

rca\_add\_sub

Corso di ASE anno 18/19

Gruppo 14

PREVITERA GABRIELE

PENNONE MIRKO

PENNA SIMONE



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	full_adder Entity Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.2	half_adder Entity Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.3	overflow_checker Entity Reference . . . . .	6
3.3.1	Detailed Description . . . . .	7
3.3.2	Member Data Documentation . . . . .	7
3.3.2.1	STD_LOGIC_1164 . . . . .	7
3.4	rippleCarry_adder Entity Reference . . . . .	7
3.4.1	Detailed Description . . . . .	8
3.4.2	Member Data Documentation . . . . .	8
3.4.2.1	c_in . . . . .	8
3.4.2.2	c_out . . . . .	8
3.4.2.3	S . . . . .	8
3.4.2.4	STD_LOGIC_1164 . . . . .	9
3.4.2.5	width . . . . .	9
3.4.2.6	Y . . . . .	9
3.5	rippleCarry_addsub Entity Reference . . . . .	9
3.5.1	Detailed Description . . . . .	10
3.5.2	Member Data Documentation . . . . .	10
3.5.2.1	B . . . . .	10
3.5.2.2	overflow . . . . .	10
3.5.2.3	S . . . . .	10
3.5.2.4	subtract . . . . .	10

<b>4 File Documentation</b>	<b>11</b>
4.1 full_adder.vhd File Reference . . . . .	11
4.1.1 Detailed Description . . . . .	11
4.2 half_adder.vhd File Reference . . . . .	11
4.2.1 Detailed Description . . . . .	12
4.3 rippleCarry_addsub.vhd File Reference . . . . .	12
4.3.1 Detailed Description . . . . .	12
<b>Index</b>	<b>13</b>

# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

entity <a href="#">full_adder</a> . . . . .	5
entity <a href="#">half_adder</a>	
Definisco il componente e la sua interfaccia . . . . .	6
entity <a href="#">overflow_checker</a> . . . . .	6
entity <a href="#">rippleCarry_adder</a> . . . . .	7
entity <a href="#">rippleCarry_addsub</a> . . . . .	9



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">full_adder.vhd</a>	Implementazione di un full adder in data flow . . . . .	11
<a href="#">half_adder.vhd</a>	Implementazione di un half adder dataflow . . . . .	11
<a href="#">rippleCarry_addsub.vhd</a>	Sommatore che effettua sia addizione che sottrazione di due operandi (settando subtract) . . .	12





## Chapter 3

# Class Documentation

### 3.1 full\_adder Entity Reference

#### Libraries

- [IEEE](#)

#### Use Clauses

- [STD\\_LOGIC\\_1164](#)

#### Ports

- **x in STD\_LOGIC**  
*full\_adder input : addendo*
- **y in STD\_LOGIC**  
*full\_adder input : addendo*
- **c\_in in STD\_LOGIC**  
*full\_adder input : carry in ingresso*
- **s out STD\_LOGIC**  
*full\_adder output : somma*
- **c\_out out STD\_LOGIC**  
*full\_adder output : carry*

#### 3.1.1 Detailed Description

Descrizione Somma i 3 bit in ingresso (2 addendi e 1 carry in ingresso).  
In uscita abbiamo il risultato della somma sul bit S e il riporto sul bit C.

The documentation for this class was generated from the following file:

- [full\\_adder.vhd](#)

## 3.2 half\_adder Entity Reference

definisco il componente e la sua interfaccia

### Libraries

- [IEEE](#)  
*architecture dataflow of [full\\_adder](#) end*

### Use Clauses

- [STD\\_LOGIC\\_1164](#)

### Ports

- **X in STD\_LOGIC**  
*[half\\_adder](#) input : addendo*
- **Y in STD\_LOGIC**  
*[half\\_adder](#) input : addendo*
- **C out STD\_LOGIC**  
*[half\\_adder](#) output : carry*
- **S out STD\_LOGIC**  
*[half\\_adder](#) output : somma*

### 3.2.1 Detailed Description

definisco il componente e la sua interfaccia

Descrizione Somma i due bit in ingresso.

In uscita abbiamo il risultato della somma sul bit S e il riporto sul bit C.

The documentation for this class was generated from the following file:

- [half\\_adder.vhd](#)

## 3.3 overflow\_checker Entity Reference

### Libraries

- [IEEE](#)  
*architecture dataflow of [half\\_adder](#) end*

### Use Clauses

- [STD\\_LOGIC\\_1164](#)

## Ports

- **a** in STD\_LOGIC  
*bit più significativo (segno) di A*
- **b** in STD\_LOGIC  
*bit più significativo (segno) di B*
- **subtract** in STD\_LOGIC  
*bit di operazione: 1 se sottrazione, 0 se addizione*
- **s** in STD\_LOGIC  
*bit più significativo (segno) di S*
- **overflow** out STD\_LOGIC  
*bit alto se ho una condizione di overflow*

### 3.3.1 Detailed Description

Descrizione La macchina controlla se vi è overflow nel risultato confrontando le cifre più significative (segno) dei due operandi e del risultato con subtract. Ho overflow in caso di:

- somma di due positivi con risultato negativo
- somma di due negativi con risultato positivo
- differenza di positivo e negativo con risultato negativo
- differenza di negativo e positivo con risultato positivo

### 3.3.2 Member Data Documentation

#### 3.3.2.1 STD\_LOGIC\_1164

[STD\\_LOGIC\\_1164](#) [Package]

#### Dependencies:

[full\\_adder](#)

The documentation for this class was generated from the following file:

- [overflow\\_checker.vhd](#)

## 3.4 rippleCarry\_adder Entity Reference

### Libraries

- [IEEE](#)  
*architecture behavioural of [overflow\\_checker](#) end*

## Use Clauses

- [STD\\_LOGIC\\_1164](#)

## Generics

- [width](#) **NATURAL** := 8

## Ports

- [X](#) in **STD\_LOGIC\_VECTOR**([width](#) - 1 downto 0 )
- [Y](#) in **STD\_LOGIC\_VECTOR**([width](#) - 1 downto 0 )
- [c\\_in](#) in **STD\_LOGIC**
- [S](#) out **STD\_LOGIC\_VECTOR**([width](#) - 1 downto 0 )
- [c\\_out](#) out **STD\_LOGIC**

*rippleCarry\_adder* output: carry

### 3.4.1 Detailed Description

Descrizione Somma le 2 stringe di bit in ingresso (2 addendi ) e 1 bit (carry in ingresso). Caratterizzato da una serie di [full\\_adder](#) in cascata che propagano il riporto.

In uscita abbiamo il risultato della somma sul bit S e il riporto sul bit C.

### 3.4.2 Member Data Documentation

#### 3.4.2.1 c\_in

[c\\_in](#) in **STD\_LOGIC** [Port]

*rippleCarry\_adder* input: addendo

#### 3.4.2.2 c\_out

[c\\_out](#) out **STD\_LOGIC** [Port]

*rippleCarry\_adder* output: carry

*rippleCarry\_adder* output: somma

#### 3.4.2.3 S

[S](#) out **STD\_LOGIC\_VECTOR**([width](#) - 1 downto 0 ) [Port]

*rippleCarry\_adder* input : carry in ingresso

#### 3.4.2.4 STD\_LOGIC\_1164

`STD_LOGIC_1164` [Package]

##### Dependencies:

`full_adder`

#### 3.4.2.5 width

`width NATURAL := 8` [Generic]

usato per definire il parallelismo del `rippleCarry_adder`

#### 3.4.2.6 Y

`Y in STD_LOGIC_VECTOR(width - 1 downto 0)` [Port]

`rippleCarry_adder` input: addendo

The documentation for this class was generated from the following file:

- `rippleCarry_adder.vhd`

## 3.5 rippleCarry\_addsub Entity Reference

### Libraries

- `IEEE`

### Use Clauses

- `STD_LOGIC_1164`

### Generics

- `width NATURAL := 8`  
*usato per definire il parallelismo del sommatore*

### Ports

- `A in STD_LOGIC_VECTOR(width - 1 downto 0)`
- `B in STD_LOGIC_VECTOR(width - 1 downto 0)`
- `subtract in STD_LOGIC`
- `S out STD_LOGIC_VECTOR(width - 1 downto 0)`
- `overflow out STD_LOGIC`  
*`rippleCarry_addsub` output: condizione di overflow*

### 3.5.1 Detailed Description

Descrizione Effettua la somma o la sottrazione di due stringhe di bit (A e B) di lunghezza width in ingresso. Il bit SUBTRACT in ingresso sarà 0 in caso di una somma, 1 in caso di una differenza. Il secondo operando del RCA (Y) è determinato dalla XOR tra SUBTRACT e B: se subtract = 1, ne farà il complemento. Il subtract è portato anche come c\_in del RCA, in modo tale che, se 1, B diventerà  $!B + 1 = -B$ , e dunque il RCA farà  $(A - B)$ . Se subtract è 0, Y sarà B e c\_in sarà 0 (eseguirà  $A + B$ ). L'overflow è alto se ho una somma di numeri positivi e risultato negativo, somma di numeri negativi e risultato positivo, differenza positivo e negativo con risultato negativo o differenza negativo e positivo con risultato positivo. Usiamo una macchina [overflow\\_checker](#) che fa tale controllo usando i bit più significativi (segno) di A, B, S e il bit di subtract.

### 3.5.2 Member Data Documentation

#### 3.5.2.1 B

`B in STD_LOGIC_VECTOR(width - 1 downto 0 )` [Port]

[rippleCarry\\_addsub](#) input: addendo

#### 3.5.2.2 overflow

`overflow out STD_LOGIC` [Port]

[rippleCarry\\_addsub](#) output: condizione di overflow

[rippleCarry\\_addsub](#) output: risultato

#### 3.5.2.3 S

`S out STD_LOGIC_VECTOR(width - 1 downto 0 )` [Port]

[rippleCarry\\_addsub](#) input: subtract sarà 0 per somma, 1 per differenza

#### 3.5.2.4 subtract

`subtract in STD_LOGIC` [Port]

[rippleCarry\\_addsub](#) input: addendo

The documentation for this class was generated from the following file:

- [rippleCarry\\_addsub.vhd](#)

## Chapter 4

# File Documentation

### 4.1 full\_adder.vhd File Reference

Implementazione di un full adder in data flow.

#### Entities

- [full\\_adder](#) entity

#### 4.1.1 Detailed Description

Implementazione di un full adder in data flow.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

### 4.2 half\_adder.vhd File Reference

Implementazione di un half adder dataflow.

## Entities

- [half\\_adder](#) entity  
*definisco il componente e la sua interfaccia*

### 4.2.1 Detailed Description

Implementazione di un half adder dataflow.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

## 4.3 rippleCarry\_addsub.vhd File Reference

sommatore che effettua sia addizione che sottrazione di due operandi (settando subtract)

## Entities

- [rippleCarry\\_addsub](#) entity

### 4.3.1 Detailed Description

sommatore che effettua sia addizione che sottrazione di due operandi (settando subtract)

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings



# Index

## B

rippleCarry\_addsub, [10](#)

## c\_in

rippleCarry\_adder, [8](#)

## c\_out

rippleCarry\_adder, [8](#)

full\_adder, [5](#)

full\_adder.vhd, [11](#)

half\_adder, [6](#)

half\_adder.vhd, [11](#)

## overflow

rippleCarry\_addsub, [10](#)

overflow\_checker, [6](#)

STD\_LOGIC\_1164, [7](#)

rippleCarry\_adder, [7](#)

c\_in, [8](#)

c\_out, [8](#)

S, [8](#)

STD\_LOGIC\_1164, [8](#)

width, [9](#)

Y, [9](#)

rippleCarry\_addsub, [9](#)

B, [10](#)

overflow, [10](#)

S, [10](#)

subtract, [10](#)

rippleCarry\_addsub.vhd, [12](#)

## S

rippleCarry\_adder, [8](#)

rippleCarry\_addsub, [10](#)

STD\_LOGIC\_1164

overflow\_checker, [7](#)

rippleCarry\_adder, [8](#)

## subtract

rippleCarry\_addsub, [10](#)

## width

rippleCarry\_adder, [9](#)

## Y

rippleCarry\_adder, [9](#)