulas for the necessary parameters (bitwidths, datawidths) are given in section Parameter Description.

Signal Description

Table 5: Signal Description

Pin	Direction	Bitwidth	Description
CLOCK	IN	1	Decoder Clock
RESET_N	IN	1	Low active asynchronous reset
INIT	IN	1	Initialization signal
SYNC	IN	1	Synchronization signal for a new code symbol (symbol clock)
R	IN	CODE_RATE *SOFT_BITS	Soft Input: CODE_RATE code bits ($R_c = 1/\text{CODE_RATE}$) each quantized with SOFT_BITS bits in Offset-Binary Format
DATA_OUT_PMM	IN	PML	Path Metric from Path Metric Memory PMM
DATA_OUT_SMU	IN	DW_SMU	Output Data of Survivor Memory Unit (SMU)
WE_PMM_N	OUT	1	Low active write enable for PMM
A_PMM	OUT	AW_PMM	Address for PMM
DATA_IN_PMM	OUT	PML	Path Metric to be stored in PMM
WE_SMU_N	OUT	1	Low active write enable for SMU
A_SMU	OUT	AW_SMU	Address for SMU
DATA_IN_SMU	OUT	DW_SMU	Input data for SMU

The input port **CLOCK** is the decoder system clock. The necessary clock frequency is derived in section Parameter Description and depends on the chosen code parameters and the codesymbol rate.

The input port **RESET_N** is a low active asynchronous reset for the **FHG_VITERBI_SER**. An asynchronous reset should be performed at the beginning of the decoding process.

The input port **INIT** is a high active initialization signal. It initializes the path metric memory **PMM** what should be done at least once at the beginning of the decoding process. For this purpose the signal **INIT** has to be high for at least one clock cycle. The initialization then starts with the first rising edge of the **CLOCK** signal (see timing diagram in section Waveforms and Timing Tables).

The input port **SYNC** is the synchronization signal for the code symbols. If **SYNC** is high, this indicates new valid soft-input bits **R** for a new code symbol. This signal can be e. g. the clock of the analog-to-digital converter at the demodulator output (see timing diagram in section Waveforms and Timing Tables).

The input port **R** consists of the soft-input data. Each of the **CODE_RATE** code bits is quantized with **SOFT_BITS** bit in Offset-Binary Format. The input code bits are numbered from code bit **CODE_RATE-1** down to code bit **0** each starting with the **MSB** and ending with the **LSB**.

The input port **DATA_OUT_PMM** reads the path metrics of bitwidth **PML** from the Path Metric Memory **PMM**, whereas the output port **DATA_IN_PMM** contains the corresponding new path metrics to be stored. The write and read operations of **PMM** are controlled by the write enable signal **WE_PMM_N** and the address signal **A_PMM** (bitwidth **AW_PMM**). **WE_PMM_N** is low active, i. e. if **WE_PMM_N** is low, the data at the input of the path metric memory (**DATA_IN_PMM**) are written with the next rising edge of the **CLOCK** signal (see timing diagram in section Waveforms and Timing Tables).

The input port **DATA_OUT_SMU** (bitwidth **DW_SMU**) is the data output of the Survivor Memory Unit **SMU** corresponding to the address **A_SMU** (bitwidth **AW_SMU**). The write and read operations for the **SMU** are controlled by the low active write enable **WE_SMU_N** and the address **A_W_SMU** (bitwidth **AW_SMU**), i. e. if **WE_SMU_N** is low, the input data **DATA_IN_SMU** (bitwidth **DW_SMU**) for the **SMU** are written with the rising edge of **CLOCK** into the address **A_SMU** (see timing diagram in section Waveforms and Timing Tables).

The output port **X_DEC** of the **FHG_VITERBI_SER** provides the decoded bit stream calculated by the Viterbi decoder. One bit is decoded every decoding cycle. The number of clock cycles per decoding cycle is derived in section Clocking of the **FHG_VITERBI_SER**.

Parameter Description

Table 6: Parameter Description

Name	Type	Default Value	Value Range	Description
CONSTRAINT_LENGTH	Integer	none		Constraint Length
CODE_RATE	Integer	none		Code rate $R_c = 1/\text{CODE_RATE}$
SOFT_BITS	Integer	none		Number of Soft-Input Bits per Code Bit
SURVIVOR_LENGTH	Integer	none		Survivor Length
GEN_POLY	Integer	none		Generatorpolynomials $\text{GEN_POLY} = (g_{\text{CODE_RATE}-1}, \dots, g_0)_2$ $g_i, i=0, \dots, \text{CODE_RATE}-1$
INVERT	Integer	none		Optional Inversion of Code Bits $\text{INVERT} = (\text{inv}_{\text{CODE_RATE}-1}, \dots, \text{inv}_0)_2$ $\text{inv}_i, i = 0, \dots, \text{CODE_RATE}-1$
AW_PMM	Integer	none		Address bits for Path Metric Memories (PMMs)
PML	Integer	none		Bitwidth of Path Metric (Path Metric Length)
AW_SMU	Integer	none		Address bits for Survivor Memory Unit (SMU)
DW_SMU	Integer	none		Bitwidth of SMU Data

The parameter CONSTRAINT_LENGTH is the constraint length of the convolutional code. Its value can be any convenient length.

The parameter CODE_RATE is the number of code bits generated for each information bit by the convolutional encoder resulting in a code rate $R_c=1/\text{CODE_RATE}$.

The parameter SOFT_BITS is the number of soft-quantized bits provided by the demodulator output for each code bit. Usually, one (hard decision) to three bits (soft decision) are used for quantization dependent on the desired bit-error rate (BER) which should be achieved under the condition of a given SNR per bit of the communication channel

The parameter SURVIVOR_LENGTH characterizes the survivor length used by the Viterbi decoder. As a rule of thumb, the survivor length should be chosen four to six times the constraint length ($\text{SURVIVOR_LENGTH} = (4..6) * \text{CONSTRAINT_LENGTH}$). The larger the survivor length is chosen the smaller is the resulting BER.

The generator polynomials of the convolutional code are given by GEN_POLY. The generators for each of the CODE_RATE code bits given in binary form (CONSTRAINT_LENGTH bits per generator) are put in one bit vector. Thereby, the generator polynomials are numbered from code bit CODE_RATE-1 down to code bit 0. GEN_POLY is the resulting bit vector of length CODE_RATE*CONSTRAINT_LENGTH converted to integer.

The parameter INVERT allows an optional inversion of the code bits generated by the convolutional encoder. If an inversion of one of the CODE_RATE code bits is desired, the corresponding bit in INVERT has to be set to 1, otherwise 0 has to be chosen. The bits are numbered from code bit CODE_RATE-1 down to code bit 0. INVERT is the resulting bit vector of length CODE_RATE converted to integer.

The number of address bits for the path metric memories is given by AW_PMM. It is calculated by

$$AW_PMM = CONSTRAINT_LENGTH - 1.$$

The path metric length PML depends on the code rate, the constraint length, and the number of soft-input bits. It can be calculated by the formulas

$$\Delta\Gamma_{max} = CODE_RATE * (CONSTRAINT_LENGTH - 1) * (2^{SOFT_BITS} - 1)$$

and

$$PML = \text{ceil}(\log_2(\Delta\Gamma_{max} + 1)) + 1$$

where $\Delta\Gamma_{max}$ denotes the maximum difference between two path metrics and $\text{ceil}(x)$ denotes the ceiling function (smallest integer value equal to or greater than x).

The address width AW_SMU of the SMU depends on the constraint length, the survivor length and the data width DW_SMU (see below) of the SMU. It is calculated by the formula

$$AW_SMU = \text{ceil}\left(\log_2\left(\frac{2^{CONSTRAINT_LENGTH-1}}{DW_SMU}\right)\right) + \text{ceil}(\log_2(SURVIVOR_LENGTH))$$

where $\text{ceil}(x)$ denotes the ceiling function (smallest integer value equal to or greater than x).

The data width of the SMU input is given by the parameter DW_SMU. DW_SMU has to be a divisor of the number of states ($= 2^{CONSTRAINT_LENGTH-1}$) of the convolutional code and an integer value greater than or equal to 2. With the choice of DW_SMU the data width of the SMU can be adapted to the requirements of the specific application.

In section Application Note an example is given how the parameter set has to be chosen for a special application.

Interfaces

The FHGVITERBI_SER has two interfaces with external memories (RAMs) for path metrics (PMM) and survivor paths (SMU).

The path metric interface consists of the low active write enable signal WE_PMM_N, the address bus A_PMM, the data input bus DATA_IN_PMM, and the data output bus DATA_OUT_PMM.

The interface to the survivor path memory consists of the low active write enable signal WE_SMU_N, the address bus A_SMU, the data input bus DATA_IN_SMU, and the data output bus DATA_OUT_SMU.

Clocking of the FHG_VITERBI_SER

The FHG_VITERBI_SER has to be provided with a CLOCK signal whose minimum clock frequency depends on the symbol rate, the constraint length of the convolutional code, the data-width for the SMU, and the survivor length. The minimum clock frequency can be derived by calculating

$$\begin{aligned}x &= 2^{\text{CONSTRAINT_LENGTH}} \\y &= \min \left\{ 2 + \text{floor} \left(\frac{\text{SURVIVOR_LENGTH} - 2 * \text{DW_SMU}}{2 * \text{DW_SMU} - 1} \right), \frac{2^{\text{CONSTRAINT_LENGTH} - 1}}{\text{DW_SMU}} \right\} \\z &= \text{SURVIVOR_LENGTH} + 2 * \text{DW_SMU} + y \\f_{min} &= \max(x, z) \times f_{symbol}\end{aligned}$$

where $\max(x, y)$ denotes the maximum of the two calculated expressions and $\text{floor}(x)$ denotes the floor-function (largest integer smaller than or equal to x). The calculated factor is the required minimum number of clock cycles for one decoding cycle. The factor $\max(x, y)$ is the number of clock cycles needed for one decoding cycle.

VHDL Usage through Component Instantiation

```
library IEEE, FHG_VITERBIDW;
use IEEE.std_logic_1164.all;
use FHG_VITERBIDW.FHG_VITERBI.all;

entity MY_VITERBI is
  port ( CLOCK          : in std_logic;
        RESET_N        : in std_logic;
        INIT            : in std_logic;
        SYNC            : in std_logic;
        R               : in std_logic_vector(5 downto 0);
        DATA_OUT_PMM  : in std_logic_vector(6 downto 0);
        DATA_OUT_SMU  : in std_logic_vector(1 downto 0);
        WE_PMM_N       : out std_logic;
        A_PMM          : out std_logic_vector(2 downto 0);
        DATA_IN_PMM   : out std_logic_vector(6 downto 0);
        WE_SMU_N       : out std_logic;
        A_SMU          : out std_logic_vector(6 downto 0);
        DATA_IN_SMU   : out std_logic_vector(1 downto 0);
        X_DEC          : out std_logic
        );
end MY_VITERBI;
```

architecture MY_VITERBI_RTL of MY_VITERBI is

```
constant GEN_POLY          : integer := 253;
constant INVERT            : integer := 2;
constant CONSTRAINT_LENGTH : integer := 4;
constant CODE_RATE        : integer := 2;
constant SOFT_BITS        : integer := 3;
constant SURVIVOR_LENGTH  : integer := 20;
constant AW_PMM           : integer := 3;
constant PML              : integer := 7;
constant AW_SMU           : integer := 7;
constant DW_SMU           : integer := 2;
```

begin

```
MY_VITERBI_CORE : FHG_VITERBI_SER
  generic map ( GEN_POLY          => GEN_POLY,
               INVERT            => INVERT,
               CONSTRAINT_LENGTH => CONSTRAINT_LENGTH,
               CODE_RATE         => CODE_RATE,
               SOFT_BITS         => SOFT_BITS,
               SURVIVOR_LENGTH   => SURVIVOR_LENGTH,
               AW_PMM            => AW_PMM,
               PML               => PML,
               AW_SMU            => AW_SMU,
               DW_SMU            => DW_SMU )
  port map ( CLOCK                => CLOCK,
            RESET_N              => RESET_N,
            INIT                 => INIT,
            SYNC                 => SYNC,
            R                    => R,
            DATA_OUT_PMM        => DATA_OUT_PMM,
            DATA_OUT_SMU        => DATA_OUT_SMU,
            WE_PMM_N             => WE_PMM_N,
            A_PMM                => A_PMM,
            DATA_IN_PMM         => DATA_IN_PMM,
            WE_SMU_N             => WE_SMU_N,
            A_SMU                => A_SMU,
            DATA_IN_SMU         => DATA_IN_SMU,
            X_DEC                => X_DEC
  );
```

end MY_VITERBI_RTL;

Waveforms and Timing Tables

Figure 5: Initialization

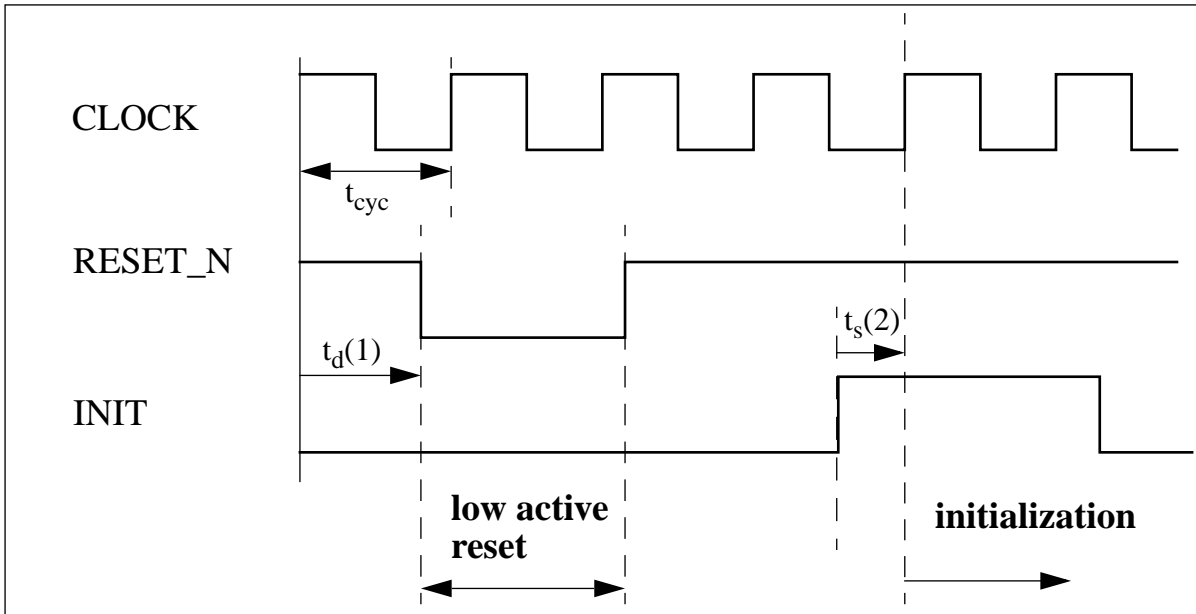


Table 7: Timing Table of Initialization Process

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{\text{rise}}-\text{RESET_N}_{\text{fall}})$	Delay time for RESET_N to rising clock edge				ns
$t_s(2)$	$t_s(\text{INIT}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Setup time for INIT to rising clock edge				ns

Figure 6: Soft-Input Synchronization

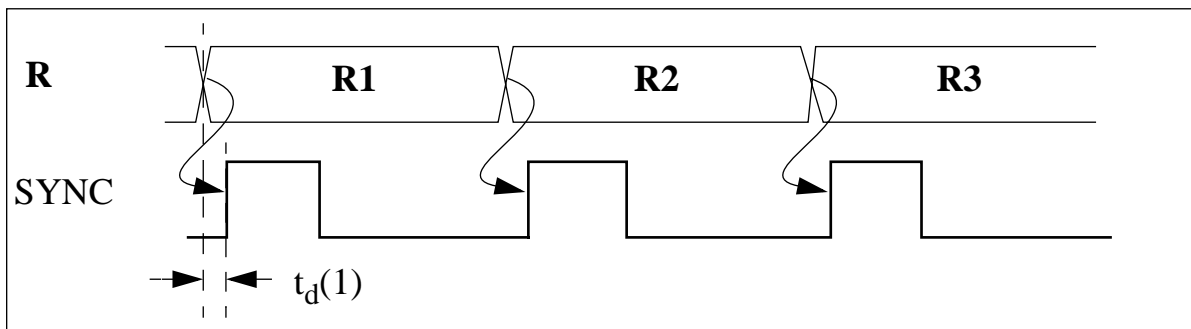


Table 8: Timing table of Soft-Input Synchronization

Symbol	Name	Item	Notes	Min	Max	Unit
t_d	$t(R_{\text{valid}}\text{-}SYNC_{\text{rise}})$	Delay time from valid soft-input R to rising SYNC signal				ns

Figure 7: Waveform of external write and read access of path metric memory

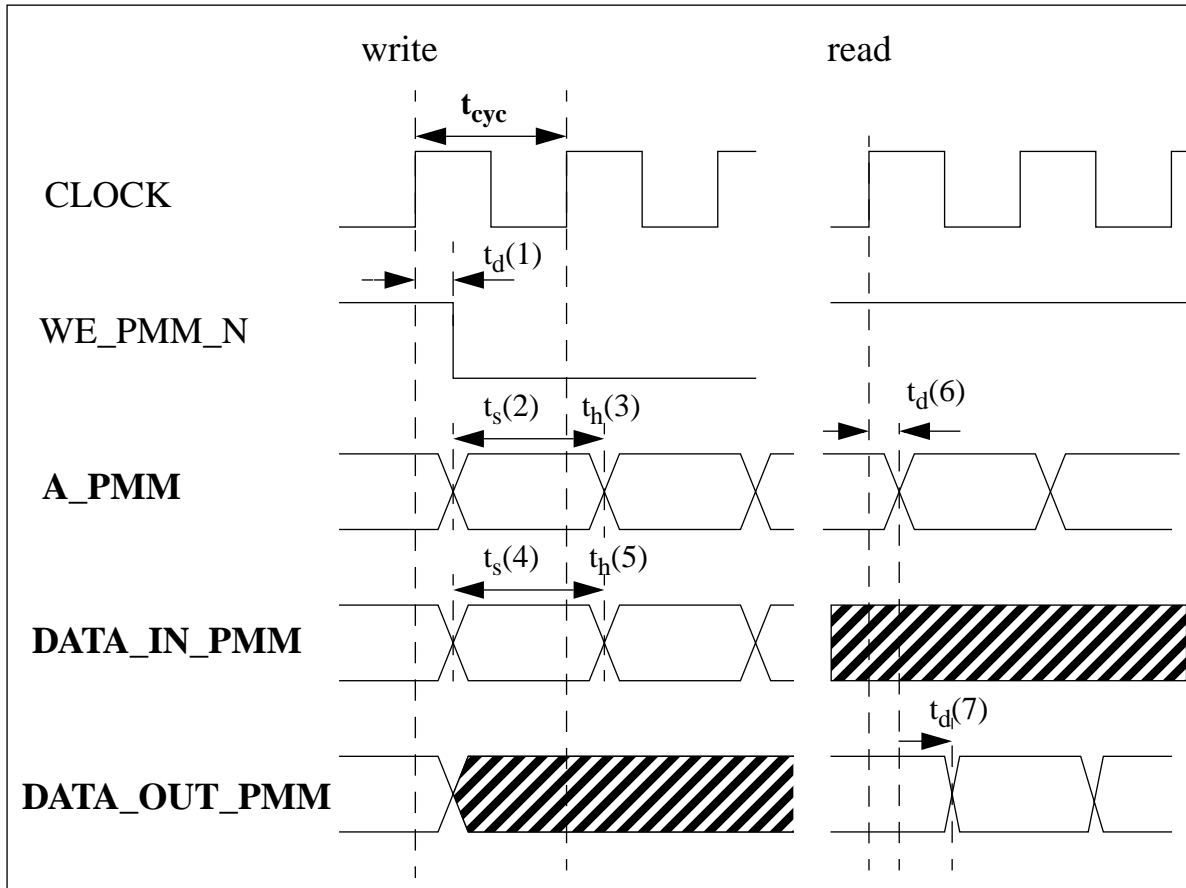


Table 9: Timing table of an external access of path metric memory

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{rise}-\text{CLOCK}_{rise})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{rise}-\text{WE_PMM_N}_{fall})$	Delay time for falling edge of WE_PMM_N from rising clock edge				ns
$t_s(2)$	$t_s(\text{A_PMM}_{valid}-\text{CLOCK}_{rise})$	Setup time for A_PMM to rising clock edge				ns
$t_h(3)$	$t_h(\text{CLOCK}_{rise}-\text{A_PMM}_{valid})$	Hold time for A_PMM				ns
$t_s(4)$	$t_s(\text{DATA_IN_PMM}_{valid}-\text{CLOCK}_{rise})$	Setup time for DATA_IN_PMM to rising clock edge				ns
$t_h(5)$	$t_h(\text{CLOCK}_{rise}-\text{A_PMM}_{valid})$	Hold time for DATA_IN_PMM				ns
$t_d(6)$	$t_d(\text{CLOCK}_{rise}-\text{A_PMM}_{valid})$	Delay time for A_PMM from rising clock edge				ns
$t_d(7)$	$t_d(\text{A_PMM}_{valid}-\text{DATA_OUT_PMM}_{valid})$	Address access time				ns

Figure 8: Waveform of external read and write accesses of external survivor memory

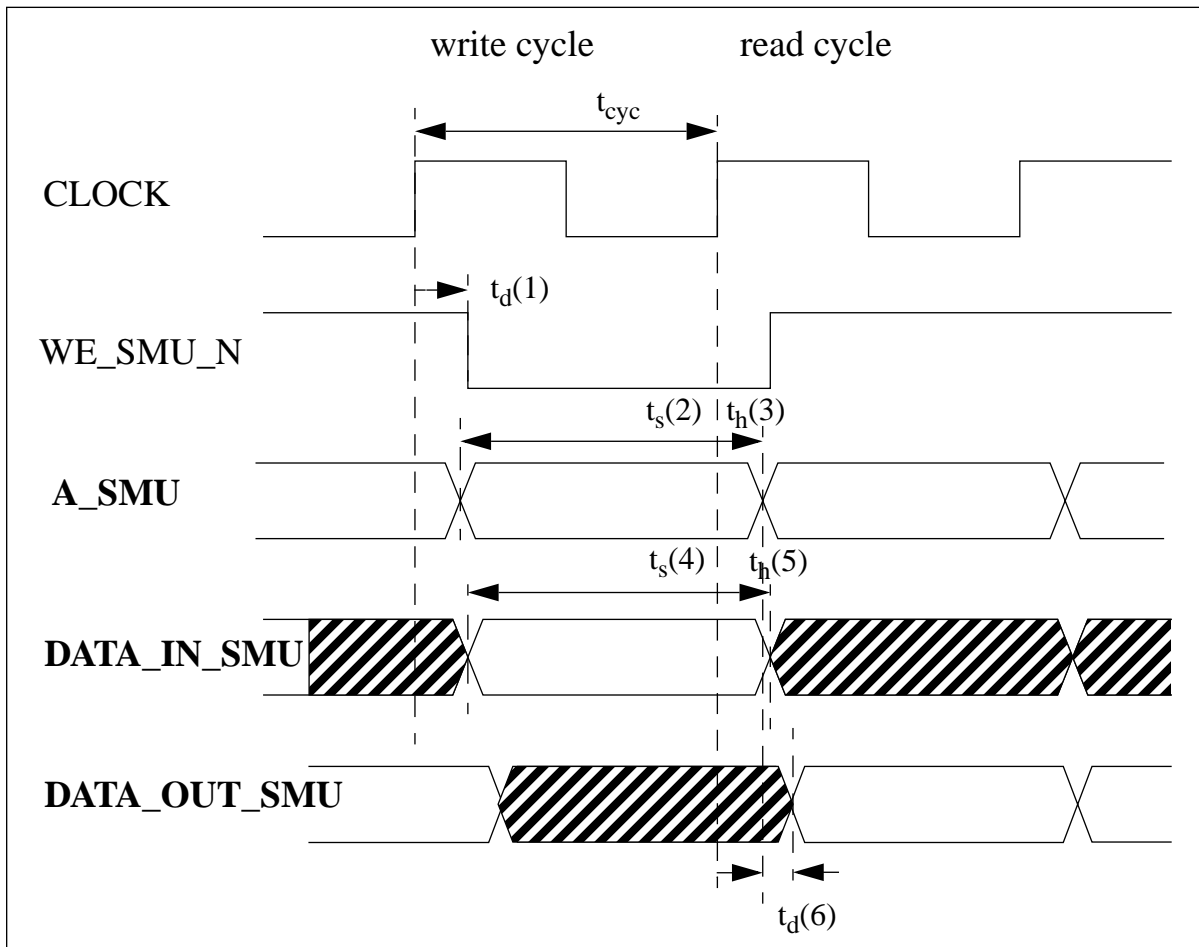


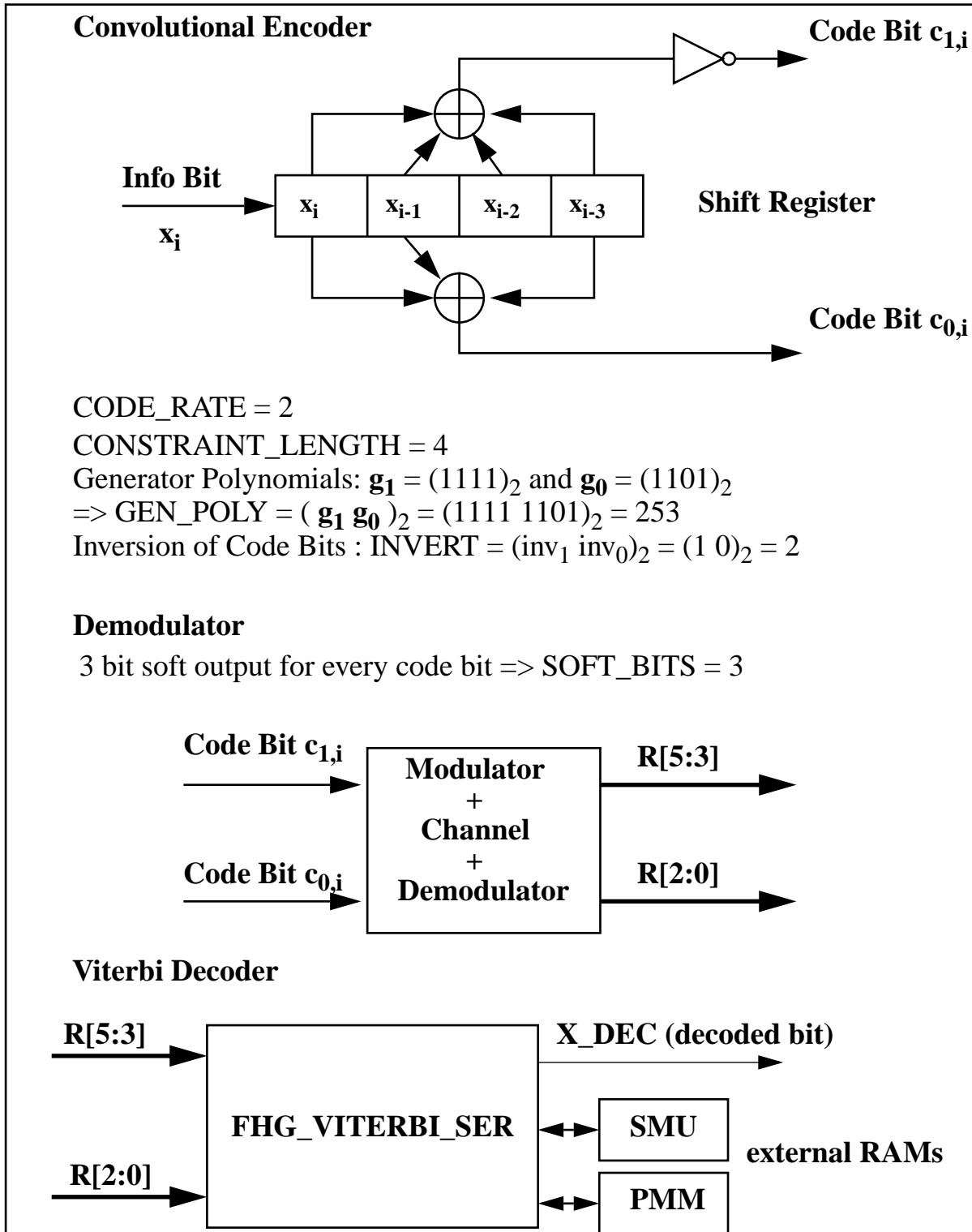
Table 10: Timing table of an external access of survivor memory

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{\text{rise}}-\text{WE_SMU_N}_{\text{fall}})$	Delay time falling edge of WE_SMU_N from rising clock edge				ns
$t_s(2)$	$t_s(\text{A_SMU}_{\text{valid}}-\text{CLOCK}_{\text{rise}})$	Setup time for A_SMU to rising clock edge				ns
$t_h(3)$	$t_h(\text{CLOCK}_{\text{rise}}-\text{A_SMU}_{\text{valid}})$	Hold time for A_SMU				ns
$t_s(4)$	$t_s(\text{DATA_IN_SMU}_{\text{valid}}-\text{CLOCK}_{\text{rise}})$	Setup time for DATA_IN_SMU to rising clock edge				ns
$t_h(5)$	$t_h(\text{CLOCK}_{\text{rise}}-\text{A_SMU}_{\text{valid}})$	Hold time for DATA_IN_SMU				ns
$t_d(6)$	$t_d(\text{A_SMU}_{\text{valid}}-\text{DATA_OUT_PMM}_{\text{valid}})$	Address access time				ns

Application Note

The following example shows the parameters of the FHG_VITERBI_SER for a special application. This parameter set is used in section VHDL Usage through Component Instantiation

Figure 9: Application example



(Application example continued)

$$AW_PMM = 3$$

$$PML = 7$$

number of states = 8 => choose e.g. $DW_SMU = 2$

$$SURVIVOR_LENGTH = 5 * CONSTRAINT_LENGTH = 20$$

$$\Rightarrow AW_SMU = 7$$

Minimum clock frequency: $x = 2^4 = 16$; $y = \min(7, 4) = 4$

$$f_{\min} = \max\{16; 20+4+4\} * f_{\text{symbol}} = 28 * f_{\text{symbol}}$$

4. FHG_VITERBI_PIPE

Purpose

The FHG_VITERBI_PIPE is a fully parameterized and synthesizable rate-1/n Viterbi decoder for low data rates. Its parameter set (constraint length, code rate, generator polynomials, number of soft bits, survivor length) enables the designer to implement Viterbi decoders for any rate-1/n convolutional code with constraint lengths $K \geq 5$ and thus provides versatility for the design of corresponding applications.

Features

- Fully Parametrized Viterbi Decoder for Rate-1/n Convolutional Codes
- Fully Synthesizable
- Constraint Length, Generator Polynomials, and Code Rate parameterized (optional inversion of code bits)
- Survivor Length of Viterbi Decoder parameterized
- Wordlength of Soft-Input parameterized
- Soft Input in Offset-Binary and Two's Complement Format
- Up to 700 kilosymbols/sec decoding rate (Constraint Length $K=7$, Code Rate $R_c=1/2$, 0.8 μ Technology)
- Registered Inputs and Outputs
- External RAMs for Path Metric Memory and Survivor Memory Unit required

Design Kit

- Technology Independent Implementation as Synopsys Design Ware Components
- VHDL/Verilog Source Code Simulation Models
- VHDL/Verilog Compliance Test Suite
- Auxiliary Simulation Models for User Testbenches
- Synthesis and Testsynthesis Scripts
- Design Support, Netlist Synthesis Service, and Consulting available

Requirements

Simulation

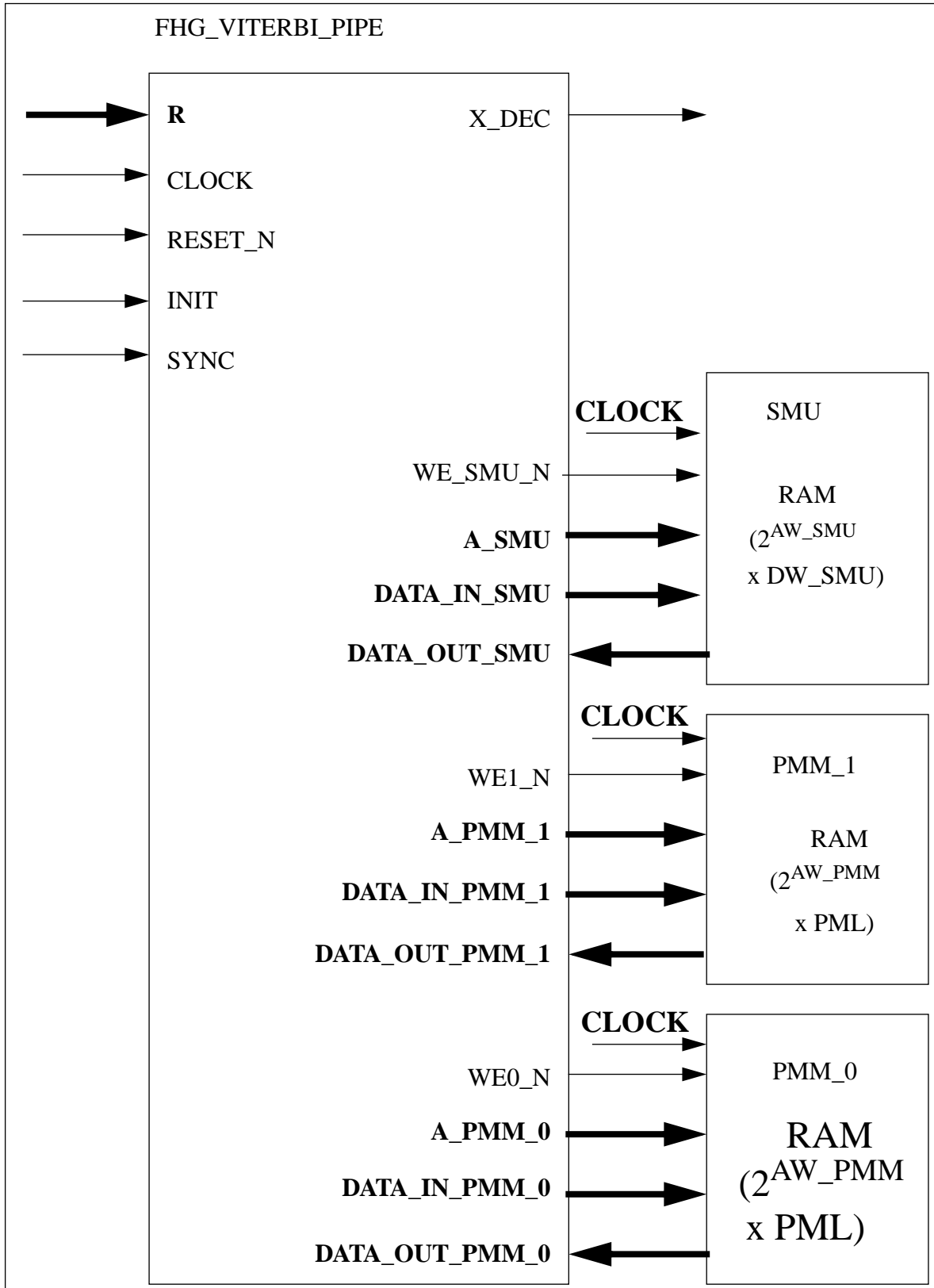
- VHDL IEEE-1076 Simulator
- Verilog IEEE-1364 Simulator

Synthesis

- Synopsys Design Compiler

Block Diagram

Figure 10: Block Diagram



General Information

The FHG_VITERBI_PIPE and its environment is a *fully synchronous design*. Therefore, all blocks of the design (FHG_VITERBI_PIPE, SMU, PMM_0, PMM_1) have to be provided with the same CLOCK signal. It is assumed that the soft-quantized bits for all code bits are applied parallelly to the input of the Viterbi decoder (as is the case e.g. for QPSK). If this is not the case, a serial-to-parallel conversion of the received data is required.

The memories for path metrics (Path Metric Memories PMM_0 and PMM_1) and survivor paths (Survivor Memory Unit SMU) have to be implemented as external RAMs with corresponding memory sizes and datawidths. PMM_0, PMM_1, and SMU have to be synchronous Single-Port-RAMs with a clock input, a low active write enable input, and an address input, i. e. if the write enable signals are low, the input data are stored with the next rising edge of the clock. Formulas for the necessary parameters (bitwidths, datawidths) are given in section Parameter Description.

Signal Description

Table 11: Signal Description

Pin	Direction	Bitwidth	Description
CLOCK	IN	1	Decoder Clock
RESET_N	IN	1	Low active asynchronous reset
INIT	IN	1	Initialization signal
SYNC	IN	1	Synchronization signal for a new code symbol
R	IN	CODE_RATE *SOFT_BITS	Soft Input: CODE_RATE code bits ($R_c = 1/\text{CODE_RATE}$) each quantized with SOFT_BITS bits in Offset Binary Format
DATA_OUT_PMM_0	IN	PML	Path Metric (PML bits) from Path Metric Memory PMM_0
DATA_OUT_PMM_1	IN	PML	Path Metric (PML bits) from Path Metric Memory PMM_1
DATA_OUT_SMU	IN	DW_SMU	Output Data from Survivor Memory Unit (SMU)
WE0_N	OUT	1	Low active write enable for PMM_0
A_PMM_0	OUT	AW_PMM	Address for PMM_0
DATA_IN_PMM_0	OUT	PML	Path Metric (PML bit) to be stored in PMM_0
WE1_N	OUT	1	Low active write enable for PMM_1
A_PMM_1	OUT	AW_PMM	Address for PMM_1
DATA_IN_PMM_1	OUT	PML	Path Metric (PML bit) to be stored in PMM_1
WE_SMU_N	OUT	1	Low active write enable for SMU
A_SMU	OUT	AW_SMU	Address for SMU
DATA_IN_SMU	OUT	DW_SMU	Input data for SMU

The input port CLOCK is the decoder system clock. The necessary clock frequency is derived in section Parameter Description and depends on the chosen code parameters and the codesymbol rate.

The input port `RESET_N` is a low active asynchronous reset for the `FHG_VITERBI_PIPE`. An asynchronous reset should be performed at the beginning of the decoding process.

The input port `INIT` is a high active initialization signal. It initializes the path metric memories `PMM_0` and `PMM_1` what should be done at least once at the beginning of the decoding process. For this purpose, the signal `INIT` has to be high for at least one clock cycle. The initialization then starts with the first rising edge of the `CLOCK` signal. (see timing diagram in section Waveforms and Timing Tables).

The input port `SYNC` is the synchronization signal for the code symbols. If `SYNC` is high, this indicates new valid soft-input bits `R` provided by the demodulator for a new code symbol. This signal can be e. g. the clock of the Analog-to-Digital Converter at the demodulator output (see timing diagram in section Waveforms and Timing Tables).

The input port `R` consists of the soft-input data. Each of the `CODE_RATE` code bits is quantized with `SOFT_BITS` bit in Offset-Binary Format. The input code bits are numbered from code bit `CODE_RATE-1` down to code bit `0` each starting with the MSB and ending with the LSB.

The input ports `DATA_OUT_PMM_0` and `DATA_OUT_PMM_1` are the path metrics of bitwidth PML read from the Path Metric Memories `PMM_0` and `PMM_1`, whereas the output ports `DATA_IN_PMM_0` and `DATA_IN_PMM_1` are the corresponding new path metrics to be stored. The write and read operations of `PMM_0` and `PMM_1` are controlled by the write enable signals `WE0_N` and `WE1_N` and the address signals `A_PMM_0` and `A_PMM_1` (bitwidth `AW_PMM`). `WE0_N` and `WE1_N` are low active, i. e. if `WE0_N` or `WE1_N` are low, the data at the input of the path metric memories (`DATA_IN_PMM_0` and `DATA_IN_PMM_1`) are written with the next rising edge of the `CLOCK` signal. The modes of `PMM_0` and `PMM_1` change in succeeding decoding cycles. If, for example, in one decoding cycle path metrics were read from `PMM_0` and written in `PMM_1`, the next decoding cycle path metrics are read from `PMM_1` and are written in `PMM_0`, and vice versa (see timing diagram in section Waveforms and Timing Tables).

The input port `DATA_OUT_SMU` (bitwidth `DW_SMU`) is the data output of the Survivor Memory Unit `SMU` corresponding to the read address `A_SMU` (bitwidth `AW_SMU`). The write and read operations for the `SMU` are controlled by the low active write enable `WE_SMU_N` and the address `A_SMU` (bitwidth `AW_SMU`), i. e. the input data `DATA_IN_SMU` (bitwidth `DW_SMU`) for the `SMU` are written with the rising edge of `CLOCK` into the address `A_SMU` (see timing diagram in section Waveforms and Timing Tables).

The output port `X_DEC` of the `FHG_VITERBI_PIPE` provides the decoded bit stream calculated by the Viterbi Decoder. One bit is decoded every decoding cycle. The number of clock cycles per decoding cycle is derived in section Clocking of the `FHG_VITERBI_PIPE`.

Parameter Description

Table 12: Parameter Description

Name	Type	Default Value	Value Range	Description
CONSTRAINT_LENGTH	Integer	none	≥ 5	Constraint Length
CODE_RATE	Integer	none		Code rate $R_c = 1/\text{CODE_RATE}$
SOFT_BITS	Integer	none		Number of Soft-Input Bits per Code Bit
SURVIVOR_LENGTH	Integer	none		Survivor Length
GEN_POLY	Integer	none		Generator polynomials $\text{GEN_POLY} = (g_{\text{CODE_RATE}-1}, \dots, g_0)_2$ $g_i, i=0, \dots, \text{CODE_RATE}-1$
INVERT	Integer	none		Optional Inversion of Code Bits $\text{INVERT} = (\text{inv}_{\text{CODE_RATE}-1}, \dots, \text{inv}_0)_2$ $\text{inv}_i, i = 0, \dots, \text{CODE_RATE}-1$
AW_PMM	Integer	none		Address bits for Path Metric Memories (PMMs)
PML	Integer	none		Bitwidth of Path Metric (Path Metric Length)
AW_SMU	Integer	none		Address bits for Survivor Memory Unit (SMU)
DW_SMU	Integer	none	≥ 2	Bitwidth of SMU Data

The parameter `CONSTRAINT_LENGTH` is the constraint length of the convolutional code. For the use of the `FHG_VITERBI_PIPE` the constraint length has to be greater than or equal to five.

The parameter `CODE_RATE` is the number of code bits generated for each information bit by the convolutional encoder resulting in a code rate $R_c=1/\text{CODE_RATE}$.

The parameter `SOFT_BITS` is the number of soft-quantized bits for each code bit. Usually, one (hard decision) to three (soft decision) bits are used for quantization dependent on the desired Bit Error Rate (BER) which should be achieved under the condition of a given SNR per bit of the communication channel.

The parameter `SURVIVOR_LENGTH` characterizes the survivor length used by the Viterbi Decoder. As a rule of thumb, the survivor length should be chosen four to six times the constraint length ($\text{SURVIVOR_LENGTH} = (4..6) * \text{CONSTRAINT_LENGTH}$). The larger the survivor length is chosen the lower is the resulting BER.

The generator polynomials of the convolutional code are given by GEN_POLY. The generators for each of the CODE_RATE code bits given in binary form (CONSTRAINT_LENGTH bits per generator) are put in one bit vector. Thereby, the generator polynomials are numbered from code bit CODE_RATE-1 down to code bit 0. GEN_POLY is the resulting bit vector of length CODE_RATE*CONSTRAINT_LENGTH converted to integer.

The parameter INVERT allows an optional inversion of the code bits generated by the convolutional encoder. If an inversion of one of the CODE_RATE code bits is desired, the corresponding bit in INVERT has to be set to 1, otherwise 0 has to be chosen. The bits are numbered from code bit CODE_RATE-1 down to code bit 0. INVERT is the resulting bit vector of length CODE_RATE converted to integer.

The number of address bits for the path metric memories is given by AW_PMM. It is calculated by

$$AW_PMM = CONSTRAINT_LENGTH - 1.$$

The path metric length PML depends on the code rate, the constraint length, and the number of soft-input bits. It can be calculated by the formulas

$$\Delta\Gamma_{max} = CODE_RATE * (CONSTRAINT_LENGTH - 1) * (2^{SOFT_BITS} - 1)$$

and

$$PML = \text{ceil}(\log_2(\Delta\Gamma_{max} + 1)) + 1,$$

where $\text{ceil}(x)$ denotes the ceiling function (smallest integer value equal to or greater than x).

The address width AW_SMU of the SMU depends on the constraint length, the survivor length and the data width DW_SMU (see below) of the SMU. It is calculated by the formula

$$AW_SMU = \text{ceil}\left(\log_2\left(\frac{2^{CONSTRAINT_LENGTH-1}}{DW_SMU}\right)\right) + \text{ceil}(\log_2(SURVIVOR_LENGTH))$$

where $\text{ceil}(x)$ denotes the ceiling function (smallest integer value equal to or greater than x).

The data width of the SMU input is given by the parameter DW_SMU. DW_SMU has to be a divisor of the number of states ($= 2^{CONSTRAINT_LENGTH-1}$) of the convolutional code and an integer value greater than or equal to 2. With the choice of DW_SMU the data width of the SMU can be adapted to the requirements of the specific application.

In section Application Note an example is given how the parameter set has to be chosen for a special application.

Interfaces

The FHGVITERBI_PIPE has three interfaces with external memories (RAMs) for path metrics (PMM_0 and PMM_1) and survivor paths (SMU).

The two path metric interfaces consist of the low active write enable signals WE0_N and WE1_N, the address busses A_PMM_0 and A_PMM_1, the data input busses DATA_IN_PMM_0 and DATA_IN_PMM_1, and the data output busses DATA_OUT_PMM_0 and DATA_OUT_PMM_1.

The interface to the survivor path memory consists of the low active write enable signal WE_SMU_N, the address bus A_SMU, the data input bus DATA_IN_SMU, and the data output bus DATA_OUT_SMU.

Clocking of the FHG_VITERBI_PIPE

The FHG_VITERBI_PIPE has to be provided with a CLOCK signal whose minimum clock frequency depends on the symbol rate, the constraint length of the convolutional code, the data width of the SMU, and the chosen survivor length. The minimum clock frequency can be simply derived by calculating

$$f_{min} = \max \{ (2^{\text{CONSTRAINT_LENGTH}-1} + 6); (\text{SURVIVOR_LENGTH} + \text{DW_SMU} + 7) \} \times f_{symbol}$$

where $\max(x, y)$ denotes the maximum of the two expressions enclosed in braces. The calculated factor is the required number of clock cycles for one decoding cycle.

VHDL Usage through Component Instantiation

```
library IEEE, FHG_VITERBIDW;
use IEEE.std_logic_1164.all;
use FHG_VITERBIDW.FHG_VITERBI.all;

entity MY_VITERBI is
  port ( CLOCK          : in std_logic;
        RESET_N        : in std_logic;
        INIT            : in std_logic;
        SYNC            : in std_logic;
        R               : in std_logic_vector(5 downto 0);
        DATA_OUT_PMM_0 : in std_logic_vector(6 downto 0);
        DATA_OUT_PMM_1 : in std_logic_vector(6 downto 0);
        DATA_OUT_SMU   : in std_logic_vector(1 downto 0);
        WE0_N           : out std_logic;
        A_PMM_0         : out std_logic_vector(3 downto 0);
        DATA_IN_PMM_0  : out std_logic_vector(6 downto 0);
        WE1_N           : out std_logic;
        A_PMM_1         : out std_logic_vector(3 downto 0);
        DATA_IN_PMM_1  : out std_logic_vector(6 downto 0);
        WE_SMU_N        : out std_logic;
        A_SMU           : out std_logic_vector(7 downto 0);
        DATA_IN_SMU    : out std_logic_vector(1 downto 0);
        X_DEC           : out std_logic
        );
end MY_VITERBI;
```

architecture MY_VITERBI_RTL of MY_VITERBI is

```
constant GEN_POLY          : integer := 637;
constant INVERT            : integer := 2;
constant CONSTRAINT_LENGTH : integer := 5;
constant CODE_RATE        : integer := 2;
constant SOFT_BITS        : integer := 3;
constant SURVIVOR_LENGTH  : integer := 25;
constant AW_PMM           : integer := 4;
constant PML              : integer := 7;
constant AW_SMU           : integer := 8;
constant DW_SMU           : integer := 2;
```

begin

```
MY_VITERBI_CORE : FHG_VITERBI_PIPE
  generic map ( GEN_POLY          => GEN_POLY,
               INVERT            => INVERT,
               CONSTRAINT_LENGTH => CONSTRAINT_LENGTH,
               CODE_RATE        => CODE_RATE,
               SOFT_BITS        => SOFT_BITS,
               SURVIVOR_LENGTH  => SURVIVOR_LENGTH,
               AW_PMM           => AW_PMM,
               PML              => PML,
               AW_SMU           => AW_SMU,
               DW_SMU           => DW_SMU )
  port map ( CLOCK                => CLOCK,
            RESET_N              => RESET_N,
            INIT                 => INIT,
            SYNC                 => SYNC,
            R                    => R,
            DATA_OUT_PMM_0     => DATA_OUT_PMM_0,
            DATA_OUT_PMM_1     => DATA_OUT_PMM_1,
            DATA_OUT_SMU       => DATA_OUT_SMU,
            WE0_N               => WE0_N,
            A_PMM_0             => A_PMM_0,
            DATA_IN_PMM_0     => DATA_IN_PMM_0,
            WE1_N               => WE1_N,
            A_PMM_1             => A_PMM_1,
            DATA_IN_PMM_1     => DATA_IN_PMM_1,
            WE_SMU_N           => WE_SMU_N,
            A_SMU               => A_SMU,
            DATA_IN_SMU       => DATA_IN_SMU,
            X_DEC               => X_DEC
            );
```

end MY_VITERBI_RTL;

Waveforms and Timing Tables

Figure 11: Initialization

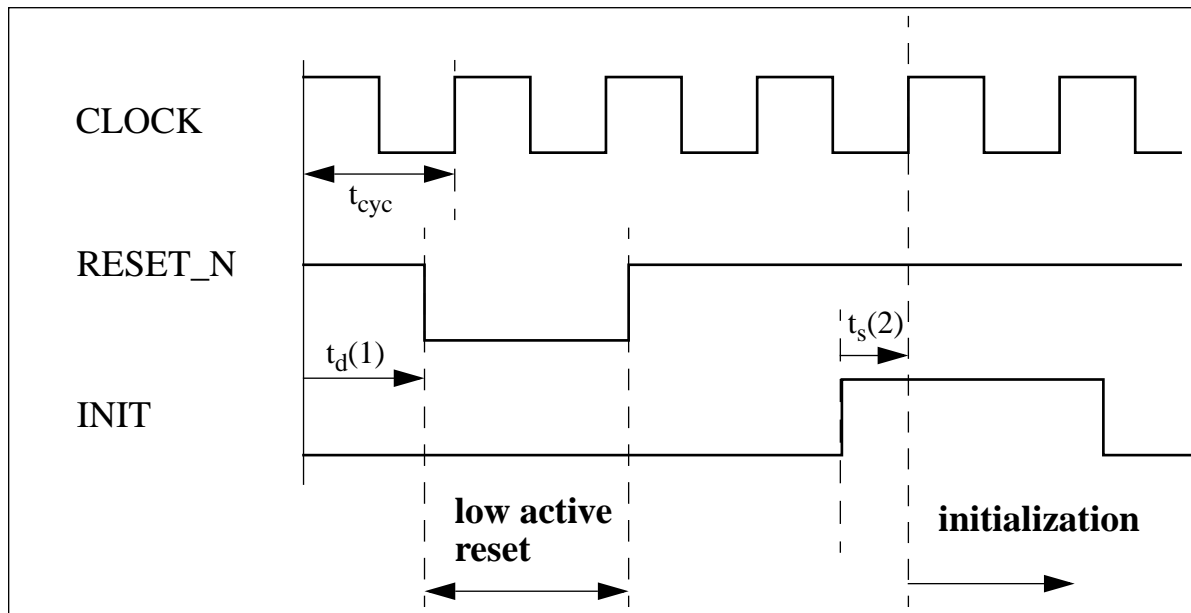


Table 13: Timing Table of Initialization Process

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{\text{rise}}-\text{RESET_N}_{\text{fall}})$	Delay time for RESET_N to rising clock edge				ns
$t_s(2)$	$t_s(\text{INIT}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Setup time for INIT to rising clock edge				ns

Figure 12: Soft-Input Synchronization

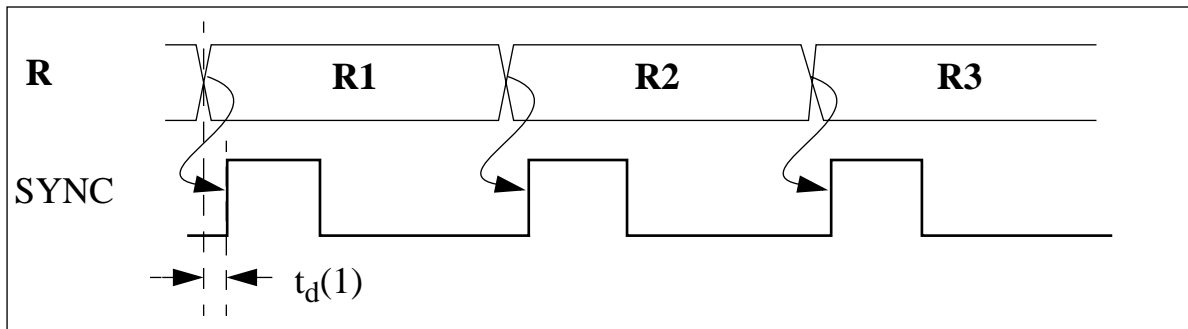


Table 14: Timing table of Soft-Input Synchronization

Symbol	Name	Item	Notes	Min	Max	Unit
t_d	$t(R_{\text{valid}}\text{-}SYNC_{\text{rise}})$	Delay time from valid soft-input R to rising SYNC signal				ns

Figure 13: Waveform of external write and read access for path metric memories

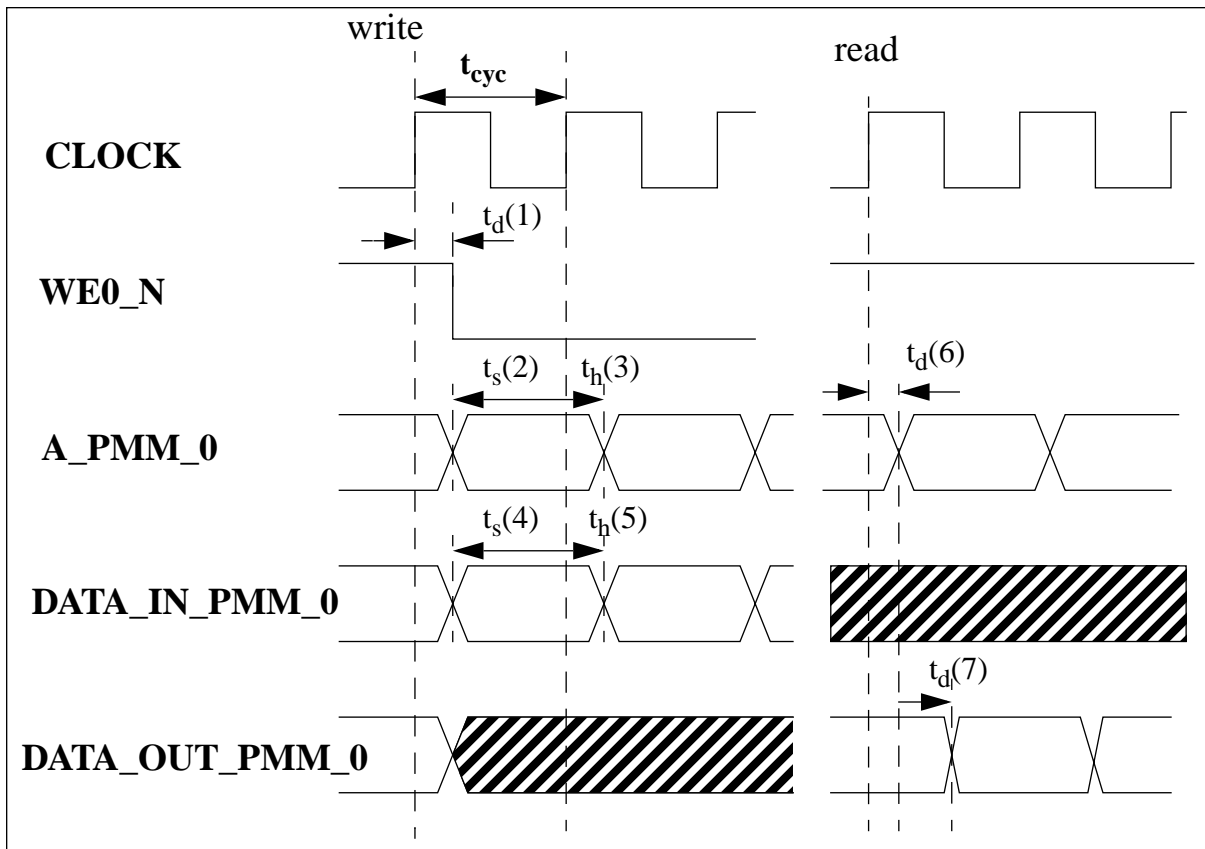


Table 15: Timing table of an external data read access

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{\text{rise}}-\text{WE0_N}_{\text{fall}})$	Delay time for WE0_N from rising clock edge				ns
$t_s(2)$	$t_s(\text{A_PMM_0}_{\text{valid}}-\text{CLOCK}_{\text{rise}})$	Setup time for A_PMM_0 to rising clock edge				ns
$t_h(3)$	$t_h(\text{CLOCK}_{\text{rise}}-\text{A_PMM_0}_{\text{valid}})$	Hold time for A_PMM_0				ns
$t_s(4)$	$t_s(\text{DATA_IN_PMM_0}_{\text{valid}}-\text{CLOCK}_{\text{rise}})$	Setup time for DATA_IN_PMM_0 to rising clock edge				ns
$t_h(5)$	$t_h(\text{CLOCK}_{\text{rise}}-\text{A_PMM_0}_{\text{valid}})$	Hold time for DATA_IN_PMM_0				ns
$t_d(6)$	$t_d(\text{CLOCK}_{\text{rise}}-\text{A_PMM_0}_{\text{valid}})$	Delay time for A_PMM_0 from rising clock edge				ns
$t_d(7)$	$t_d(\text{A_PMM_0}_{\text{valid}}-\text{DATA_OUT_PMM_0}_{\text{valid}})$	Address access time of PMM_0				ns

Figure 14: Waveform of external read and write accesses of external survivor memory

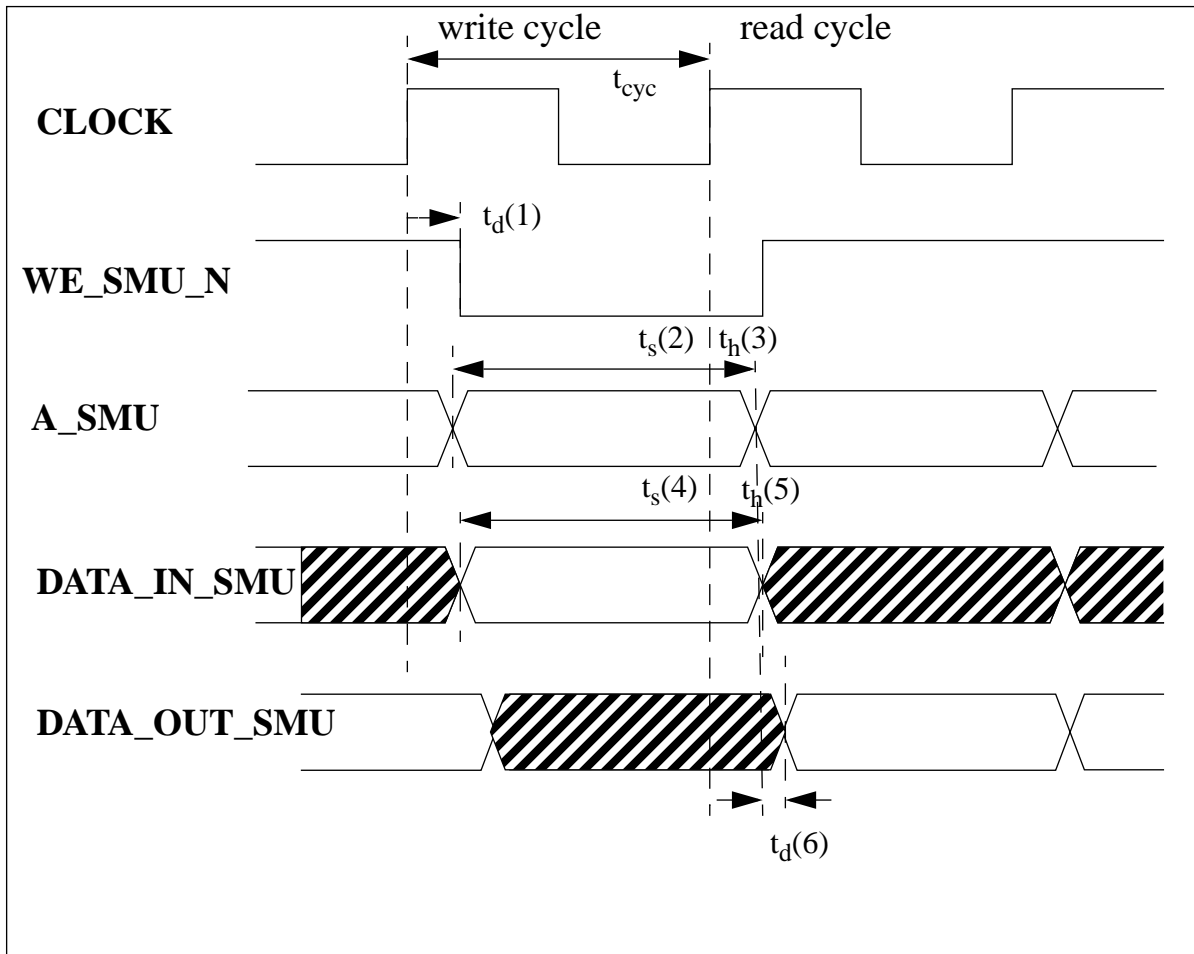


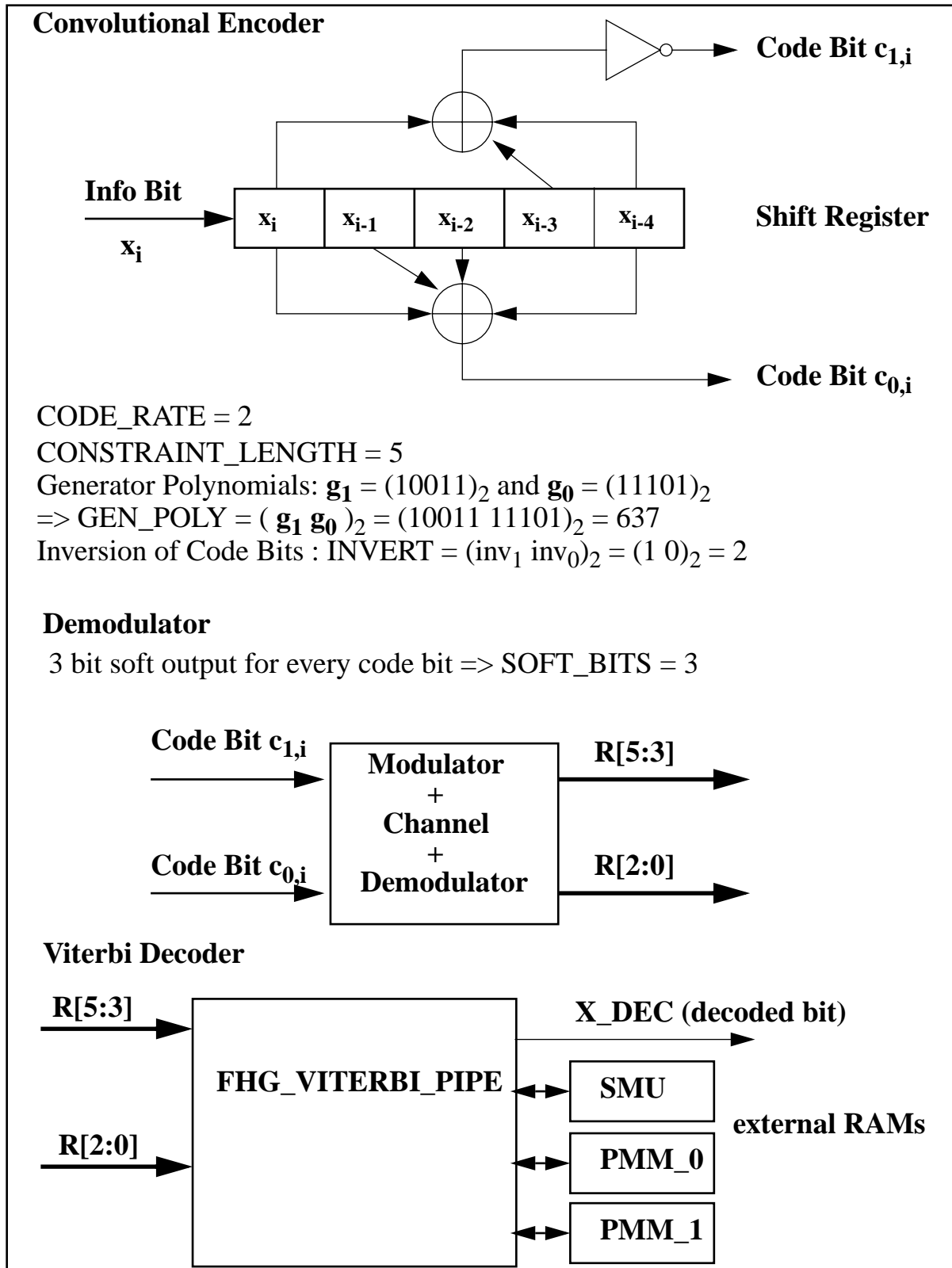
Table 16: Timing table of an external access of survivor memory

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{rise}-\text{CLOCK}_{rise})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{rise}-\text{WE_SMU_N}_{fall})$	Delay time for WE_SMU_N from rising clock edge				ns
$t_s(2)$	$t_s(\text{A_SMU}_{valid}-\text{CLOCK}_{rise})$	Setup time for A_SMU to rising clock edge				ns
$t_h(3)$	$t_h(\text{CLOCK}_{rise}-\text{A_SMU}_{valid})$	Hold time for A_SMU				ns
$t_s(4)$	$t_s(\text{DATA_IN_SMU}_{valid}-\text{CLOCK}_{rise})$	Setup time for DATA_IN_SMU to rising clock edge				ns
$t_h(5)$	$t_h(\text{CLOCK}_{rise}-\text{A_SMU}_{valid})$	Hold time for DATA_IN_SMU				ns
$t_d(6)$	$t_d(\text{A_SMU}_{valid}-\text{DATA_OUT_PMM}_{valid})$	Address access time of SMU				ns

Application Note

The following example shows the parameters of the FHG_VITERBI_PIPE for a special application. This parameter set is used in section VHDL Usage through Component Instantiation.

Figure 15: Application example



(Application example continued)

$$AW_PMM = 4$$

$$PML = 7$$

number of states = 16 => choose e.g. $DW_SMU = 2$

$$SURVIVOR_LENGTH = 5 * CONSTRAINT_LENGTH = 25$$

$$\Rightarrow AW_SMU = 8$$

$$\text{Min. clock frequency: } f_{\min} = \max\{2^4+6; 25+2+7\} * f_{\text{symbol}} = 34 * f_{\text{symbol}}$$

5. FHG_VITERBI_SP

Purpose

The FHG_VITERBI_SP is a fully parameterized and synthesizable rate-1/n Viterbi decoder. Its parameter set (constraint length, code rate, generator polynomials, number of soft bits, survivor length) enables the designer to implement Viterbi decoders for any rate-1/n convolutional code for a wide range of symbol rates and thus provides versatility for the design of corresponding applications.

Features

- Fully Synthesizable Parametrized Viterbi Decoder for Rate-1/n Convolutional Codes
- Constraint Length, Generator Polynomials, and Code Rate parameterized (optional inversion of code bits)
- Number of parallel Butterfly ACS Computers parameterized
- Survivor Length of Viterbi Decoder parameterized
- Wordlength of Soft-Input parameterized
- Soft-Input in Offset-Binary and Two's Complement Format
- Symbol Rates in a range from 500 kilosymbols/sec up to 10 MSymbols/sec (Constraint Length $K=7$, Code Rate $R_c=1/2$, 0.8 μ m Technology)
- Registered Inputs and Outputs
- External RAMs for Path Metrics and Survivor Memory Unit required

Design Kit

- Technology Independent Implementation as Synopsys Design Ware Components
- VHDL/Verilog Source Code Simulation Models
- VHDL/Verilog Compliance Test Suite
- Auxiliary Simulation Models for User Testbenches
- Synthesis and Testsynthesis Scripts
- Design Support, Netlist Synthesis Service, and Consulting available

Requirements

Simulation

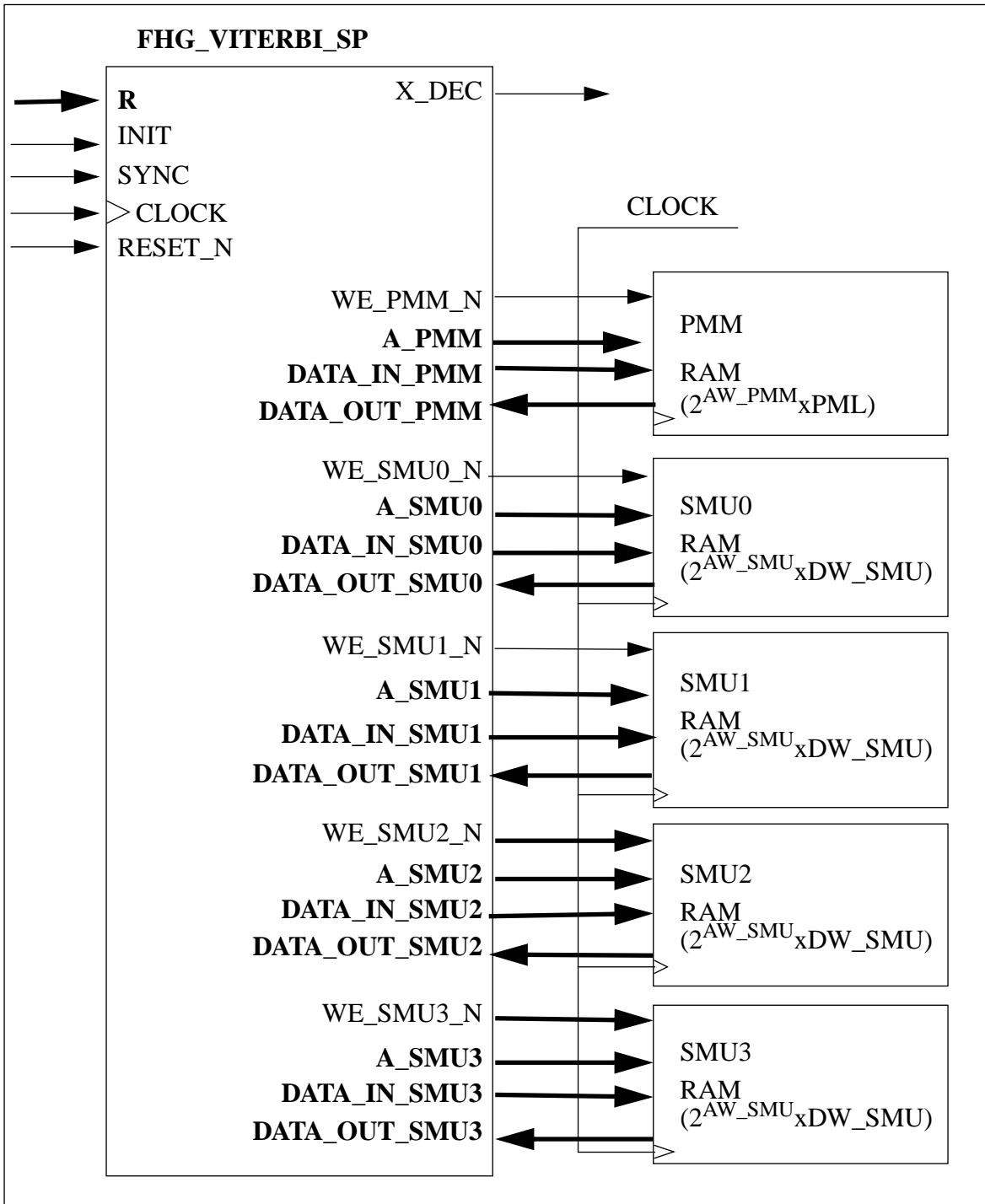
- VHDL IEEE-1076 Simulator
- Verilog IEEE-1364 Simulator

Synthesis

- Synopsys Design Compiler

Block Diagram

Figure 16: Block Diagram



General Information

The FHG_VITERBI_SP and its environment is a fully synchronous design. Therefore, all blocks of the design (FHG_VITERBI_SP, SMU0, SMU1, SMU2, SMU3, PMM) have to be provided with the same CLOCK signal. It is assumed that the soft-quantized bits for all code bits are applied parallelly to the input of the Viterbi decoder (as is the case e.g. for QPSK). If this is not the case, a serial-to-parallel conversion is required.

The Survivor Memory Unit ist divided into four memory banks SMU0, SMU1, SMU2, and SMU3 because of the usage of a special decoding algorithm. The memories for the path metrics (Path Metric Memory PMM) and the survivor paths (Survivor Memory Units SMU0, SMU1, SMU2, SMU3) have to be implemented as external RAMs with corresponding memory sizes and datawidths. PMM and the four path memories SMU0, SMU1, SMU2, and SMU3 have to be synchronous Single-Port-RAMs with a clock input, a low active write enable input, and an address input, i. e. if the write enable signals are low, the input data are stored with the next rising edge of the clock. Formulas for the necessary parameters (bitwidths, datawidths) are given in section Parameter Description.

Signal Description

Table 17: Signal Description

Pin	Porttyp	Bitbreite	Beschreibung
CLOCK	IN	1	Decoder Clock
RESET_N	IN	1	Low active asynchronous reset
INIT	IN	1	Initialization signal
SYNC	IN	1	Synchronization signal for a new code symbol (symbol clock)
R	IN	CODE_RATE *SOFT_BITS	Soft Input: CODE_RATE code bits ($R_c = 1/\text{CODE_RATE}$) each quantized with SOFT_BITS bits in Off-set Binary Format
DATA_OUT_PMM	IN	CALCS*PML	Path Metrics from Path Metric Memory PMM
DATA_OUT_SMU0 DATA_OUT_SMU1 DATA_OUT_SMU2 DATA_OUT_SMU3	IN	DW_SMU	Output Data from Survivor Memory Units (SMU0, SMU1, SMU2, SMU3)
WE_PMM_N	OUT	1	Low active write enable for PMM
A_PMM	OUT	AW_PMM	Address for PMM
DATA_IN_PMM	OUT	CALCS*PML	Path Metric (PML bit) to be stored in PMM
WE_SMU0_N WE_SMU1_N WE_SMU2_N WE_SMU2_N	OUT	1	Low active write enable for SMU0, SMU1, SMU2, SMU3
A_SMU0 A_SMU1 A_SMU2 A_SMU3	OUT	AW_SMU	Addresses for SMU0, SMU1, SMU2, SMU3
DATA_IN_SMU0 DATA_IN_SMU1 DATA_IN_SMU2 DATA_IN_SMU3	OUT	DW_SMU	Input data for SMU0, SMU1, SMU2, SMU3

The input port CLOCK is the decoder system clock. The necessary clock frequency is derived in section Parameter Description and depends on the chosen code parameters and the codesymbol rate.

The input port `RESET_N` is a low active asynchronous reset for the `FHG_VITERBI_SER`. An asynchronous reset should be performed at the beginning of the decoding process.

The input port `INIT` is a high active initialization signal. It initializes the path metric memory `PMM` what should be done at least once at the beginning of the decoding process. For this purpose, the signal `INIT` has to be high for at least one clock cycle. The initialization then starts with the first rising edge of the `CLOCK` signal (see timing diagram in section Waveforms and Timing Tables).

The input port `SYNC` is the synchronization signal for the code symbols. If `SYNC` is high, this indicates new valid soft-input bits `R` provided by the demodulator for a new code symbol. This signal can be e. g. the clock of the analog-to-digital converter at the demodulator output (see timing diagram in section Waveforms and Timing Tables).

The input port `R` consists of the soft-input data. Each of the `CODE_RATE` code bits is quantized with `SOFT_BITS` bit in Offset-Binary Format. The input code bits are numbered from code bit `CODE_RATE-1` down to code bit `0` each starting with the MSB and ending with the LSB.

The input port `DATA_OUT_PMM` reads the path metrics of bitwidth `CALCS*PML` from the Path Metric Memory `PMM`, whereas the output port `DATA_IN_PMM` contains the corresponding new path metrics to be stored. The write and read operations of `PMM` are controlled by the write enable signal `WE_PMM_N` and the address signal `A_PMM` (bitwidth `AW_PMM`). `WE_PMM_N` is low active, i. e. if `WE_PMM_N` is low, the data at the input of the path metric memory (`DATA_IN_PMM`) are written with the next rising edge of the `CLOCK` signal (see timing diagram in section Waveforms and Timing Tables).

The input ports `DATA_OUT_SMU0`, `DATA_OUT_SMU1`, `DATA_OUT_SMU2`, and `DATA_OUT_SMU3` (bitwidth `DW_SMU`) are the data outputs of the four memory banks of the Survivor Memory Unit corresponding to the addresses `A_SMU0`, `A_SMU1`, `A_SMU2`, and `A_SMU3` (bitwidth `AW_SMU`). The write and read operations for the four survivor memory banks are controlled by the low active write enable signals `WE_SMU0_N`, `WE_SMU1_N`, `WE_SMU2_N`, and `WE_SMU3_N`, and the addresses `A_SMU0`, `A_SMU1`, `A_SMU2`, and `A_SMU3` (bitwidth `AW_SMU`), i. e. if the write enable signals are low, the input data for the memory banks are written with the rising edge of `CLOCK` into the corresponding addresses (see timing diagram in section Waveforms and Timing Tables)

The output port `X_DEC` provides the decoded bit stream calculated by the Viterbi decoder. One bit is decoded every decoding cycle. The number of clock cycles per decoding cycle is derived in section Clocking of the `FHG_VITERBI_SP`.

Parameter Description

Table 18: Parameter Description

Name	Type	Default Value	Value Range	Description
CONSTRAINT_LENGTH	Integer	none		Constraint Length
CODE_RATE	Integer	none		Code rate $R_c = 1/\text{CODE_RATE}$
SOFT_BITS	Integer	none		Number of Soft-Input Bits per Code Bit
CALCS	Integer	none	$\{2..2^{\text{CL}-2}\}$	number of Butterfly-ACS-Calculators (power of 2)
SURVIVOR_LENGTH	Integer	none		Survivor Length
GEN_POLY	Integer	none		Generator polynomials $\text{GEN_POLY} = (g_{\text{CODE_RATE}-1}, \dots, g_0)_2$ $g_i, i=0, \dots, \text{CODE_RATE}-1$
INVERT	Integer	none		Optional Inversion of Code Bits $\text{INVERT} = (\text{inv}_{\text{CODE_RATE}-1}, \dots, \text{inv}_0)_2$ $\text{inv}_i, i = 0, \dots, \text{CODE_RATE}-1$
AW_PMM	Integer	none		Address bits for Path Metric Memories (PMMs)
PML	Integer	none		Bitwidth of Path Metric (Path Metric Length)
AW_SMU	Integer	none		Address bits for Survivor Memory Unit (SMU)
DW_SMU	Integer	none	≥ 2	Bitwidth of SMU Data

The parameter CONSTRAINT_LENGTH is the constraint length of the convolutional code. Its value can be any convenient length.

The parameter CODE_RATE is the number of code bits generated for each information bit by the convolutional encoder resulting in a code rate $R_c=1/\text{CODE_RATE}$.

The parameter SOFT_BITS is the number of soft-quantized bits provided for each code bit. Usually, one (hard decision) to three bits (soft decision) are used for quantization dependent on the desired bit-error rate (BER) which should be achieved under the condition of a given SNR per bit of the communication channel.

The parameter CALCS is the number of parallel butterfly ACS computers to calculate the ACS equations of the Viterbi Algorithm. The possible range is from CALCS=2 to CALCS = $2^{\text{CONSTRAINT_LENGTH}-2}$. Thus, for the decoding of one code symbol STATES/(2*CALCS) clock cycles are necessary. Thus the parameter CALCS allows a trade-off between the desired symbol rate and the area needed on the chip. The less ACS computers are needed, the less is the required chip area.

The parameter SURVIVOR_LENGTH characterizes the survivor length used by the Viterbi decoder. As a rule of thumb, the survivor length should be chosen four to six times the constraint length (SURVIVOR_LENGTH = (4..6)*CONSTRAINT_LENGTH). The larger the survivor length is chosen the lower is the resulting BER.

The generator polynomials of the convolutional code are given by GEN_POLY. The generators for each of the CODE_RATE code bits given in binary form (CONSTRAINT_LENGTH bits per generator) are put in one bit vector. Thereby, the generator polynomials are numbered from code bit CODE_RATE-1 down to code bit 0. GEN_POLY is the resulting bit vector of length CODE_RATE*CONSTRAINT_LENGTH converted to integer.

The parameter INVERT allows an optional inversion of the code bits generated by the convolutional encoder. If an inversion of one of the CODE_RATE code bits is desired, the corresponding bit in INVERT has to be set to 1, otherwise 0 has to be chosen. The bits are numbered from code bit CODE_RATE-1 down to code bit 0. INVERT is the resulting bit vector of length CODE_RATE converted to integer.

The number of address bits for the path metric memories is given by AW_PMM. With CALCS= $2^{\text{AW_CALCS}}$ it is calculated by

$$\text{AW_PMM} = \text{CONSTRAINT_LENGTH} - 1 - \text{AW_CALCS}.$$

The path metric length PML depends on the code rate, the constraint length, and the number of soft-input bits. It can be calculated by the formulas

$$\Delta\Gamma_{\max} = \text{CODE_RATE} * (\text{CONSTRAINT_LENGTH} - 1) * (2^{\text{SOFT_BITS}} - 1)$$

and

$$\text{PML} = \text{ceil}(\log_2(\Delta\Gamma_{\max} + 1)) + 1,$$

where $\Delta\Gamma_{\max}$ denotes the maximum difference between two path metrics and $\text{ceil}(x)$ denotes the ceiling function (smallest integer value equal to or greater than x).

The datawidth of the memory banks of the Survivor Memory Unit is determined by the number of parallel ACS computers (parameter CALCS). It is simply calculated by

$$\text{DW_SMU} = 2 * \text{CALCS}.$$

The address width AW_SMU of the SMU depends on the constraint length, the survivor length, and the datawidth DW_SMU of the SMU. It is calculated by the formula

$$AW_SMU = AW_PMM - 1 + \text{ceil}(\log_2(\text{SURVIVOR_LENGTH}/2))$$

where $\text{ceil}(x)$ denotes the ceiling function (smallest integer value equal to or greater than x).

In section Application Note an example is given how the parameter set has to be chosen for a special application.

Interfaces

The FHG_VITERBI_PIPE has five interfaces with external memories (RAMs) for path metrics (PMM) and survivor paths (SMU_0, SMU_1, SMU_2, SMU_3).

The path metric interface consists of the low active write enable signal WE_PMM_N, the address bus A_PMM, the data input bus DATA_IN_PMM, and the data output bus DATA_OUT_PMM.

The interfaces with the survivor path memories consist of the low active write enable signals WE_SMU(0..3)_N, the address busses A_SMU(0..3), the data input busses DATA_IN_SMU(0..3), and the data output busses DATA_OUT_SMU(0..3).

Clocking of the FHG_VITERBI_SP

The FHG_VITERBI_SP has to be provided with a CLOCK signal whose clock frequency depends on the symbol rate, the constraint length of the convolutional code, and the number of parallel ACS computers. The clock frequency can be derived by calculating

$$f_{CLOCK} = 2^{\text{CONSTRAINT_LENGTH-AW_CALCS}} \times f_{symbol}$$

VHDL Usage through Component Instantiation

```
library IEEE, FHG_VITERBIDW;
use IEEE.std_logic_1164.all;
use FHG_VITERBIDW.FHG_VITERBI.all;

entity MY_VITERBI is
  port ( CLOCK          : in std_logic;
        RESET_N        : in std_logic;
        INIT            : in std_logic;
        SYNC            : in std_logic;
        R               : in std_logic_vector(5 downto 0);
        WE_PMM_N        : out std_logic;
        A_PMM           : out std_logic_vector(2 downto 0);
        DATA_IN_PMM    : out std_logic_vector(13 downto 0);
        DATA_OUT_PMM   : in std_logic_vector(13 downto 0);
        WE_SMU0_N       : out std_logic;
        A_SMU0          : out std_logic_vector(5 downto 0);
        DATA_IN_SMU0   : out std_logic_vector(3 downto 0);
        DATA_OUT_SMU0 : in std_logic_vector(3 downto 0);
        WE_SMU1_N       : out std_logic;
        A_SMU1          : out std_logic_vector(5 downto 0);
        DATA_IN_SMU1   : out std_logic_vector(3 downto 0);
        DATA_OUT_SMU1 : in std_logic_vector(3 downto 0);
        WE_SMU2_N       : out std_logic; low)
        A_SMU2          : out std_logic_vector(5 downto 0);
        DATA_IN_SMU2   : out std_logic_vector(3 downto 0);
        DATA_OUT_SMU2 : in std_logic_vector(3 downto 0);
        WE_SMU3_N       : out std_logic; low)
        A_SMU3          : out std_logic_vector(5 downto 0);
        DATA_IN_SMU3   : out std_logic_vector(3 downto 0);
        DATA_OUT_SMU3 : in std_logic_vector(3 downto 0);
        X_DEC           : out std_logic
  );
end MY_VITERBI;

architecture MY_VITERBI_RTL of MY_VITERBI is

  constant GEN_POLY      : integer := 637;
  constant INVERT        : integer := 2;
  constant CONSTRAINT_LENGTH : integer := 5;
  constant CODE_RATE     : integer := 2;
  constant SOFT_BITS     : integer := 3;
  constant CALCS         : integer := 2;
  constant SURVIVOR_LENGTH : integer := 30;
  constant AW_PMM        : integer := 3;
  constant PML           : integer := 7;
  constant AW_SMU        : integer := 6;
  constant DW_SMU        : integer := 4;
```

begin

```
MY_VITERBI_CORE : FHG_VITERBI_SP
generic map ( GEN_POLY          => GEN_POLY,
              INVERT            => INVERT,
              CONSTRAINT_LENGTH => CONSTRAINT_LENGTH,
              CODE_RATE         => CODE_RATE,
              SOFT_BITS         => SOFT_BITS,
              CALCS             => CALCS,
              SURVIVOR_LENGTH  => SURVIVOR_LENGTH,
              AW_PMM           => AW_PMM,
              PML              => PML,
              AW_SMU           => AW_SMU,
              DW_SMU           => DW_SMU
            )
port map ( CLOCK          => CLOCK,
          RESET_N        => RESET_N,
          INIT           => INIT,
          SYNC           => SYNC,
          R              => R,
          WE_PMM_N       => WE_PMM_N,
          A_PMM          => A_PMM,
          DATA_IN_PMM   => DATA_IN_PMM,
          DATA_OUT_PMM  => DATA_OUT_PMM,
          WE_SMU0_N      => WE_SMU0_N,
          A_SMU0         => A_SMU0,
          DATA_IN_SMU0  => DATA_IN_SMU0,
          DATA_OUT_SMU0 => DATA_OUT_SMU0,
          WE_SMU1_N      => WE_SMU1_N,
          A_SMU1         => A_SMU1,
          DATA_IN_SMU1  => DATA_IN_SMU1,
          DATA_OUT_SMU1 => DATA_OUT_SMU1,
          WE_SMU2_N      => WE_SMU2_N,
          A_SMU2         => A_SMU2,
          DATA_IN_SMU2  => DATA_IN_SMU2,
          DATA_OUT_SMU2 => DATA_OUT_SMU2,
          WE_SMU3_N      => WE_SMU3_N,
          A_SMU3         => A_SMU3,
          DATA_IN_SMU3  => DATA_IN_SMU3,
          DATA_OUT_SMU3 => DATA_OUT_SMU3,
          X_DEC          => X_DEC
        );

end MY_VITERBI_RTL;
```

Waveforms and Timing Tables

Figure 17: Initialization

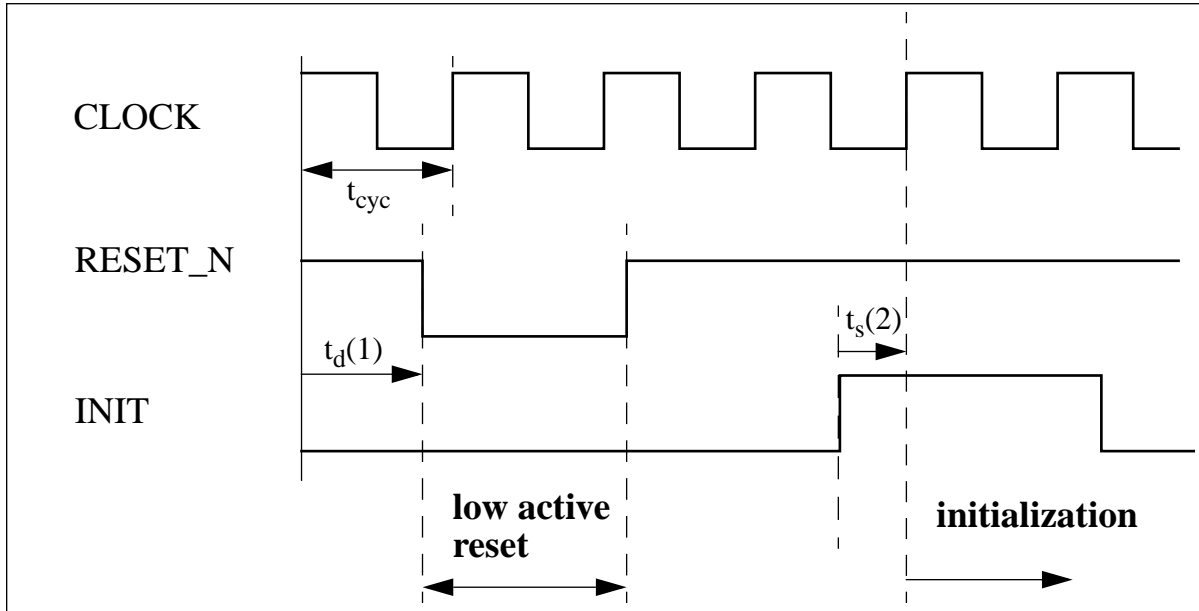


Table 19: Timing Table of Initialization Process

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{\text{rise}}-\text{RESET_N}_{\text{fall}})$	Delay time for RESET_N to rising clock edge				ns
$t_s(2)$	$t_s(\text{INIT}_{\text{rise}}-\text{CLOCK}_{\text{rise}})$	Setup time for INIT to rising clock edge				ns

Figure 18: Soft-Input Synchronization

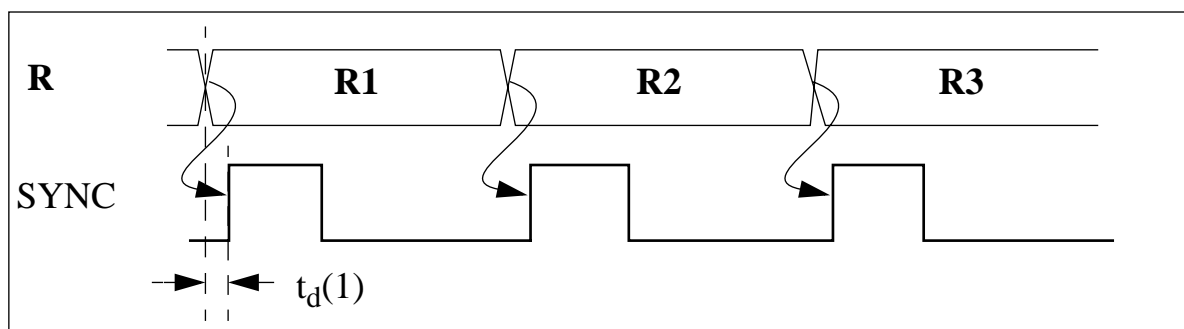


Table 20: Timing table of Soft-Input Synchronization

Symbol	Name	Item	Notes	Min	Max	Unit
t_d	$t(R_{\text{valid}}\text{-}SYNC_{\text{rise}})$	Delay time from valid soft-input R to rising SYNC signal				ns

Figure 19: Waveform of external write and read access of path metric memory

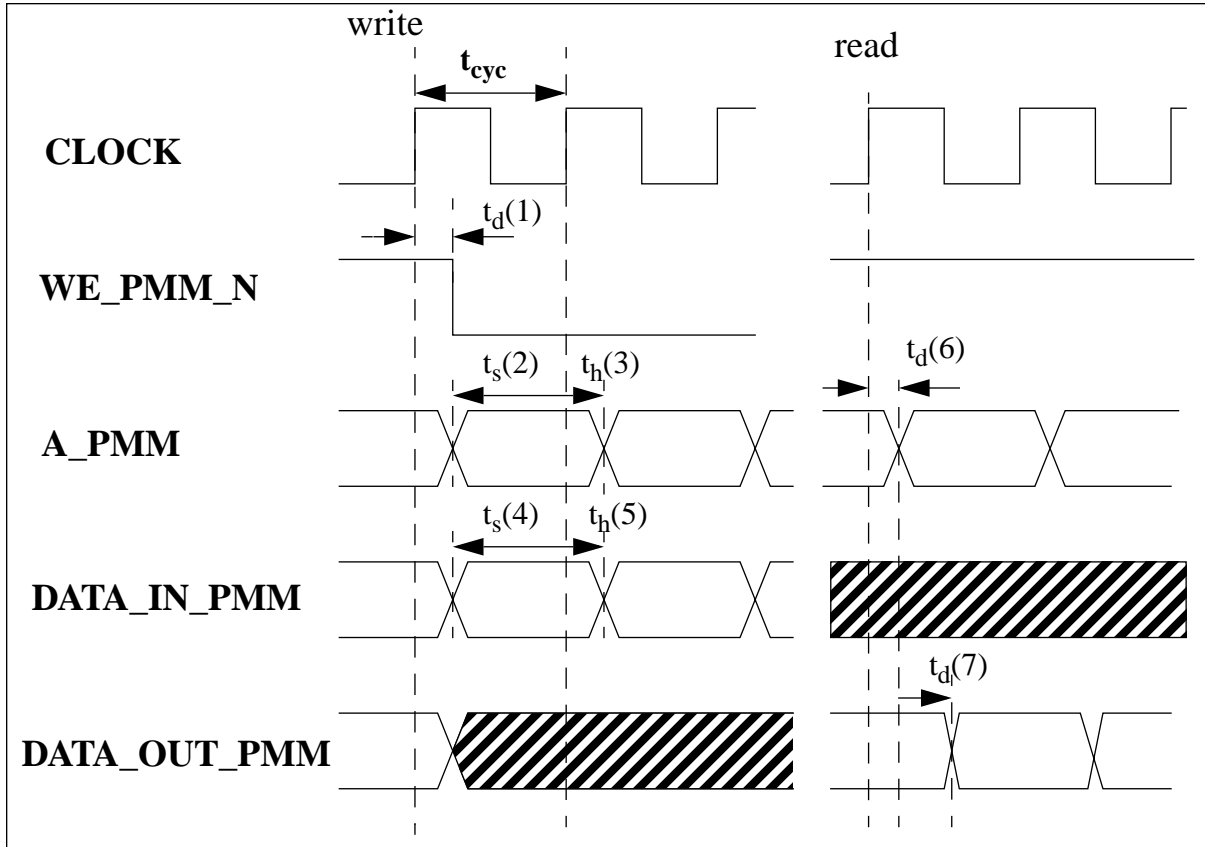


Table 21: Timing table of an external access of path metric memory

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{rise}-\text{CLOCK}_{rise})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{rise}-\text{WE_PMM_N}_{fall})$	Delay time for falling edge of WE_PMM_N from rising clock edge				ns
$t_s(2)$	$t_s(\text{A_PMM}_{valid}-\text{CLOCK}_{rise})$	Setup time for A_PMM to rising clock edge				ns
$t_h(3)$	$t_h(\text{CLOCK}_{rise}-\text{A_PMM}_{valid})$	Hold time for A_PMM				ns
$t_s(4)$	$t_s(\text{DATA_IN_PMM}_{valid}-\text{CLOCK}_{rise})$	Setup time for DATA_IN_PMM to rising clock edge				ns
$t_h(5)$	$t_h(\text{CLOCK}_{rise}-\text{A_PMM}_{valid})$	Hold time for DATA_IN_PMM				ns
$t_d(6)$	$t_d(\text{CLOCK}_{rise}-\text{A_PMM}_{valid})$	Delay time for A_PMM from rising clock edge				ns
$t_d(7)$	$t_d(\text{A_PMM}_{valid}-\text{DATA_OUT_PMM}_{valid})$	Address access time of PMM				ns

Figure 20: Waveform of external read and write accesses of survivor memory

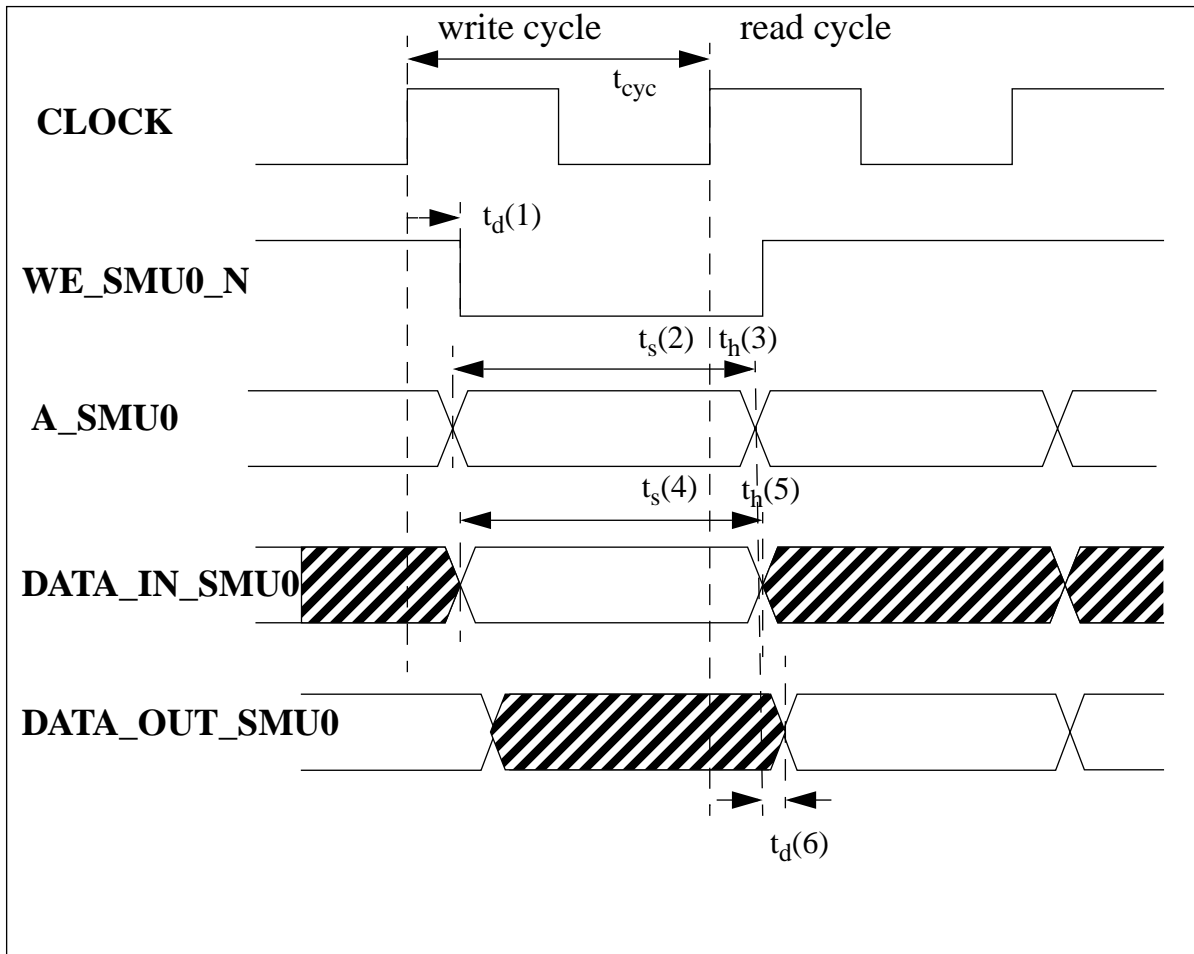


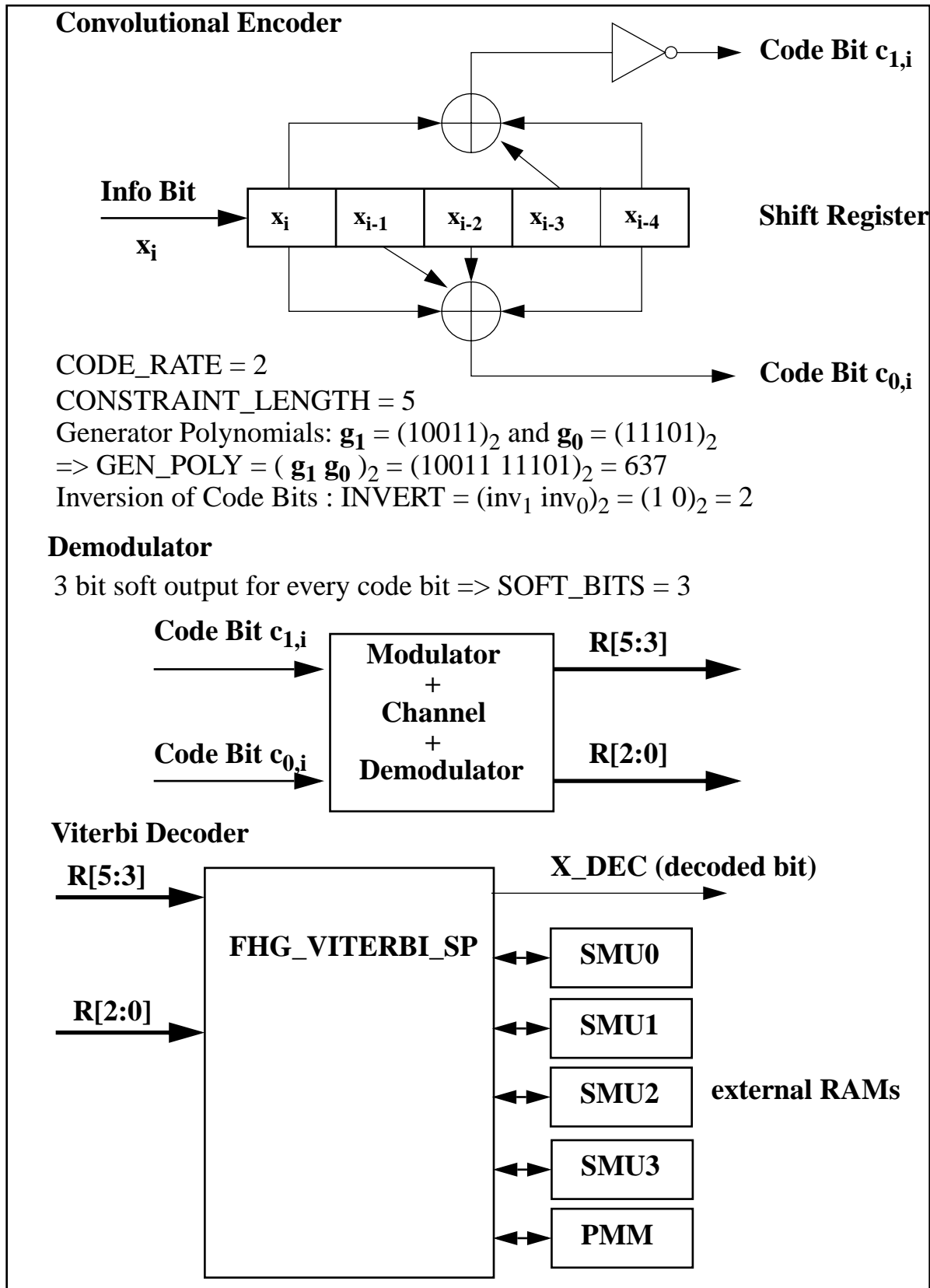
Table 22: Timing table of an external access of survivor memory

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLOCK}_{rise}-\text{CLOCK}_{rise})$	Clock cycle				ns
$t_d(1)$	$t_d(\text{CLOCK}_{rise}-\text{WE_SMU0_N}_{fall})$	Delay time for WE_SMU0_N from rising clock edge				ns
$t_s(2)$	$t_s(\text{A_SMU0}_{valid}-\text{CLOCK}_{rise})$	Setup time for A_SMU0 to rising clock edge				ns
$t_h(3)$	$t_h(\text{CLOCK}_{rise}-\text{A_SMU0}_{valid})$	Hold time for A_SMU0				ns
$t_s(4)$	$t_s(\text{DATA_IN_SMU0}_{valid}-\text{CLOCK}_{rise})$	Setup time for DATA_IN_SMU0 to rising clock edge				ns
$t_h(5)$	$t_h(\text{CLOCK}_{rise}-\text{A_SMU0}_{valid})$	Hold time for DATA_IN_SMU0				ns
$t_d(6)$	$t_d(\text{A_SMU0}_{valid}-\text{DATA_OUT_PMM}_{valid})$	Address access time of SMU0				ns

Application Note

The following example shows the parameters of the FHG_VITERBI_SP for a special application. This parameter set is used in section VHDL Usage through Component Instantiation.

Figure 21: Application example



(Application example continued)

number of states = 16 => choose e.g. CALCS = 2 => AW_CALCS = 1

AW_PMM = 3

PML = 7

DW_SMU = 2 * CALCS = 4

SURVIVOR_LENGTH = 6 * CONSTRAINT_LENGTH = 30

=> AW_SMU = 6

Min. clock frequency: $f_{\min} = 16 * f_{\text{symbol}}$