Reed-Muller Decoder/Encoder in VHDL

Generato da Doxygen 1.8.8

Ven 12 Mag 2017 10:22:26

Indice

Capitolo 1

Lista dei test

Classe RMDecoder
RM(1,m), m=3

Classe RMEncoder
RM(1,m), m=3

Capitolo 2

Design Unit Index

2.1 Design Unit Hierarchy

Questo elenco di ereditarietà è ordinato approssimativamente, ma non completamente, in ordine alfabetico:

Capitolo 3

Design Unit Index

3.1 Design Unit List

Here is a list of all design unit members with links to the Entities they belong to:

Capitolo 4

Indice dei file

4.1 Elenco dei file

Questo è un elenco dei file documentati con una loro breve descrizione:

Capitolo 5

Documentazione delle classi

5.1 adder_block Entity Reference

Implementazione VHDL Structural di un generico livello del componente generic_adder. Tale livello è costituito da 2^level addizionatori che lavorano in parallelo, di dimensione dipendente dal livello corrente: N = number_bit_for↩ _operand + log2(number_operand)-level-1.

Diagramma delle classi per adder_block

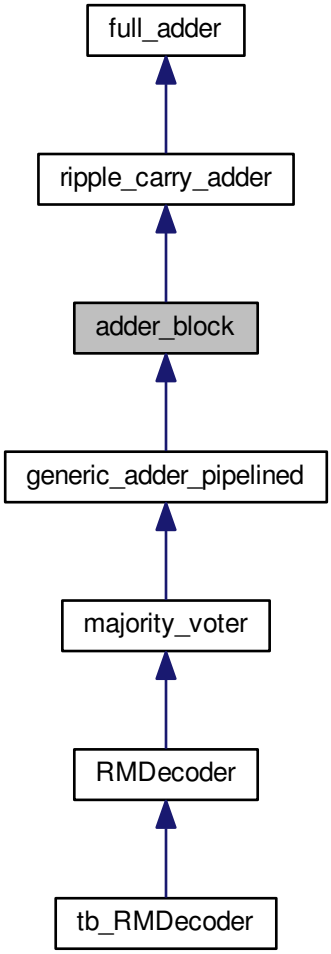
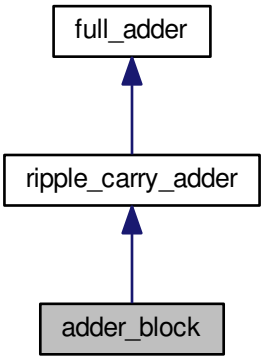


Diagramma di collaborazione per adder_block:



Entities

- Structural architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164
- NUMERIC_STD
- MATH_REAL

Generics

- number_operand **NATURAL**
- number_bit_for_operand **NATURAL**
- level **NATURAL**

Ports

- data_in **in STD_LOGIC_VECTOR(number_operand∗number_bit_for_operand- 1 downto 0)**
- data_out **out STD_LOGIC_VECTOR(number_operand∗number_bit_for_operand- 1 downto 0)**

5.1.1 Descrizione dettagliata

Implementazione VHDL Structural di un generico livello del componente generic_adder. Tale livello è costituito da 2^level addizionatori che lavorano in parallelo, di dimensione dipendente dal livello corrente: N = number_bit_for↩ _operand + log2(number_operand)-level-1.

Parametri

number_operand[in]	parametro che determina il numero di operandi dell' addizionatore
number_bit_for_operand[in]	parametro che determina il numero di bit di ogni operando
level	parametro che determina il livello dell'addizionatore che si sta costruendo
data_in[in]	stringa di bit di input di dimensione (number_operand ∗ number_bit_for_operand) che è la concatenzaione delle somme parziali del livello precedente
data_out[out]	stringa di bit di uscita di dimensione (number_operand ∗ number_bit_for_operand) che è la concatenzaione delle somme parziali per il livello successivo

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/adder_block.vhd

5.2 Behavioral Architecture Reference

Processes

- PROCESS_0**(clock , reset_n , load , data_in)**

Signals

- tmp **std_logic_vector(width- 1 downto 0):=(others=>' 0 ')**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/GenericBuffer.vhd

5.3 Behavioral Architecture Reference

Processes

- clock_process**( )**
- stim_process**( )**

Components

- RMDecoder

Constants

- m **natural:= 4**
- encoded **matrix(0 to 2 ∗∗(m+ 1)- 1):=((x" 0000 "),(x" 5555 "),(x" 3333 "),(x" 6666 "),(x" 0f0f "),(x" 5a5a "),(x" 3c3c "),(x" 6969 "),(x" 00ff "),(x" 55aa "),(x" 33cc "),(x" 6699 "),(x" 0ff0 "),(x" 5aa5 "),(x" 3cc3 "),(x" 6996 "),(x" ffff "),(x" aaaa "),(x" cccc "),(x" 9999 "),(x" f0f0 "),(x" a5a5 "),(x" c3c3 "),(x" 9696 "),(x" ff00 "),(x" aa55 "),(x" cc33 "),(x" 9966 "),(x" f00f "),(x" a55a "),(x" c33c "),(x" 9669 ")**
- clock_period **time:= 10 ns**
- generator_matrix_01 **boolean:=true**

Types

- **matrix**array(naturalrange<>)ofstd_logic_vector(2 **m- 1 downto 0)

Signals

- clock **std_logic:=' 0 '**
- reset_n **std_logic:=' 0 '**
- data_in **std_logic_vector(2 **m- 1 downto 0):=(others=>' 0 ')**
- data_out **std_logic_vector(mdownto 0):=(others=>' 0 ')**

Instantiations

- uut **RMDecoder**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/tb_RMDecoder.vhd

5.4 Behavioral Architecture Reference

Processes

- stim_process**( )**

Components

- RMEncoder

Constants

- period **time:= 10 ns**
- m **natural:= 4**
- generator_matrix_01 **boolean:=true**

Signals

- clock **std_logic:=' 0 '**
- data_in **std_logic_vector(mdownto 0):=(others=>' 0 ')**
- data_out **std_logic_vector(2 **m- 1 downto 0):=(others=>' 0 ')**

Instantiations

- uut **RMEncoder**

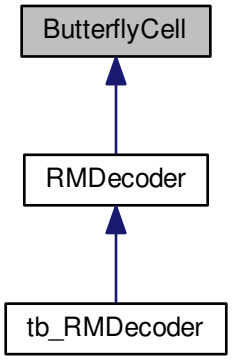
La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMEncoder/tb_RMEncoder.vhd

5.5 ButterflyCell Entity Reference

implementazione VHDL dello swap usato nel decodificatore di Reed-Muller

Diagramma delle classi per ButterflyCell



Entities

- Structural architecture

Libraries

- ieee

Use Clauses

- std_logic_1164

Generics

- m **natural:= 3**

Ports

- data_in **in std_logic_vector(2 ∗∗m- 1 downto 0)**
- swapped **out std_logic_vector(2 ∗∗m- 1 downto 0)**

5.5.1 Descrizione dettagliata

implementazione VHDL dello swap usato nel decodificatore di Reed-Muller

Il componente ButterflyCell implementa la rete di swap necessarie all'implementazione del decodificatore a maggioranza per i codici di Reed-Muller RM(1, m). Il componente ha un'implementazione parametrica, il che permette di usare lo stesso componente qualsiasi sia il parametro "m".

Parametri

m	parametro "m" del codice di Reed-Muller usato.
data_in[in]	vettore contenente il codice di Reed-Muller da swappare. Il parallelismo e' 2^m - 1
swapped[out]	vettore che conterra' il risultato delle operazioni di swapping del vettore data_in

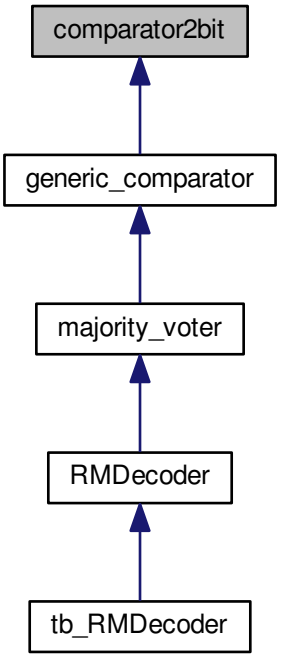
La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/ButterflyCell.vhd

5.6 comparator2bit Entity Reference

Implementazione VHDL Data Flow di un comparatore a 2 bit che tiene conto anche del risutato di un confronto precedente.

Diagramma delle classi per comparator2bit



Entities

- DataFlow architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164

Ports

- a **in STD_LOGIC**
- b **in STD_LOGIC**
- res_in **in STD_LOGIC**
- res_out **out STD_LOGIC**

5.6.1 Descrizione dettagliata

Implementazione VHDL Data Flow di un comparatore a 2 bit che tiene conto anche del risutato di un confronto precedente.

Parametri

a[in]	ingresso 1
b[in]	ingresso 2
res_in[in]	risultato del confronto precedente
res_out[in]	risultato del confronto

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/comparator2bit.vhd

5.7 DataFlow Architecture Reference

Signals

- tmp1 **std_logic:=' 0 '**
- tmp2 **std_logic:=' 0 '**
- tmp3 **std_logic:=' 0 '**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/comparator2bit.vhd

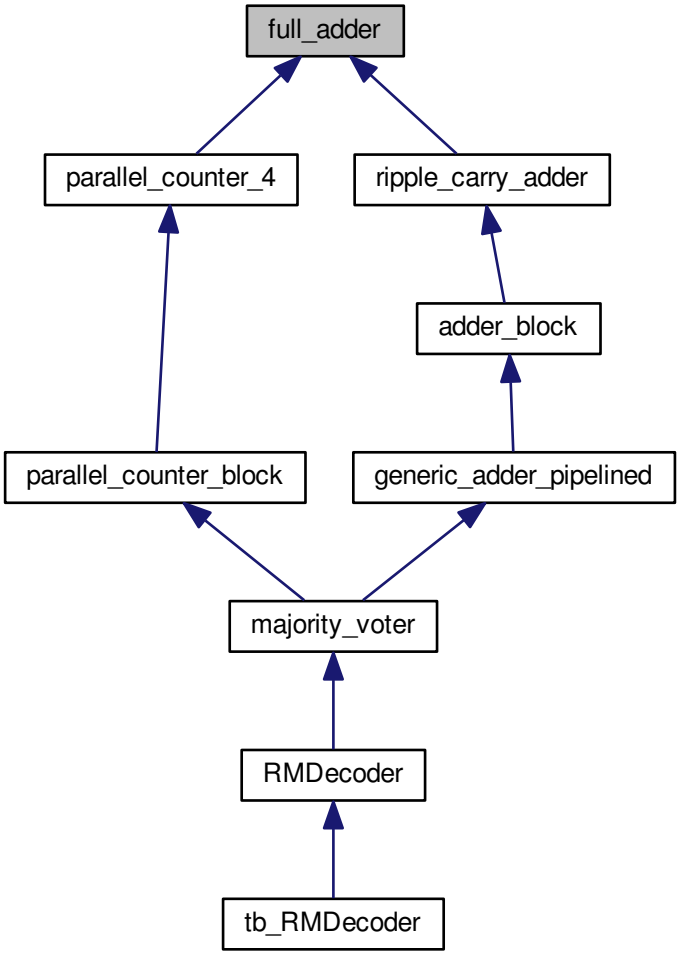
5.8 DataFlow Architecture Reference

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/full_adder.vhd

5.9 full_adder Entity Reference

Diagramma delle classi per full_adder



Entities

- DataFlow architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164

Ports

- add_1 **in STD_LOGIC**
- add_2 **in STD_LOGIC**
- carry_in **in STD_LOGIC**
- carry_out **out STD_LOGIC**
- sum **out STD_LOGIC**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/full_adder.vhd

5.10 generic_adder_pipelined Entity Reference

Implementazione VHDL Structural di un addizionatore generico pipelined : M operandi di N bit.

Diagramma delle classi per generic_adder_pipelined

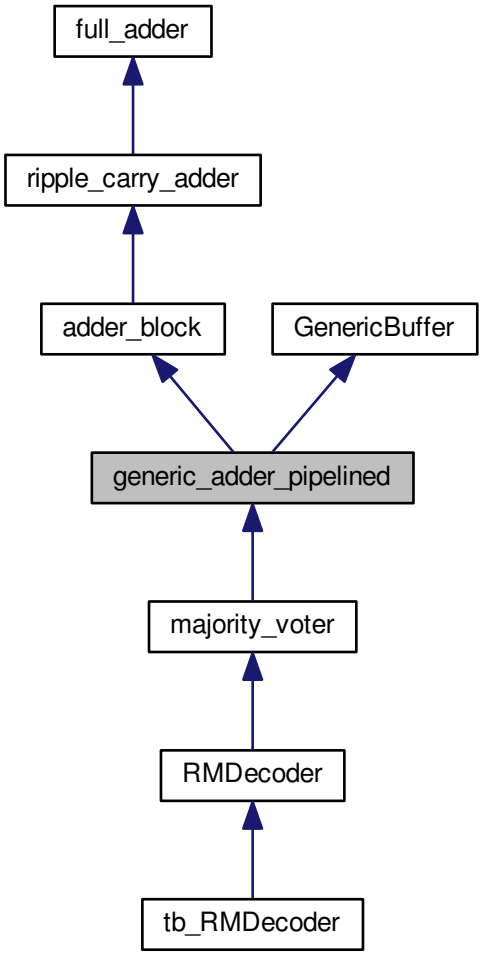
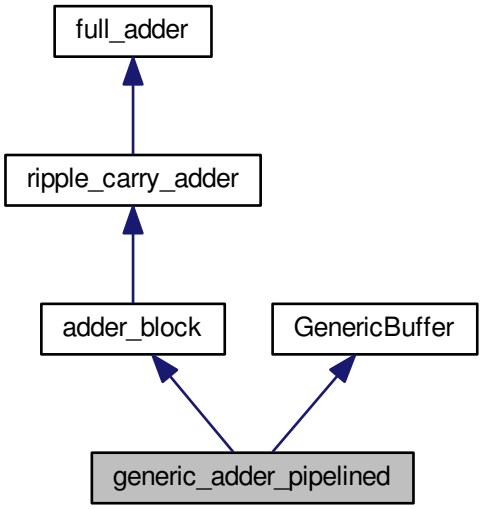


Diagramma di collaborazione per generic_adder_pipelined:



Entities

- Structural architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164
- NUMERIC_STD
- MATH_REAL

Generics

- number_operand **NATURAL:=** 2
- number_bit_for_operand **NATURAL:=** 3

Ports

- clk **in STD_LOGIC**
- reset_n **in STD_LOGIC**
- data_in **in STD_LOGIC_VECTOR((number_operand∗number_bit_for_operand)-** 1 **downto** 0 **)**
- data_out **out STD_LOGIC_VECTOR((number_bit_for_operand+natural(log2(real(number_operand))))-** 1 **downto** 0 **)**

5.10.1 Descrizione dettagliata

Implementazione VHDL Structural di un addizionatore generico pipelined : M operandi di N bit.

Parametri

number_↩ operand[in]	parametro che determina il numero di operandi dell' addizionatore
number_bit_for↩ _operand[in]	parametro che determina il numero di bit di ogni operando
clk[in]	segnale di clock
reset_n[in]	segnale di reset asincrono, attivo basso
data_in[in]	stringa di bit di input di dimensione (number_operand ∗ number_bit_for_operand) che è la concatenzaione degli operandi da sommare
data_out[out]	stringa di bit di uscita di dimensione (number_bit_for_operand + natural(log2(real(number_↩ operand)))) che è la somma totale degli operandi di ingresso

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/generic_adder_pipelined.vhd

5.11 generic_comparator Entity Reference

Implementazione VHDL Structural di un generico comparatore a maggioranza di due stringhe di width bit. Tale implementazione genera una catena di "width" comparatori a 2 bit.

Diagramma delle classi per generic_comparator

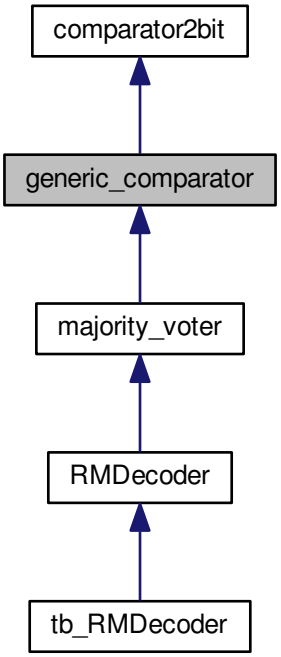
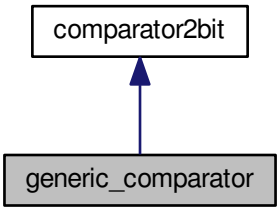


Diagramma di collaborazione per generic_comparator:



Entities

- Structural architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164

Generics

- width **NATURAL:= 8**

Ports

- data_in **in STD_LOGIC_VECTOR(width- 1 downto 0)**
- data_cmp **in STD_LOGIC_VECTOR(width- 1 downto 0)**
- data_out **out STD_LOGIC**

5.11.1 Descrizione dettagliata

Implementazione VHDL Structural di un generico comparatore a maggioranza di due stringhe di width bit. Tale implementazione genera una catena di "width" comparatori a 2 bit.

Parametri

width[in]	parametro che determina la dimensione del comparatore
data_in[in]	stringa di bit di input di dimensione width
data_cmp[in]	stringa di bit di input di dimensione width

data_out[out]	risultato del confronto: data_out = 1 se data_in > data_cmp data_out = 0 altrimenti

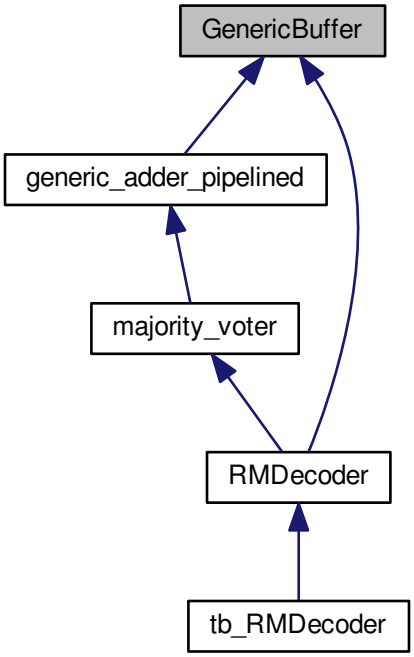
La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/generic_comparator.vhd

5.12 GenericBuffer Entity Reference

Registro di dimensione generica.

Diagramma delle classi per GenericBuffer



Entities

- Behavioral architecture

Libraries

- ieee

Use Clauses

- std_logic_1164

Generics

- width **natural:= 8**
- edge **std_logic:=' 1 '**

Ports

- clock **in std_logic**
- reset_n **in std_logic**
- load **in std_logic**
- data_in **in std_logic_vector(width- 1 downto 0)**
- data_out **out std_logic_vector(width- 1 downto 0)**

5.12.1 Descrizione dettagliata

Registro di dimensione generica.

Parametri

width[in]	numero di bit del registro
edge[in]	fronte di attivo del clock: • '1': fronte di salita • '0': fronte di discesa
clock[in]	segnale di clock
reset_n[in]	reset asincrono, attivo basso
load[in]	segnale di load, quando '1' l'uscita (data_out) segue l'ingresso (data_in)
data_in[in]	ingresso del registro
data_out[out]	uscita del registro

5.12.2 Documentazione dei membri dato

5.12.2.1 ieee [Library]

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩ unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

17-04-2017

Copyright

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/GenericBuffer.vhd

5.13 majority_voter Entity Reference

Implementazione VHDL Structural del majority voter.

Diagramma delle classi per majority_voter

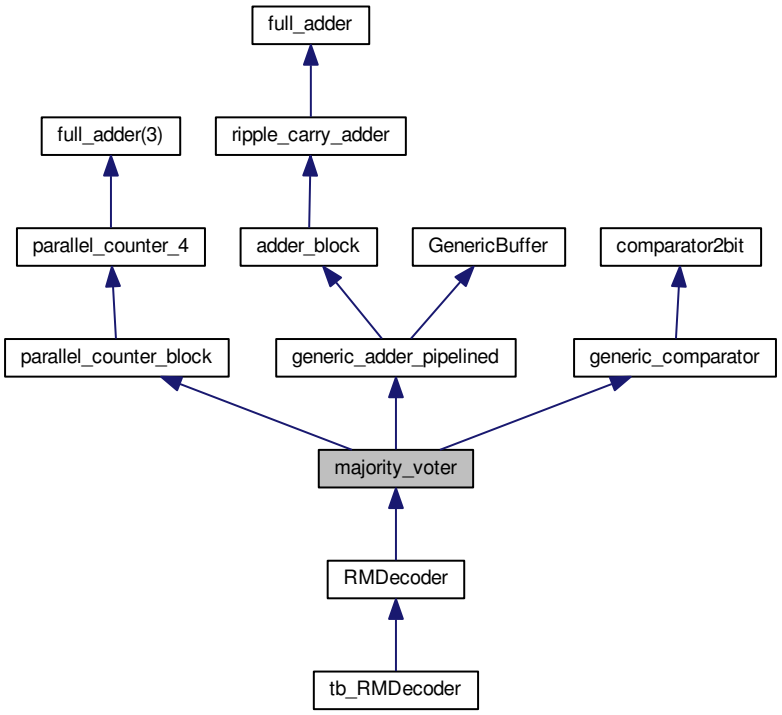
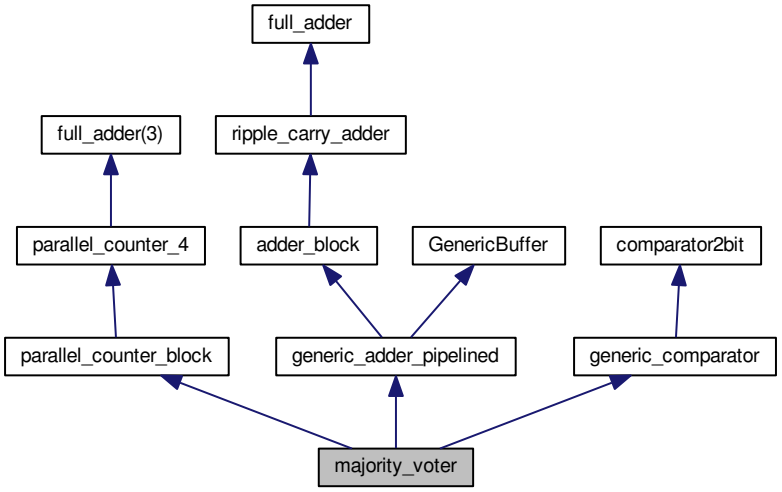


Diagramma di collaborazione per majority_voter:



Entities

- Structural architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164
- NUMERIC_STD
- MATH_REAL

Generics

- width **NATURAL:= 64**

Ports

- clk **in STD_LOGIC**
- reset_n **in STD_LOGIC**
- data_in **in STD_LOGIC_VECTOR(width- 1 downto 0)**
- majority **out STD_LOGIC**

5.13.1 Descrizione dettagliata

Implementazione VHDL Structural del majority voter.

Parametri

width[in]	parametro che determina la dimensione dell'input del componente, width >= 4
clk[in]	segnale di clock
reset_n[in]	segnale di reset asincrono, attivo basso
data_in[in]	stringa di bit di input su cui il componente lavora
majority[out]	bit di uscita : majority = "0" => nella stringa di input #bit = 1 >= #bit = 0 majority = "1" => nella stringa di input #bit = 1 > #bit = 0

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/majority_voter.vhd

5.14 parallel_counter_4 Entity Reference

Diagramma delle classi per parallel_counter_4

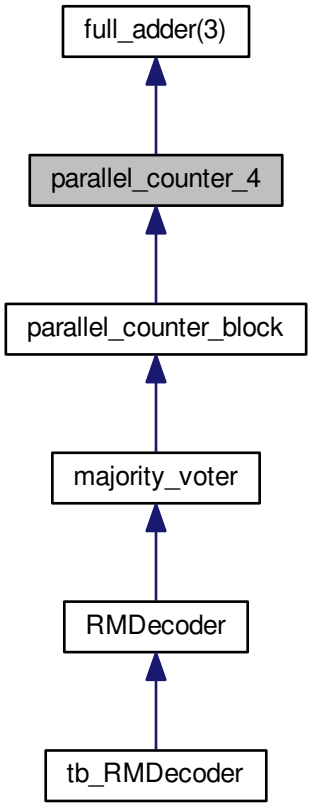
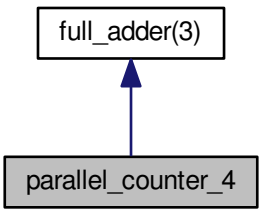


Diagramma di collaborazione per parallel_counter_4:



Entities

- Structural architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164

Ports

- X **in STD_LOGIC_VECTOR(3 downto 0)**
- C0 **out STD_LOGIC**
- C1 **out STD_LOGIC**
- C2 **out STD_LOGIC**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/parallel_counter_4.vhd

5.15 parallel_counter_block Entity Reference

Implementazione VHDL Structural del Modulo 1 : genera width/4 contatori paralleli a 4 bit. Data una stringa di input di width bit, multipla di 4, assegna a ogni contatore un nibble. Ogni contatore parallelo a 4 bit codifica in binario il numero di 1 presente nel nibble di competenza.

Diagramma delle classi per parallel_counter_block

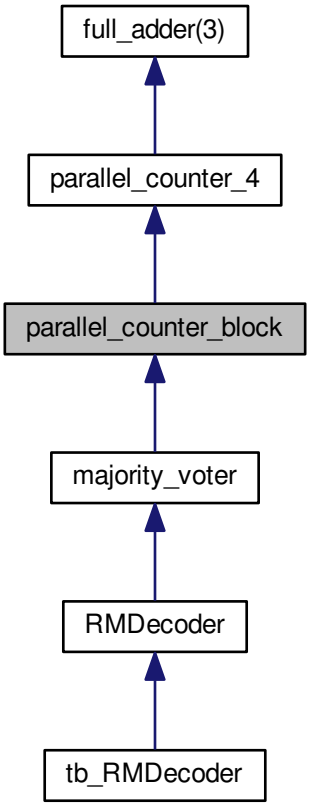
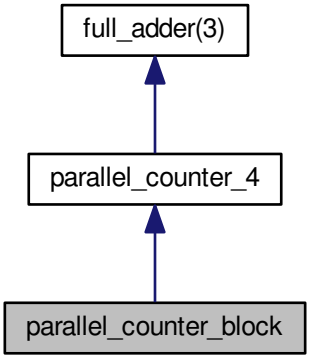


Diagramma di collaborazione per parallel_counter_block:



Entities

- Structural architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164

Generics

- width **NATURAL:=** 8

Ports

- data_in **in STD_LOGIC_VECTOR(width-** 1 **downto** 0 **)**
- data_out **out STD_LOGIC_VECTOR((width-(width/** 4 **))-** 1 **downto** 0 **)**

5.15.1 Descrizione dettagliata

Implementazione VHDL Structural del Modulo 1 : genera width/4 contatori paralleli a 4 bit. Data una stringa di input di width bit, multipla di 4, assegna a ogni contatore un nibble. Ogni contatore parallelo a 4 bit codifica in binario il numero di 1 presente nel nibble di competenza.

Parametri

width[in]	parametro che determina la dimensione dell'input del componente, width multiplo di 4
data_in[in]	stringa di bit di input di dimensione width su cui il componente lavora
data_out[out]	stringa di bit di uscita di dimensione width-(width/4) che è la concatenazione degli output dei singoli contatori paralleli a 4 bit => concatenzaione di stringhe da 3 bit.

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/parallel_counter_block.vhd

5.16 ripple_carry_adder Entity Reference

Implementazione VHDL Structural di un Ripple Carry Adder generico a N bit.

Diagramma delle classi per ripple_carry_adder

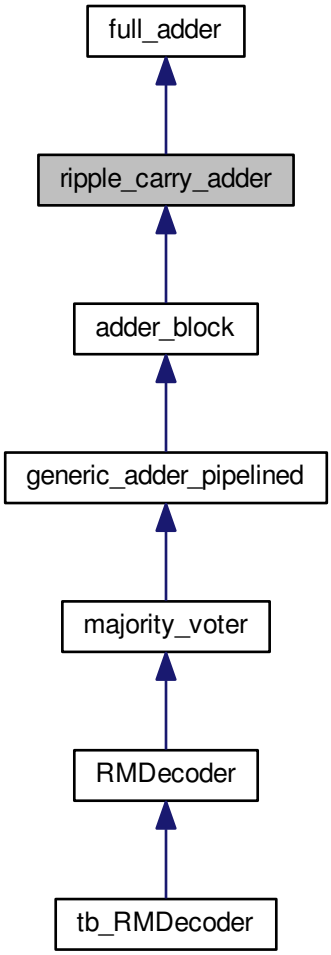
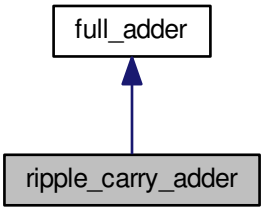


Diagramma di collaborazione per ripple_carry_adder:



Entities

- structural architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164

Generics

- N **natural:=** 4

Ports

- x **in STD_LOGIC_VECTOR(N- 1 downto 0)**
- y **in STD_LOGIC_VECTOR(N- 1 downto 0)**
- carry_in **in STD_LOGIC**
- carry_out **out STD_LOGIC**
- sum **out STD_LOGIC_VECTOR(N- 1 downto 0)**

5.16.1 Descrizione dettagliata

Implementazione VHDL Structural di un Ripple Carry Adder generico a N bit.

Parametri

N[in]	parametro che determina il numero di bit per addendo
x[in]	addendo 1
x[in]	addendo 2
carry_in[in]	carry in ingresso
carry_out[out]	carry in uscita
sum[out]	somma dei due addendi

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/ripple_carry_adder.vhd

5.17 RMDecoder Entity Reference

Implementazione VHDL del decodificatore per codici di Reed-Muller(1,m)

Diagramma delle classi per RMDecoder

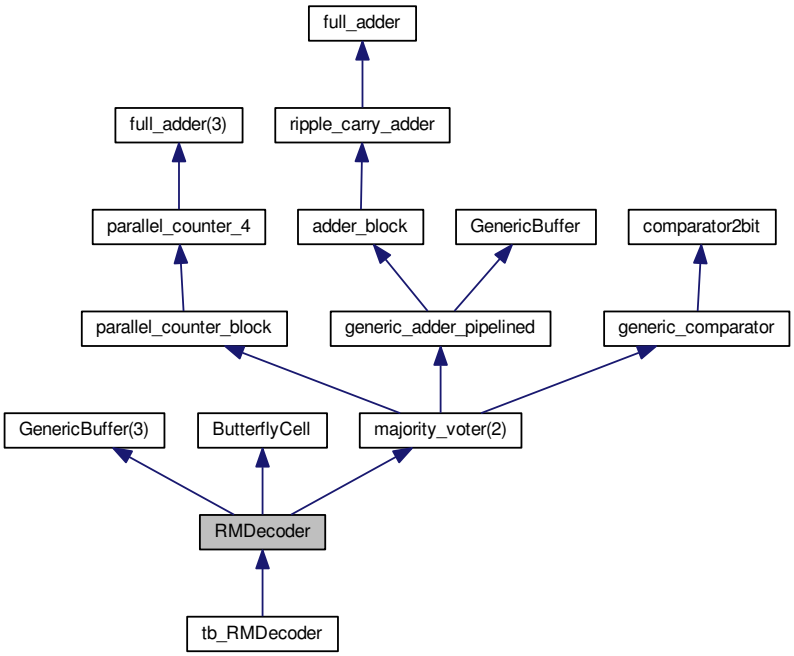
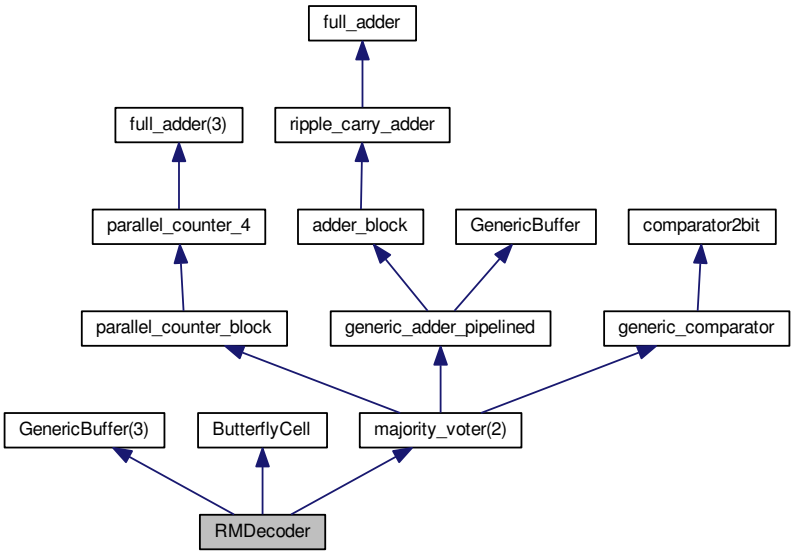


Diagramma di collaborazione per RMDecoder:



Entities

- Structural architecture

Libraries

- ieee

Use Clauses

- std_logic_1164
- numeric_std

Generics

- m **natural:= 6**
- generator_matrix_01 **boolean:=true**

Ports

- clock **in std_logic**
- reset_n **in std_logic**
- data_in **in std_logic_vector(2 ∗∗m- 1 downto 0)**
- data_out **out std_logic_vector(mdownto 0)**

5.17.1 Descrizione dettagliata

Implementazione VHDL del decodificatore per codici di Reed-Muller(1,m)

Tale implementazione fa uso della tecnica con majority-voter ed e' pipelined, con numero di stadi delle pipe variabile in base al particolare codice di Reed-Muller. Il numero totale di stadi della pipe, in funzione di "m", e' 2m-4. Ad esempio, per codici RM(1,7), il numero di stadi della pipe e' 10. Il componente, in questo modo, manifesta, questo si, una latenza di 2m-4 colpi di clock, ma e' potenzialmente in grado di completare una decodifica per colpo di clock. Il seguente esempio istanzia un encoder ed un decoder. L'output dell'encoder viene posto in ingresso al decoder. L'input dell'encoder viene controllato attraverso un VIO. Lo stesso VIO viene usato anche per monitorare l'uscita dell'encoder e l'uscita del decoder, oltre che per controllare il segnale di reset di quest'ultimo.

```
encoder : RMEncoder
    Generic map (   m                   => m,
                    generator_matrix_01 => generator_matrix_01)
    Port map (      data_in             => encoder_data_in,
                    data_out            => encoder_data_out);

decoder : RMDecoder
    Generic map (   m                   => m,
                    generator_matrix_01 => generator_matrix_01)
    Port map (      clock               => clock,
                    reset_n             => reset_n,
                    data_in             => encoder_data_out,
                    data_out            => decoder_data_out);

vio : vio_0
    Port map (  clk             => clock,
                probe_in0       => encoder_data_out,
                probe_in1       => decoder_data_out,
                probe_out0      => encoder_data_in,
                probe_out1(0)   => reset_n);
```

Avvertimento

I codici devono essere stati ottenuti con una matrice di generazione in forma canonica. Vedi il parametro generator_matrix_01.

Parametri

m[in]	parametro "m" del codice di Reed-Muller; incide sulla dimensione, in bit, dell'input e dell' output del componente: l'input sara' 2^m bit, mentre l'output m+1 bit. Oltre che stabilire il particolare codice che e' possibile decifrare, incide sul numero di stadi della pipe di cui il decoder e' composto. Il numero totale di stadi della pipe, in funzione di "m", e' 2(m-3)+1. Ad esempio, per codici RM(1,7), il numero di stadi e' 9.
generator_matrix_01[in]	permette di scegliere la matrice di generazione da usare in fase di decodifica. Scegliendo generator_matrix_01 => true, verra' usata una matrice 1111111111111111 0000000011111111 0000111100001111 0011001100110011 0101010101010101 Se, invece, generator_matrix_01 => false, verra' usata una matrice 1111111111111111 1111111100000000 1111000011110000 1100110011001100 1010101010101010
clock[in]	segnale di clock
reset_n[in]	segnale di reset asincrono, attivo basso
data_in[in]	codice di Reed-Muller RM(1, m) da decodificare, di lunghezza 2^{m} bit
data_out[out]	stringa di bit corrispondente al codice di Reed-Muller RM(1, m) decodificato, di lunghezza pari ad m bit

Test RM(1,m), m=3

data_in	data_out
x"00"	x"0"
x"55"	x"1"
x"33"	x"2"
x"66"	x"3"
x"0f"	x"4"
x"5a"	x"5"
x"3c"	x"6"
x"69"	x"7"
x"ff"	x"8"
x"aa"	x"9"
x"cc"	x"a"
x"99"	x"b"
x"f0"	x"c"
x"a5"	x"d"
x"c3"	x"e"
x"96"	x"f"

RM(1,m), m=4

data_in	data_out
x"0000"	x"00"
x"5555"	x"01"
x"3333"	x"02"
x"6666"	x"03"
x"0f0f"	x"04"

x"5a5a"	x"05"
x"3c3c"	x"06"
x"6969"	x"07"
x"00ff"	x"08"
x"55aa"	x"09"
x"33cc"	x"0a"
x"6699"	x"0b"
x"0ff0"	x"0c"
x"5aa5"	x"0d"
x"3cc3"	x"0e"
x"6996"	x"0f"
x"ffff"	x"10"
x"aaaa"	x"11"
x"cccc"	x"12"
x"9999"	x"13"
x"f0f0"	x"14"
x"a5a5"	x"15"
x"c3c3"	x"16"
x"9696"	x"17"
x"ff00"	x"18"
x"aa55"	x"19"
x"cc33"	x"1a"
x"9966"	x"1b"
x"f00f"	x"1c"
x"a55a"	x"1d"
x"c33c"	x"1e"
x"9669"	x"1f"

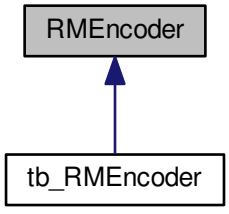
La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/RMDecoder.vhd

5.18 RMEncoder Entity Reference

Implementazione VHDL del codificatore per codici di Reed-Muller(1,m)

Diagramma delle classi per RMEncoder



Entities

- Structural architecture

Libraries

- ieee

Use Clauses

- std_logic_1164
- numeric_std

Generics

- m **natural:= 6**
- generator_matrix_01 **boolean:=true**

Ports

- data_in **in std_logic_vector(mdownto 0)**
- data_out **out std_logic_vector(2 ∗∗m- 1 downto 0)**

5.18.1 Descrizione dettagliata

Implementazione VHDL del codificatore per codici di Reed-Muller(1,m)

Il seguente esempio istanzia un encoder ed un decoder. L'output dell'encoder viene posto in ingresso al decoder. L'input dell'encoder viene controllato attraverso un VIO. Lo stesso VIO viene usato anche per monitorare l'uscita dell'encoder e l'uscita del decoder, oltre che per controllare il segnale di reset di quest'ultimo.

```
encoder : RMEncoder
    Generic map (   m                   => m,
                    generator_matrix_01 => generator_matrix_01)
    Port map (      data_in             => encoder_data_in,
                    data_out            => encoder_data_out);

decoder : RMDecoder
    Generic map (   m                   => m,
                    generator_matrix_01 => generator_matrix_01)
    Port map (      clock               => clock,
                    reset_n             => reset_n,
                    data_in             => encoder_data_out,
                    data_out            => decoder_data_out);

vio : vio_0
    Port map (  clk             => clock,
                probe_in0       => encoder_data_out,
                probe_in1       => decoder_data_out,
                probe_out0      => encoder_data_in,
                probe_out1(0)   => reset_n);
```

Avvertimento

I codici devono essere stati ottenuti con una matrice di generazione in forma canonica. Vedi il parametro generator_matrix_01.

Parametri

m[in]	parametro "m" del codice di Reed-Muller; incide sulla dimensione, in bit, dell'input e dell' output del componente: l'input sara' m+1 bit, mentre l'output 2^m bit.

generator_↩ matrix_01[in]	permette di scegliere la matrice di generazione da usare in fase di decodifica. Scegliendo generator_matrix_01 => true, verra' usata una matrice 1111111111111111 0000000011111111 0000111100001111 0011001100110011 0101010101010101 Se, invece, generator_matrix_01 => false, verra' usata una matrice 1111111111111111 1111111100000000 1111000011110000 1100110011001100 1010101010101010
data_in[in]	stringa di bit, di lunghezza m+1 bit, da codificare come codice di Reed-Muller RM(1, m)
data_out[out]	stringa di bit corrispondente al codice di Reed-Muller RM(1, m) decodificato, di lunghezza pari a 2^m bit

Test RM(1,m), m=3

data_in	data_out
x"0"	x"00"
x"1"	x"55"
x"2"	x"33"
x"3"	x"66"
x"4"	x"0f"
x"5"	x"5a"
x"6"	x"3c"
x"7"	x"69"
x"8"	x"ff"
x"9"	x"aa"
x"a"	x"cc"
x"b"	x"99"
x"c"	x"f0"
x"d"	x"a5"
x"e"	x"c3"
x"f"	x"96"

RM(1,m), m=4

data_in	data_out
x"00"	x"0000"
x"01"	x"5555"
x"02"	x"3333"
x"03"	x"6666"
x"04"	x"0f0f"
x"05"	x"5a5a"
x"06"	x"3c3c"
x"07"	x"6969"
x"08"	x"00ff"
x"09"	x"55aa"
x"0a"	x"33cc"
x"0b"	x"6699"

x"0c"	x"0ff0"
x"0d"	x"5aa5"
x"0e"	x"3cc3"
x"0f"	x"6996"
x"10"	x"ffff"
x"11"	x"aaaa"
x"12"	x"cccc"
x"13"	x"9999"
x"14"	x"f0f0"
x"15"	x"a5a5"
x"16"	x"c3c3"
x"17"	x"9696"
x"18"	x"ff00"
x"19"	x"aa55"
x"1a"	x"cc33"
x"1b"	x"9966"
x"1c"	x"f00f"
x"1d"	x"a55a"
x"1e"	x"c33c"
x"1f"	x"9669"

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMEncoder/RMEncoder.vhd

5.19 Structural Architecture Reference

Components

- ripple_carry_adder

Constants

- N **natural:=number_bit_for_operand+natural(log2(real(number_operand)))-level- 1**

Signals

- tmp_in **std_logic_vector(number_operand∗number_bit_for_operand- 1 downto 0):=(others=>' 0 ')**
- tmp_out **std_logic_vector(number_operand∗number_bit_for_operand- 1 downto 0):=(others=>' 0 ')**

Instantiations

- adder_inst **ripple_carry_adder**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/adder_block.vhd

5.20 Structural Architecture Reference

Components

- full_adder

Signals

- tmp_cout_0 **std_logic:=' 0 '**
- tmp_cout_1 **std_logic:=' 0 '**
- tmp_cout_2 **std_logic:=' 0 '**
- tmp_sum_0 **std_logic:=' 0 '**
- tmp_sum_1 **std_logic:=' 0 '**
- tmp_sum_2 **std_logic:=' 0 '**

Instantiations

- full_adder_inst_0 **full_adder**
- full_adder_inst_1 **full_adder**
- full_adder_inst_2 **full_adder**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/parallel_counter_4.vhd

5.21 Structural Architecture Reference

Signals

- cell0_swapped **std_logic_vector(2 ∗∗(m- 1)- 1 downto 0):=(others=>' 0 ')**
- cell1_swapped **std_logic_vector(2 ∗∗(m- 1)- 1 downto 0):=(others=>' 0 ')**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/ButterflyCell.vhd

5.22 structural Architecture Reference

Components

- full_adder

Signals

- tmp_carry **std_logic_vector(Ndownto 0):=(others=>' 0 ')**

Instantiations

- full_adder_inst **full_adder**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/ripple_carry_adder.vhd

5.23 Structural Architecture Reference

Components

- GenericBuffer
- ButterflyCell
- majority_voter

Constants

- zero **std_logic_vector(2 ∗∗m- 1 downto 0):=(others=>' 0 ')**

Types

- **std_logic_matrix1array(naturalrange<>)ofstd_logic_vector(2 ∗∗m- 1 downto 0)**
- **std_logic_matrix2array(naturalrange<>)ofstd_logic_vector(2 ∗∗(m- 1)- 1 downto 0)**
- **std_logic_matrix3array(naturalrange<>)ofstd_logic_vector(m- 1 downto 0)**

Signals

- generation_matrix **std_logic_matrix1(0 tom)**
- buffered_data_in **std_logic_matrix1(0 tom- 3)**
- swapped_data **std_logic_matrix1(0 tom- 1)**
- coupled_xor **std_logic_matrix2(0 tom- 1)**
- majority **std_logic_vector(m- 1 downto 0):=(others=>' 0 ')**
- majority_m **std_logic:=' 0 '**
- pipe_majority **std_logic_matrix3(0 tom- 2)**
- am_matrix **std_logic_matrix1(0 tom- 1)**
- am_xored_matrix **std_logic_matrix1(0 tom)**

Instantiations

- data_in_buffer **GenericBuffer**
- cell **ButterflyCell**
- voter **majority_voter**
- pipe_buffer **GenericBuffer**
- am_voter **majority_voter**
- buffer_data_out **GenericBuffer**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/RMDecoder.vhd

5.24 Structural Architecture Reference

Constants

- zero **std_logic_vector(2 ∗∗m- 1 downto 0):=(others=>' 0 ')**

Types

- **std_logic_matrix1array(naturalrange<>)ofstd_logic_vector(2 **m- 1 downto 0)**
- **std_logic_matrix2array(naturalrange<>)ofstd_logic_vector(2 **(m- 1)- 1 downto 0)**

Signals

- generation_matrix **std_logic_matrix1(0 tom)**
- am_matrix **std_logic_matrix1(0 tom)**
- am_xored_matrix **std_logic_matrix1(0 tom- 1)**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMEncoder/RMEncoder.vhd

5.25 Structural Architecture Reference

Components

- comparator2bit

Signals

- tmp_res **std_logic_vector(widthdownto 0):=(others=>' 0 ')**

Instantiations

- comparator2bit_inst **comparator2bit**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/generic_comparator.vhd

5.26 Structural Architecture Reference

Components

- adder_block
- GenericBuffer

Types

- **tmp_sum_arrayarray(naturalrange<>)ofstd_logic_vector(number_operand*number_bit_for_↩ operand- 1 downto 0)**

Signals

- tmp_sum **tmp_sum_array(natural(log2(real(number_operand)))downto 0)**
- tmp_sum_buffer **tmp_sum_array(natural(log2(real(number_operand)))- 1 downto 0)**

Instantiations

- adder_block_inst **adder_block**
- buffer_between_adder_block_inst **GenericBuffer**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/generic_adder_pipelined.vhd

5.27 Structural Architecture Reference

Components

- parallel_counter_block
- generic_adder_pipelined
- generic_comparator

Signals

- parallel_counter_block_in **std_logic_vector(width- 1 downto 0):=(others=>' 0 ')**
- parallel_counter_block_out **std_logic_vector((width-(width/ 4))- 1 downto 0):=(others=>' 0 ')**
- generic_adder_in **std_logic_vector((width-(width/ 4))- 1 downto 0):=(others=>' 0 ')**
- generic_adder_out **std_logic_vector(3 +natural(log2(real((width/ 4))))- 1 downto 0):=(others=>' 0 ')**
- data_compare_in **std_logic_vector(3 +natural(log2(real((width/ 4))))- 1 downto 0):=(others=>' 0 ')**
- data_compare_cmp **std_logic_vector(3 +natural(log2(real((width/ 4))))- 1 downto 0):=(others=>' 0 ')**
- data_compare_out **std_logic:=' 0 '**

Instantiations

- parallel_counter_block_inst **parallel_counter_block**
- generic_adder_pipelined_inst **generic_adder_pipelined**
- generic_comparator_inst **generic_comparator**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/majority_voter.vhd

5.28 Structural Architecture Reference

Components

- parallel_counter_4

Signals

- tmp_in_parallel_conter4 **std_logic_vector(width- 1 downto 0)**
- tmp_out_parellel_counter4 **std_logic_vector((width-(width/ 4))- 1 downto 0):=(others=>' 0 ')**

Instantiations

- parallel_counter4_inst **parallel_counter_4**

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/MajorityVoter/parallel_counter_block.vhd

5.29 tb_RMDecoder Entity Reference

Diagramma delle classi per tb_RMDecoder

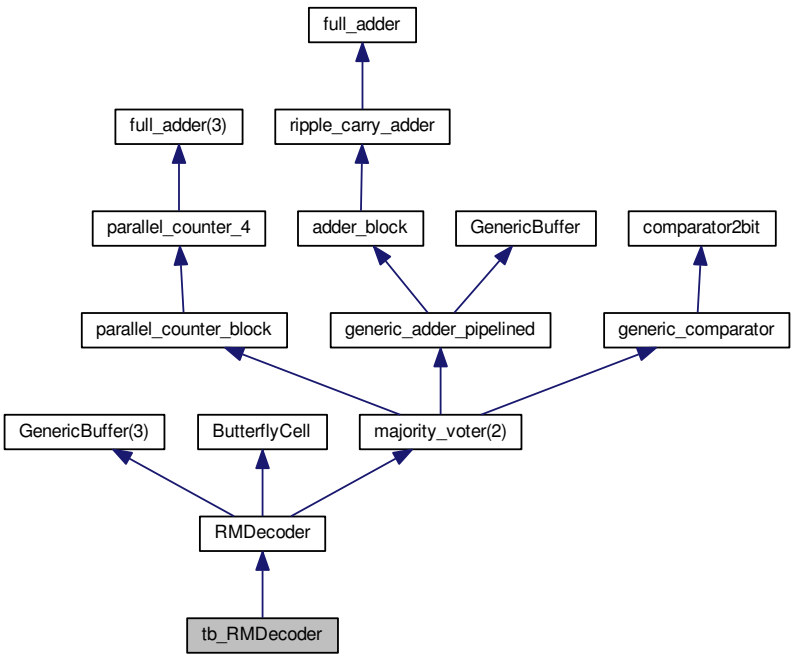
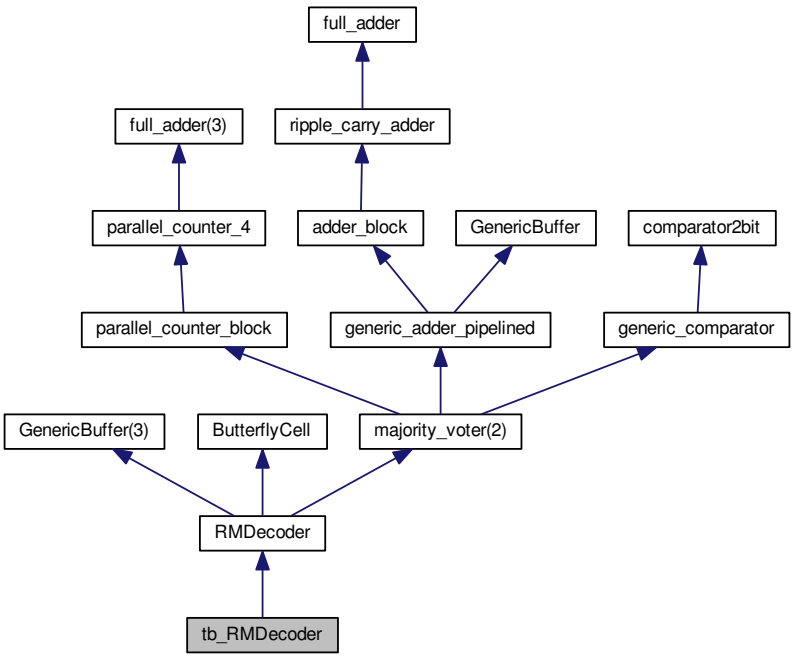


Diagramma di collaborazione per tb_RMDecoder:



Entities

- Behavioral architecture

Libraries

- ieee

: *Salvatore Barone* <salvator.barone@gmail.com, salvator.barone@studenti.unina.it> ↩
: *03-05-2017 : tb_RMDecoder.vhd*

Use Clauses

- std_logic_1164

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMDecoder/tb_RMDecoder.vhd

5.30 tb_RMEncoder Entity Reference

Diagramma delle classi per tb_RMEncoder

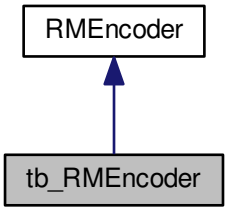
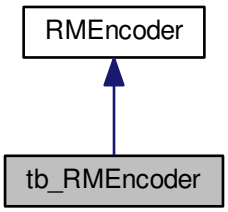


Diagramma di collaborazione per tb_RMEncoder:



Entities

- Behavioral architecture

Libraries

- IEEE

Use Clauses

- STD_LOGIC_1164
- numeric_std

La documentazione per questa classe è stata generata a partire dal seguente file:

- Src/RMEncoder/tb_RMEncoder.vhd

Capitolo 6

Documentazione dei file

6.1 Riferimenti per il file Src/RMDecoder/ButterflyCell.vhd

Entities

- ButterflyCell entity

  implementazione VHDL dello swap usato nel decodificatore di Reed-Muller
- Structural architecture

6.1.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

13 04 2017

Copyright

6.2 Riferimenti per il file Src/RMDecoder/MajorityVoter/adder_block.vhd

Entities

- adder_block entity

  Implementazione VHDL Structural di un generico livello del componente generic_adder. Tale livello è costituito da 2^level addizionatori che lavorano in parallelo, di dimensione dipendente dal livello corrente: N = number_bit_for_↩operand + log2(number_operand)-level-1.
- Structural architecture

6.2.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

15.04.2017 19:51:29

Copyright

6.3 Riferimenti per il file Src/RMDecoder/MajorityVoter/comparator2bit.vhd

Entities

- comparator2bit entity

  Implementazione VHDL Data Flow di un comparatore a 2 bit che tiene conto anche del risutato di un confronto precedente.
- DataFlow architecture

6.3.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

13.04.2017 20:27:06

Copyright

6.4 Riferimenti per il file Src/RMDecoder/MajorityVoter/full_adder.vhd

Entities

- full_adder entity
- DataFlow architecture

6.4.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

12/19/2015 17:27:35

Copyright

6.5 Riferimenti per il file Src/RMDecoder/MajorityVoter/generic_adder_pipelined.vhd

Entities

- generic_adder_pipelined entity

  Implementazione VHDL Structural di un addizionatore generico pipelined : M operandi di N bit.
- Structural architecture

6.5.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

15.04.2017 11:59:47

Copyright

6.6 Riferimenti per il file Src/RMDecoder/MajorityVoter/generic_comparator.vhd

Entities

- generic_comparator entity

Implementazione VHDL Structural di un generico comparatore a maggioranza di due stringhe di width bit. Tale implementazione genera una catena di "width" comparatori a 2 bit.

- Structural architecture

6.6.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

13.04.2017 20:27:06

Copyright

6.7 Riferimenti per il file Src/RMDecoder/MajorityVoter/majority_voter.vhd

Entities

- majority_voter entity

  Implementazione VHDL Structural del majority voter.
- Structural architecture

6.7.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

14.04.2017 00:12:12

Copyright

6.8 Riferimenti per il file Src/RMDecoder/MajorityVoter/parallel_counter_4.vhd

Entities

- parallel_counter_4 entity
- Structural architecture

6.8.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩ unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

12/19/2015 17:27:35

Copyright

6.9 Riferimenti per il file Src/RMDecoder/MajorityVoter/parallel_counter_block.vhd

Entities

- parallel_counter_block entity

  Implementazione VHDL Structural del Modulo 1 : genera width/4 contatori paralleli a 4 bit. Data una stringa di input di width bit, multipla di 4, assegna a ogni contatore un nibble. Ogni contatore parallelo a 4 bit codifica in binario il numero di 1 presente nel nibble di competenza.
- Structural architecture

6.9.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.↩ unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

14.04.2017 00:32:13

Copyright

6.10 Riferimenti per il file Src/RMDecoder/MajorityVoter/ripple_carry_adder.vhd

Entities

- ripple_carry_adder entity

  Implementazione VHDL Structural di un Ripple Carry Adder generico a N bit.
- structural architecture

6.10.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

19.12.2015 17:27:35

Copyright

6.11 Riferimenti per il file Src/RMDecoder/RMDecoder.vhd

Entities

- RMDecoder entity

  Implementazione VHDL del decodificatore per codici di Reed-Muller(1,m)
- Structural architecture

6.11.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

13-04-2017

- implementazione VHDL del decodificatore a maggioranza utilizzabile per la decodifica dei codici di Reed-Muller RM(1, m).

Copyright

6.12 Riferimenti per il file Src/RMEncoder/RMEncoder.vhd

Entities

- RMEncoder entity

  Implementazione VHDL del codificatore per codici di Reed-Muller(1,m)
- Structural architecture

6.12.1 Descrizione dettagliata

Autore

Salvatore Barone salvator.barone@studenti.unina.it Alfonso Di Martino alf.dimartino@studenti.unina.it Pietro Liguori pi.liguori@studenti.unina.it

Data

13-04-2017

- implementazione VHDL del codificatore utilizzabile per la codifica dei codici di Reed-Muller RM(1, m).

Copyright

Indice analitico