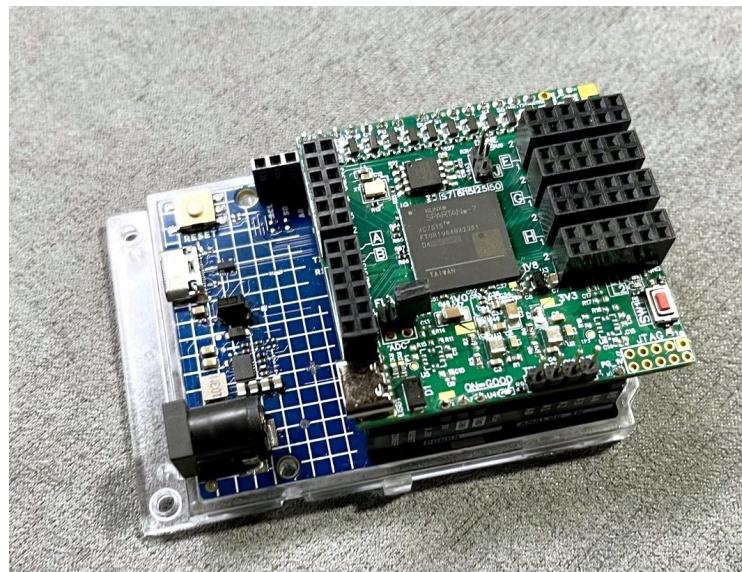


# Guida Rapida

&

## Guida CLI ARDUINO® per il MODULO FPGA SPARTAN7



### Indice generale

1 Descrizione.....	3
2 Primo avvio.....	5
2.1 Primo progetto VHDL.....	8
2.2 Upload bitstream (metodo 1).....	19
2.3 Upload bitstream (metodo 2).....	23
2.4 Upload binary FLASH SNOR (metodo 1).....	26
2.5 Upload binary FLASH SNOR (metodo 2).....	30
3 Guida alle opzioni del menu numerico nella CLI.....	34
3.1 "1. Upload bitsteam: UART to SPI".....	35
3.2 "2. Upload bitsteam: XMODEM to SPI".....	37
3.3 "3. FPGA reset cycle".....	38
3.4 "4. FPGA status DONE" .....	38
3.5 "5. UART bridge" .....	39
3.6 "6. Reset Arduino board or CTRL+R".....	39
3.7 "7. chipErase [Opcode C7h]".....	40
3.8 "8. CRC32 calc".....	40
3.9 "9. Upload bin-file: UART mode" .....	42
3.10 "10. Upload bin-file: XMODEM mode" .....	43
3.11 "11. deviceID [Opcode 9Fh RDID]".....	44
3.12 "12. Enable Advanced SPI-NOR menu" .....	44
3.13 "13. Set SPI".....	45
3.14 " 14. blockErase 4096/32768/65536 [Opcode 20h/52h/D8h]".....	45

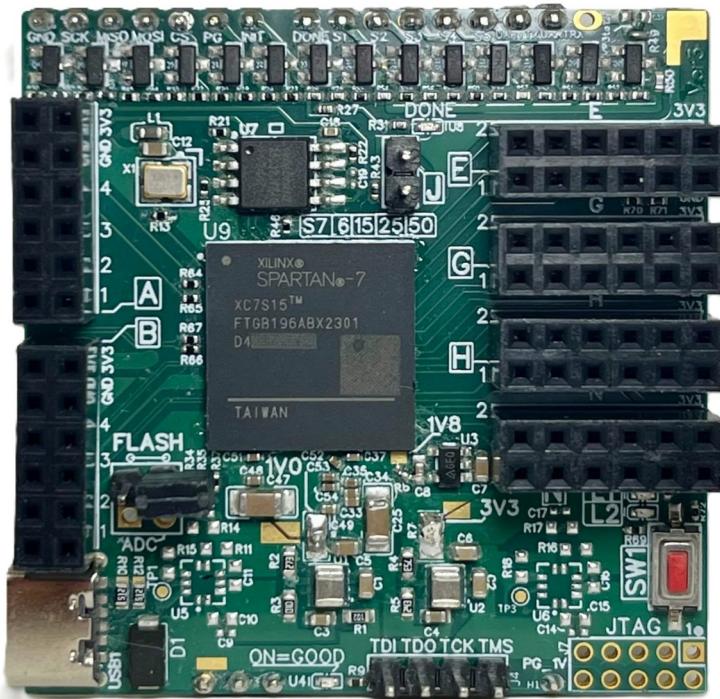
3.15 "15. writeEnable [Opcode 06h WREN]".....	45
3.16 "16. readFlash [Opcode 03h RDSR]".....	46
3.17 "17. fast-read Flash [Opcode 0Bh FRD]".....	47
3.18 "18. statusRegister [Opcode=0x05 RDSR]".....	47
3.19 "19. writeData [Opcode 02h PP]".....	48
3.20 "20. deviceID [Opcode 90h REMS]".....	49
3.21 "21. Download SNOR-Flash: UART mode".....	50
3.22 "23. Custom SPI Transaction".....	52
3.23 "22. Download SNOR-Flash: XMODEM mode".....	54
3.24 "24. FPGA Reset: Hold".....	57
3.25 "25. FPGA Reset: Realese".....	57
<b>4 Appendice.....</b>	<b>58</b>
4.1 Crediti.....	58
4.2 Revision.....	59
4.3 Configurazione Tera Term.....	60
4.4 Installazione Vivado 2024.2 Windows 11.....	63

## 1 Descrizione

Il "MODULO FPGA SPARTAN7" può essere collegato ad una scheda Arduino R4, con la quale è possibile programmare la FPGA, comunicare con UART, scrivere la SNOR FLASH e molto altro, quindi sulla board Arduino verrà scaricato un software sviluppato per poter eseguire queste operazioni. Lo scopo di questo documento è spiegare il funzionamento del software a bordo di Arduino che comunicherà con l'utente attraverso la seriale UART-USB di Arduino con una interfaccia utente CLI.

### Cosa Serve:

- MODULO FPGA SPARTAN7 [PCB Version=3][Rev00 vedi qui [4.2](#)]



- Arduino® IDE 2.3.0 :  
<https://www.arduino.cc/en/software/>
- Software per Arduino [FPGA\_Schield-cli.ino Ver 0.0]  
[https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/software/FPGA\\_Shield-cli.ino](https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/software/FPGA_Shield-cli.ino)
- Tera Term [Version 5.4.0]  
<https://teratermproject.github.io/>  
<https://github.com/TeraTermProject/teraterm/releases>
- File binario o bitstream FPGA Xilinx SPARTAN 7 .
- File pinout del "MODULO FPGA SPARTAN7" :  
[https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/pinout/pinout\\_ver003\\_rev00.xdc](https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/pinout/pinout_ver003_rev00.xdc)
- Vivado (Vedi come installare Vivado 4.4) [Referance version 2024.2]
- Cavo USB-C (>USB 2.0).

Questo progetto non è affiliato né approvato da Arduino.

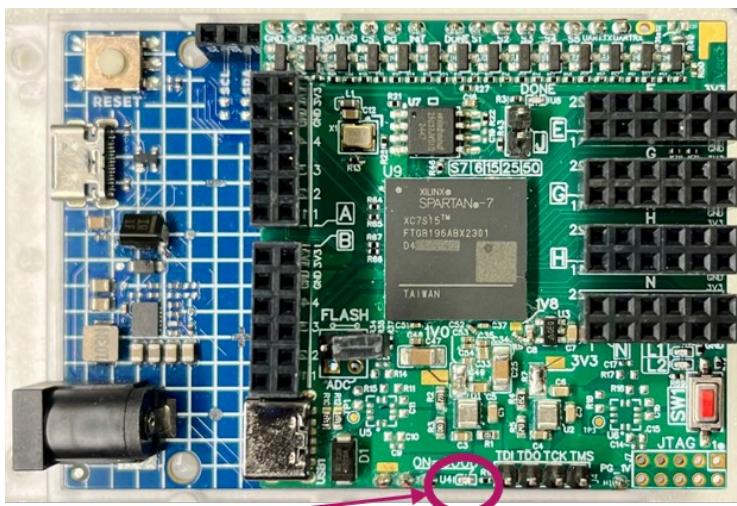
## 2 Primo avvio

Come prima cosa andremo a vedere il software che utilizzeremo per comunicare con la schield "MODULO FPGA SPARTAN7" come prima cosa dobbiamo scaricare il software dalla repository presente su <https://github.com/> e scaricarla nella scheda Arduino R4.

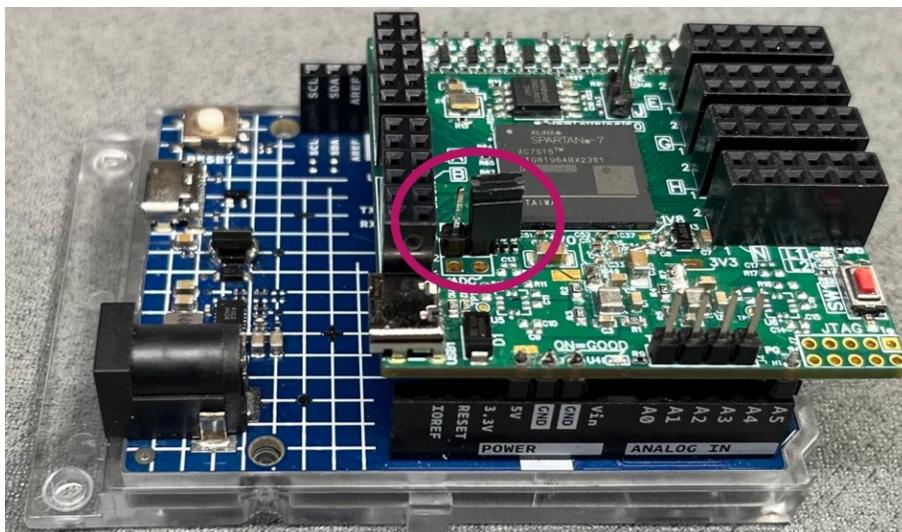
- Scaricare il file :  
[https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/software/FPGA\\_Shield-cli.ino](https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/software/FPGA_Shield-cli.ino)
- Collegare il "MODULO FPGA SPARTAN7" ad Arduino e con una porta USB-C al PC.  
Attenzione collega il modulo prima di collegare la porta USB-C al PC per prevenire eventuali cortocircuiti durante l'inserimento della schield, rispetta il PG del "MODULO FPGA SPARTAN7" deve essere collegato al pin A5 di Arduino.



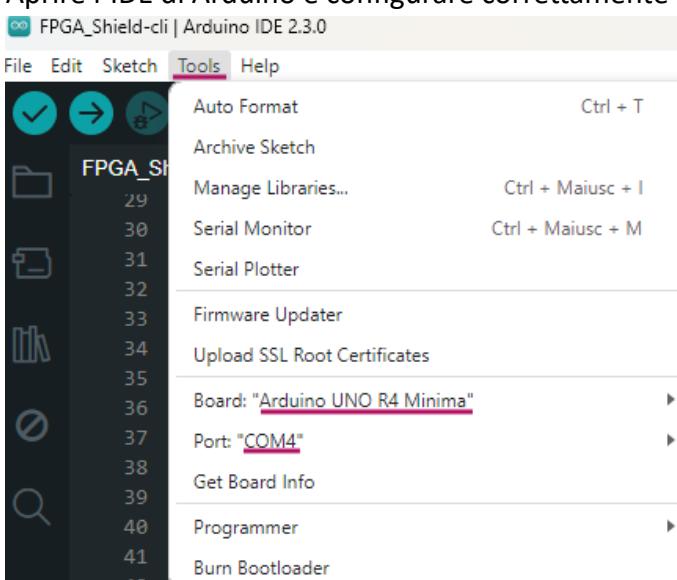
- Se il collegamento è avvenuto con successo e il modulo è alimentato correttamente il led "U4" sarà acceso.



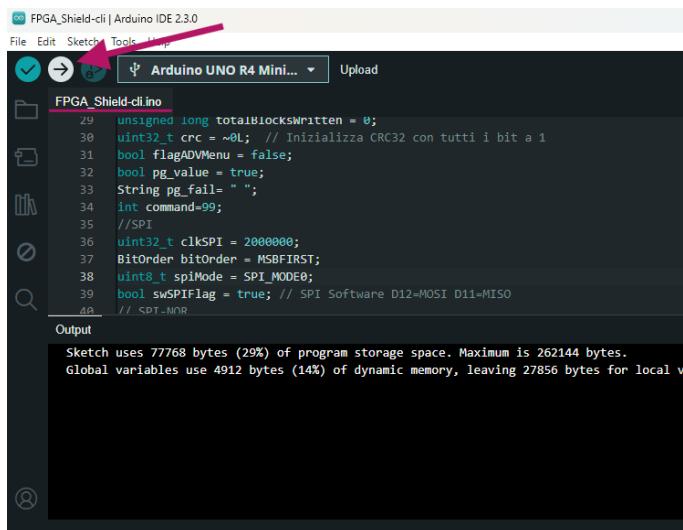
- Apri il jumper. (Imposta la FPGA in Slave-SPI mode):



- Aprire l'IDE di Arduino e configurare correttamente COMx e board Type:



- Aprire il file "FPGA\_Shield-cli.ino" e premere "Upload" sull'IDE di Arduino .



- Ora aprire Tera Term e collegarsi alla COMx di Arduino. Le configurazione corrette di Tera Term le trovi qui [4.3](#).

```

VT COM4 - Tera Term VT
File Edit Setup Control Window Help
MENU Main Ver0.0
1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R
    MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU
>

```

Premendo il carattere "Invio" sulla tastiera verrà stampato il "Menu", questa interfaccia è chiamata anche CLI.

Il menu è diviso in due parti, la prima parte "MENU Main" dove sono presenti le opzioni relative alla programmazione, al reset, status, UART bridge ecc del "MODULO FPGA SPARTAN7" queste opzioni saranno quelle che useremo più spesso durante la progettazione.

Mentre la seconda parte del menu "MENU SPI-NOR" riguarda soprattutto la comunicazione con la SPI-NOR e altre funzionalità di debug.

Poi troviamo il riquadro relativo alla versione software.

Tutte le opzione del menu saranno dettagliate nei capitoli successivi

Nei seguenti sottocapitoli faremo il nostro primo progetto VHDL e genereremo il primo "bitstream" con il quale programmeremo la FPGA con due metodi differenti, poi creeremo il file "binario" da scrivere nella FLASH-SNOR, renderà la nostra shield FPGA indipendente dal caricamento da parte di Arduino.

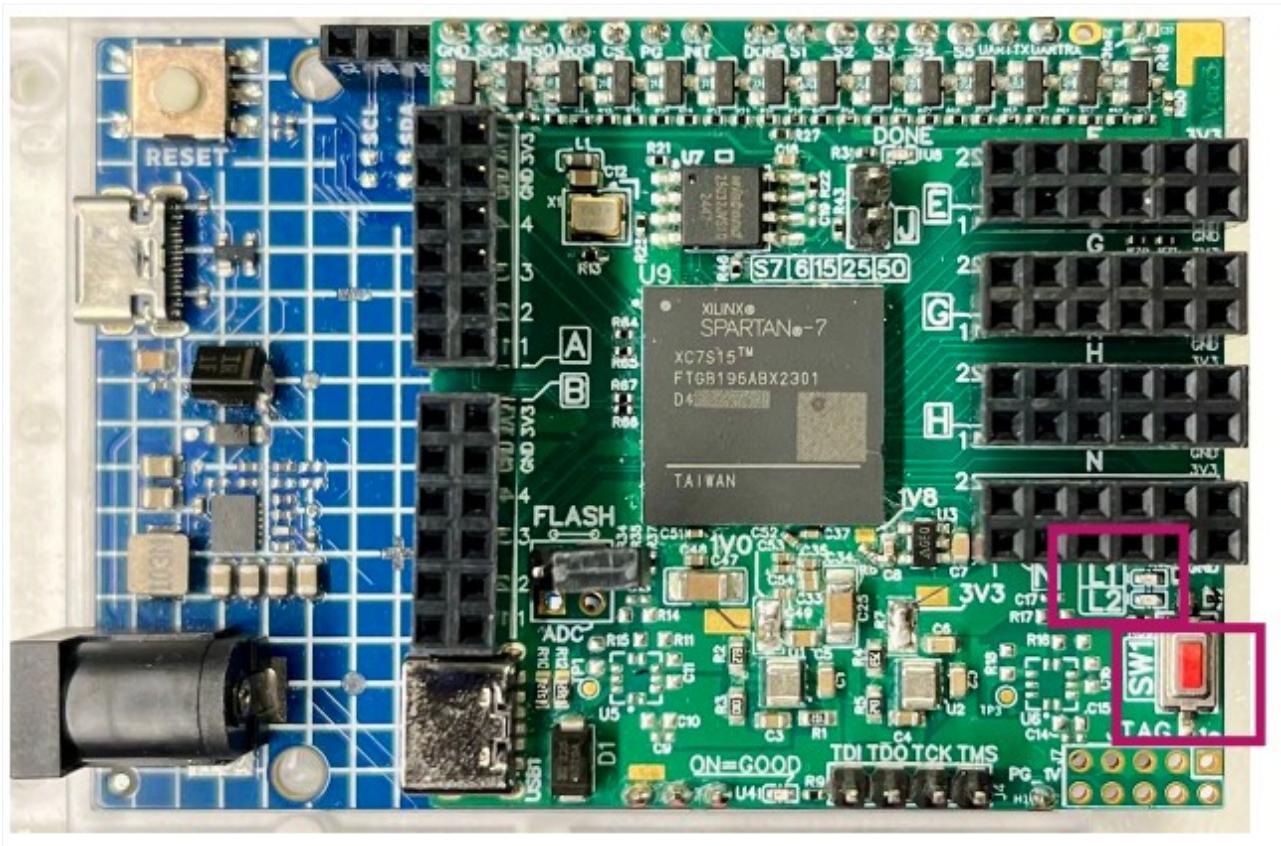
Qui i passi che seguiremo:

- Primo progetto VHDL [2.1](#).
- Upload bitstream (metodo 1) [2.2](#).
- Upload bitstream (metodo 2) [2.3](#).
- Upload binary file in FLASH-SNOR (metodo 1) [2.4](#).
- Upload binary file in FLASH-SNOR (metodo 2) [2.5](#).

Prima di continuare è necessario aver scaricato Vivado sul proprio PC ti rimando al capitolo [4.4](#) nel quale troverai una guida su come installare Vivado.

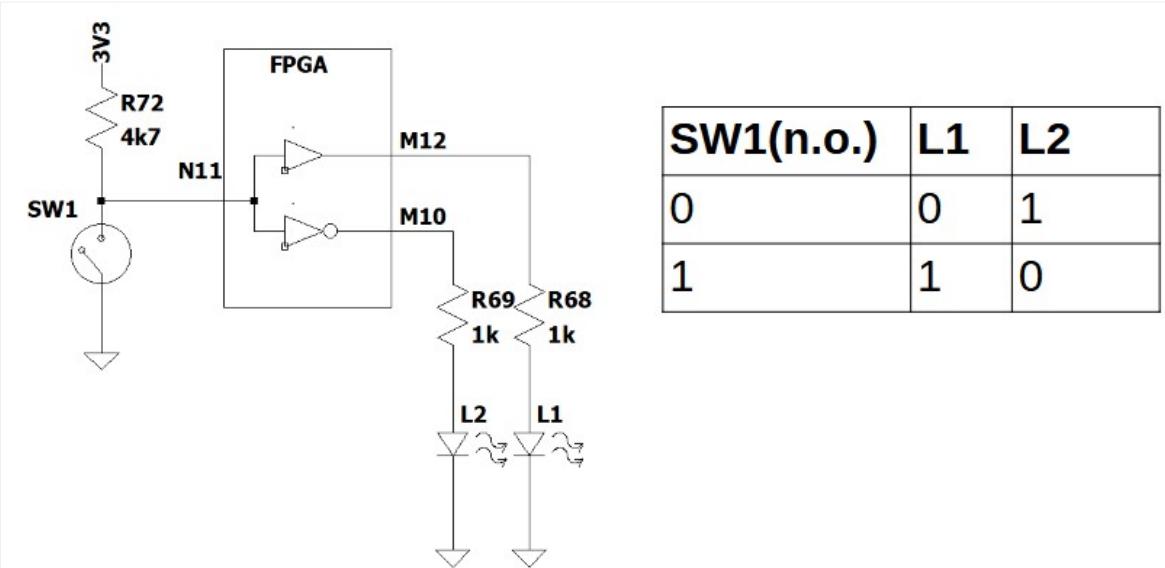
## 2.1 Primo progetto VHDL

Come primo progetto VHDL andremo ad usare i due led presenti sul "MODULO FPGA SPARTAN7" L1 e L2 e il pulsante SW1.



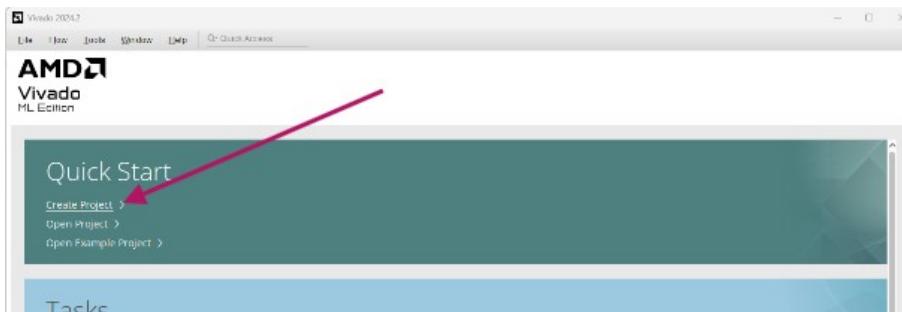
Andremo a comandare i due led presenti sulla board con il pulsante con la seguente logica:

- Quando il pulsante "SW1" non sarà premuto la FPGA legge un valore logico "1" avendo la pull-up esterna (on-board), il led "L1" sarà acceso e il led "L2" sarà spento.
- Quando il pulsante "SW1" viene premuto la FPGA legge un valore logico "0", il led "L2" sarà acceso e il led "L1" sarà spento.

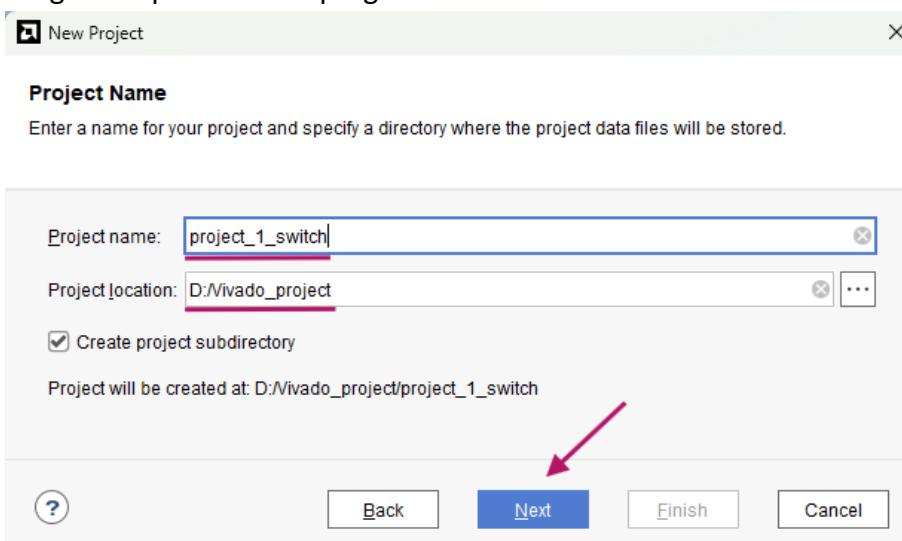


Definito il comportamento, bisogna creare un progetto VHDL con Vivado

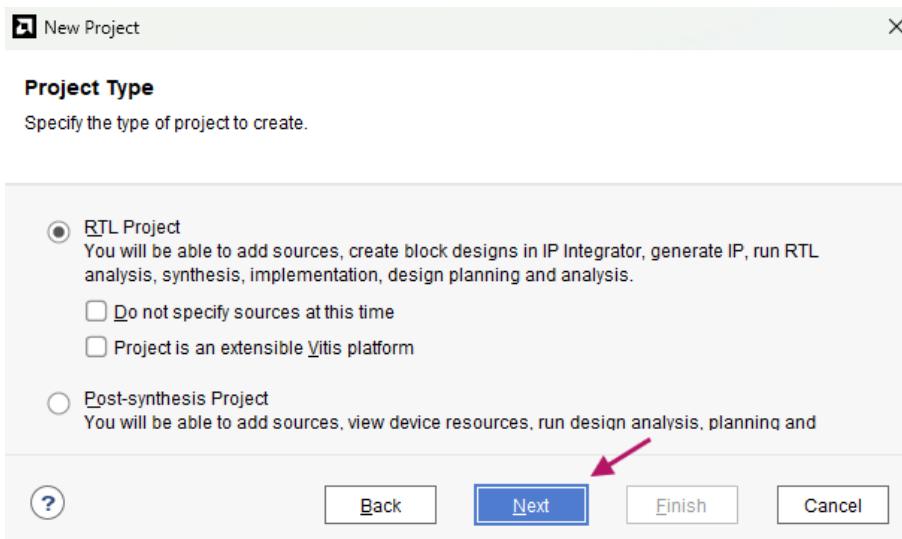
- Aprire Vivado e "Create Project":



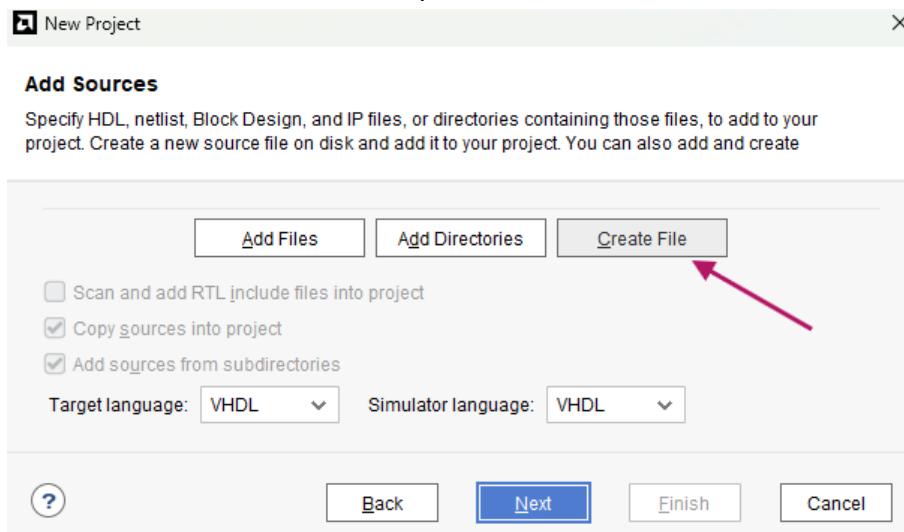
- Scegliere il percorso del progetto:



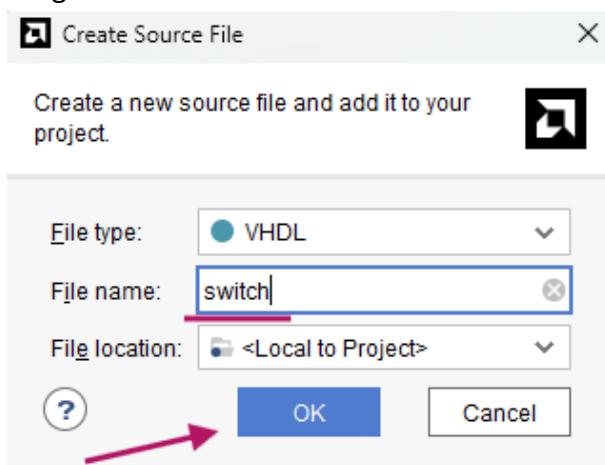
- Scegliere "RTL Project":



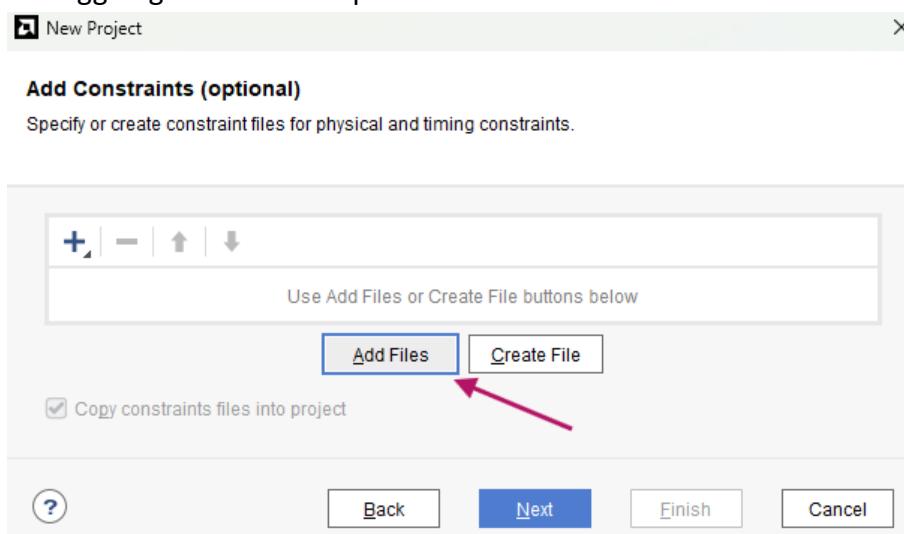
- Creare il file dove andremo a copiare il nostro codice VHDL



- Scegliere il nome del file VHDL



- Ora aggiungere il file della pinout della FPGA che ha come estensione .xdc



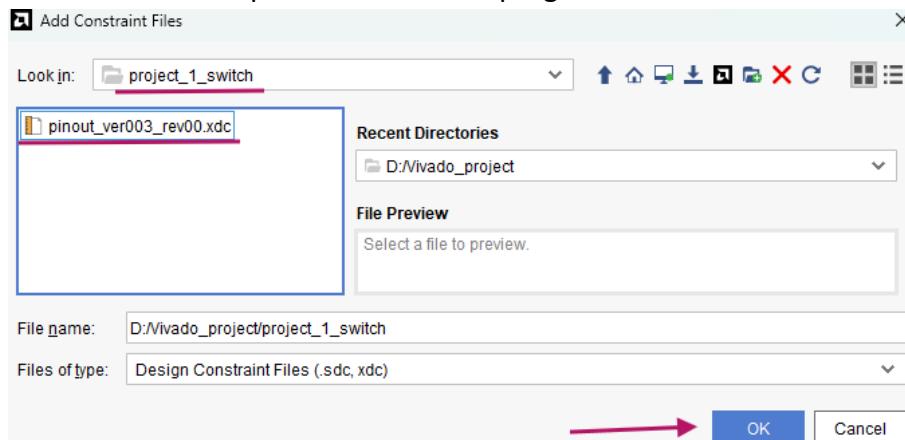
- Ora scarica dalla repository il file .xdc e copialo nella cartella del progetto:

File .xdc: [https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/pinout/pinout\\_ver003\\_rev00.xdc](https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/pinout/pinout_ver003_rev00.xdc)

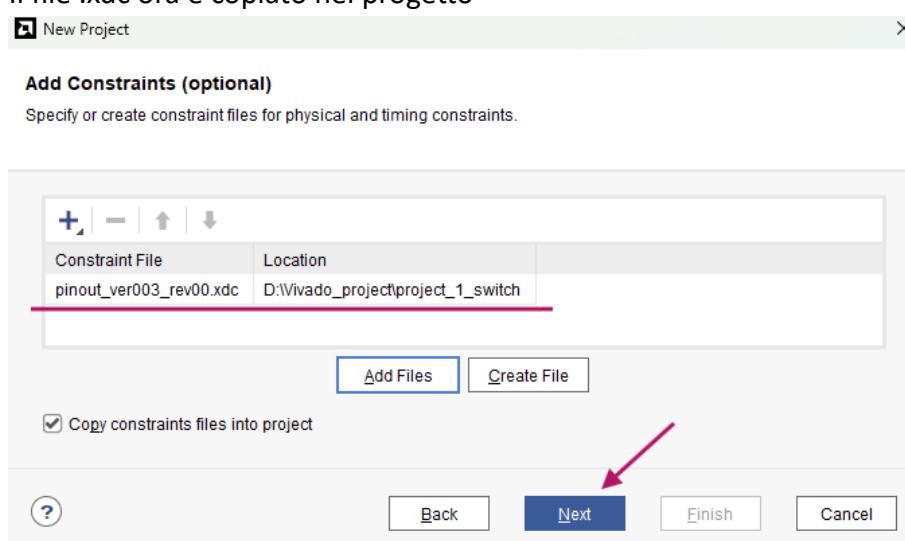
Cartella del progetto ad esempio:

D:\Vivado\_project\project\_1\_switch

- Ora se abbiamo copiato il file .xdc nel progetto lo troveremo in Vivado, selezionalo e premi "OK"

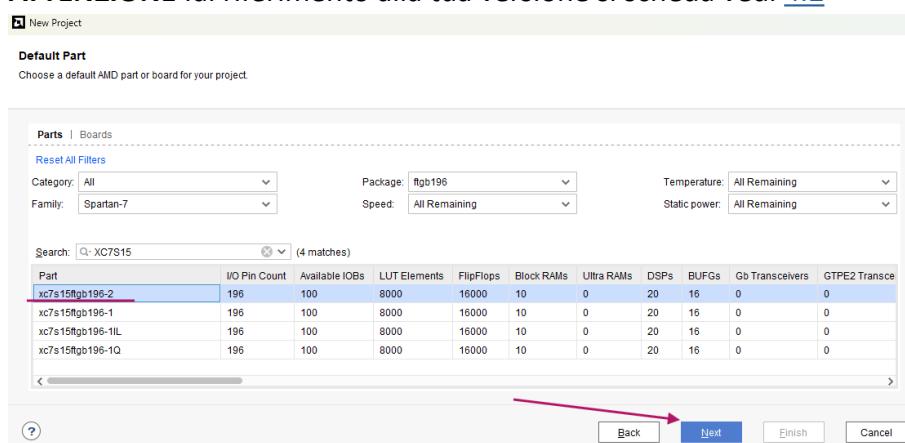


- Il file .xdc ora è copiato nel progetto

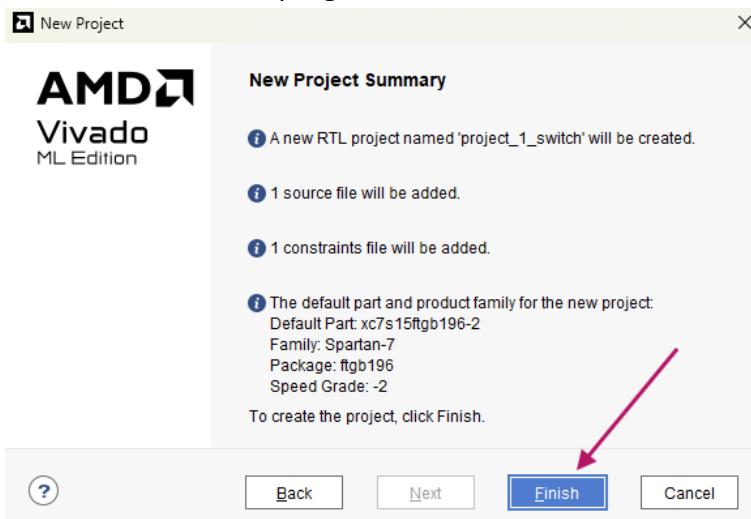


- Ora bisogna scegliere il part-number della FPGA.

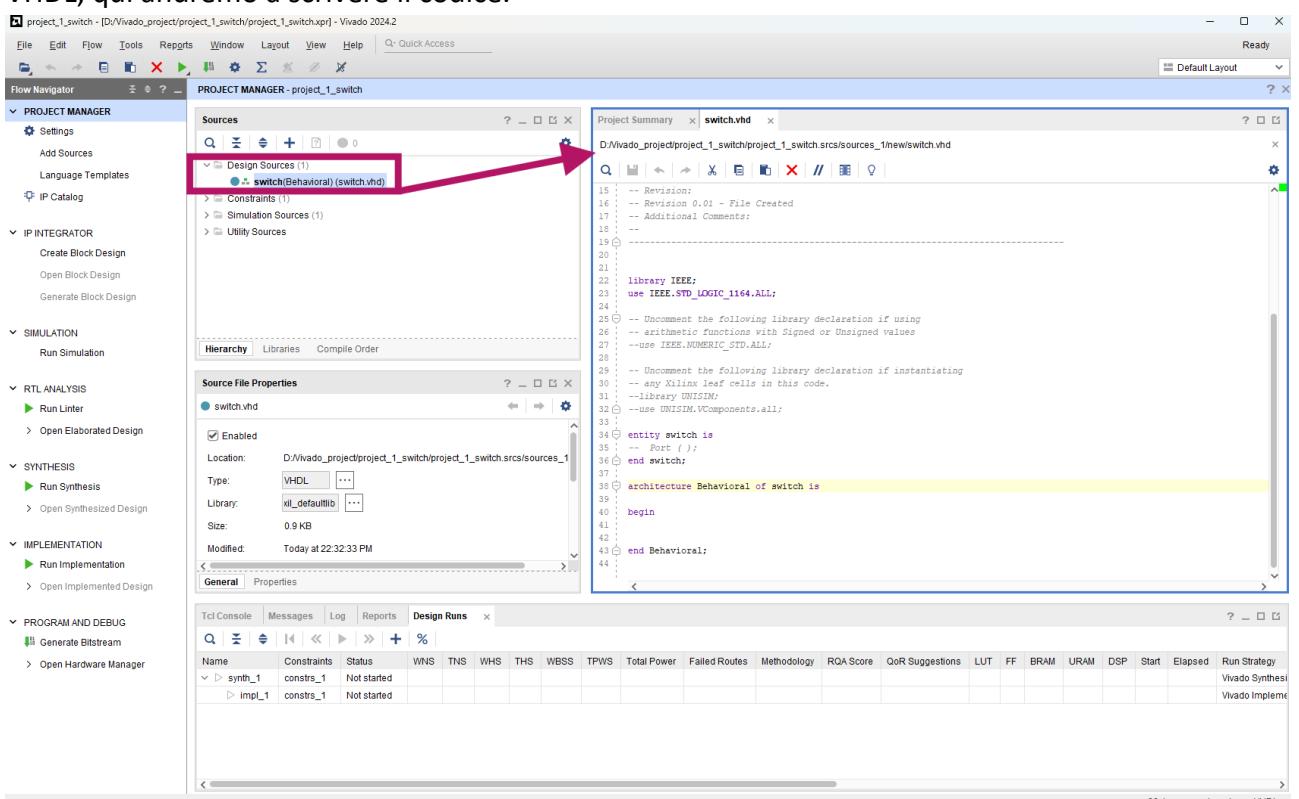
**ATTENZIONE** fai riferimento alla tua versione si scheda vedi [4.2](#)



- Premendo "Finish" il progetto verrà creato:



- Ora si aprirà il progetto, se si fa bi-click sul file "switch.vhd" si aprirà il default template del codice VHDL, qui andremo a scrivere il codice.



- Questo è il file "switch.vhd" che descrive il comportamento dei led :

<https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.vhd>

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity switch is
    Port (
        L1      : out std_logic ;
        L2      : out std_logic ;
        SW1     : in  std_logic
    );
end switch;

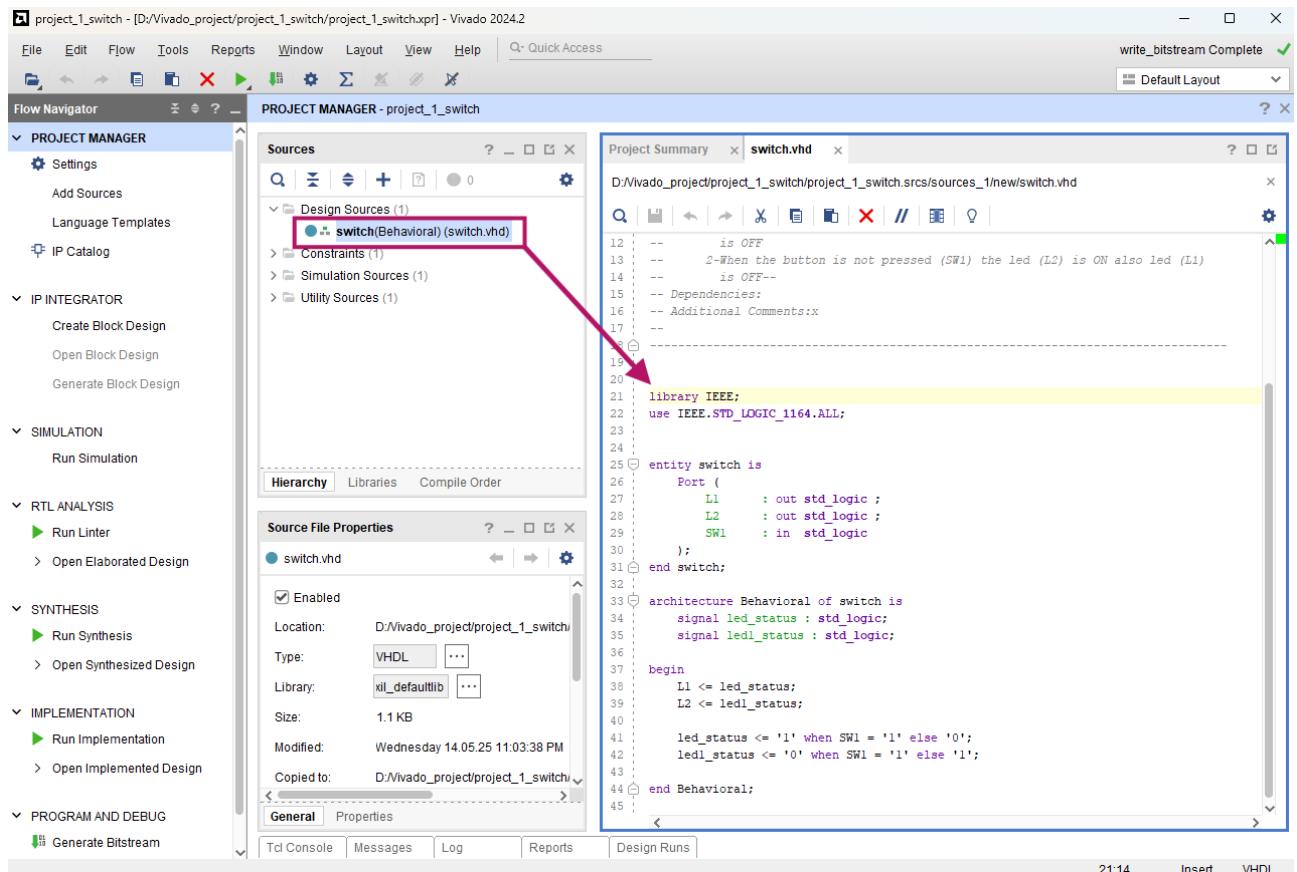
architecture Behavioral of switch is
    signal led_status : std_logic;
    signal led1_status : std_logic;

begin
    L1 <= led_status;
    L2 <= led1_status;

    led_status <= '1' when SW1 = '1' else '0';
    led1_status <= '0' when SW1 = '1' else '1';

end Behavioral;
```

- Ora copia/importa il file nel progetto VivadoAprire il file



- Ora dobbiamo associare gli "out" e gli "in" del codice VHDL a i pin fisici della FPGA. Per fare questo si devo modificare il file .xdc che troverai già modificato nella repository:  
[https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/pinout\\_ver003\\_rev00.xdc](https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/pinout_ver003_rev00.xdc)

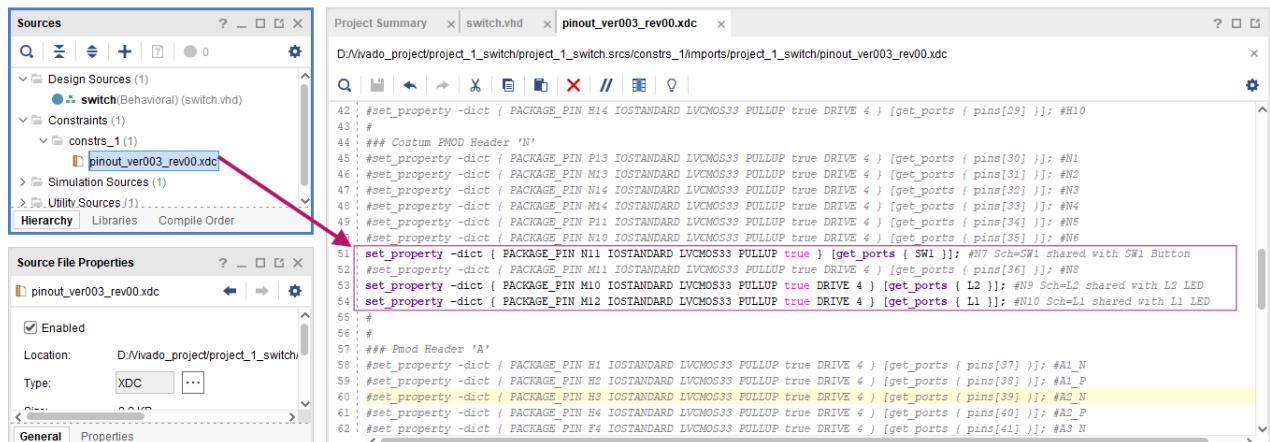
Questo sono le stringhe che dobbiamo scommentare:

```
set_property -dict { PACKAGE_PIN N11 IOSTANDARD LVCMOS33 PULLUP
true } [get_ports { SW1 }]; #N7 Sch=SW1 shared with SW1 Button

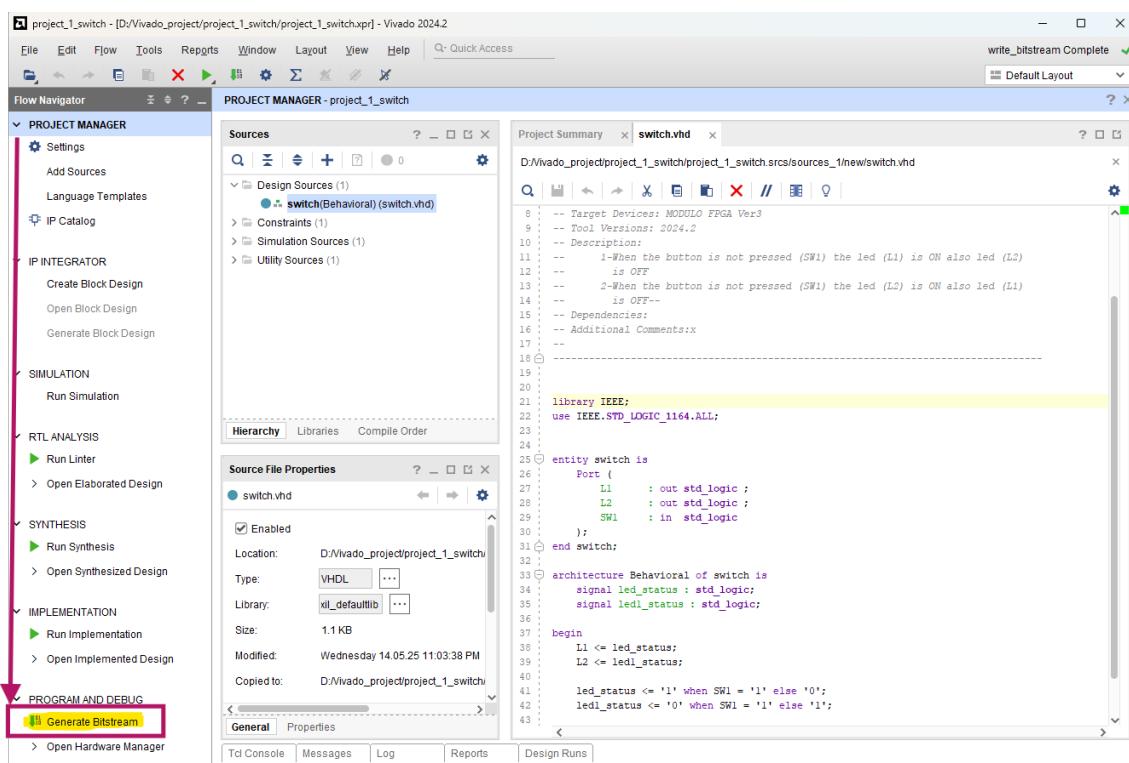
set_property -dict { PACKAGE_PIN M10 IOSTANDARD LVCMOS33 PULLUP
true DRIVE 4 } [get_ports { L2 }]; #N9 Sch=L2 shared with L2 LED

set_property -dict { PACKAGE_PIN M12 IOSTANDARD LVCMOS33 PULLUP
true DRIVE 4 } [get_ports { L1 }]; #N10 Sch=L1 shared with L1 LED
```

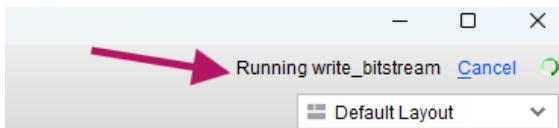
Risulterà così:



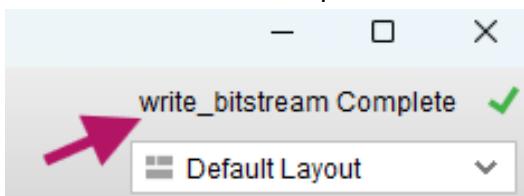
- Siamo al penultimo passo, ora possiamo generare il bitstream file che conterrà il tuo primo codice scritto in VHDL. Clicca "Generate Bitstream"



- In alto a destra di Vivado vedrai lo stato di completamento

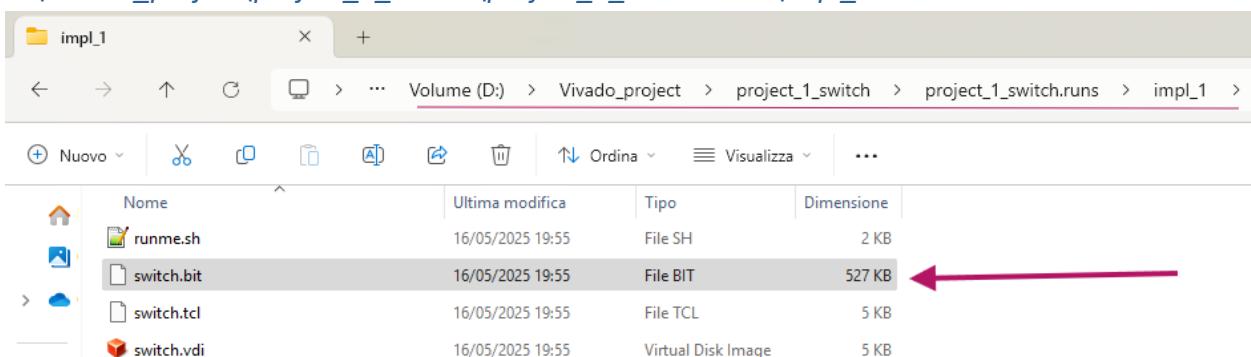


- Se si conclude con esito positivo in alto a destra sarà presente:



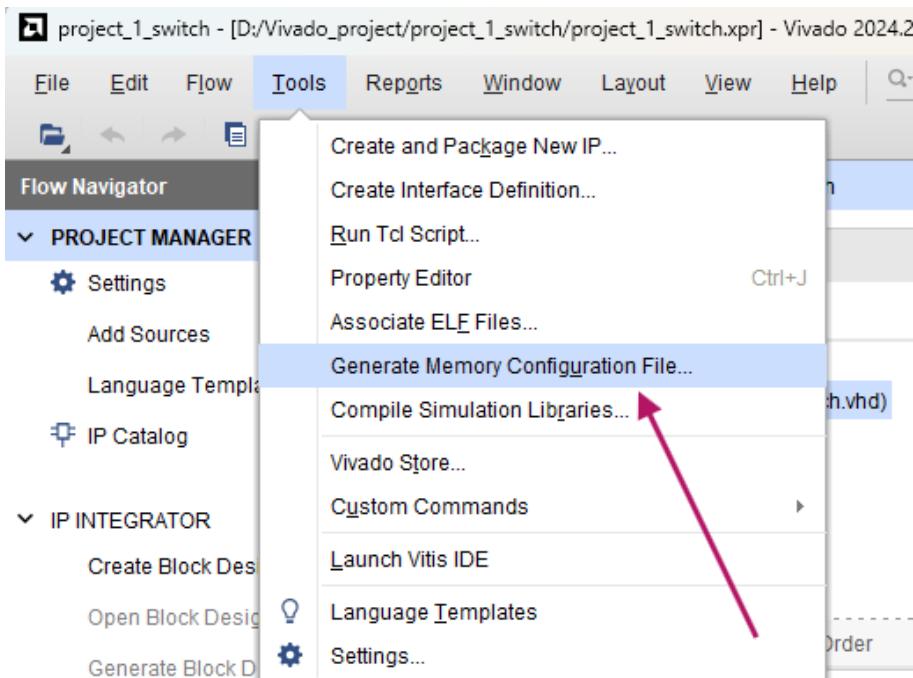
- Il file di "bitstream" si trova nella cartella del progetto, ad esempio:

[D:\Vivado\\_project\project\\_1\\_switch\project\\_1\\_switch.runs\impl\\_1](D:\Vivado_project\project_1_switch\project_1_switch.runs\impl_1)

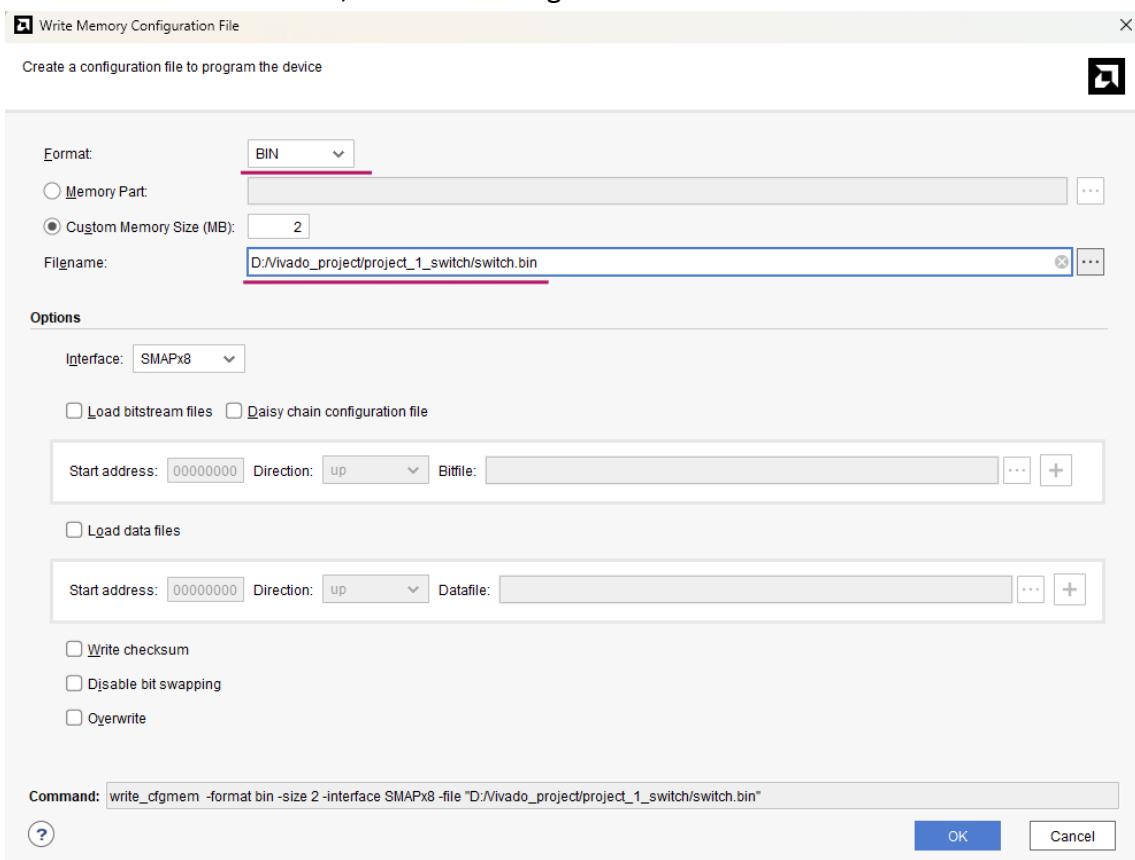


Il file "bitstream" con estensione .bit lo useremo per caricare la FPGA ogni volta che viene tolta l'alimentazione o al reset, mentre per programmare "MODULO FPGA SPARTAN7" ad ogni suo riavvio è necessario caricare il file .bin sulla SNOR, qui sotto i passaggi per generare il "binario" dal file "bitstream".

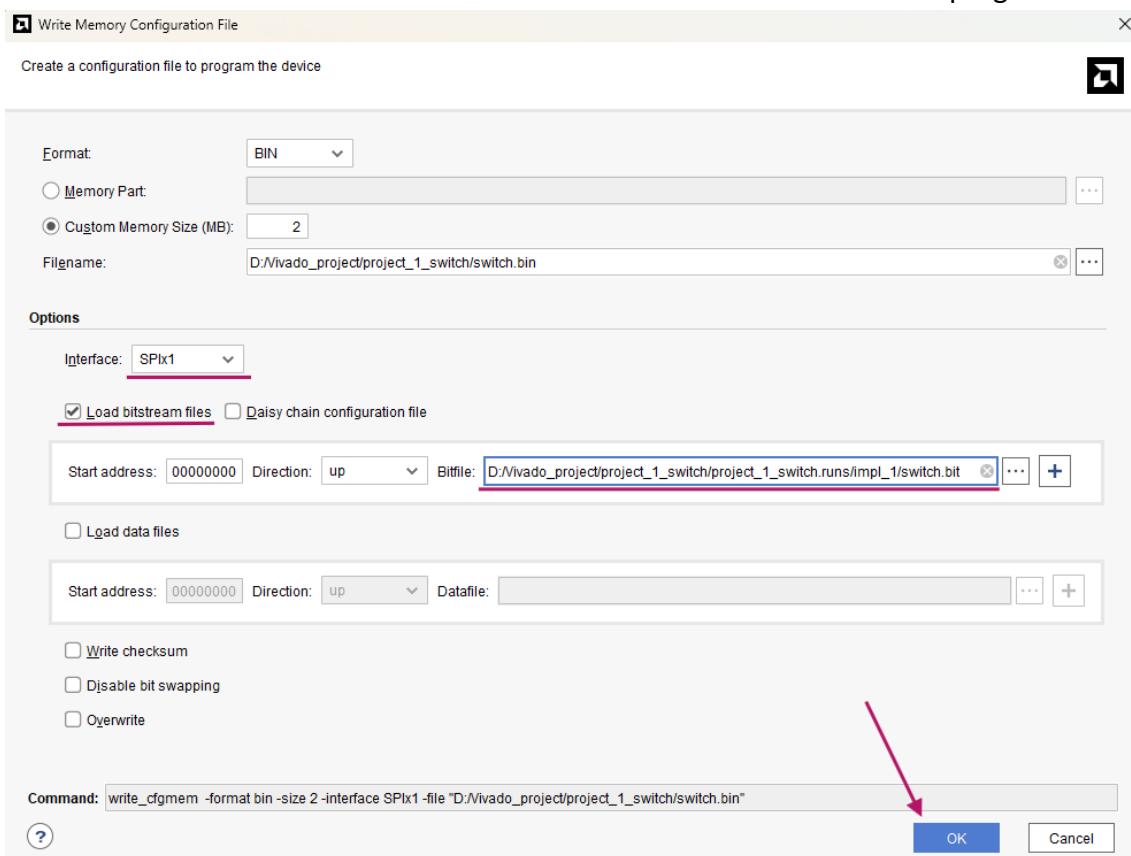
- Per creare il file binario andare sotto "Tools"



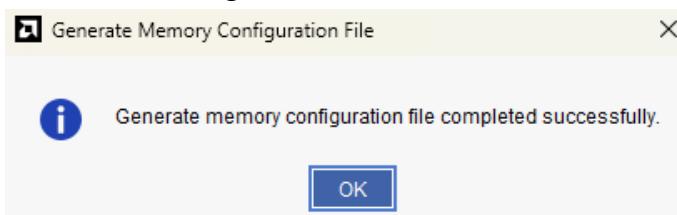
- Seleziona il formato "BIN", Size 2MB e scegli il nome del file:



- Selezionare "SPIx1" e seleziona il bitstream "switch.bit" nella cartella del progetto



- Ora il file è stato generato



## 2.2 Upload bitstream (metodo 1)

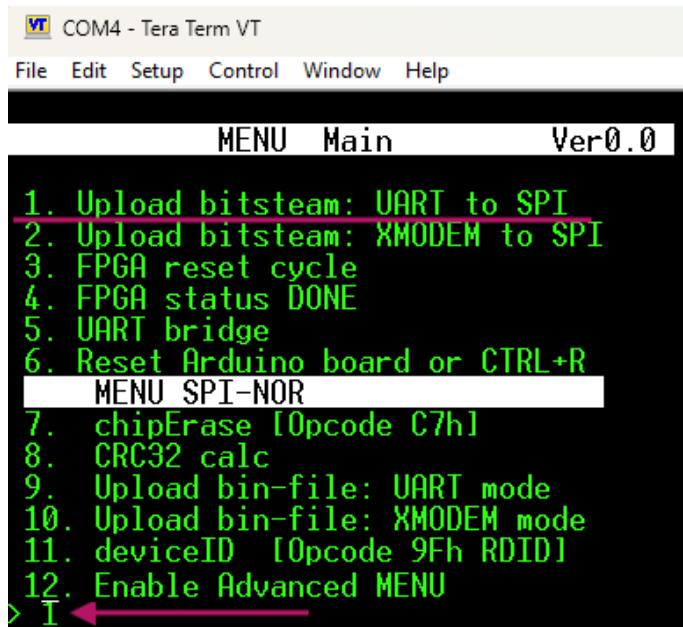
In questo capitolo scarichiamo il file bitstream con l'ausilio di Arduino R4 e Tera Term.

Il file presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, in questo esempio useremo Tera Term in quanto soddisfa i due requisiti fondamentali:

1- L'uso del Flow Control XON/XOFF.

2-L'invio di massimo 256byte alla volta.

- Selezionare l'opzione 1 del "Menu" e premere invio:



- Arduino attende l'invio del file bitstream:

```
> 1
FPGA STATUS: RESET <hold>
FPGA STATUS: RESET <release>
FPGA STATUS: DONE <LOW,not programmed>
FPGA STATUS: INIT_B LOW <attend>
FPGA STATUS: INIT_B LOW <ready>
FPGA STATUS: INIT_B HIGH <attend>
FPGA STATUS: INIT_B HIGH <ready>

***** TERATEM *****
* General Setting:
*   -Setup>Terminal> Receive=CR; Transmit=CR
*   -Setup>Serial port>
*     Speed=115200; Data=8b; Parity=none; Stop bits=1
*     Flowcontrol=Xon/Xoff; Transmit delay=0msec
* Version:
*   -Tested with version 5.4.0
* Transfer setting:
*   -File>Send file>
*     Filename: <file>
*     Binary:[checked]
*     delay type: <per sendsize>
*     [send size(bytes): 256 ]
*     delay time(ms): <0 or 1>

Minimum file size is 256byte!
Do not press any button on keyboard!
Ready to receive data from UART...
```

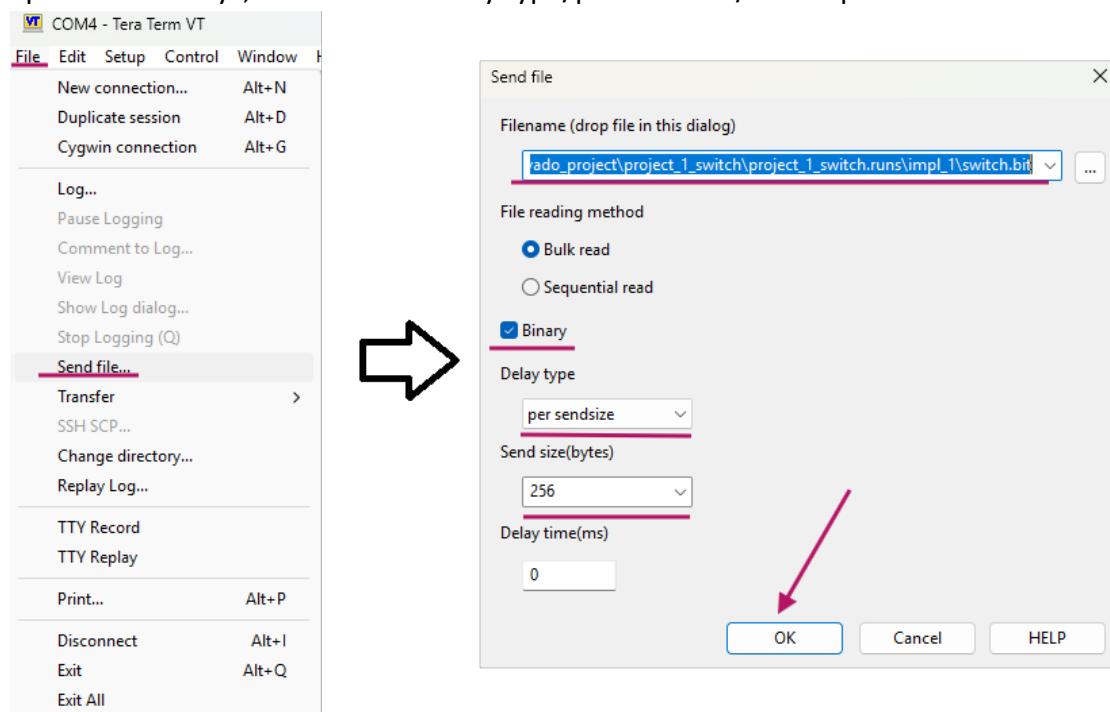
- In Tera Term andare su "File/Send file..." poi selezionare il file bitstream che trovi sotto:

*D:\Vivado\_project\project\_1\_switch\project\_1\_switch.runs\impl\_1\switch.bit*

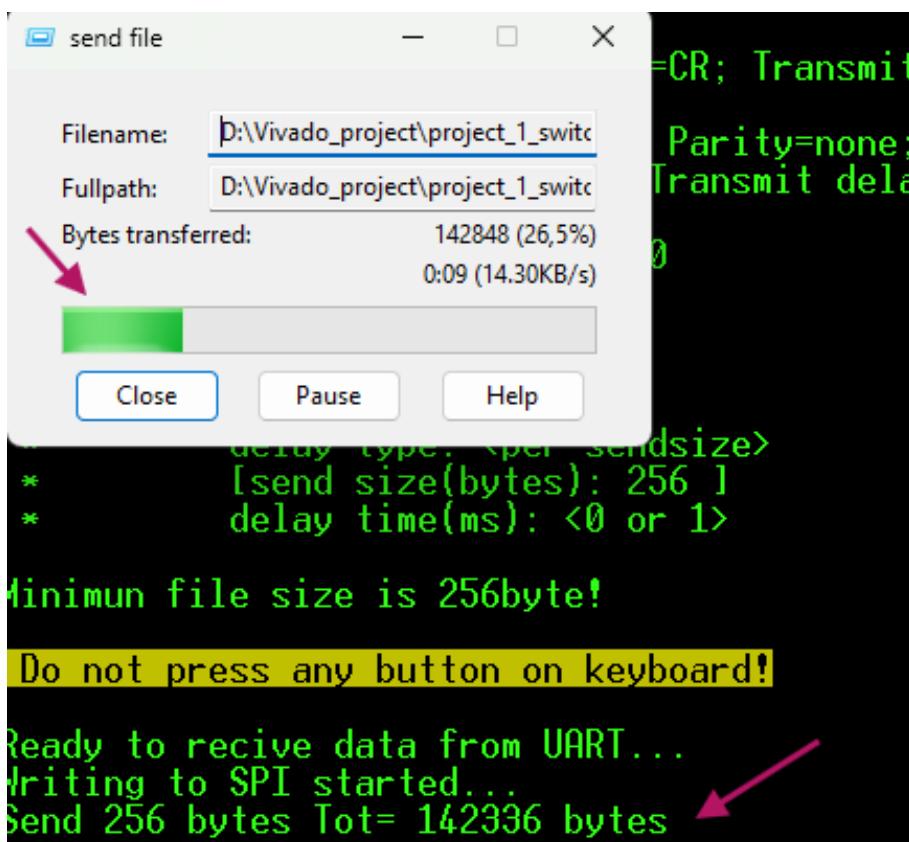
O

*https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.bit*

Spuntare "Binary", selezionare "Delay type/per sendsize/256" e premere OK



- Dopo aver premuto ok si avvia la trasmissione:



- Al termine della trasmissione verifica che la FPGA sia programmata verificando la stampa come nell'immagine sottostante "programmed".

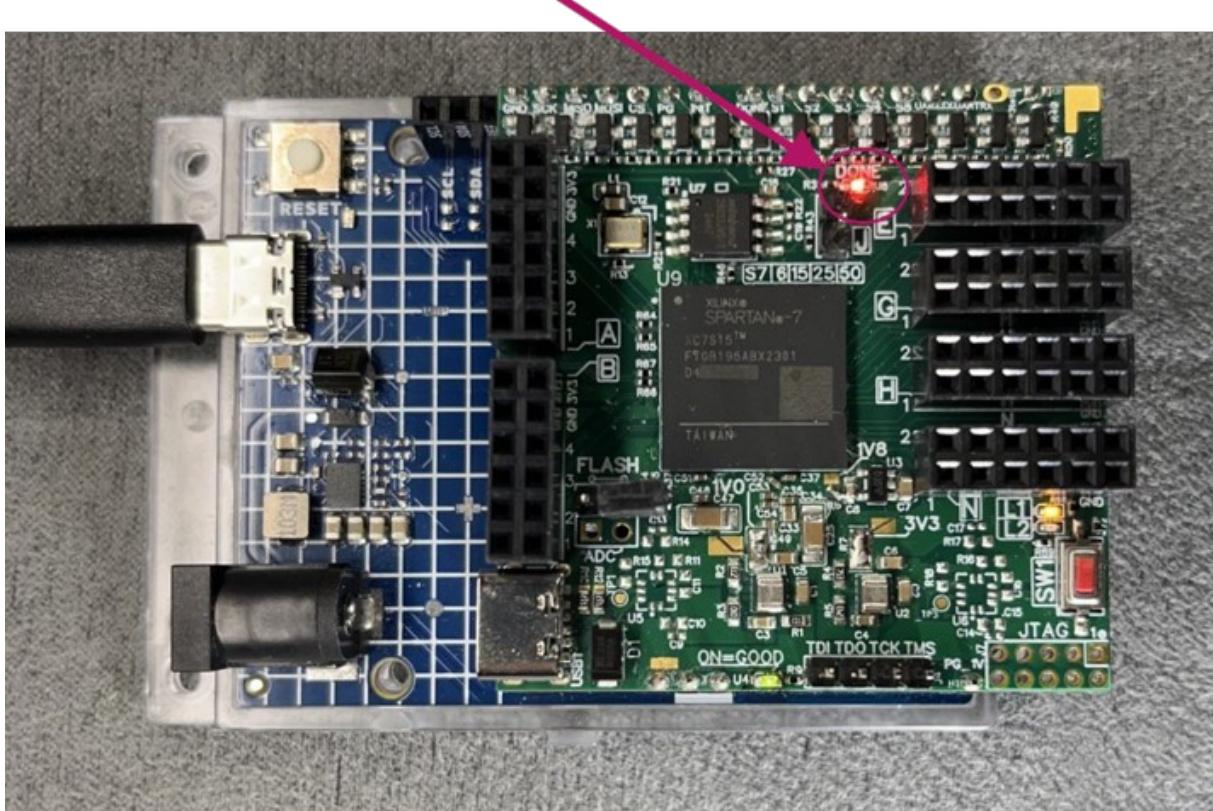
```
Do not press any button on keyboard!

Ready to receive data from UART...
Writing to SPI started...
Send 256 bytes Tot= 538880 bytes
Writing last buffer: 66
Last buffer size: 66 bytes!
Total blocks written: 2106
Total bytes written: 538946
CRC32: C984BCDB ←
Data transfer to SPI completed!

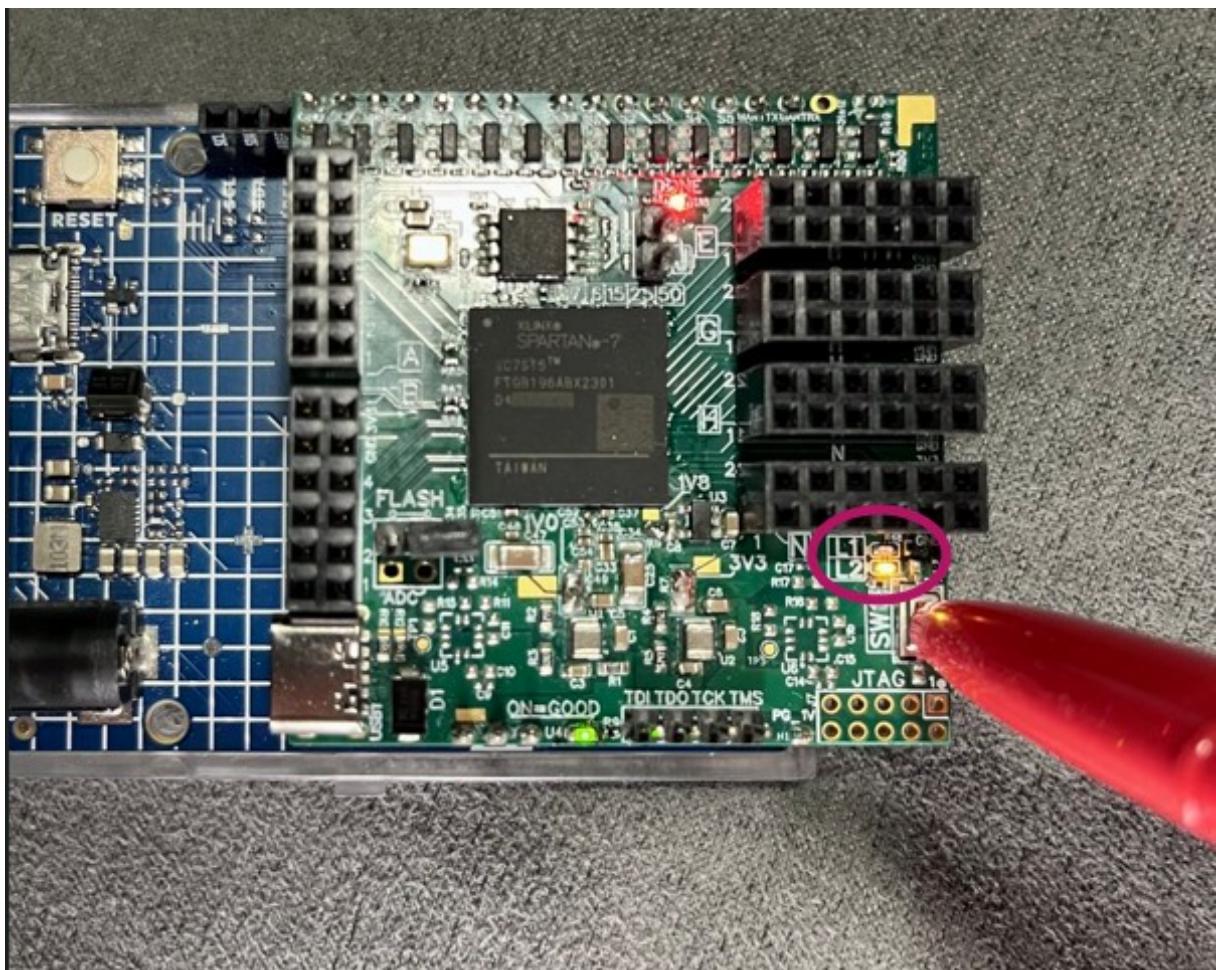
Exec. : 20520ms
FPGA STATUS: DONE <attend>
FPGA STATUS: DONE <programmed> ←

Exec. : 191ms
>
```

- Guarda la tua board... la FPGA è carica! "ON FIRE"



- Il led "L1" è acceso mentre "L2" è spento, come da progetto.
- Ora premi il tasto "SW1" e "L1" è spento mentre "L2" è acceso, come da progetto:



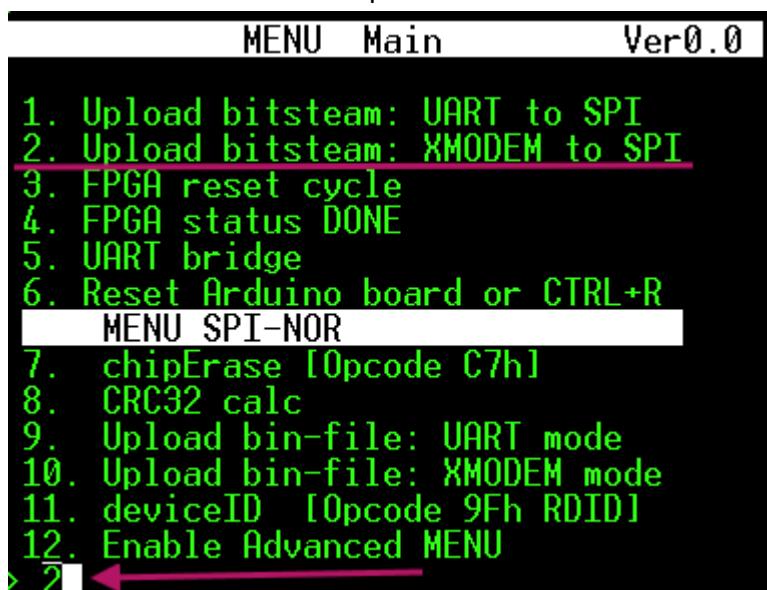
FINE

## 2.3 Upload bitstream (metodo 2)

In questo capitolo scarichiamo il file bitstream con l'ausilio di Arduino R4 e Tera Term.

Il file presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, in questo esempio useremo Tera Term in quanto mette a disposizione il trasferimento XMODEM, a differenza del capitolo 2.2 è necessario solo specificare la dimensione del file da scaricare.

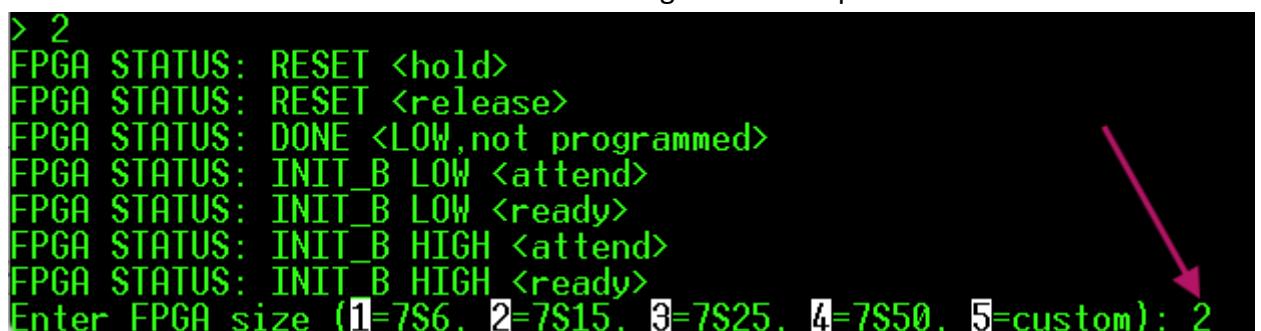
- Selezionare 2 del "Menu" e premere invio:



```

MENU Main Ver0.0
1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R
    MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU
> 2
  
```

- Selezionare la FPGA montata sulla board "S715" digitando "2" e poi "INVIO":



```

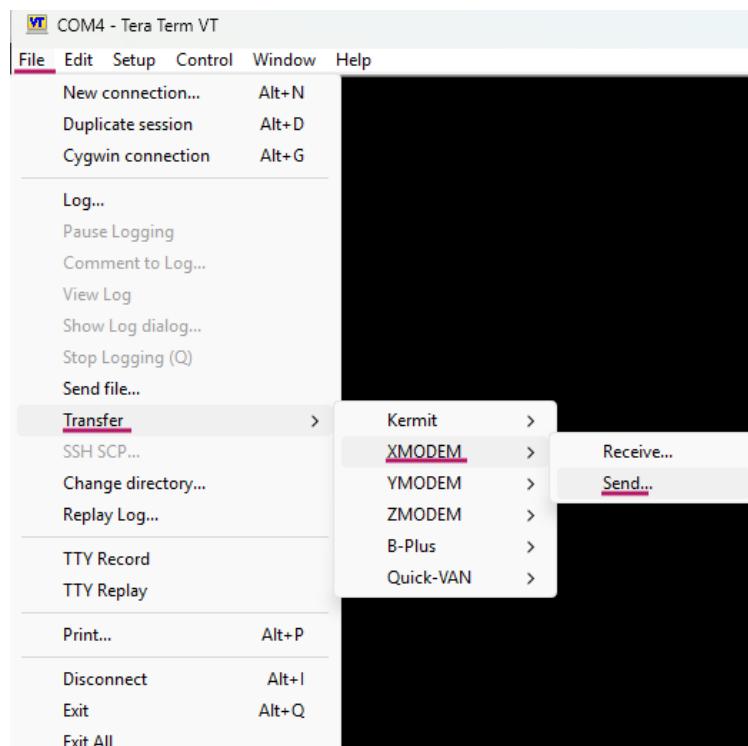
> 2
FPGA STATUS: RESET <hold>
FPGA STATUS: RESET <release>
FPGA STATUS: DONE <LOW,not programmed>
FPGA STATUS: INIT_B LOW <attend>
FPGA STATUS: INIT_B LOW <ready>
FPGA STATUS: INIT_B HIGH <attend>
FPGA STATUS: INIT_B HIGH <ready>
Enter FPGA size (1=7S6, 2=7S15, 3=7S25, 4=7S50, 5=custom): 2
  
```

- In Tera Term andare su "File/Transfer/XMODEM/Send..." poi selezionare il file bitstream che trovi sotto:

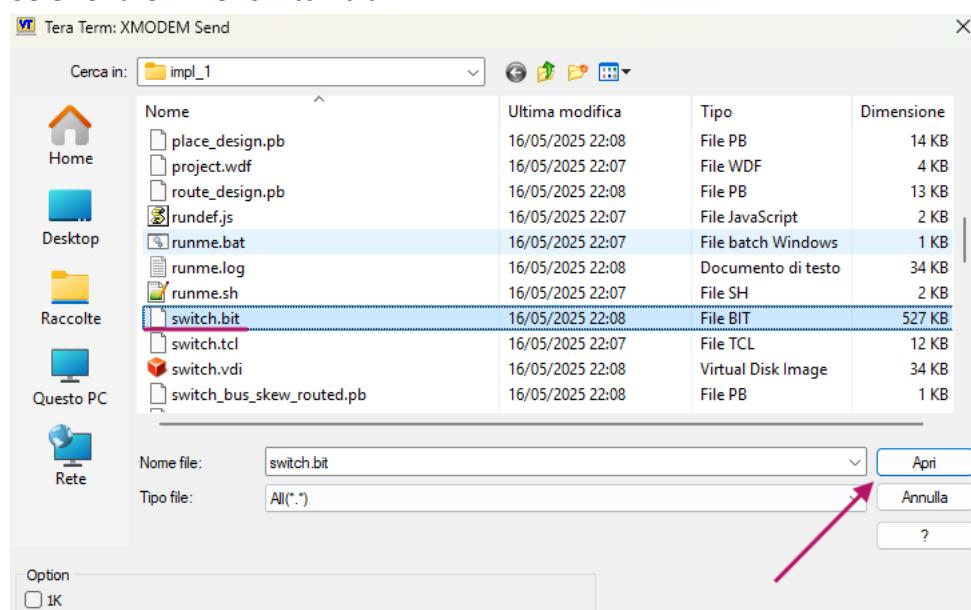
*D:\Vivado\_project\project\_1\_switch\project\_1\_switch.runs\impl\_1\switch.bit*

O

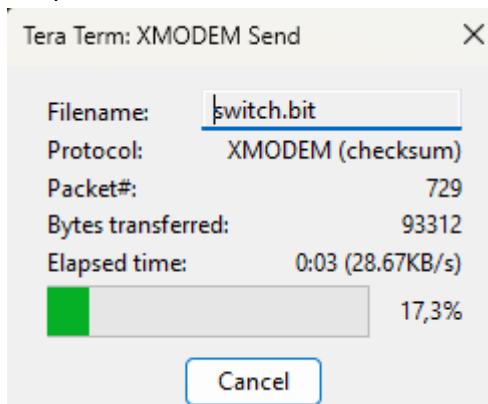
*https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.bit*



- Selezionare il file "switch.bit"



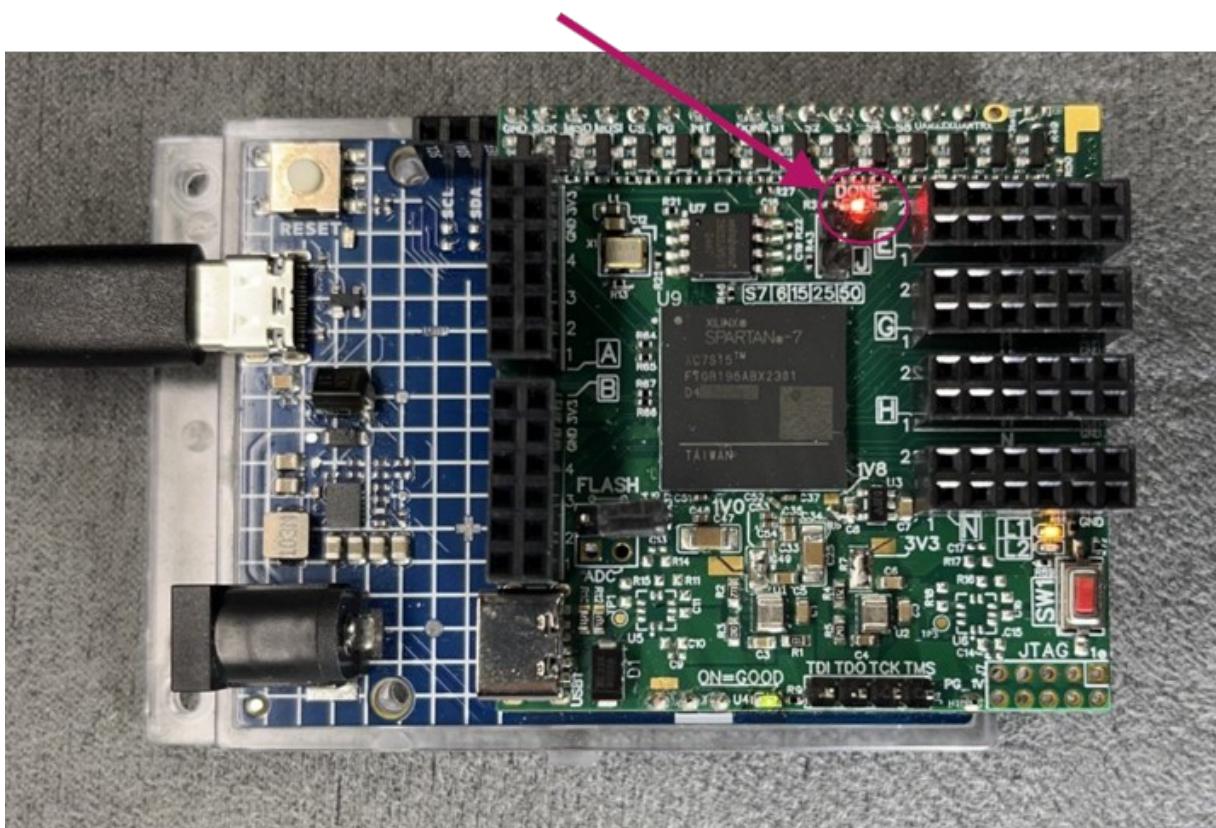
- Si aprirà una finestra che mostra la barra di caricamento del file:



- Al termine la FPGA sarà carica:

```
File Edit Setup Control Window Help
XMODEM Transfer Completed!
Total bytes written: 538844
Exec. : 13589ms
FPGA STATUS: DONE <attend>
FPGA STATUS: DONE <programmed>
Exec. : 190ms
> █
```

- Guarda la tua board... la FPGA è carica! "ON FIRE"

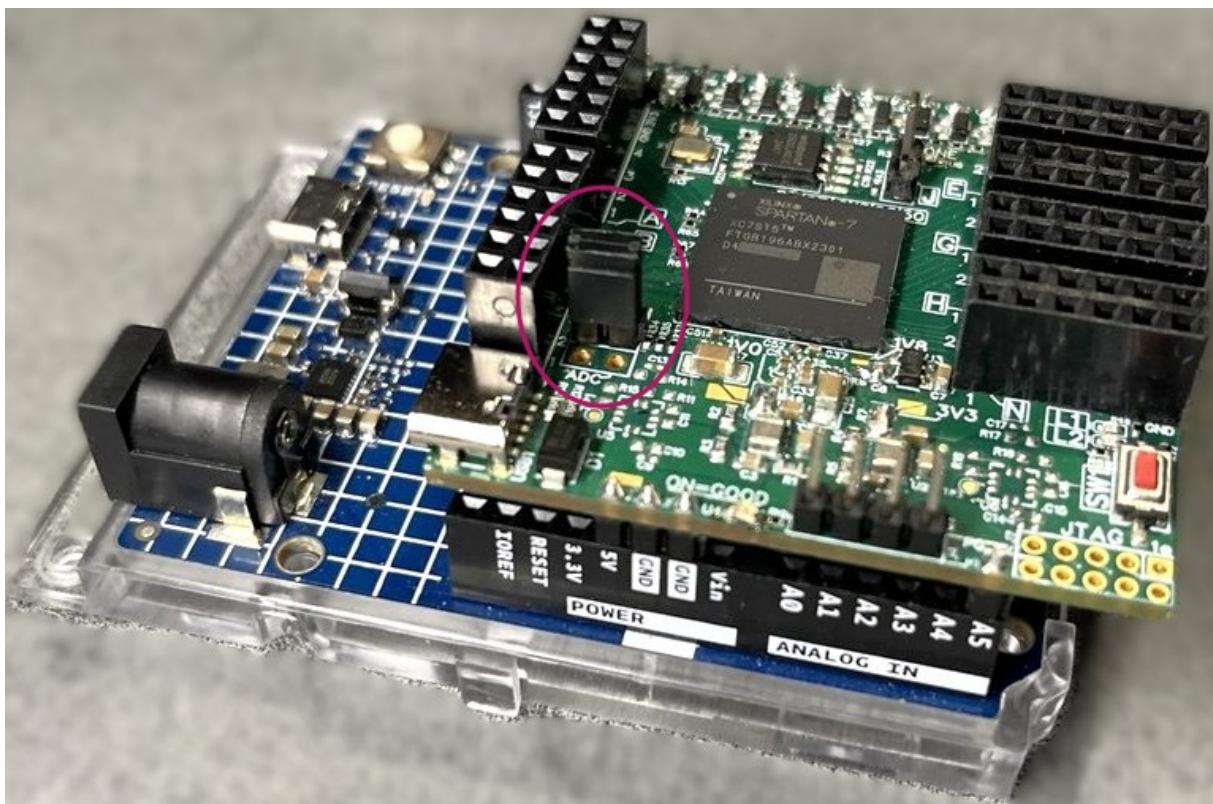


## 2.4 Upload binary FLASH SNOR (metodo 1)

In questo capitolo scaricheremo nella FLASH-SNOR presente sul "["MODULO FPGA SPARTAN7"](#)" il file binario con l'ausilio di Arduino R4 e Tera Term.

È importante cancellare la FLASH-SNOR ad ogni nuova copia del file binario.

- Chiudere il jumper "FLASH"



Chiudendo il jumper "FLASH" la FPGA entra in modalità "stand-alone" ovvero carica in automatico al power-on o al reset il binario dalla SNOR, in questo caso non dobbiamo più caricare la FPGA ogni volta e potremmo usare il "["MODULO FPGA SPARTAN7"](#)" stand-alone alimentandolo dalla USB-C presente sul modulo.

Il file presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, in questo esempio useremo Tera Term in quanto soddisfa i due requisiti fondamentali:

1- L'uso del Flow Control XON/XOFF.

2-L'invio di massimo 256byte alla volta.

- Cancellare la FLASH-SNOR, poi premere "Y" e dare invio:

```

MENU Main          Ver0.0

1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R
    MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU
> 7 ←

```

- La FLASH ora è vuota

```

FPGA STATUS: RESET <hold>
You want erase whole chip [Y/N]: y
Chip Erase in progress...
.....
Chip Erase completed!

Exec. : 8000 ms
FPGA STATUS: RESET <release>
> 

```

- Ora possiamo caricare il file binario nella FLASH-SNOR premi il tasto "9" e dai invio:

```

> 9
FPGA STATUS: RESET <hold>

Execute chip Erase or erase blocks equal to the file size!
Enter start address (hex): 

```

Qui ci verrà ricordato di cancellare la FLASH prima di scriverla e poi ci chiede da quale indirizzo vogliamo scrivere il file binario, partiremo dallo 0

```

> 9
FPGA STATUS: RESET <hold>

Execute chip Erase or erase blocks equal to the file size!
Enter start address (hex): 0 ←

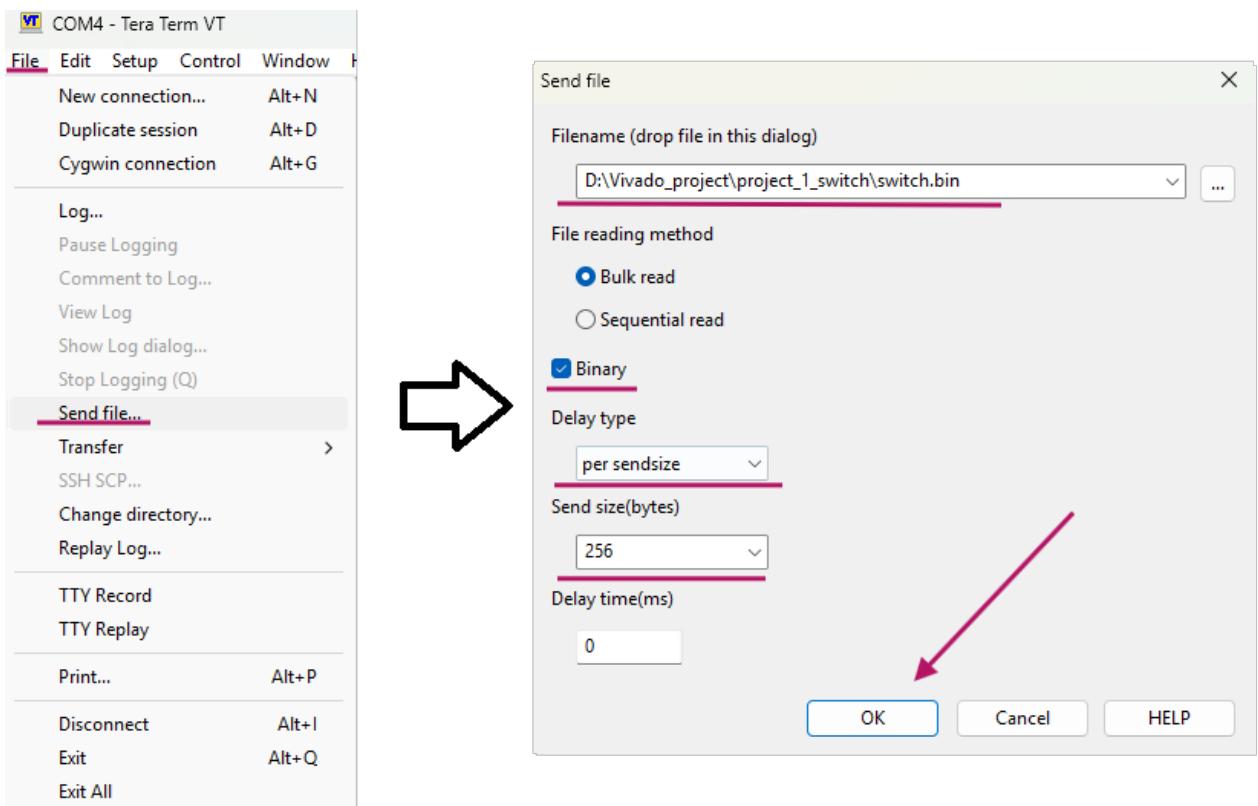
```

- Ora Arduino è pronto per ricevere il file binario che si trova anche in repository:  
<https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.bin>

```
Enter start address (hex): 0
***** TERATEM *****
* General Setting:
*   -Setup>Terminal> Receive=CR; Transmit=CR
*   -Setup>Serial port>
*     Speed=115200; Data=8b; Parity=none; Stop bits=1
*     Flowcontrol=Xon/Xoff; Transmit delay=0msec
* Version:
*   -Tested with verion 5.4.0
* Transfer setting:
*   -File>Send file>
*     Filename: <file>
*     Binary:[checked]
*     delay type: <per sendsize>
*     [send size(bytes): 256 ]
*     delay time(ms): <0 or 1>

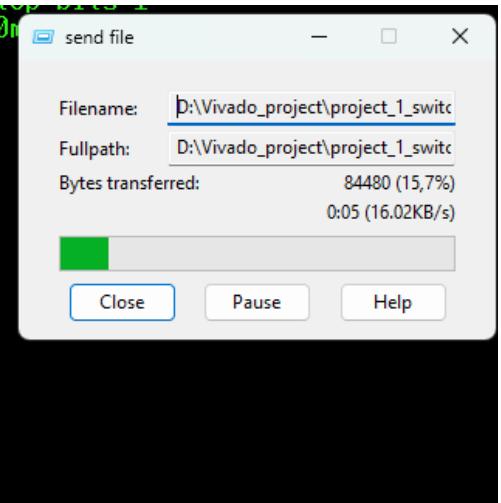
Minimun file size is 256byte!
Do not press any button on keyboard!
Ready to recive data from UART...
```

- In Tera Term andare su "File/Send file..." poi selezionare il file bitstream che trovi sotto:  
Spuntare "Binary", selezionare "Delay type/per sendsize/256" e premere OK



- Si avvierà il trasferimento

```
*      Flowcontrol=Xon/Xoff; Transmit delay=0ms
* Version:
*   -Tested with verion 5.4.0
* Tranfer setting:
*   -File>Send file>
*     Filename: <file>
*     Binary:[checked]
*     delay type: <per sendsize>
*     [send size(bytes): 256 ]
*     delay time(ms): <0 or 1>
Minimun file size is 256byte!
Do not press any button on keyboard!
Ready to recive data from UART...
Writing to SPI NOR started...
Write 256 bytes - Block #327
```



- Al termine della scrittura in "FLASH-SNOR" potremo vedere che la FPGA è programmata "ON FIRE"

```
Writing to SPI NOR started.  
Write 256 bytes - Block #2104  
Writing last byte buffer: 220  
Last buffer size220 byte!  
File saved into SPI NOR!  
Total blocks written: 2105  
Total bytes written: 538844  
CRC32: 2AA4D7AA  
  
Exec. : 27784 ms  
FPGA STATUS: RESET <release>  
> █
```

Altrimenti per vedere lo stato della FPGA seleziona l'opzione "4"

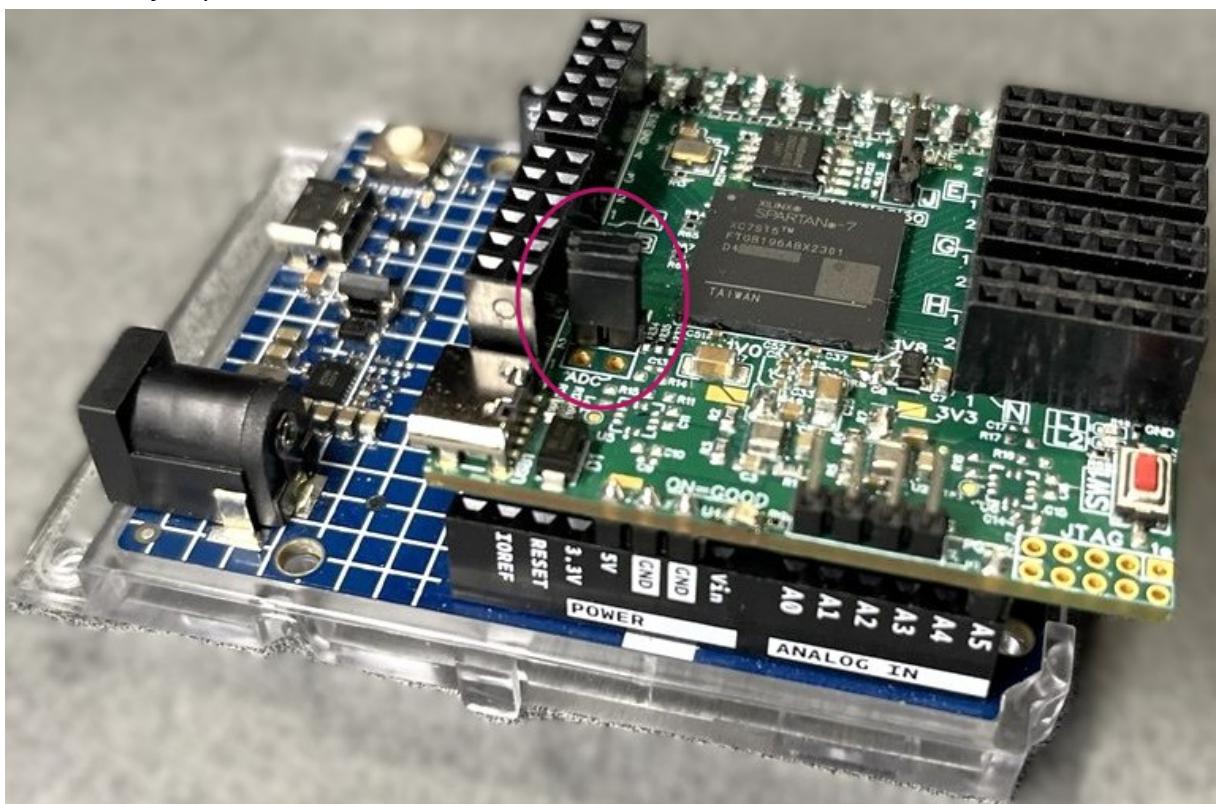
```
> 4  
FPGA STATUS: DONE <programmed>  
>
```

## 2.5 Upload binary FLASH SNOR (metodo 2)

In questo capitolo scarichiamo nella FLASH-SNOR presente sul "MODULO FPGA SPARTAN7" il file binario con l'ausilio di Arduino R4 e Tera Term.

È importante cancellare la FLASH-SNOR ad ogni nuova copia del file binario.

- Chiudere il jumper "FLASH"



Chiudendo il jumper "FLASH" la FPGA entra in modalità "stand-alone" ovvero carica in automatico al power-on o al reset il binario dalla SNOR, in questo caso non dobbiamo più caricare la FPGA ogni volta e potremmo usare il "MODULO FPGA SPARTAN7" stand-alone alimentandolo dalla USB-C presente sul modulo.

Il file presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, in questo esempio useremo Tera Term in quanto mette a disposizione il trasferimento XMODEM, a differenza del capitolo [2.4](#) è necessario solo specificare la dimensione del file da scaricare.

- Cancellare la FLASH-SNOR, poi premere "Y" e dare invio:

```

MENU Main          Ver0.0
1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R
    MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU
> 7 ←

```

- La FLASH ora è vuota

```

FPGA STATUS: RESET <hold>
You want erase whole chip [Y/N]: y
Chip Erase in progress...
.....
Chip Erase completed!
Exec. : 8000 ms
FPGA STATUS: RESET <release>
> 

```

- Ora possiamo caricare il file binario nella FLASH-SNOR premi il tasto "10" e dai invio:

```

> 10
FPGA STATUS: RESET <hold>
  Opcode 02h

Execute chip Erase or erase blocks equal to the file size!
Enter FPGA size (1=7$6, 2=7$15, 3=7$25, 4=7$50, 5=custom): 

```

Qui ci verrà ricordato di cancellare la FLASH prima di scriverla.

Selezionare la FPGA montata sulla board "S715" digitando "2" e poi "INVIO":



```

Execute chip Erase or erase blocks equal to the file size!
Enter FPGA size (1=7$6, 2=7$15, 3=7$25, 4=7$50, 5=custom): 2

```

- Ora inseriamo da che indirizzo vogliamo scrivere la FLASH-SNOR, in questo caso "0"

```
Enter FPGA size (1=7S6, 2=7S15, 3=7S25, 4=7S50, 5=custom): 2
Parsed file size byte: 538844
Enter start address (hex): 0
```



è pronto per la ricezione del file

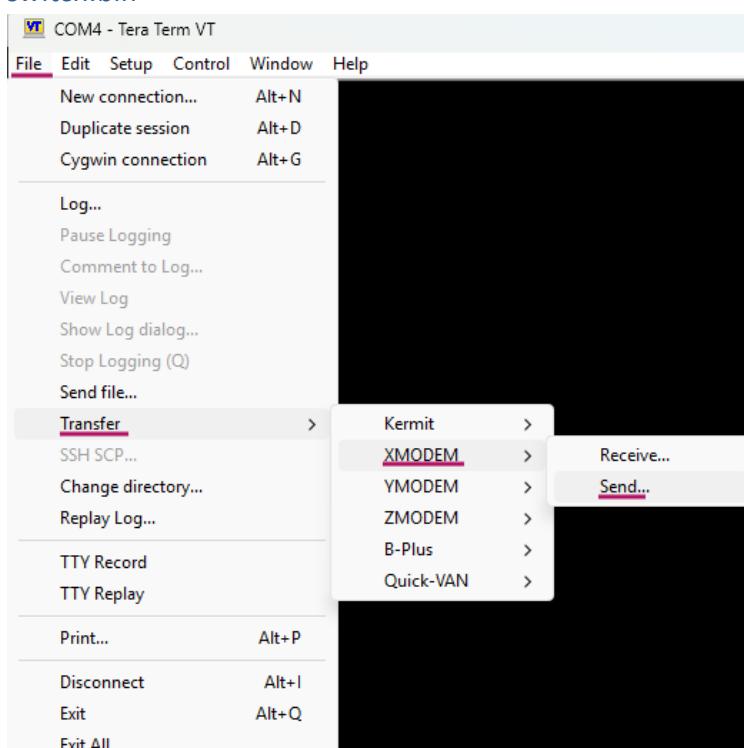
```
Enter start address (hex): 0
Do not press any button on keyboard!
Ready to receive...
```

- In Tera Term andare su "File/Transfer/XMODEM/Send..." poi selezionare il file bitstram che trovi sotto:

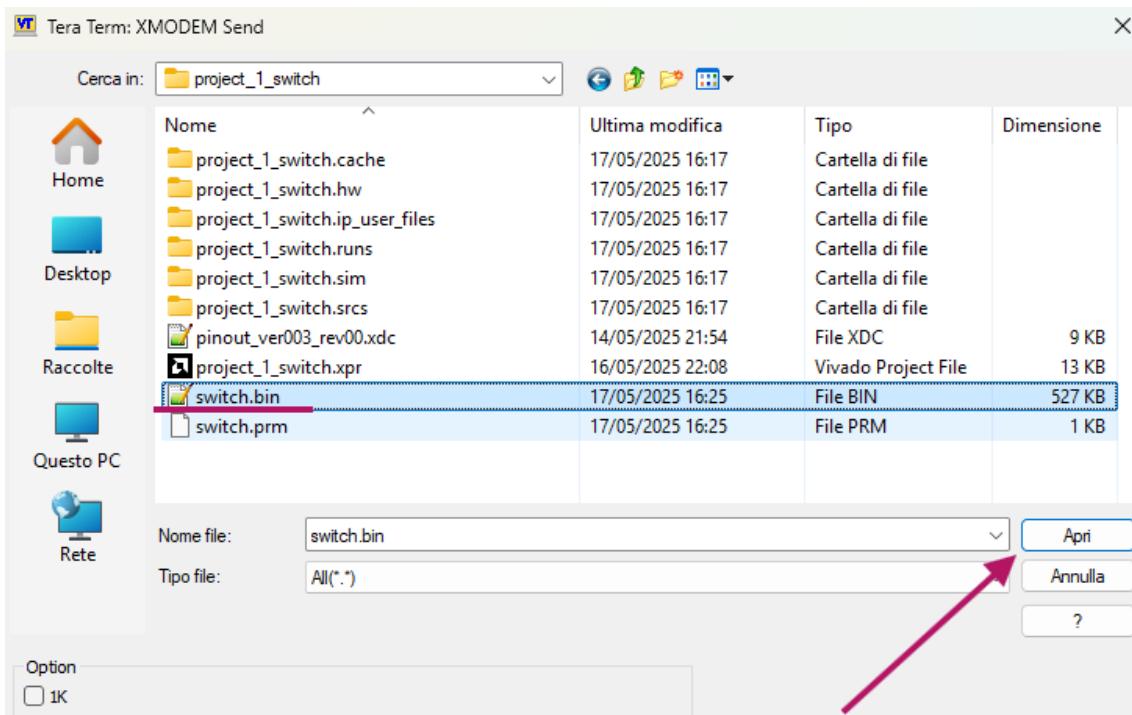
*D:\Vivado\_project\project\_1\_switch\project\_1\_switch.runs\impl\_1\switch.bin*

o

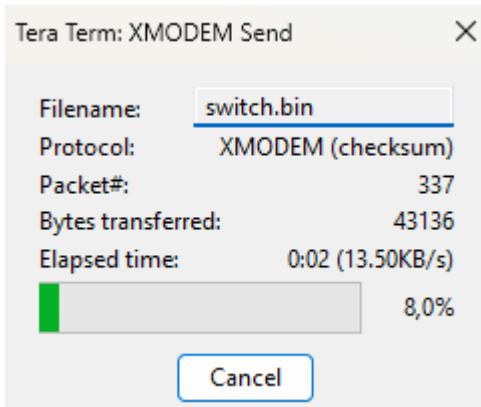
*https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.bin*



- Selezionare il file "switch.bin"



- Si aprirà una finestra di avanzamento di Upload



- Al termine della scrittura in "FLASH-SNOR" potremo vedere che la FPGA è programmata "ON FIRE"

```
Ready to receive...          XMODEM Transfer Completed!
File saved into SPI NOR!
Total bytes written: 538844
Exec. : 25787ms
FPGA STATUS: RESET <release>
>
```

Altrimenti per vedere lo stato della FPGA seleziona l`opzione "4"

```
> 4
FPGA STATUS: DONE <programmed>
>
```

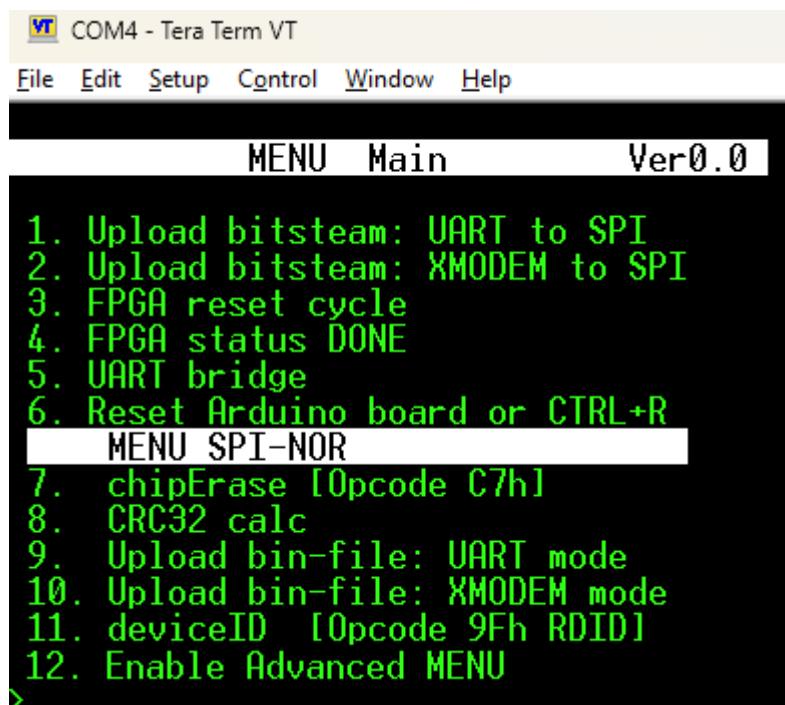
### 3 Guida alle opzioni del menu numerico nella CLI

Questo capitolo è una guida dettagliata al software realizzato su piattaforma Arduino. Ogni funzione verrà descritta nel dettaglio, seguendo l'ordine del menu numerico implementato nell'interfaccia a riga di comando.

Il programma per Arduino R4 è disponibile al seguente repository:

[https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/software/FPGA\\_Shield-cli.ino](https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/software/FPGA_Shield-cli.ino)

Dopo aver uploadato il programma in Arduino collegati con Tera Term(vedi [4.3](#) per le configurazioni)



```

VT COM4 - Tera Term VT
File Edit Setup Control Window Help

MENU Main Ver0.0

1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R
    MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU
>

```

Il programma si presenta come una CLI (Command Line Interface) che consente all'utente di interagire tramite un menu numerico.

Ogni volta che si preme Invio, il menu viene ristampato, permettendo all'utente di visualizzare nuovamente le opzioni disponibili.

Premendo Ctrl+C, è possibile resettare Arduino nella maggior parte delle situazioni, interrompendo l'esecuzione del programma in corso.

Le funzioni principali che utilizzeremo sono state raggruppate sotto la voce "MENU Main", mentre sotto la voce "Menu SPI-NOR" troviamo le opzioni relative alla SNOR FLASH presente sul "[MODULO FPGA SPARTAN7](#)" e altre funzionalità di debug. Sono riportati anche gli Opcode utilizzati per la comunicazione con SNOR-FLASH, che sono dei comandi standard delle memorie SNOR.

La comunicazione tra Arduino e la FLASH-SNOR avviene tramite SPI la stessa SPI è usata dalla FPGA quando è in "Master Mode" ovvero quando il ponticello "FLASH" è chiuso, per evitare interferenze la FPGA è messa sotto reset durante la lettura della SNOR per poi rilasciare il reset alla fine dell'operazione, le operazione di reset enable e reset disable sono riportate in CLI.

Nei sotto-capitoli successivi troverai la descrizione per ogni voce del MENU

### 3.1 "1. Upload bitsteam: UART to SPI"

Questa opzione consente di caricare un file bitstream per la FPGA via UART e di scriverlo direttamente sulla trama SPI FPGA. Ricorda che dopo il reset della FPGA la FPGA si sprogramma e non verrà mantenuta memoria.

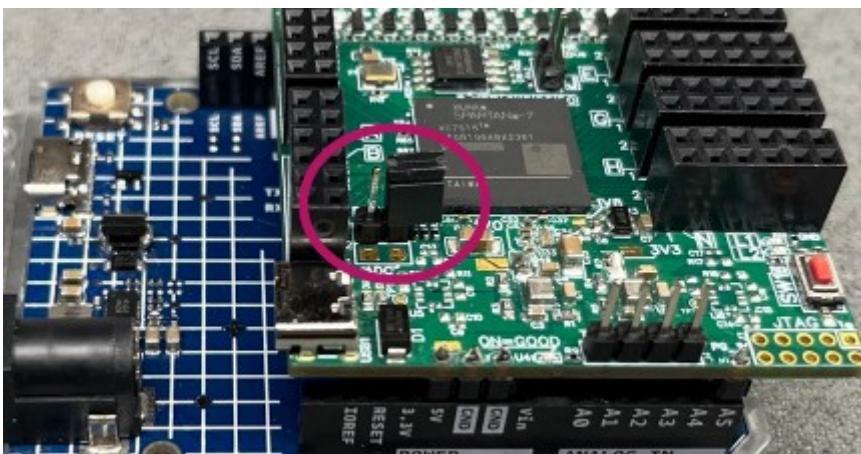
Il file presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, questa funzione è supportata da Tera Term in quanto soddisfa i due requisiti fondamentali:

- 1- L'uso del Flow Control XON/XOFF.
- 2- L'invio di massimo 256byte alla volta.

Il "MODULO FPGA SPARTAN7" deve essere in "Slave-SPI" altrimenti l'operazione verrà interrotta in automatico mostrando il seguente messaggio:

```
> 1
FPGA STATUS: RESET <hold>
FPGA STATUS: RESET <release>
FPGA STATUS: DONE <LOW,not programmed>
! FPGA in MASTER MODE remove jumper FLASH. Aborting...
```

Per configurare il modulo in "Slave-SPI" bisogna aprire il ponticello



Tutti i punti passo dopo posso sono stati li puoi trovare al capitolo [2.2](#).

#### Possibili problemi:

- Ricordarsi di non usare cavi USB-C troppo lunghi.
- Nel caso di problemi di alimentazione il "MODULO FPGA SPARTAN7" invierà tramite un Power Good collegato al pin A5 di Arduino il segnale di allarme. Questo può succedere se superiamo la corrente erogabile dalla porta USB, ricorda che nei progetti più avanzati la FPGA richiede molta corrente ed è consigliabile alimentare il modulo dalla porta USB-C dedicata.

Verifica che sulla seriale non ci sia questo messaggio:

```
> WARNING FAIL PG
```

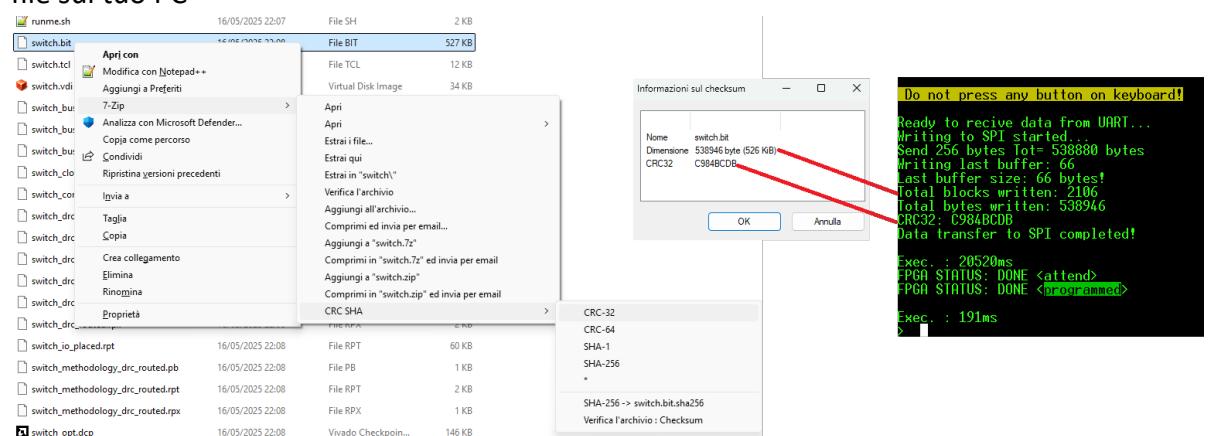
Lo stato di FAIL del PG verrà registrato in Arduino e verrà stampato insieme al MENU, il messaggio si cancellerà al reset di Arduino.

```

MENU Main Ver0.0
WARNING FAIL PG
1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R
    MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU
>

```

- Per verificare se i dati ricevuti da Arduino sono corretti... alla fine della programmazione verranno stampate la dimensioni del File e il CRC32 del File, verifica con 7zip se corrisponde con il file sul tuo PC



## 3.2 "2. Upload bitsteam: XMODEM to SPI"

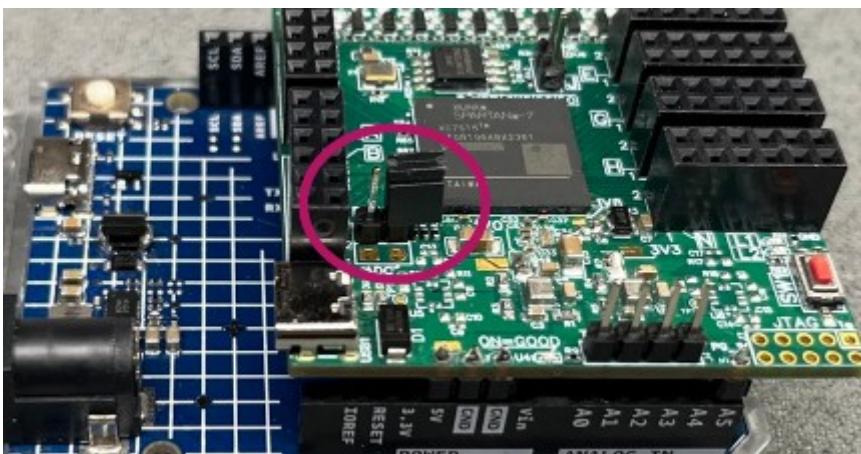
Questa opzione consente di caricare un file bitstream per la FPGA via UART e di scriverlo direttamente sulla trama SPI FPGA. Ricorda che dopo il reset della FPGA la FPGA si sprogramma.

Il file presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, useremo Tera Term in quanto mette a disposizione il trasferimento XMODEM, il quale rileva l'integrità dei dati trasmessi con checksum( non affidabile come un CRC32).

Il "MODULO FPGA SPARTAN7" deve essere in "Slave-SPI" altrimenti l'operazione verrà interrotta in automatico mostrando il seguente messaggio:

```
> 1
FPGA STATUS: RESET <hold>
FPGA STATUS: RESET <release>
FPGA STATUS: DONE <LOW,not programmed>
! FPGA in MASTER MODE remove jumper FLASH. Aborting...
```

Per configurare il modulo in "Slave-SPI" bisogna aprire il ponticello



Tutti i punti passo dopo posso sono stati li puoi trovare al capitolo [2.3](#).

### Possibili problemi:

- Ricordarsi di non usare cavi USB-C troppo lunghi.
- Nel caso di problemi di alimentazione il "MODULO FPGA SPARTAN7" invierà tramite un Power Good collegato al pin A5 di Arduino il segnale di allarme. Questo può succedere se superiamo la corrente erogabile dallo porta USB, ricorda che nei progetti più avanzati la FPGA richiede molta corrente ed è consigliabile alimentare il modulo dalla porta USB-C dedicata.

Verifica che sulla seriale non ci sia questo messaggio:

```
> WARNING FAIL PG
```

Lo stato di FAIL del PG verrà registrato in Arduino e verrà stampato insieme al MENU, il messaggio si cancellerà al reset di Arduino.

```

MENU Main Ver0.0
WARNING FAIL PG
1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R
    MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU IOR
>

```

### 3.3 "3. FPGA reset cycle"

Con il comando numero 3 si effettuerà un reset del "MODULO FPGA SPARTAN7" tramite il pin D9 (PB) e poi verrà rilasciato, se il modulo è in "Master-Spi" e nella FLASH-SNOR è presente un binario la FPGA si programmerà altrimenti no.

### 3.4 "4. FPGA status DONE"

Lo stato della FPGA è riportato dal pin del "MODULO FPGA SPARTAN7" D7(DONE).

Se è programmata :

```

> 4
FPGA STATUS: DONE <programmed>
>

```

Se non è programmata:

```

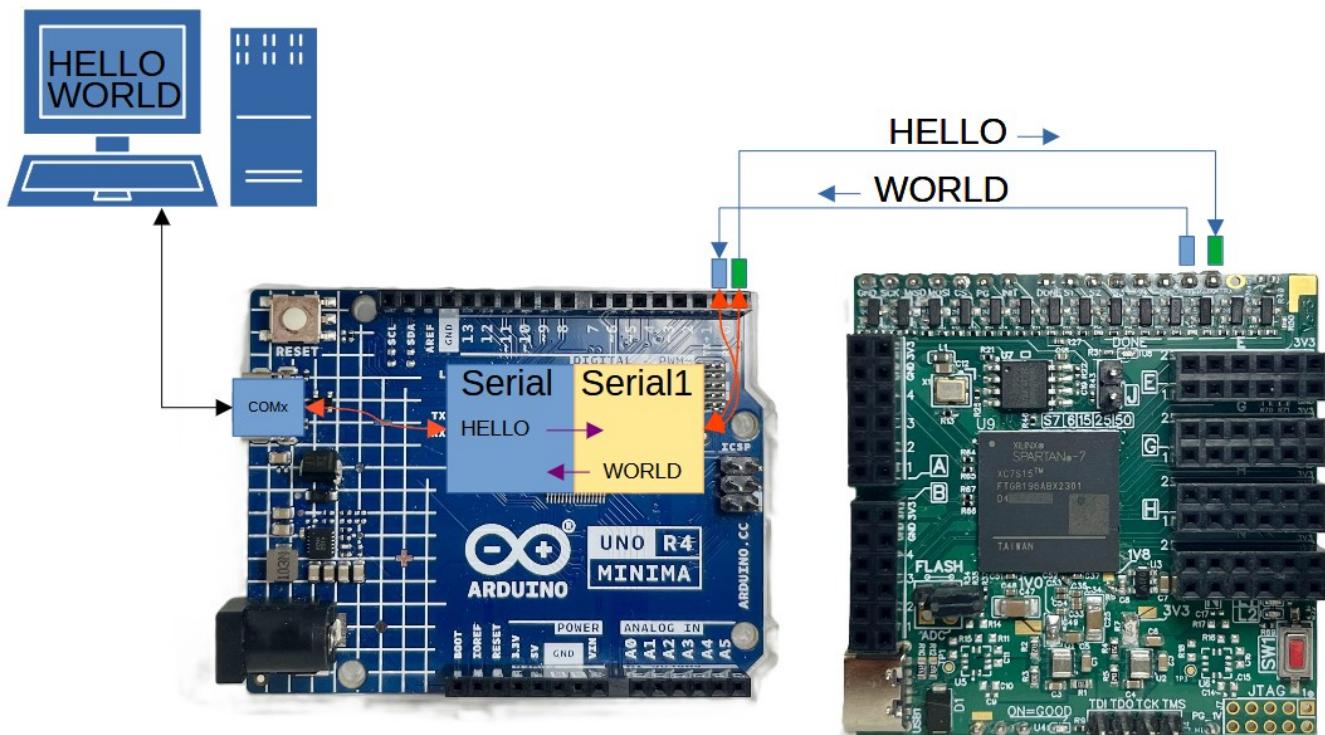
> 4
FPGA STATUS: DONE <LOW,not programmed>
>

```

### 3.5 "5. UART bridge"

Sul "MODULO FPGA SPARTAN7" sono presenti due pin UARTRX e UARTTX collegati a D0 e D1 di Arduino.

I pin D0 e D1 di Arduino sono segnali UART indipendenti dalla porta collegata al PC USBC-UART. In questa modalità si perderà l'uso del MENU e verranno copiati i dati da e per UARTRX e UARTTX del "MODULO FPGA SPARTAN7" sulla UART-USBC del PC le velocità sono per entrambe le porte 115200.



Per uscire da questa modalità premi la combinazione di tasti Ctrl+C

```
> 5
Exit Ctrl+C
Ctrl+C detected
> |
```

### 3.6 "6. Reset Arduino board or CTRL+R"

Per resettare Arduino R4 è possibile eseguirlo da MENU e con la combinazione "CTRL+R"

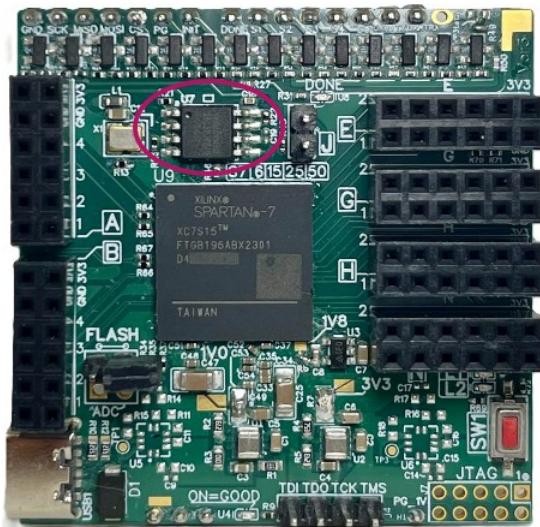
Il reset usa la funzione di "NVIC\_SystemReset()" di Arduino per il core Renesas.

```
> 6
Resetting Arduino...
```

La combinazione "CTRL+R" è disabilitata durante le operazioni di INVIO e RICEZIONE dati tramite UART.

### 3.7 "7. chipErase [Opcode C7h]"

Queste è una funzione relativa alla FLASH-SNOR presente "MODULO FPGA SPARTAN7" che permette di cancellare l'intera memoria, questa funzione è da richiamare prima di usare le funzioni di scrittura della FLASH-SNOR presente nei capitoli [2.4](#) e [2.5](#).



- La FLASH ora è vuota

### 3.8 "8. CRC32 calc"

Questa funzione calcola il CRC32 di una sezione di memoria FLASH-SNOR nel quale normalmente scriviamo il binario della FPGA vedi capitoli [2.4](#) e [2.5](#).

Questa funzione è utile per debug per verificare l'integrità del file binario scritto nella FLASH-SNOR ed è possibile confrontarlo con il file sul PC utilizzando ad esempio 7-zip

- Selezionare lo start address, il default è 0

> 8  
FPGA STATUS: RESET <hold>  
Enter start address (hex): 0

- Ora definire la dimensione del file da leggere, se non si conosce la dimensione si possono usare i default proposti

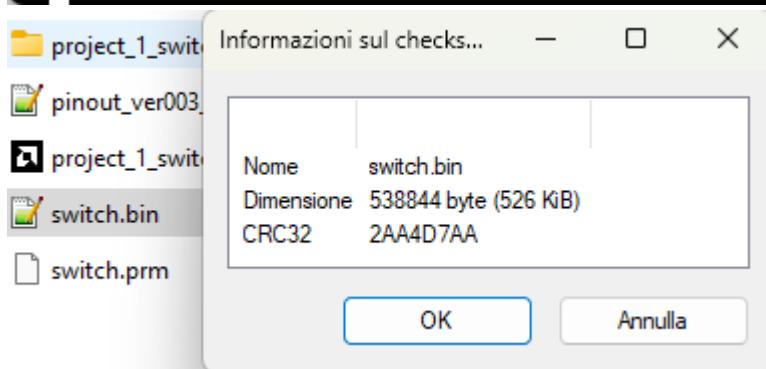
```
> 8
FPGA STATUS: RESET <hold>
Enter start address (hex): 0
Enter FPGA size (1=7$6, 2=7$15, 3=7$25, 4=7$50, 5=custom): 2
```

- Verrà indicata la dimensione del file che si sta comparando, il tempo di esecuzione è circa 76s per 500Kb

```
Enter FPGA size (1=7S6, 2=7S15, 3=7S25, 4=7S50, 5=custom): 2
Parsed file size byte: 538844 ←
Opcode 03h
Ready to receive...Calculating CRC32...
```

- Ora puoi comparare il CRC32 calcolato con il CRC32 del file presente sul PC usando 7-zip

```
Parsed file size byte: 538844
Opcode 03h
Ready to receive...Calculating CRC32...
CRC32 : 2AA4D7AA ←
Exec. : 76737 ms
FPGA STATUS: RESET <release>
>
```



La comunicazione tra Arduino e la FLASH-SNOR avviene tramite SPI la stessa SPI è usata dalla FPGA quando è in "Master Mode" ovvero quando il ponticello "FLASH" è chiuso, per evitare interferenze la FPGA è messa sotto reset durante la lettura della SNOR per poi rilasciare il reset alla fine dell'operazione

### 3.9 "9. Upload bin-file: UART mode"

Il "MODULO FPGA SPARTAN7" come detto in precedenza ha una memoria non volatile FLASH-SNOR nella quale è possibile scrivere il proprio file binario.

Il file binario presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, in questo esempio useremo Tera Term in quanto soddisfa i due requisiti fondamentali:

- 1- L'uso del Flow Control XON/XOFF.
- 2-L'invio di massimo 256byte alla volta.

Successivamente alla scrittura della FLASH-SNOR per caricare in automento il binario bisogna impostare il modulo in "Master SPI Mode" in questa modalità il caricamento del binario FPGA verrà eseguito in automati rendendo inoltre "MODULO FPGA SPARTAN7" "stand-alone".

È importante cancellare la FLASH-SNOR ad ogni nuova copia del file binario.

La procedura passo-passo è descritta al capitolo [2.4](#).

Ora è possibile usare la scheda anche senza l'uso di Arduino fornendo l'alimentazione della USB-C.



Altrimenti se si usa il "MODULO FPGA SPARTAN7" con Arduino per forzare il caricamento dalla FLASH-SNOR, chiudere il ponticello "FLASH" ed eseguire un ciclo di reset veri [3.3](#).

### 3.10 "10. Upload bin-file: XMODEM mode"

Il "MODULO FPGA SPARTAN7" come detto in precedenza ha una memoria non volatile FLASH-SNOR nella quale è possibile scrivere il proprio file binario.

Il file presente sul PC verrà inviato ad Arduino tramite seriale e Arduino convertirà i dati seriali UART in dati SPI, in questo esempio useremo Tera Term in quanto mette a disposizione il trasferimento XMODEM, questo è un metodo alternativo descritto nel capitolo [3.9](#).

È importante cancellare la FLASH-SNOR ad ogni nuova copia del file binario.

La procedura passo-passo è descritta al capitolo [2.5](#).

Ora è possibile usare la scheda anche senza l'uso di Arduino fornendo l'alimentazione della USB-C.



Altrimenti se si usa il "MODULO FPGA SPARTAN7" con Arduino per forzare il caricamento dalla FLASH-SNOR, chiudere il ponticello "FLASH" ed eseguire un ciclo di reset veri [3.3](#).

### 3.11 "11. deviceID [Opcode 9Fh RDID]"

Per verificare la corretta comunicazione tra Arduino e il "MODULO FPGA SPARTAN7" è possibile verificare il device ID della FLASH-SNOR presente sul modulo. Questa funzione è pensata solo per il debug e non verrà utilizzata in altre procedure.

Come hai visto vengono anche indicati gli Opcode utilizzati per la comunicazione , che non sono altro che dei comandi standard per le FLASH-SNOR.

- Comando 11 lettura device ID SNOR

```
> 11
FPGA STATUS: RESET <hold>
Manufacturer ID: 0xEE
Memory Type: 0x40
Memory Capacity: 0x16
FPGA STATUS: RESET <release>
>
```

### **3.12 "12. Enable Advanced SPI-NOR menu"**

Esistono altre funzioni di debug nascoste che si possono abilitare con la CLI opzione "12", queste funzioni sono di debug o avanzate. E sono dettagliate nei capitoli successivi.

- Abilitando le funzioni di debug il MENU avrà delle nuove voci:

> 12 Advanced MENU: Enabled ←

MENU Main Ver0.0

1. Upload bitsteam: UART to SPI
2. Upload bitsteam: XMODEM to SPI
3. FPGA reset cycle
4. FPGA status DONE
5. UART bridge
6. Reset Arduino board or CTRL+R  
MENU SPI-NOR
7. chipErase [Opcode C7h]
8. CRC32 calc
9. Upload bin-file: UART mode
10. Upload bin-file: XMODEM mode
11. deviceID [Opcode 9Fh RDID]
12. Enable Advanced MENU  
MENU Advanced ←
13. Set SPI
14. blockErase 4096/32768/65536 [Opcode 20h/52h/D8h]
15. writeEnable [Opcode 06h WREN]
16. readFlash [Opcode 03h RDSR]
17. fast-read Flash [Opcode 0Bh FRD]
18. statusRegister [Opcode=0x05 RDSR]
19. writeData [Opcode 02h PP]
20. deviceID [Opcode 90h REMS]
21. Download SNOR-Flash: UART mode
22. Download SNOR-Flash: XMODEM mode
23. Custom SPI Transaction
24. FPGA Reset: Hold
25. FPGA Reset: Release

- Per disabilitare il "MENU Advanced" invia 12 sulla CLI

> 12 Advanced MENU: Disabled  
> █

### 3.13 "13. Set SPI"

Vengono usate due tipologie di SPI una Hardware durante l'Upload del bitstream(opzioni [3.1](#) e [3.2](#)) mentre una SPI Software durante l'Upload bin-file nella FLASH-SNOR. L'unica SPI che è possibile configurare è la SPI Hardware.

La configurazione di default della SPI Hardware è la seguente:

Speed: 2000000Hz - MSBFIRST – SPI\_MODE0

- Proviamo a cambiare la configurazione della SPI, al reset di Arduino la configurazione è quella di default. In questo esempio cambiamo solo la frequenza della speed alla massima misurata con Arduino minima R4 (12MHz)

```
> 13
Default: 2000000Hz - MSBFIRST - SPI_MODE0
Enter SPI Clock (max 20000000 Hz): 12000000
```

- Impostiamo come ordine dei bit MSBFIRST

```
Enter SPI Clock (max 20000000 Hz): 12000000
Enter SPI Bit Order (0=LSBFIRST, 1=MSBFIRST): 1
```

- Infine la modalità di campionamento SPI MODE 0:

```
Enter SPI Clock (max 20000000 Hz): 12000000
Enter SPI Bit Order (0=LSBFIRST, 1=MSBFIRST): 1
Enter SPI Mode (0, 1, 2, 3): 0
```

Aumentare la velocità della SPI a 12MHz riduce i tempi di caricamento del bitstream a 12,5s questo non è un grande vantaggio rispetto a i 20s, per precauzione è consigliabile usare l'impostazione di default 2MHz.

### 3.14 " 14. blockErase 4096/32768/65536 [Opcode 20h/52h/D8h]"

È stato implementato anche la cancellazione parziale della FLASH-SNOR potendo scegliere l'indirizzo di partenza, la dimensione del tipo di erase e quanti blocchi cancellare. Per verificare prima e dopo la cancellazione dei blocchi usa l'opzione dettagliata nel capitolo [3.16](#).

### 3.15 "15. writeEnable [Opcode 06h WREN]"

Le FLASH-SNOR usate per prevenire scritture involontarie hanno un registro che abilita la scrittura della FLASH, nelle procedure al di fuori di queste di debug è gestito in automatico.

```
> 15
FPGA STATUS: RESET <hold>
WREN enabled,latch/self clear
FPGA STATUS: RESET <release>
```

### 3.16 "16. readFlash [Opcode 03h RDSR]"

L'opzione di debug di lettura del contunto della FLASH-SNOR può essere utile per confrontare il file presente sul nostro PC con il file scritto in FLASH-SNOR.

- Scegliere l'indirizzo di partenza della SNOR, attenzione il numero è viene interpretato come esadecimale

> 16  
FPGA STATUS: RESET <hold>  
Enter start address (hex): 0

- Definire i numeri di byte da leggere sempre in esadecimale

FPGA STATUS: RESET <hold>  
Enter start address (hex): 0  
Enter number of bytes (hex): 100

- Output che è possibile confrontare con il file presente sul PC (Io ho usato Notepad++ con estensione HEXviewer)

### 3.17 "17. fast-read Flash [Opcode 0Bh FRD"]

L'opzione di debug di lettura del contenuto della FLASH-SNOR può essere utile per confrontare il file presente sul nostro PC con il file scritto in FLASH-SNOR. Questa funzione è simile al capitolo [3.16](#) ma usa un Opcode diverso lo stesso che usa la FPGA quando si trova in "Master SPI".

### 3.18 "18. statusRegister [Opcode=0x05 RDSR]"

Lo status register è un registro all'interno della FLASH-SNOR che descrive vari stati in cui si trova la memoria, ad esempio quando si scrive la memoria per vedere se il processo di scrittura è finito si legge lo status register ecc.

```
> 18
FPGA STATUS: RESET <hold>
Status Register: 0x2
Write Enabled
FPGA STATUS: RESET <release>
> █
```

Questo è un estratto del Datasheet

#### 7.1 Status Registers

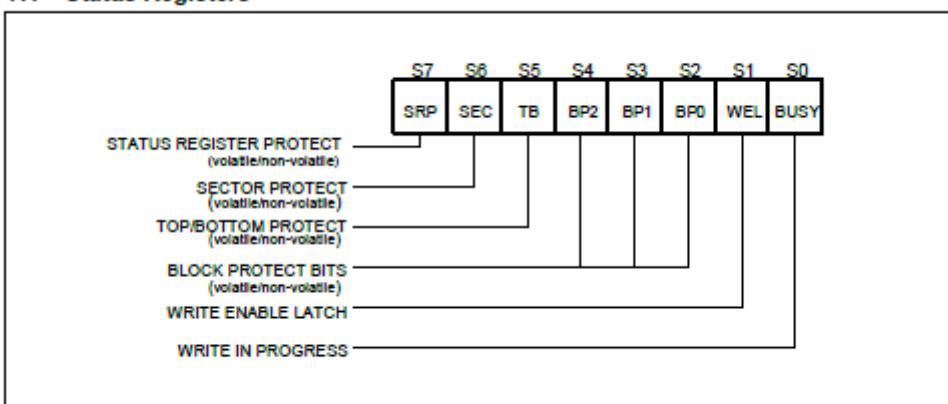


Figure 4a. Status Register-1

### 3.19 "19. writeData [Opcode 02h PP]"

Per scrivere un singolo byte è necessario prima cancellare la memoria e il minima dimensione da cancellare è un blocco (vedi 3.14) altrimenti si può cancellare l'intero chip SNOR FLASH. Per capire se il settore è cancellato il byte bisogna prima leggerlo, se è FF vuol dire che si può scrivere.

In questo capitolo useremo le funzioni spiegate nei capitoli precedenti, lo scopo di questo esempio è scrivere qualche byte in FLASH-SNOR.

- Leggi il contenuto della flash i primi 48 byte

```
> 16
FPGA STATUS: RESET <hold>
Enter start address (hex): 0
Enter number of bytes (hex): 30
Reading SNOR:
0x00000: FF | .....|
0x00010: FF | .....|.D.|
0x00020: 00 00 00 BB 11 22 00 44 FF | .....".D.|
Read complete!

Exec. : 14 ms
FPGA STATUS: RESET <release>
>
```

- Cancella il primo settore della FLASH-SNOR

```
> 14
FPGA STATUS: RESET <hold>
Enter start address (hex): 0
Enter block size (4096, 32768, 65536): 4096
Enter number of blocks: 1
Block erase 4KB from address 0x0
All blocks are erased!

Exec. : 101 ms
FPGA STATUS: RESET <release>
>
```

- Rileggi i primi 48byte, vedrai che sono tutti FF fino al 4095byte

```
> 16
FPGA STATUS: RESET <hold>
Enter start address (hex): 0
Enter number of bytes (hex): 30
Reading SNOR:
0x000000: FF FF
0x000010: FF FF
0x000020: FF FF
Read complete!

Exec. : 14 ms
FPGA STATUS: RESET <release>
>
```

- Ora è possibile scrivere un byte alla volta

```
> 19
FPGA STATUS: RESET <hold>
Erase block before write!
Enter start address (hex): 0
Enter data byte (hex): 43
Byte written to address 0x0: 0x43

> 19
FPGA STATUS: RESET <hold>
Erase block before write!
Enter start address (hex): 1
Enter data byte (hex): 49
Byte written to address 0x1: 0x49
```

```
> 19
FPGA STATUS: RESET <hold>
Erase block before write!
Enter start address (hex): 2
Enter data byte (hex): 41
Byte written to address 0x2: 0x41
> 19
FPGA STATUS: RESET <hold>
Erase block before write!
Enter start address (hex): 3
Enter data byte (hex): 4f
Byte written to address 0x3: 0x4F
```

Attenzione se si sbagli a scrivere bisogna cancellare l'intero settore

- Rileggi i primi 16byte, hai scritto la tua prima parola.

```
> 16
FPGA STATUS: RESET <hold>
Enter start address (hex): 0
Enter number of bytes (hex): 10
Reading SNOR:
0x00000: 43 49 41 4F FF ICIAO.....
Read complete!
```

- Ora spegni e riaccendi e rileggi i primi 16byte!!!

### 3.20 "20. deviceID [Opcode 90h REMS"]

Questo opzione è simile al capitolo [3.11](#) ma si differenzia dal comando inviato SPI alla FLASH-SNOR.

Questa funzione è pensata solo per il debug e non verrà utilizzata in altre procedure.

Come hai visto vengono anche indicati gli Opcode utilizzati per la comunicazione (vedi il Datasheet della FLASH-SNOR), che non sono altro che dei comandi standard per le FLASH-SNOR.

- Comando lettura device ID SNOR

```
> 20
FPGA STATUS: RESET <hold>
Manufacturer ID: 0xEF
Device ID: 0x15
FPGA STATUS: RESET <release>
>
```

### 3.21 "21. Download SNOR-Flash: UART mode"

Abbiamo già visto come scrivere scrivere il binario nella FLASH-SNOR ora vediamo come è possibile anche leggere il contenuto della memoria e copiarlo in un file sul PC.

In questo esempio il contenuto della FLASH-SNOR è il file binario presente in repository:

<https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.bin>

Per la copia del contenuto del file è necessario usare Tera Term.

- Inserire il l'indirizzo di partenza della FLASH-SNOR da leggere.

```
> 21
FPGA STATUS: RESET <hold>
***** TERATEM *****
* General Setting:
*   -Setup>Terminal> Receive=AUTO; Transmit=CR
*   -Setup>Serial port> Speed=115200; Data=8b; Parity=none; Stop bits=1;
*   Flowcontrol=Xon/Xoff; Transmit delay=0msec
* Version:
*   -Tested with verion 5.4.0
* Acquire setting:
*   -File>Log...>
*     Filename: <file>
*     Write mode: Append
*     Binary:[checked]
*Execute this configuration then set waiting time
* 1-Start log file while LED is blinking
* 2-When the LED is ON,the acquisition is in progress
* 3-When the LED is OFF,close the log file

Opcode 03h
Enter start address (hex): 0
```

- Inserire quanti byte leggere in esadecimale, ad esempio il file

<https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.bin>

Ha la dimensione di 538.844 byte che in esadecimale sono 838DC

```
Opcode 03h
Enter start address (hex): 0
Enter number of bytes (hex): 838dc
```

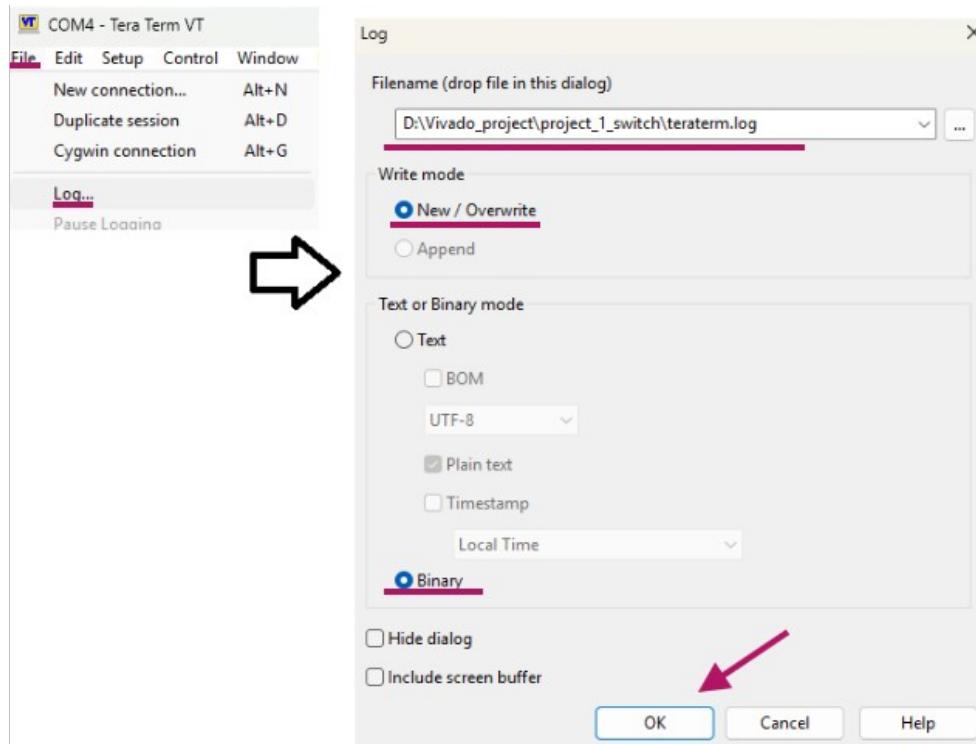
- Ora dobbiamo inserire il tempo di attesa prima che venga trasmesso il contenuto della FLASH-SNOR sulla seriale.

Questo tempo è necessario per configurare Tera Term per l'acquisizione del file da console.

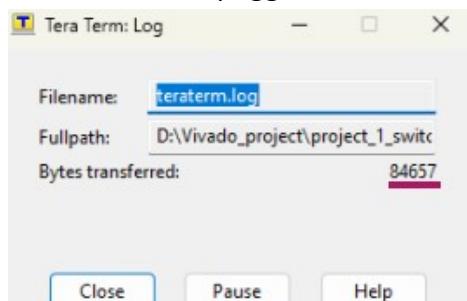
```
Opcode 03h
Enter start address (hex): 0
Enter number of bytes (hex): 838dc
Enter the waiting time before/after acquisition(s): 20
```

Il led D13 di Arduino lampeggerà per tutto questo tempo. Affrettati

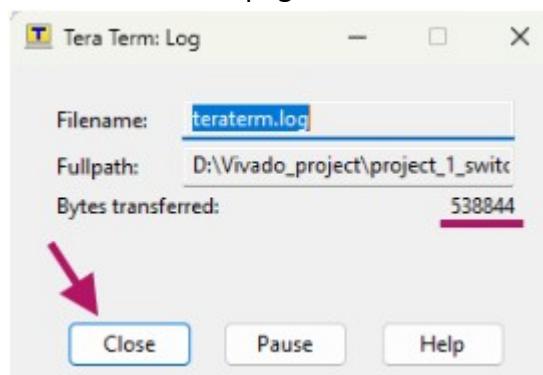
Devi essere veloce a configurare Tera Term, il led D13 lampeggia nel frattempo



- Alla fine del lampeggio del led D13 il led diventa ON fisso e vedrai questa finestra incrementare



Qual il led D13 si spegnerà avrai lo stesso tempo impostato prima per chiudere il file di .log



I byte trasferiti saranno uguali a byte inseriti prima.

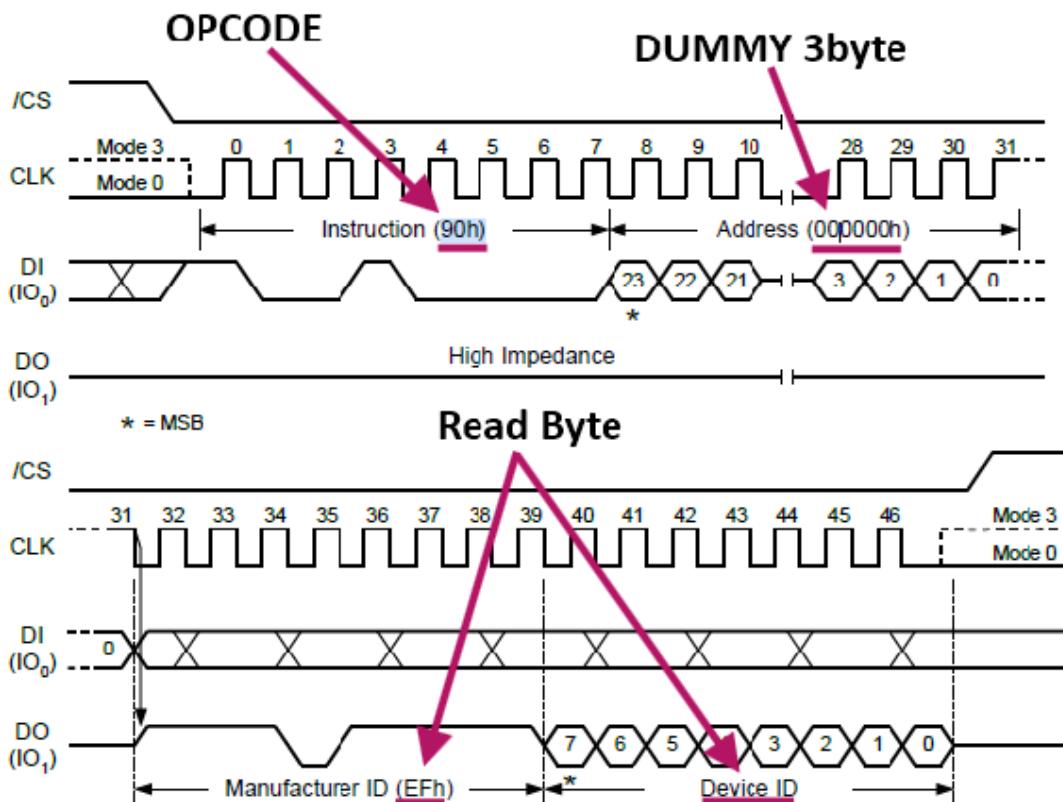
- A questo punto poi confrontare/controllare il file appena scaricato prova a verificare che sia uguale al file presente in repository usando 7zip o Notepad++

### 3.22 "23. Custom SPI Transaction"

L'opzione "Custom SPI" permette di inviare comandi personalizzati alla FLASH-SNOR ad esempio per vedere un OPCODE non integrato in questo software.

La SPI di Arduino è configurata : SCK=D13, MISO=D11 ,MOSI=D12 e CS=D10

In questo esempio leggeremo il device ID della FLASH-SNOR che ha come OpCode il 90h, qui sotto è riportato l'estratto del Datasheet relativo al OPCODE 90h



- Conifuriamo la CLI come descritto nel DS... restituisce 0xEF e 0x15

```
> 23
FPGA STATUS: RESET <hold>
Enter Write Byte (hex, e.g. opcode): 90
How many dummy bytes? 3
Dummy byte value (00=0x00, 1=0xFF): 00
How many bytes do you want to read? 2
Response: EF 15
Exec. :
1ms
FPGA STATUS: RESET <release>
>
```

- Ora possiamo confrontare il valore letto con la funzione della CLI

```
> 20
FPGA STATUS: RESET <hold>
Manufacturer ID: 0xEF
Device ID: 0x15
FPGA STATUS: RESET <release>
>
```

### 3.23 "22. Download SNOR-Flash: XMODEM mode"

Abbiamo già visto come scrivere il binario nella FLASH-SNOR ora vediamo come è possibile anche leggere il contenuto della memoria e copiarlo in un file sul PC.

Questa procedura è simile alla procedura presente nel capitolo precedente [3.21](#) ma con la semplificazione di usare il protocollo XMODEM per la ricezione del file, quindi non si deve essere particolarmente veloci a configurare Tera Term.

L'unico inconveniente è che gestisce file multipli di minimo di 128byte e multipli di 128byte, se il file non è multiplo di 128 aggiunge dei byte denominati PADDING(riempimento).

In questo esempio il contenuto della FLASH-SNOR è il file binario presente in repository:

<https://github.com/Jampag/FPGA-Shield-Arduino-compatible/blob/main/example/switch/switch.bin>

Il file ha la dimensione del file è 538.844byte non multiplo di 128byte, in questo caso il programma aggiunge dei byte per renderlo multiplo di 128byte, a questo punto il file presente sul nostro PC dovremmo tenere in considerazione questi byte o rimuoverli a mano con un editor di testo ad esempio Notepad++.

- Inserire il l'indirizzo di partenza della FLASH-SNOR da leggere.

```
> 22
FPGA STATUS: RESET <hold>
  Opcode 03h
Enter start address (hex): 0
```

- Inserire quanti byte leggere in esadecimale, ad esempio il file:

```
Enter start address (hex): 0
Enter FPGA size (1=7S6, 2=7S15, 3=7S25, 4=7S50, 5=custom): 2
Parsed file size byte: 538844
```



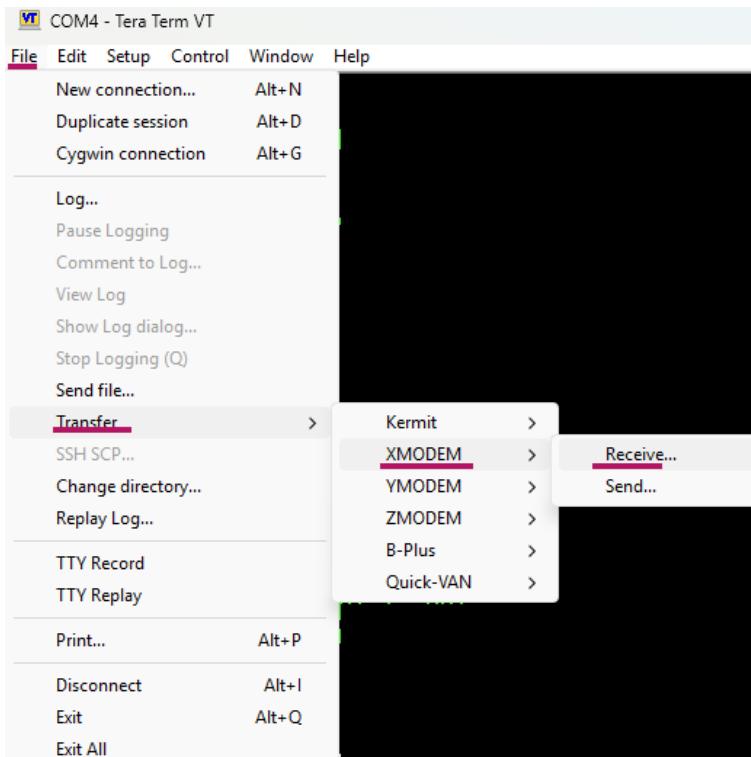
- Definire come il valore del byte di padding

```
Enter FPGA size (1=7S6, 2=7S15, 3=7S25, 4=7S50, 5=custom): 2
Parsed file size byte: 538844
```

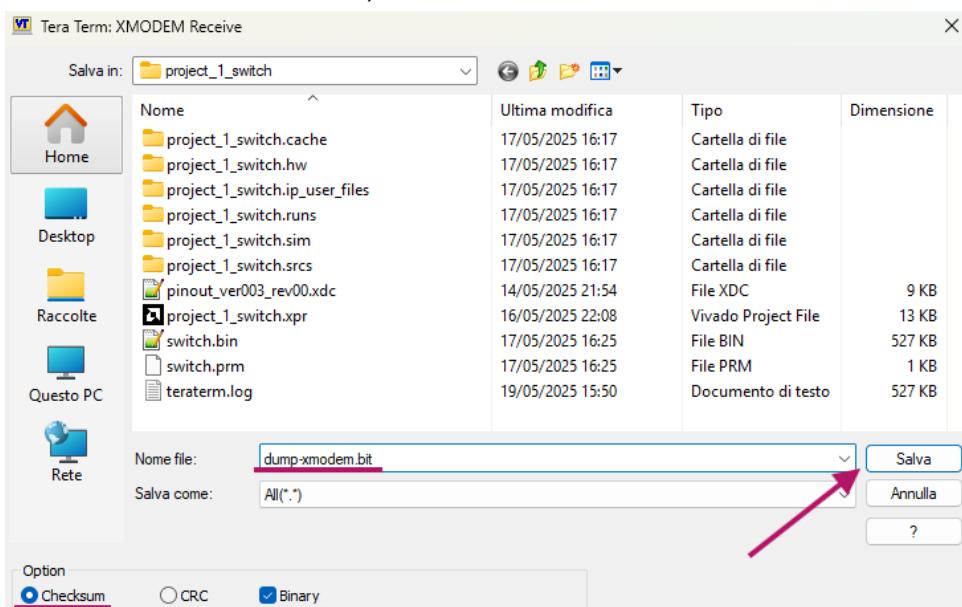
```
Warning: File size is not a multiple of 128 bytes.
Enter padding byte (hex, default 1A): 1A
```



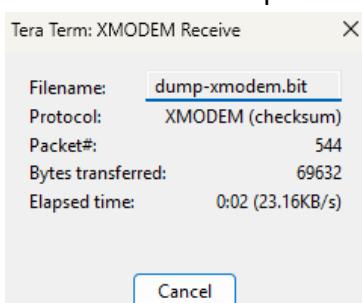
- Abilitate la ricezione tramite protocollo XMODEM



- Seleziona dove salvare il file, si avvierà in automatico la ricezione.



Succivamente l'acquisizione verrà avviata



- Fine

```
Enter padding byte (hex, default 1A): 1a
Waiting for receiver...
XMODEM Transfer Completed!
Total bytes sent: 538844

Exec. time: 22191ms
FPGA STATUS: RESET <release>
> █
```

- Se confrontiamo il file "switch.bin" con il file dump-xomdem.bit noteremo una differenza alla fine del file

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00083870	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
00083880	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
00083890	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838a0	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838b0	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838c0	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838d0	20	00	00	00	20	00	00	00	20	00	00	00	1a	1a	1a	1a	....
000838e0	1a	.....															
000838f0	1a	.....															

Normal text file

length : 538.880 lines : 3

Ln : 1 Col : 2 Sel : 1 | 1

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00083850	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
00083860	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
00083870	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
00083880	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
00083890	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838a0	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838b0	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838c0	20	00	00	00	20	00	00	00	20	00	00	00	20	00	00	00	....
000838d0	20	00	00	00	20	00	00	00	20	00	00	00	00	00	00	00	....

Normal text file

length : 538.844 lines : 3

Ln : 1 Col : 1

### 3.24 "24. FPGA Reset: Hold"

Configura il pin PG\_B "PROGRAM" corrispondete al pin D9 di arduino a 0V, questo mette la FPGA "MODULO FPGA SPARTAN7" in reset. Per rimuovere il reset vedi il capitolo [3.25](#).

- FPGA Reset ON

```
> 24
FPGA STATUS: RESET <hold>
>
```

### 3.25 "25. FPGA Reset: Realese"

Configura il pin PG\_B "PROGRAM" corrispondete al pin D9 di arduino a 5V, questo toglier il reset alla FPGA "MODULO FPGA SPARTAN7"

- FPGA Reset OFF

```
> 25
FPGA STATUS: RESET <release>
>
```

## 4 Appendix

### 4.1 Crediti

- Tera Term

Copyright (C) 1994-1998 T. Teranishi

(C) 2004-2025 TeraTerm Project

All rights reserved.

- Arduino ®

<https://www.arduino.cc/en/trademark>

Questo progetto è stato sviluppato utilizzando la piattaforma Arduino, un ambiente open-source per la prototipazione elettronica. Si ringrazia la community di Arduino e i contributori del progetto per la documentazione, gli strumenti software (IDE Arduino) e l'hardware disponibile.

Sito ufficiale: [www.arduino.cc](http://www.arduino.cc)

- Notepad++

<https://notepad-plus-plus.org/downloads/>

## 4.2 Revisione

Per verificare la revisione della board è sufficiente verificare il retro del "MODULO FPGA SPARTAN7", vedi immagine SX, inoltre le schede "MODULO FPGA SPARTAN7" supportano 4 versioni di FPGA ed è visibile sul fronte della scheda immagine DX:



Qui il riassunto delle varie versioni:

- Rev00:



CLK=100MHz, SNOR=W25Q32JV, FPGA=xc7s15ftgb196-2

- RevXX:

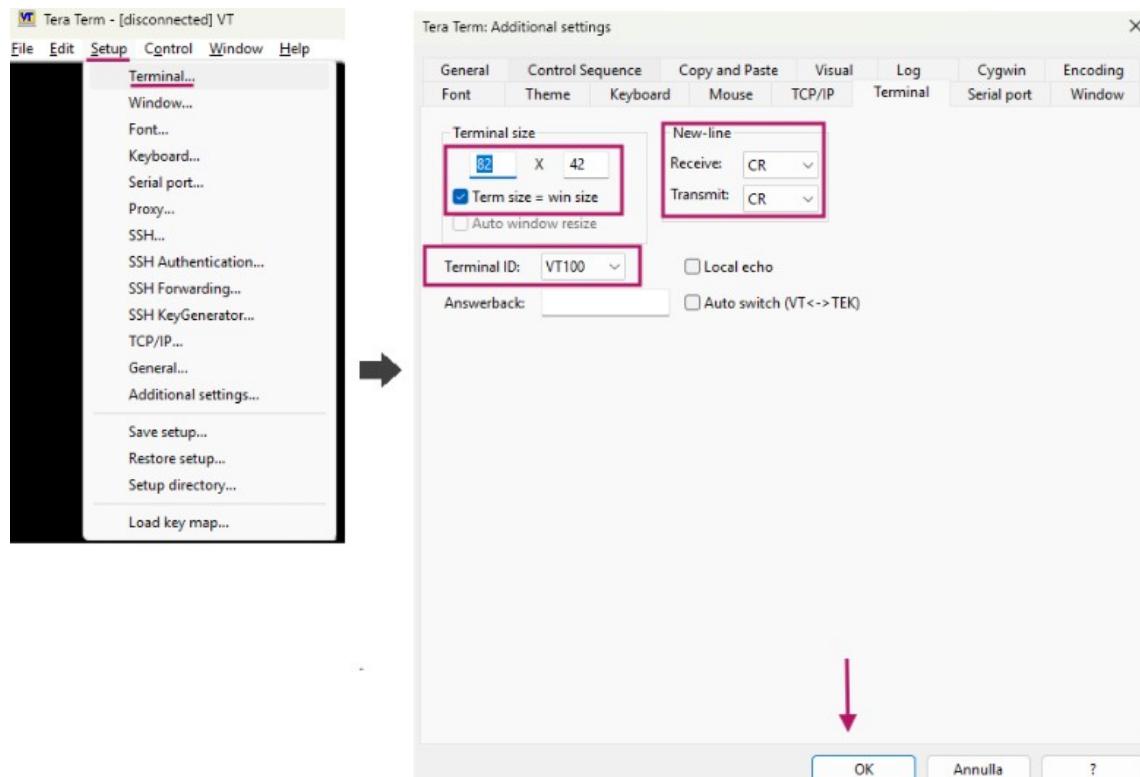
TBD

## 4.3 Configurazione Tera Term

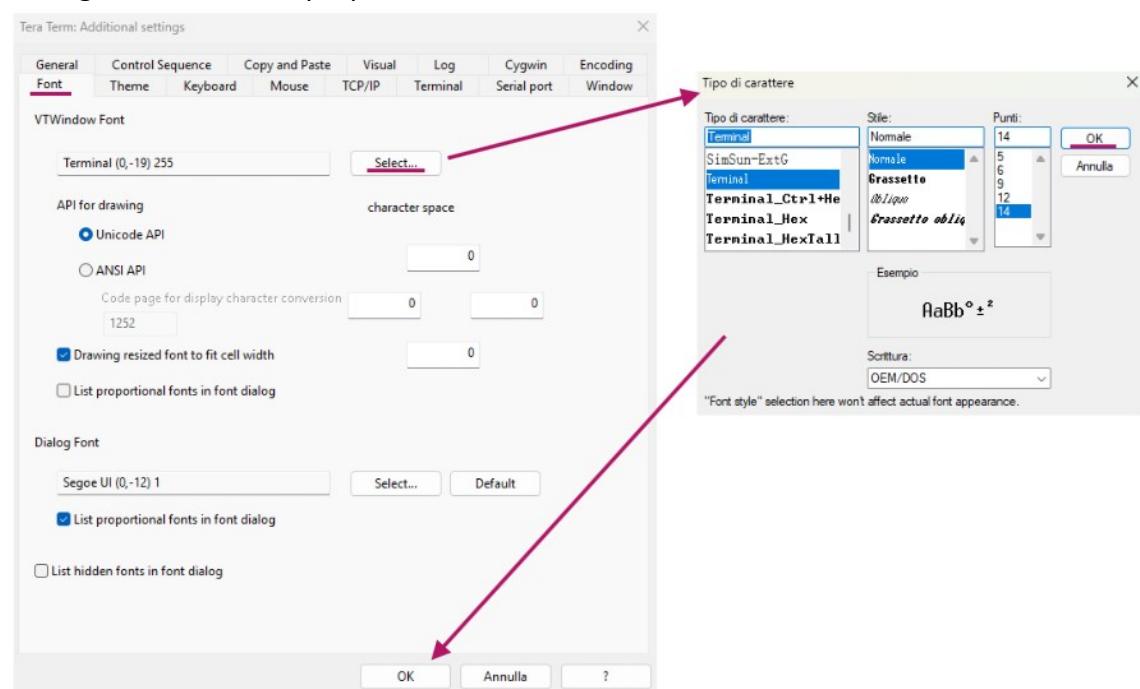
Questa configurazione riguarda Tera Term [Version 5.4.0] scaricabile da  
<https://teratermproject.github.io/>

Configurare Tera Term seguendo i passaggi sottostanti

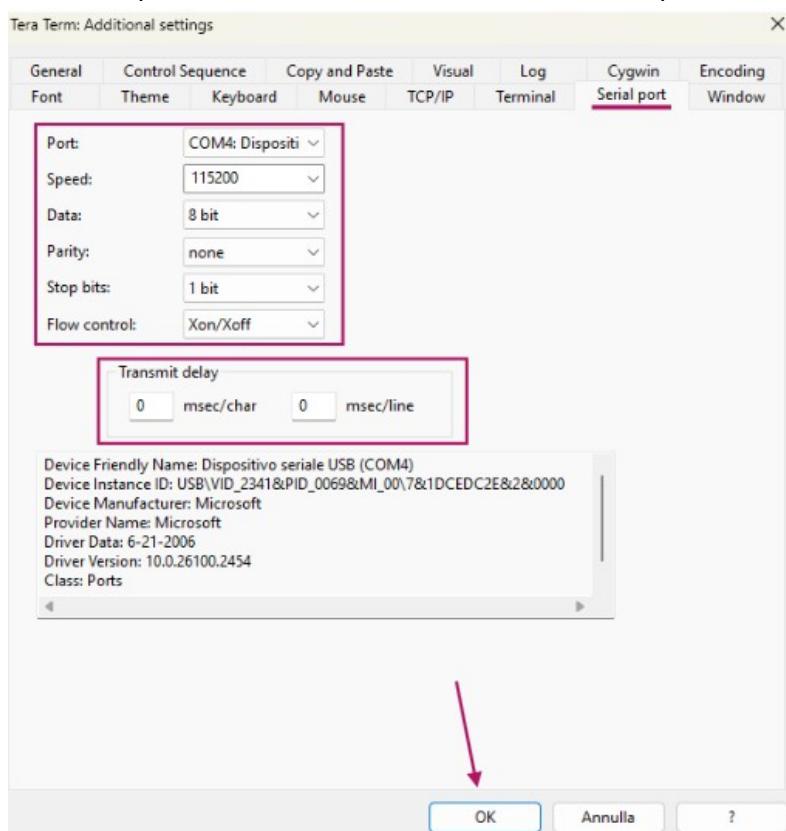
- Configurare il "Terminal"



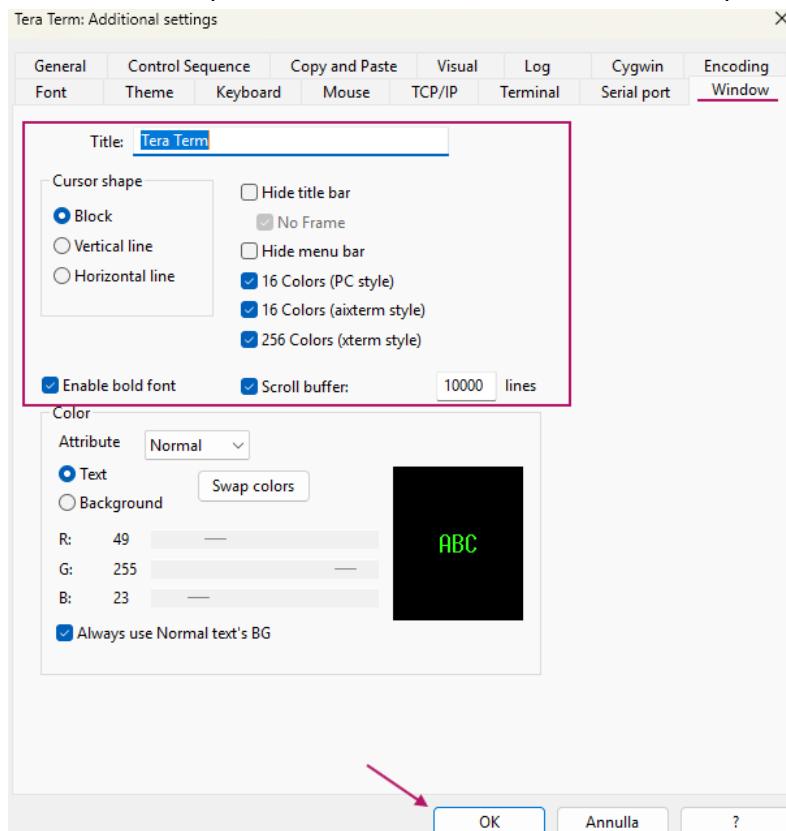
- Configurare il "Font" a propria scelta



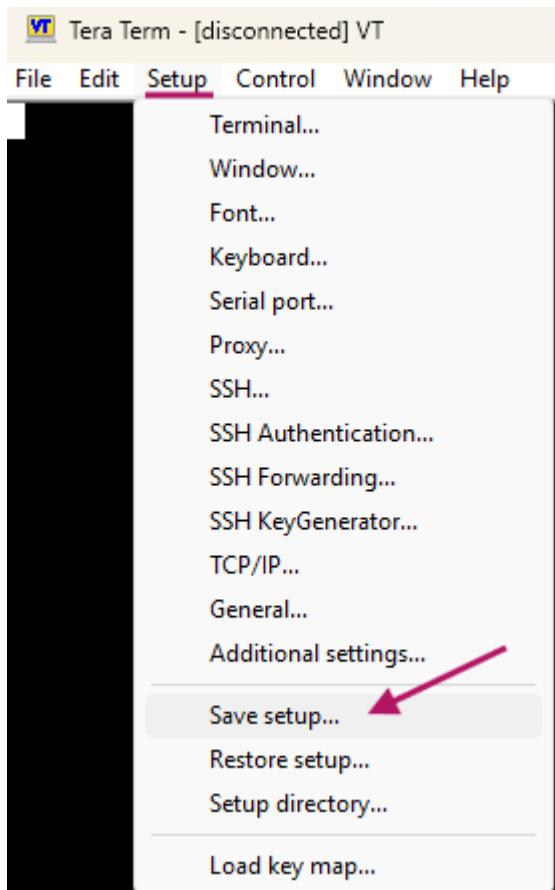
- Configurare come la "Serial port". **Attenzione la COM deve rispettare** la porta associata da Windwos per il tuo PC, nel mio caso è associata la porta COM4.



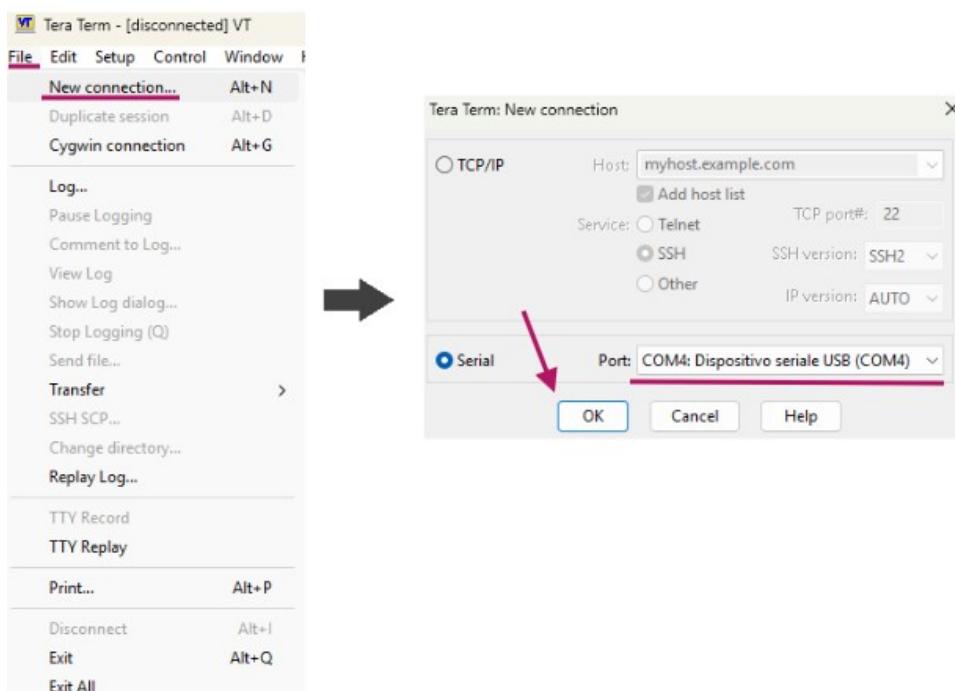
- Selezionare l'aspetto "Window", e selezionare il colore preferito



- Salvare i settaggi, alla prossima riapertura Tera Term manterrà la configurazione



- Connessione alla COMx, ricordati di selezionare la COM associata da Windows per la tua scheda Arduino:



- FINE.

## 4.4 Installazione Vivado 2024.2 Windows 11

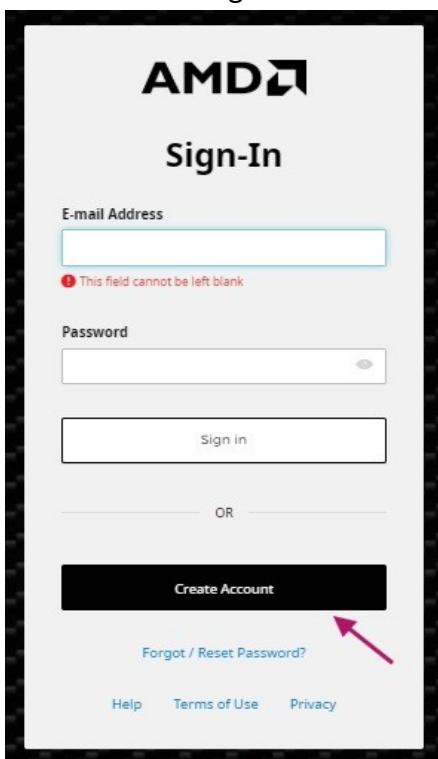
Per poter scaricare Vivado è necessario avere un account AMD

- Vai sul sito :  
<https://www.xilinx.com/support/download.html>

- Clicca su

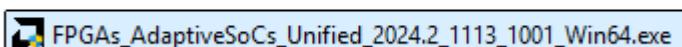


- Ora richiede la registrazione con un email, registrati e il download si avvierà:

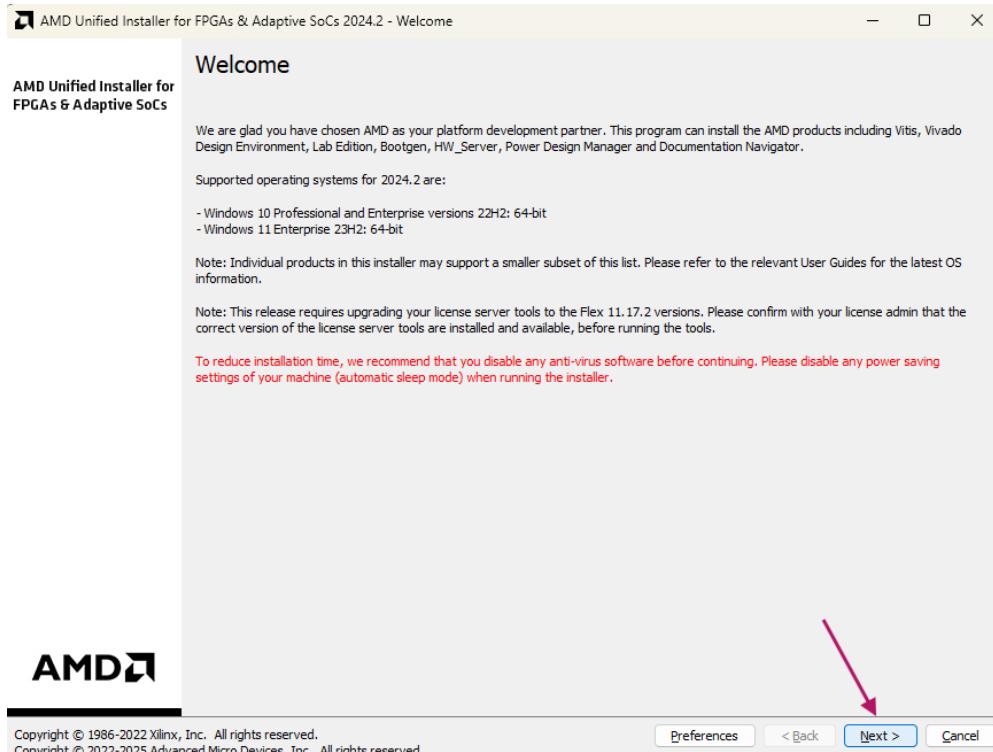


- Avviare il programma scaricato:

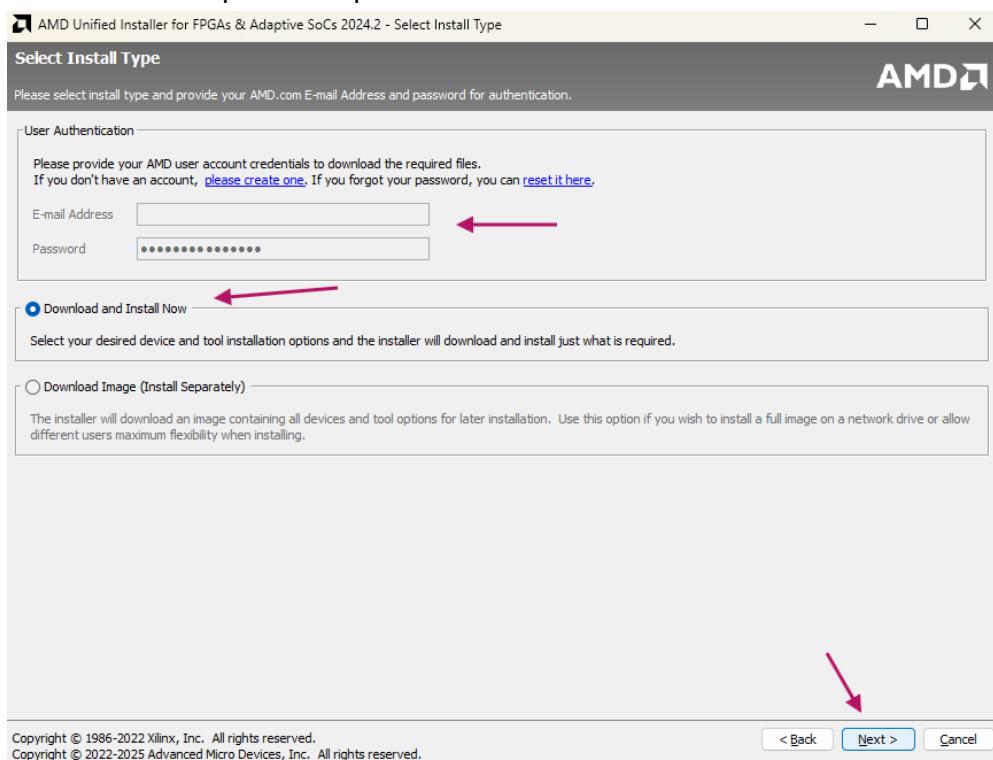
▼ Oggi



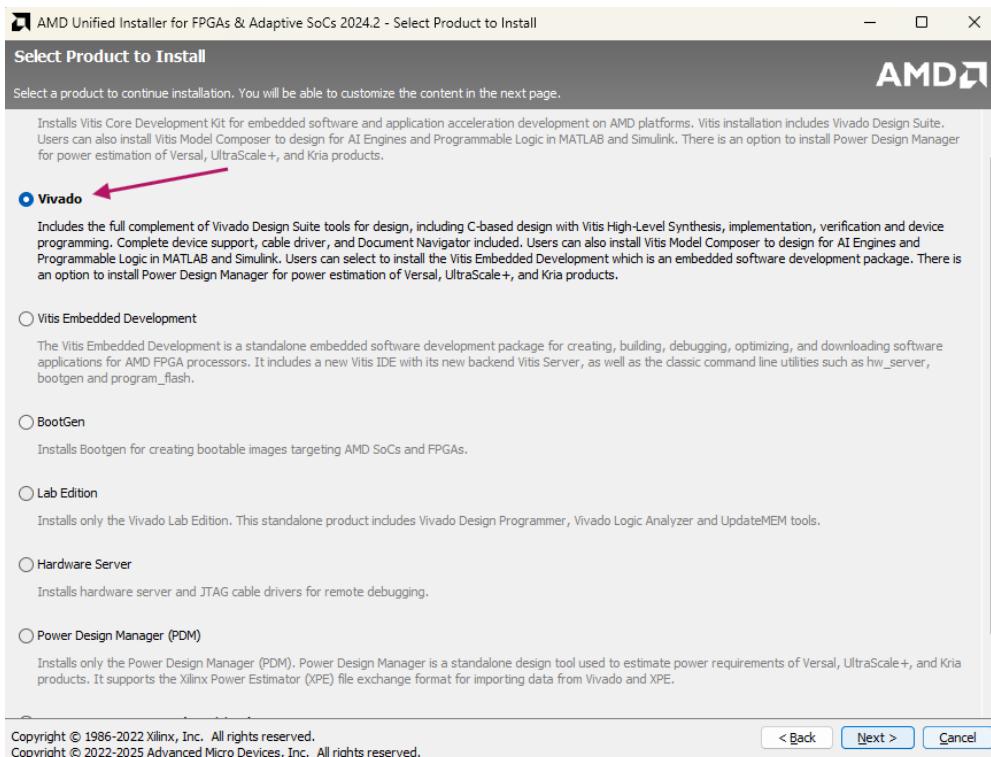
- Premere "NEXT":



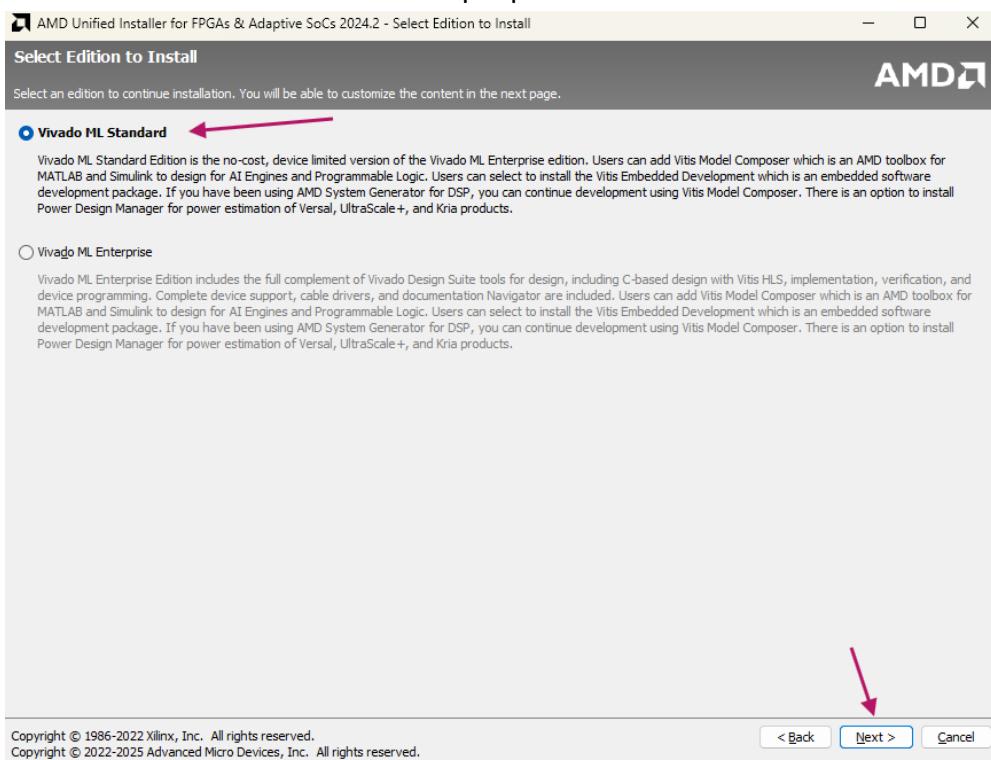
- Inserisci email e password premi "NEXT"



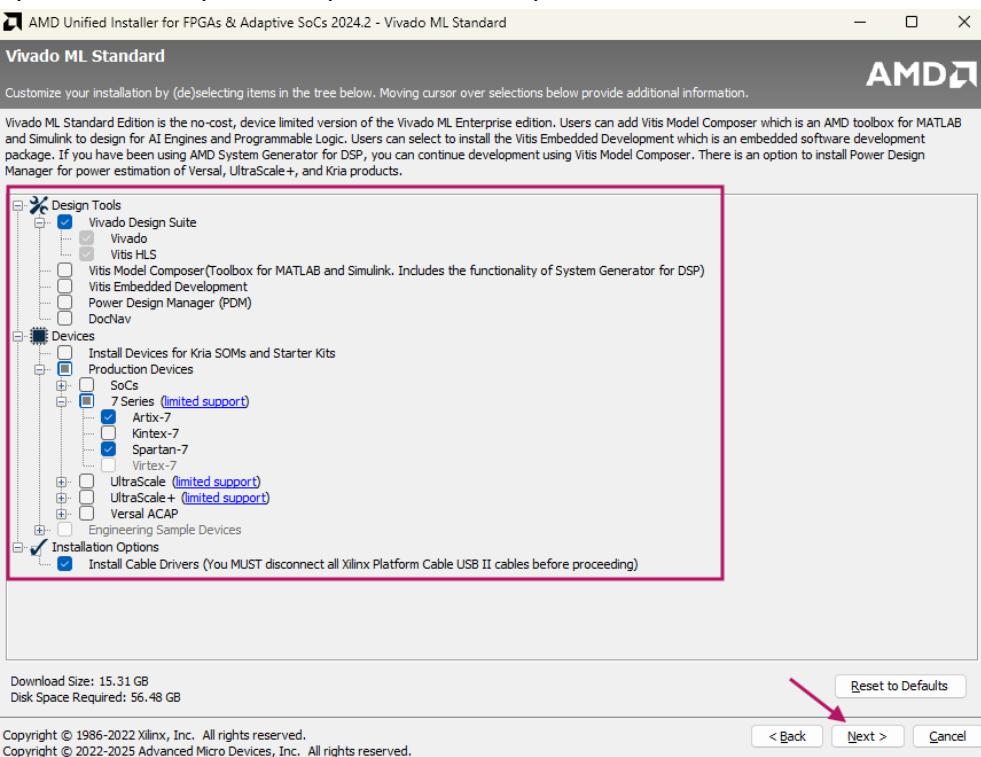
- Selezione “Vivado” e poi premi “Next”



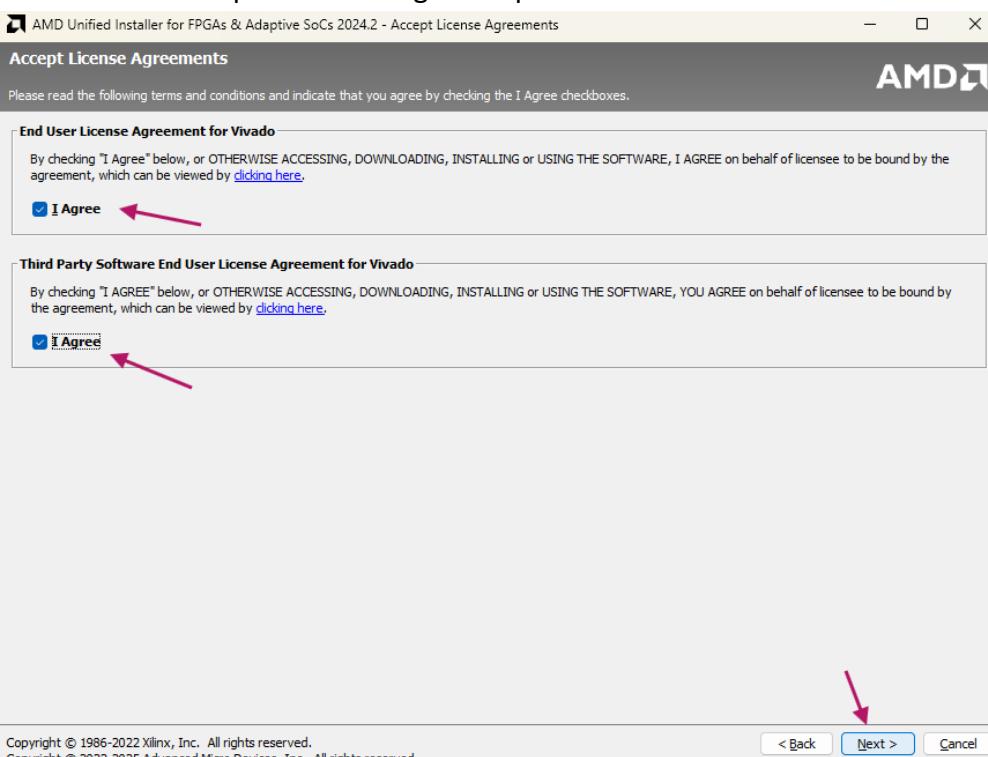
- Selezione “Vivado ML Standard” e poi premi “Next”:



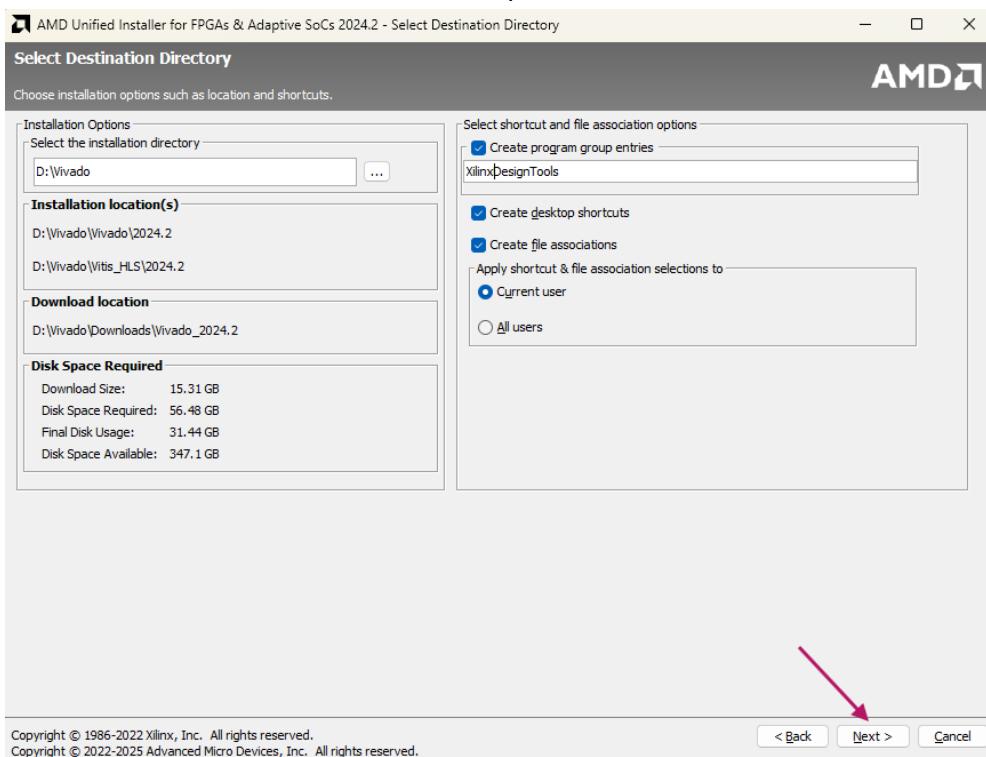
- Spuntare solo i pacchetti presenti nel riquadro rosso :



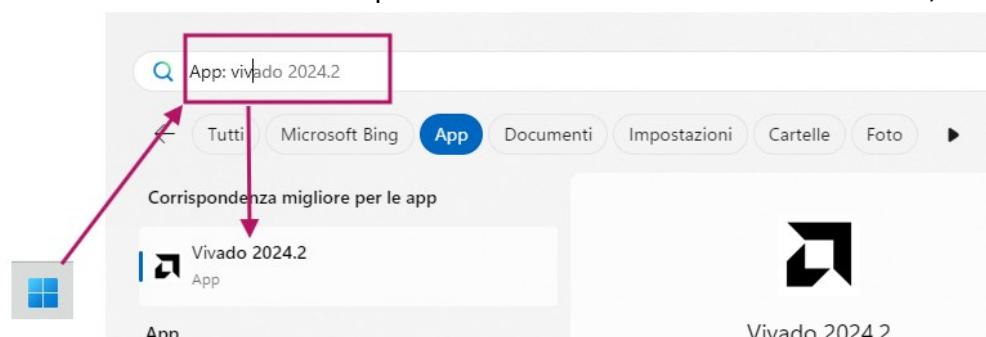
- Dare il consenso spuntando "I Agree" e premere "Next"



- Selezionare la catella di installazione e premere “Next”



- Al termine dell’installazione premere il tasto Windows e cercare Vivado, avviare “Vivavo 2024.2”:



- Ora Vivado è pronto:

