

RobertsonMultiplier

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	carrySelect_adder Entity Reference . . . . .	5
3.2	carrySelect_addSub Entity Reference . . . . .	6
3.3	carrySelect_cell Entity Reference . . . . .	6
3.4	counter_UpN_Re_Sr Entity Reference . . . . .	7
3.5	flipflop_d_risingEdge_asyncReset Entity Reference . . . . .	8
3.5.1	Detailed Description . . . . .	8
3.5.2	Member Data Documentation . . . . .	8
3.5.2.1	IEEE . . . . .	9
3.5.2.2	STD_LOGIC_1164 . . . . .	9
3.6	flipflopmux Entity Reference . . . . .	9
3.6.1	Detailed Description . . . . .	10
3.7	full_adder Entity Reference . . . . .	10
3.7.1	Detailed Description . . . . .	10
3.7.2	Member Data Documentation . . . . .	10
3.7.2.1	IEEE . . . . .	10
3.7.2.2	STD_LOGIC_1164 . . . . .	11
3.8	mux2_1 Entity Reference . . . . .	11

3.8.1	Detailed Description	11
3.8.2	Member Data Documentation	12
3.8.2.1	STD_LOGIC_1164	12
3.9	overflow_checker Entity Reference	12
3.9.1	Detailed Description	12
3.9.2	Member Data Documentation	13
3.9.2.1	STD_LOGIC_1164	13
3.10	register_d_Re_Ar Entity Reference	13
3.10.1	Detailed Description	14
3.10.2	Member Data Documentation	14
3.10.2.1	STD_LOGIC_1164	14
3.11	rippleCarry_adder Entity Reference	14
3.11.1	Detailed Description	15
3.11.2	Member Data Documentation	15
3.11.2.1	c_in	15
3.11.2.2	c_out	15
3.11.2.3	S	15
3.11.2.4	STD_LOGIC_1164	15
3.11.2.5	width	15
3.11.2.6	Y	16
3.12	robertson_control_unit Entity Reference	16
3.13	robertson_multiplier Entity Reference	17
3.14	scan_chain Entity Reference	17
3.14.1	Detailed Description	18
<b>4</b>	<b>File Documentation</b>	<b>19</b>
4.1	carrySelect_adder.vhd File Reference	19
4.1.1	Detailed Description	19
4.2	carrySelect_addSub.vhd File Reference	19
4.2.1	Detailed Description	20
4.3	carrySelect_cell.vhd File Reference	20
4.3.1	Detailed Description	20
4.4	counter_UpN_Re_Sr.vhd File Reference	21
4.4.1	Detailed Description	21
4.5	flipflopmux.vhd File Reference	21
4.5.1	Detailed Description	21
4.6	robertson_control_unit.vhd File Reference	22
4.6.1	Detailed Description	22
4.7	robertson_multiplier.vhd File Reference	22
4.7.1	Detailed Description	22
4.8	scan_chain.vhd File Reference	23
4.8.1	Detailed Description	23
<b>Index</b>		<b>25</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

entity <a href="#">carrySelect_adder</a> . . . . .	5
entity <a href="#">carrySelect_addSub</a> . . . . .	6
entity <a href="#">carrySelect_cell</a> . . . . .	6
entity <a href="#">counter_UpN_Re_Sr</a> . . . . .	7
entity <a href="#">flipflop_d_risingEdge_asyncReset</a> Flipflop_d_risingEdge_asyncReset implementa un flipflop di tipo d che commuta sul fronte di salita, con segnale di enable e reset asincrono . . . . .	8
entity <a href="#">flipflopmux</a> . . . . .	9
entity <a href="#">full_adder</a> . . . . .	10
entity <a href="#">mux2_1</a> Definisco il componente e la sua interfaccia . . . . .	11
entity <a href="#">overflow_checker</a> . . . . .	12
entity <a href="#">register_d_Re_Ar</a> Registro di diensione "width" che prende in ingresso un dato D e lo memorizza . . . . .	13
entity <a href="#">rippleCarry_adder</a> . . . . .	14
entity <a href="#">robertson_control_unit</a> . . . . .	16
entity <a href="#">robertson_multiplier</a> . . . . .	17
entity <a href="#">scan_chain</a> . . . . .	17



## Chapter 2

# File Index

### 2.1 File List

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

<a href="#">carrySelect_adder.vhd</a>	
Sommatore Carry Select . . . . .	19
<a href="#">carrySelect_addSub.vhd</a>	
Sommatore Carry Select in grado di effettuare anche l'operazione di sottrazione . . . . .	19
<a href="#">carrySelect_cell.vhd</a>	
Singolo blocco di un sommatore carry Select . . . . .	20
<a href="#">counter_UpN_Re_Sr.vhd</a>	
Contatore modulo N . . . . .	21
<a href="#">flipflopmux.vhd</a>	
Flip flop D con multiplexer . . . . .	21
<a href="#">robertson_control_unit.vhd</a>	
Unità di controllo del moltiplicatore di Robertson . . . . .	22
<a href="#">robertson_multiplier.vhd</a>	
Moltiplicatore di Robertson che implementa l'algoritmo di Robertson . . . . .	22
<a href="#">scan_chain.vhd</a>	
Registro di n flip flop D multiplexati . . . . .	23





## Chapter 3

# Class Documentation

### 3.1 carrySelect\_adder Entity Reference

#### Libraries

- [IEEE](#)

#### Use Clauses

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

#### Generics

- **M NATURAL:= 4**  
*M parallelismo dei ripplecarry adder.*
- **P NATURAL:= 2**  
*P parallelismo delle celle dell carry select Come metto M e P, marco e co fanno la stima dei tempi e mettono solo (M\*P) da cui ricavano poi M e P io direi di fare una versione con M e P espliciti e una versione come l'hanno fatta loro, ma su quella.*

#### Ports

- **A in STD\_LOGIC\_VECTOR(((M \*P)- 1 )downto 0 )**  
*input addendo*
- **B in STD\_LOGIC\_VECTOR(((M \*P)- 1 )downto 0 )**  
*input addendo*
- **c\_in in STD\_LOGIC**  
*input carry in ingresso*
- **S out STD\_LOGIC\_VECTOR(((M \*P)- 1 )downto 0 )**  
*output somma*
- **c\_out out STD\_LOGIC**  
*output carry in uscita*

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

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

## 3.2 carrySelect\_addSub Entity Reference

### Libraries

- [IEEE](#)

### Use Clauses

- [STD\\_LOGIC\\_1164](#)
- [math\\_real](#)
- [numeric\\_std](#)

### Generics

- **M NATURAL** := 4
- **P NATURAL** := 2

*P parallelismo delle celle dell carry select Come metto M e P, marco e co fanno la stima dei tempi e mettono solo width da cui ricavano poi M e P io direi di fare una versione con M e P espliciti e una versione come l'hanno fatta loro, ma su quella.*

### Ports

- **A** in **STD\_LOGIC\_VECTOR**((**M**\***P**)- 1 )downto 0 )  
*input addendo*
- **B** in **STD\_LOGIC\_VECTOR**((**M**\***P**)- 1 )downto 0 )  
*input addendo*
- **subtract** in **STD\_LOGIC**
- **S** out **STD\_LOGIC\_VECTOR**((**M**\***P**)- 1 )downto 0 )  
*output somma*
- **overflow** out **STD\_LOGIC**
- **c\_out** out **STD\_LOGIC**  
*output carry in uscita*

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

- [carrySelect\\_addSub.vhd](#)

## 3.3 carrySelect\_cell Entity Reference

### Libraries

- [IEEE](#)

### Use Clauses

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

### Generics

- `width` **NATURAL** := 4

### Ports

- `A` in **STD\_LOGIC\_VECTOR**((width- 1 )downto 0 )
- `B` 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**

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

- [carrySelect\\_cell.vhd](#)

## 3.4 counter\_UpN\_Re\_Sr Entity Reference

### Libraries

- [IEEE](#)

### Use Clauses

- [STD\\_LOGIC\\_1164](#)
- [numeric\\_std](#)
- [math\\_real](#)

### Generics

- `n` **NATURAL** := 2
- `enable_level` **STD\_LOGIC** := ' 1 '

### Ports

- `enable` in **STD\_LOGIC**  
*enable input*
- `reset_n` in **STD\_LOGIC**  
*reset input*
- `clock` in **STD\_LOGIC**  
*clock input*
- `count_hit` out **STD\_LOGIC**  
*count\_hit output*
- `COUNTS` out **STD\_LOGIC\_VECTOR**((integer(ceil(log2(real(n))))- 1 )downto 0 )  
*COUNT output.*

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

- [counter\\_UpN\\_Re\\_Sr.vhd](#)

## 3.5 flipflop\_d\_risingEdge\_asyncReset Entity Reference

[flipflop\\_d\\_risingEdge\\_asyncReset](#) implementa un flipflop di tipo d che commuta sul fronte di salita, con segnale di enable e reset asincrono.

### Libraries

- [IEEE](#)

### Use Clauses

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

### Generics

- [init\\_value](#) **STD\_LOGIC:= '0'**  
*definisce il livello iniziale del flipflop*
- [reset\\_level](#) **STD\_LOGIC:= '0'**  
*definisce il livello reset*
- [enable\\_level](#) **STD\_LOGIC:= '1'**  
*definisce il livello enable*

### Ports

- [clock](#) **in STD\_LOGIC**  
*flipflop\_d\_risingEdge\_asyncReset input : segnale di clock per sincronizzare*
- [enable](#) **in STD\_LOGIC**  
*flipflop\_d\_risingEdge\_asyncReset input : segnale enable*
- [reset](#) **in STD\_LOGIC**  
*flipflop\_d\_risingEdge\_asyncReset input : segnale reset*
- [d](#) **in STD\_LOGIC**  
*flipflop\_d\_risingEdge\_asyncReset input : input data*
- [q](#) **out STD\_LOGIC**  
*flipflop\_d\_risingEdge\_asyncReset output : output data*

#### 3.5.1 Detailed Description

[flipflop\\_d\\_risingEdge\\_asyncReset](#) implementa un flipflop di tipo d che commuta sul fronte di salita, con segnale di enable e reset asincrono.

#### 3.5.2 Member Data Documentation

### 3.5.2.1 IEEE

[IEEE](#) [Library]

FEDERICO II , CORSO DI ASE 18/19, Gruppo 14 –

### 3.5.2.2 STD\_LOGIC\_1164

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

last changes: <14/11/2018> <13/11/2018> <log> create

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

- `flipflop_d_risingEdge_asyncReset.vhd`

## 3.6 flipflopmux Entity Reference

### Libraries

- [IEEE](#)  
*architecture behavioural end*

### Use Clauses

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

### Ports

- [clock](#) in **STD\_LOGIC**  
*clock*
- [en](#) in **STD\_LOGIC**  
*enable*
- [reset\\_n](#) in **STD\_LOGIC**  
*reset*
- [scan\\_en](#) in **STD\_LOGIC**  
*segnale di selezione del multiplexer per modalità (0 = normale, 1 = controllo)*
- [d](#) in **STD\_LOGIC**  
*ingresso del flipflop in modalità normale*
- [scan\\_in](#) in **STD\_LOGIC**  
*ingresso del flipflop in modalità controllo*
- [q](#) out **STD\_LOGIC**  
*uscita del flipflop*

### 3.6.1 Detailed Description

flipflopmux è un flip flop D con multiplexer: scan\_en è il segnale di controllo del multiplexer, se scan\_en = 0 l'ingresso è d, se scan\_en = 1 l'ingresso è scan\_in.

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

- [flipflopmux.vhd](#)

## 3.7 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*

### 3.7.1 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.

### 3.7.2 Member Data Documentation

#### 3.7.2.1 IEEE

[IEEE](#) [Library]

## 3.7.2.2 STD\_LOGIC\_1164

STD\_LOGIC\_1164 [Package]

last changes: &lt;11/11/2018&gt; &lt;15/10/2018&gt; &lt;log&gt; Aggiunta doc doxygen

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

- full\_adder.vhd

## 3.8 mux2\_1 Entity Reference

definisco il componente e la sua interfaccia

## Libraries

- IEEE  
*architecture dataflow of full\_adder end*

## Use Clauses

- STD\_LOGIC\_1164

## Generics

- width natural:= 1  
*parallelismo dell' I/O del multiplexer*

## Ports

- SEL in STD\_LOGIC  
*mux2\_1 input: selezione*
- A in STD\_LOGIC\_VECTOR((width - 1)downto 0 )  
*mux2\_1 input: A*
- B in STD\_LOGIC\_VECTOR((width - 1)downto 0 )  
*mux2\_1 input: B*
- X out STD\_LOGIC\_VECTOR((width - 1)downto 0 )  
*mux2\_1 output: X*

## 3.8.1 Detailed Description

definisco il componente e la sua interfaccia

Descrizione Quando l'ingresso SEL è basso, l'uscita assume il valore del segnale A, altrimenti quando il segnale SEL è alto l'uscita assume il valore del segnale B.

### 3.8.2 Member Data Documentation

#### 3.8.2.1 STD\_LOGIC\_1164

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

last changes: <14/11/2018> <13/11/2018> <log> create

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

- mux2\_1.vhd

## 3.9 overflow\_checker Entity Reference

### Libraries

- [IEEE](#)  
*architecture dataflow of [mux2\\_1](#) 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*

#### 3.9.1 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



### 3.9.2 Member Data Documentation

#### 3.9.2.1 STD\_LOGIC\_1164

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

last changes: <11/11/2018> <15/10/2018> <log> Aggiunta doc doxygen

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

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

## 3.10 register\_d\_Re\_Ar Entity Reference

Registro di diensione "width" che prende in ingresso un dato D e lo memorizza.

### Libraries

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

### Use Clauses

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

### Generics

- [width](#) **NATURAL** := **8**  
*definisce il parallelismo del registro*
- [reset\\_level](#) **STD\_LOGIC** := '**0**'  
*definisce il livello reset*
- [enable\\_level](#) **STD\_LOGIC** := '**1**'  
*definisce il livello enable*

### Ports

- [clock](#) in **STD\_LOGIC**  
*[register\\_d\\_Re\\_Ar](#) input : segnale di clock per sincronizzare*
- [enable](#) in **STD\_LOGIC**  
*[register\\_d\\_Re\\_Ar](#) input : segnale enable*
- [reset](#) in **STD\_LOGIC**  
*[register\\_d\\_Re\\_Ar](#) input : segnale reset*
- [d](#) in **STD\_LOGIC\_VECTOR**([width](#) - **1** downto **0**)  
*[register\\_d\\_Re\\_Ar](#) input : inpput data*
- [q](#) out **STD\_LOGIC\_VECTOR**([width](#) - **1** downto **0**)  
*[register\\_d\\_Re\\_Ar](#) input : output data*

### 3.10.1 Detailed Description

Registro di diensione "width" che prende in ingresso un dato D e lo memorizza.

### 3.10.2 Member Data Documentation

#### 3.10.2.1 STD\_LOGIC\_1164

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

last changes: <16/11/2018> <16/11/2018> <log> create

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

- [register\\_d\\_Re\\_Ar.vhd](#)

## 3.11 rippleCarry\_adder Entity Reference

### Libraries

- [IEEE](#)  
*architecture behavioral of [register\\_d\\_Re\\_Ar](#) 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*

### 3.11.1 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.

### 3.11.2 Member Data Documentation

#### 3.11.2.1 c\_in

`c_in in STD_LOGIC [Port]`

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

#### 3.11.2.2 c\_out

`c_out out STD_LOGIC [Port]`

[rippleCarry\\_adder](#) output: carry

[rippleCarry\\_adder](#) output: somma

#### 3.11.2.3 S

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

[rippleCarry\\_adder](#) input : carry in ingresso

#### 3.11.2.4 STD\_LOGIC\_1164

`STD_LOGIC_1164 [Package]`

last changes: <11/11/2018> <15/10/2018> <log> Aggiunta doc doxygen

#### 3.11.2.5 width

`width NATURAL := 8 [Generic]`

usato per definire il parallelismo del [rippleCarry\\_adder](#)

## 3.11.2.6 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`

## 3.12 robertson\_control\_unit Entity Reference

### Libraries

- [IEEE](#)

### Use Clauses

- [STD\\_LOGIC\\_1164](#)
- [numeric\\_std](#)
- [math\\_real](#)

### Generics

- **N NATURAL := 8**  
*parallelismo di X*

### Ports

- `clock in STD_LOGIC`
- `start in STD_LOGIC`
- `reset_n in STD_LOGIC`
- `current_multiplicand in STD_LOGIC`  
*moltiplicando corrente*
- `counter_hit in STD_LOGIC`  
*segnala la fine della moltiplicazione*
- `stop out STD_LOGIC`
- `en_a out STD_LOGIC`  
*se scan\_en = 1 la scan chain funziona come shifter register*
- `en_q out STD_LOGIC`
- `en_m out STD_LOGIC`
- `shift out STD_LOGIC`
- `subtract out STD_LOGIC`
- `count_up out STD_LOGIC`
- `sel out STD_LOGIC`  
*sel pilota il secondo input dell'adder 0 in input è Y 1 sono tutti 0*
- `reset_a out STD_LOGIC`
- `reset_count out STD_LOGIC`  
*reset il conteggio*

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

- `robertson_control_unit.vhd`
- `robertson_control_unit_old.vhd`

## 3.13 robertson\_multiplier Entity Reference

### Libraries

- [IEEE](#)

### Use Clauses

- [STD\\_LOGIC\\_1164](#)
- [numeric\\_std](#)
- [math\\_real](#)

### Generics

- [N](#) **INTEGER:= 8**

### Ports

- [X](#) in **STD\_LOGIC\_VECTOR(N- 1 downto 0 )**
- [Y](#) in **STD\_LOGIC\_VECTOR(N- 1 downto 0 )**
- [start](#) in **STD\_LOGIC**
- [clock](#) in **STD\_LOGIC**
- [reset\\_n](#) in **STD\_LOGIC**
- [stop](#) out **STD\_LOGIC**
- [Z](#) out **STD\_LOGIC\_VECTOR(( 2 \*N)- 1 downto 0 )**

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

- [robertson\\_multiplier.vhd](#)

## 3.14 scan\_chain Entity Reference

### Libraries

- [IEEE](#)

### Use Clauses

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

### Generics

- [width](#) **integer:= 8**  
*dimensione del registro*
- [shift\\_direction](#) **std\_logic:= ' 1 '**  
*shift a sinistra*

## Ports

- **clock** in STD\_LOGIC  
*segnale clock di tempificazione*
- **en** in STD\_LOGIC  
*segnale di abilitazione 1-attivo*
- **reset\_n** in STD\_LOGIC  
*segnale di reset 0-attivo*
- **scan\_en** in STD\_LOGIC  
*segnale di selezione modalità (0 = normale, 1 = controllo)*
- **scan\_in** in STD\_LOGIC  
*primo valore scan-in*
- **d\_reg** in STD\_LOGIC\_VECTOR(**width - 1** downto **0**)  
*valore in ingresso nel registro*
- **scan\_out** out STD\_LOGIC  
*ultimo valore scan-out*
- **q\_reg** out STD\_LOGIC\_VECTOR(**width - 1** downto **0**)  
*valore in uscita del registro*

### 3.14.1 Detailed Description

Scan chain è un registro di width flipflop D multiplexati. Quando scan\_en = 0, il componente si comporta come un normale registro. Quando scan\_en = 1, diventa uno shift register che shifta ad ogni colpo di clock. La direzione dello shift è regolata dal generic shift\_direction (0 = right, 1 = left)

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

- [scan\\_chain.vhd](#)

## Chapter 4

# File Documentation

### 4.1 carrySelect\_adder.vhd File Reference

Sommatore Carry Select.

#### Entities

- [carrySelect\\_adder](#) entity

#### 4.1.1 Detailed Description

Sommatore Carry Select.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

### 4.2 carrySelect\_addSub.vhd File Reference

Sommatore Carry Select in grado di effettuare anche l'operazione di sottrazione.

## Entities

- [carrySelect\\_addSub](#) entity

### 4.2.1 Detailed Description

Sommatore Carry Select in grado di effettuare anche l'operazione di sottrazione.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

## 4.3 carrySelect\_cell.vhd File Reference

Singolo blocco di un sommatore carry Select.

## Entities

- [carrySelect\\_cell](#) entity

### 4.3.1 Detailed Description

Singolo blocco di un sommatore carry Select.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings



## 4.4 counter\_UpN\_Re\_Sr.vhd File Reference

Contatore modulo N.

### Entities

- [counter\\_UpN\\_Re\\_Sr](#) entity

### 4.4.1 Detailed Description

Contatore modulo N.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

## 4.5 flipflopmux.vhd File Reference

flip flop D con multiplexer

### Entities

- [flipflopmux](#) entity

### 4.5.1 Detailed Description

flip flop D con multiplexer

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

## 4.6 robertson\_control\_unit.vhd File Reference

Unità di controllo del moltiplicatore di Robertson.

### Entities

- [robertson\\_control\\_unit](#) entity

### 4.6.1 Detailed Description

Unità di controllo del moltiplicatore di Robertson.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

## 4.7 robertson\_multiplier.vhd File Reference

Moltiplicatore di Robertson che implementa l'algoritmo di Robertson.

### Entities

- [robertson\\_multiplier](#) entity

### 4.7.1 Detailed Description

Moltiplicatore di Robertson che implementa l'algoritmo di Robertson.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings

## 4.8 scan\_chain.vhd File Reference

Registro di n flip flop D multiplexati.

### Entities

- [scan\\_chain](#) entity

### 4.8.1 Detailed Description

Registro di n flip flop D multiplexati.

#### Author

Gabriele Previtera, Mirko Pennone, Simone Penna

#### Date

04/03/2019

#### Version

0.2

#### Dependencies:

Nothings



# Index

c\_in  
    rippleCarry\_adder, 15

c\_out  
    rippleCarry\_adder, 15

carrySelect\_addSub, 6

carrySelect\_addSub.vhd, 19

carrySelect\_adder, 5

carrySelect\_adder.vhd, 19

carrySelect\_cell, 6

carrySelect\_cell.vhd, 20

counter\_UpN\_Re\_Sr, 7

counter\_UpN\_Re\_Sr.vhd, 21

flipflop\_d\_risingEdge\_asyncReset, 8

    IEEE, 8

    STD\_LOGIC\_1164, 9

flipflopmux, 9

flipflopmux.vhd, 21

full\_adder, 10

    IEEE, 10

    STD\_LOGIC\_1164, 10

IEEE

    flipflop\_d\_risingEdge\_asyncReset, 8

    full\_adder, 10

mux2\_1, 11

    STD\_LOGIC\_1164, 12

overflow\_checker, 12

    STD\_LOGIC\_1164, 13

register\_d\_Re\_Ar, 13

    STD\_LOGIC\_1164, 14

rippleCarry\_adder, 14

    c\_in, 15

    c\_out, 15

    S, 15

    STD\_LOGIC\_1164, 15

    width, 15

    Y, 15

robertson\_control\_unit, 16

robertson\_control\_unit.vhd, 22

robertson\_multiplier, 17

robertson\_multiplier.vhd, 22

S

    rippleCarry\_adder, 15

STD\_LOGIC\_1164

    flipflop\_d\_risingEdge\_asyncReset, 9

    full\_adder, 10

    mux2\_1, 12

    overflow\_checker, 13

    register\_d\_Re\_Ar, 14

    rippleCarry\_adder, 15

scan\_chain, 17

scan\_chain.vhd, 23

width

    rippleCarry\_adder, 15

Y

    rippleCarry\_adder, 15