KITAB TEKNIK DIGITAL

Edisi 1

Sekte Tapak Kucing

Sektenya Para Legenda

Konten

1. Logic Gate, Aritmatika, Delay Propagasi
2. Flip-Flop, Finite State Machine, Delay Path
3. Case 1 : Forward Error Coding (Flip-Flop)
4. Case 2 : Q Function (Look up table)
5. Case 3 : Interleaver (Multiplexing, Scrambling).
6. Case 4 : Systolic (Finite State Machine)
7. Case 5 : Digital Filtering (Flip-flop timing)

Pendahuluan

Modul ini merupakan modul pendukung kuliah, seperti Sistem Digital, Digital Signal Processing atau bahkan bisa anda manfaatkan sebagai pendukung matakuliah System Embedded. Beberapa materi memang mengambil kasus dari permasalahan Elektronika Komunikasi, seperti Forward Error Coding, Interleaver, dan Filtering, namun modul ini sebenernya tidak menutup kemungkinan untuk pendalam materi seperti Arsitektur Komputer, karena dalam proses kerja dari sebuah komputer ketika kita hanya memprogam sebuah Microcontroller misalkan, kita serasa hanya membaca dan menulis dari alamat memori, tanpa tahu apa yang terjadi pada gerbang logika yang kita desain.

Xilinx merupakan salah satu perusahaan semikonduktor terbesar di dunia. Xilinx menawarkan salah satu produk elektronik yang sering kita sebut sebagai FPGA (Field Programmable Gate Array). Ada beberapa algoritma yang dipermudah dalam platform dari Xilinx. Misal perkalian dalam prototyping FPGA, bisa kita buat dengan algoritma Booth atau pergeseran bit, namun kita juga bisa menggunakan langsung perkalian dari Framework dari Xilinx yang bernama Vitis, sehingga untuk masalah perkalian kita tidak perlu melakukan optimisasi lagi dengan begitu kita akan lebih menghemat waktu dalam mendevelop sebuah project tanpa mengurangi kedalaman sebuah project yang kita kerjakan, dengan begitu kita tetap akan belajar lebih mendalam tanpa memikirkan sesuatu yang terlalu membuang waktu.

Dalam modul ini, kita akan menggunakan platform dari Xilinx untuk melakukan Praktikum yang bernama Vitis.[1] Untuk versi yang digunakan menggukan Vitis versi 2022.1, mungkin ada pertanyaan apakah bisa kita menggunakan versi yang lebih lama? Saya menggunakan versi ini karena pada tahun 2022 merupakan versi terbaru. Ada beberapa fitur yang tidak disediakan di versi sebelumnya, semisal : API untuk mengakses Xil_Out16, User interface yang berbeda di bagian SDK ketika kita mencoba menggunakan versi dibawah versi 2020. Untuk sekarang Vitis 2022.1 ini masih memiliki platform yang relevan dengan beberapa platform yang disediakan oleh Xilinx sehingga untuk development 7-Series hal ini cukup, namun jika anda menggunakan produk yang lebih lama dari hardware 7-Series mungkin ini tidak cocok untuk anda. Untuk installasi anda bisa menggunakan Appendix A untuk referensi.

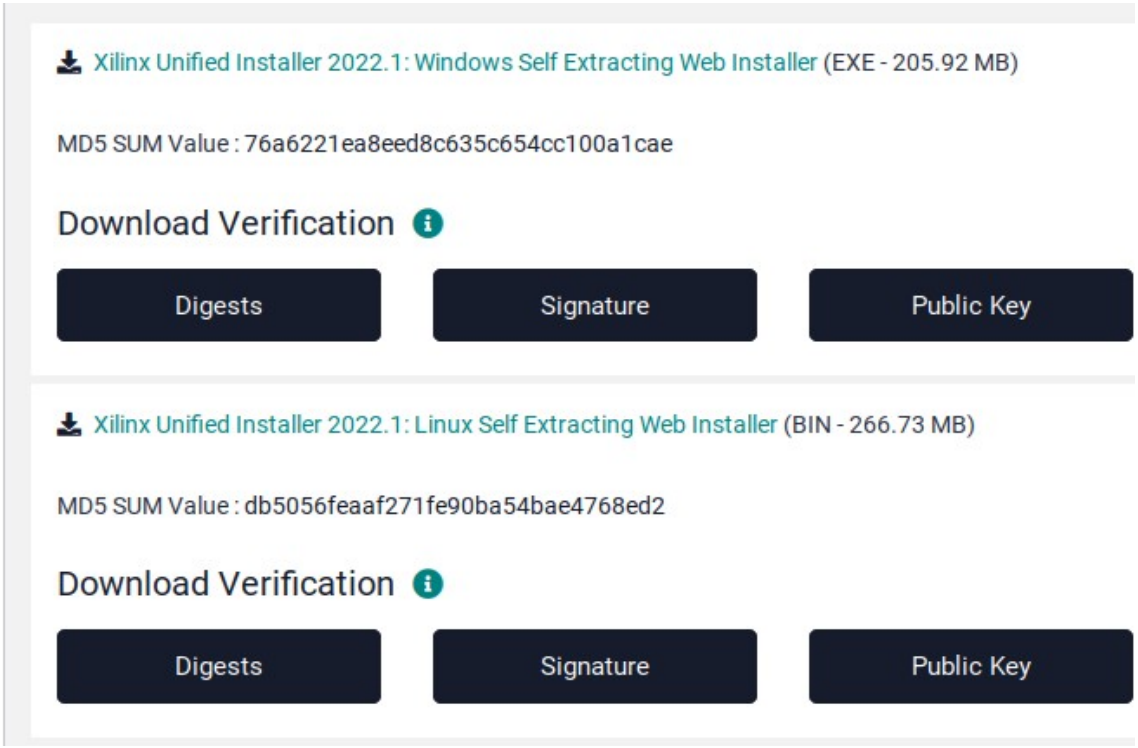
Modul ini membahas development melalui hardware seperti menggunakan verilog dan testbench verilog, high level syntesis, dan juga pengujian hardware melalui prosesor arm yang akan digunakan sebagai antarmuka antara prosesor penguji dan desain hardware (yang dibangun dengan FPGA) yang akan diuji. Pengguanaan HLS dibahas dalam modul ini karena selain untuk mempercepat development dari sebuah hadware yang didesain, pendekatan dengan HILS akan membantu juga memahami hardware tanpa bergantung pada verilog. Pengujian dengan processor arm yang tertanam pada chip akan membantu kita memahami bagaimana interface antara beberapa processor tanpa harus membuat sebuah rangkaian fisik (semisal jumper) sehingga bisa mengurangi kerja yang tidak terlalu dibutuhkan.

1 Untuk mendownload vitis anda bisa gunakan link berikut https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis.html

Gerbang Logika Aritmatika dan Delay Propagasi

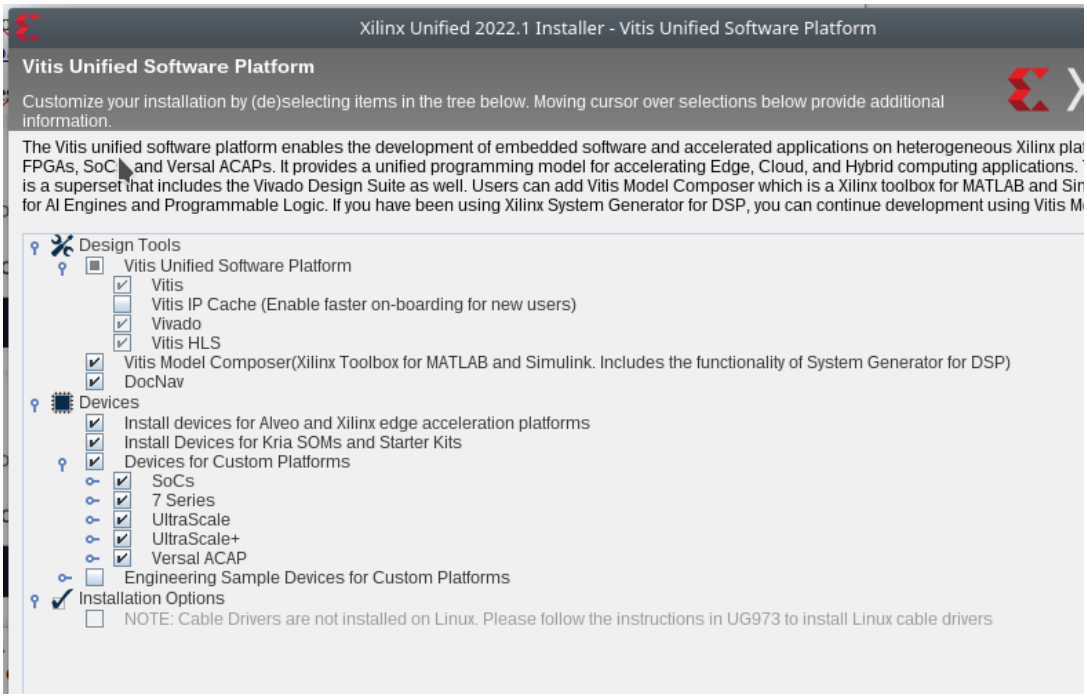
Appendix A : Instalasi Vitis

1. Donwload vitis di https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis.html
2. Pilih sesuai os anda



3. Setelah download selesai maka installasi dilakukan biasanya akan ada permintaan username berupa email dan password untuk memulai installasi
4. Pilih Vitis kemudian Next.

5. Kemudian pastikan Vitis, Vitis HLS, dan Vivado terinstall.

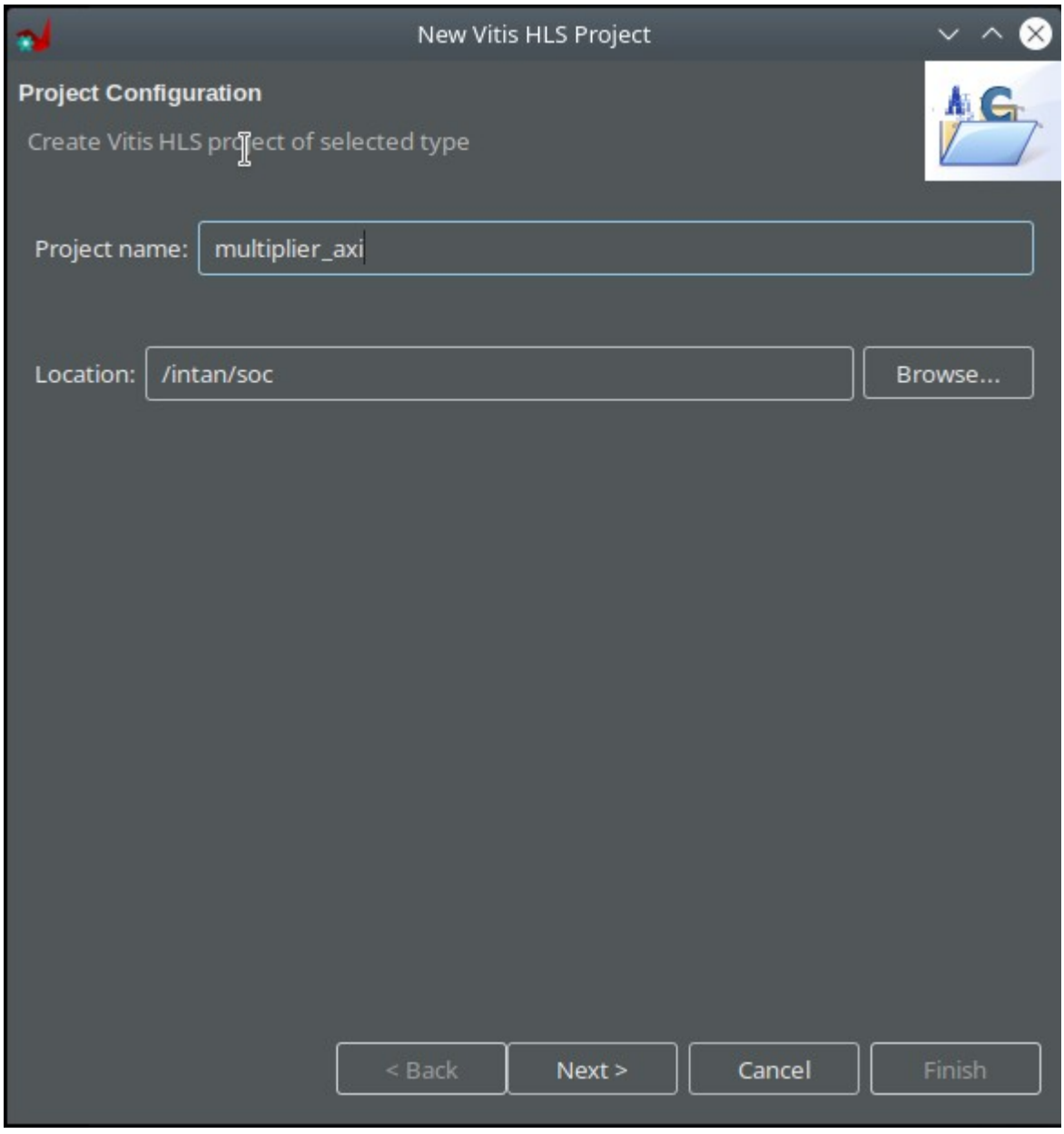


6. Setelah itu pilih direktory yang akan diinstall lalu tunggu sampai selesai.

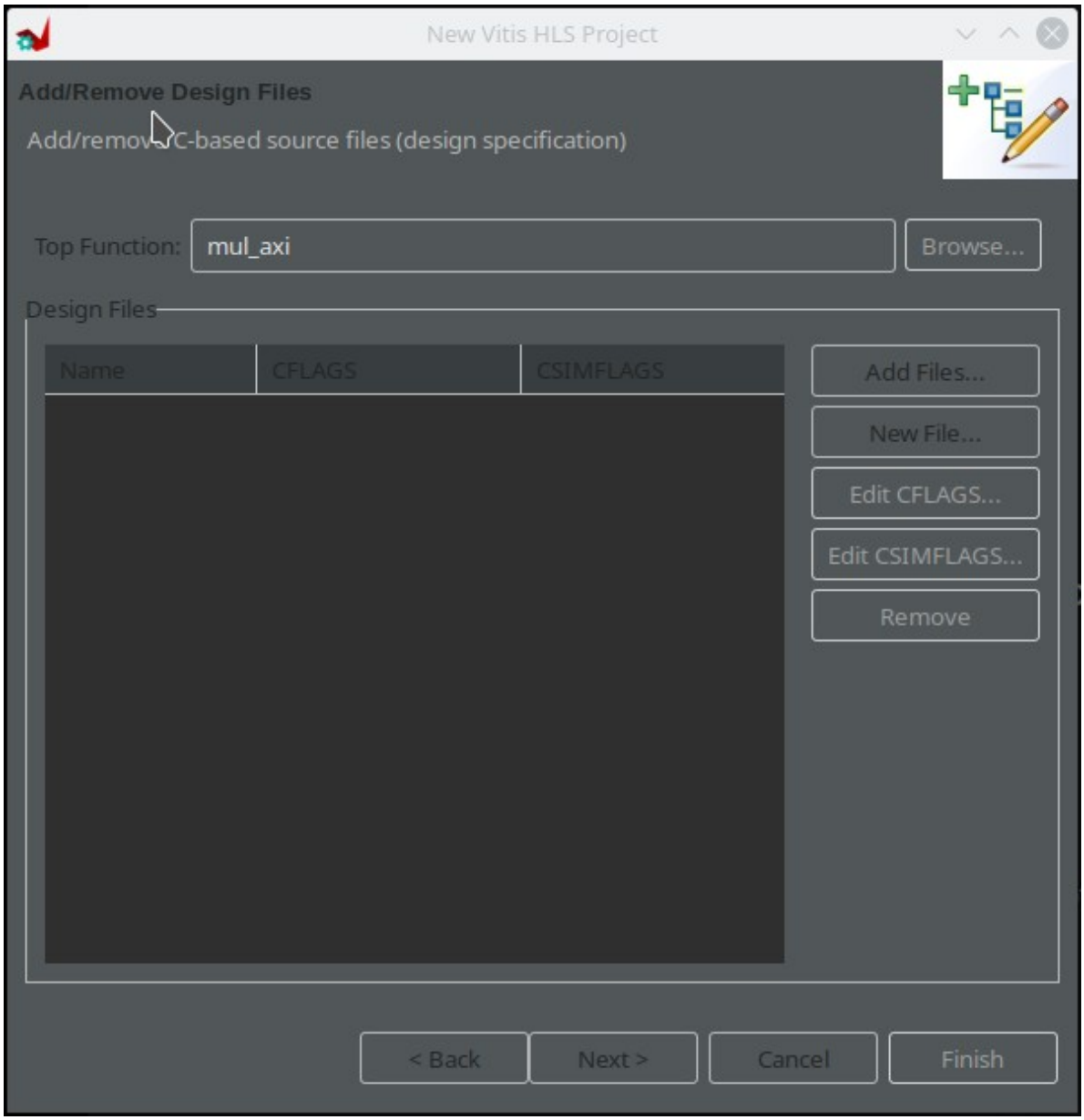
Appendix B : Vitis HLS project

High level syntesis biasanya digunakan untuk membuat sebuah development hardware menggunakan paradigma object oriented programming. Dengan menggukan hls anda bisa membuat configurasi hardware dengan C++. Untuk memulai Sintesa dengan Vitis HLS, hal yang diperlukan adalah membuka windows Vitis HLS anda.

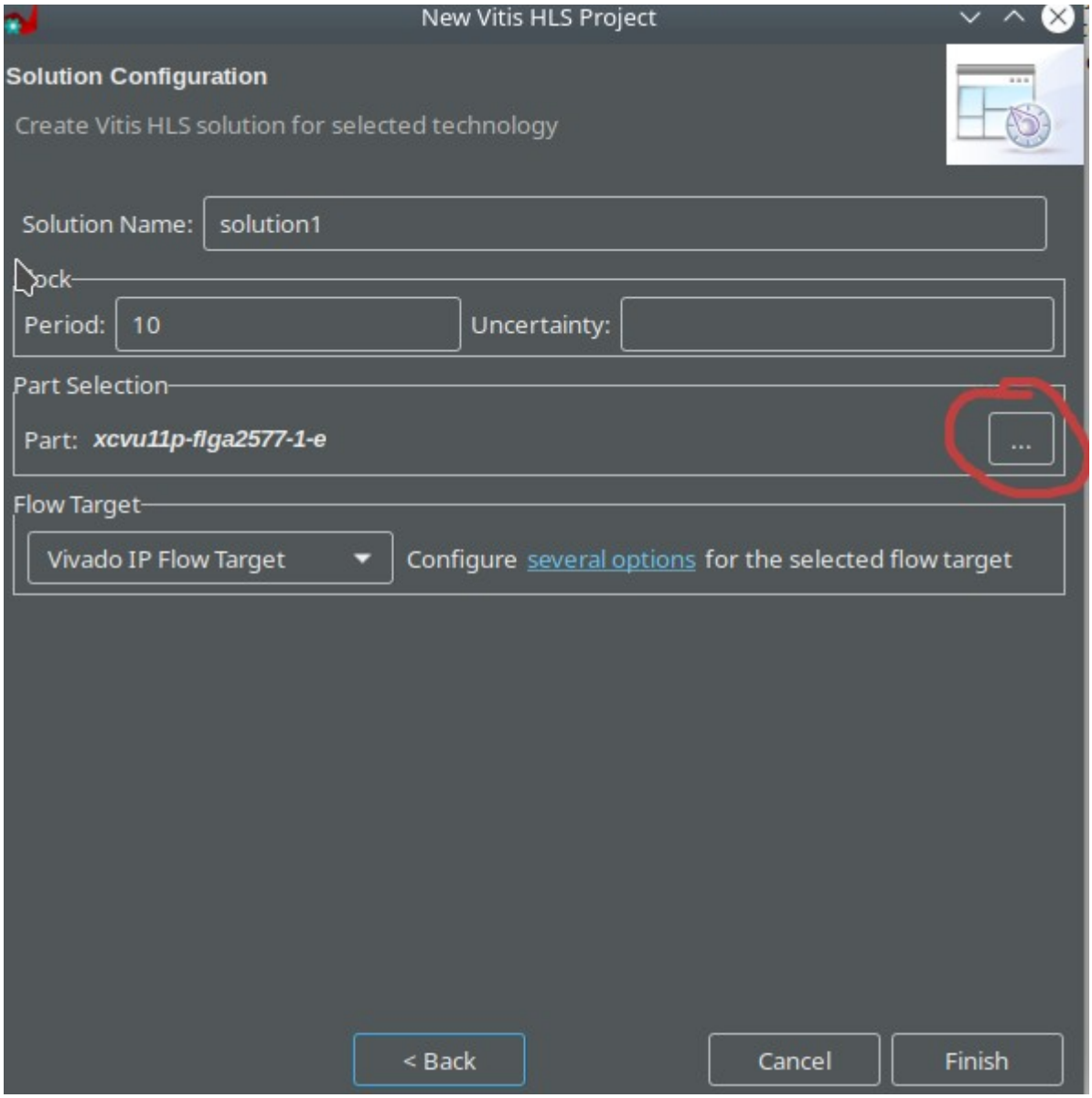
1. File → New Project



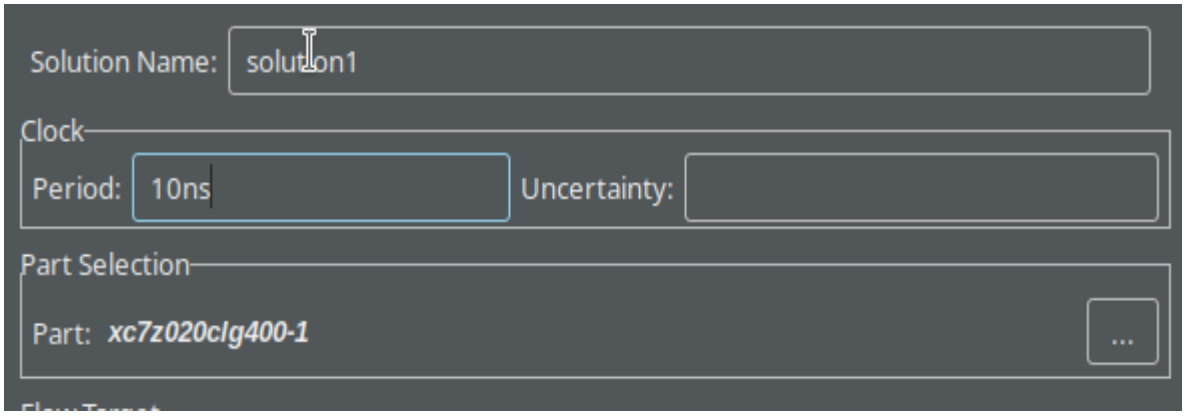
2. Berikan nama Project → Next



3. Masukkan nama Top Function yang akan saya gunakan bernama mul_axi, pada tab add/remove Design Files.
4. Kemudian skip pada Testbench Files dengan mengeklik tombol next.Untitled 1



5. Pilih Processor yang anda inginkan. Saya menggunakan xc7z020clg400-1, tambahkan periode dengan tulisan ns sehingga Period clock menjadi 10ns.



6. Periodenya kemudian click finish.

7. Click Project → New Source File → pilihlah direktory multiplier_axi lalu open → berikan nama mul_axi.cpp, nama file dari Cpp yang sama dengan top function akan menjadi code utama pada Vitis HLS.

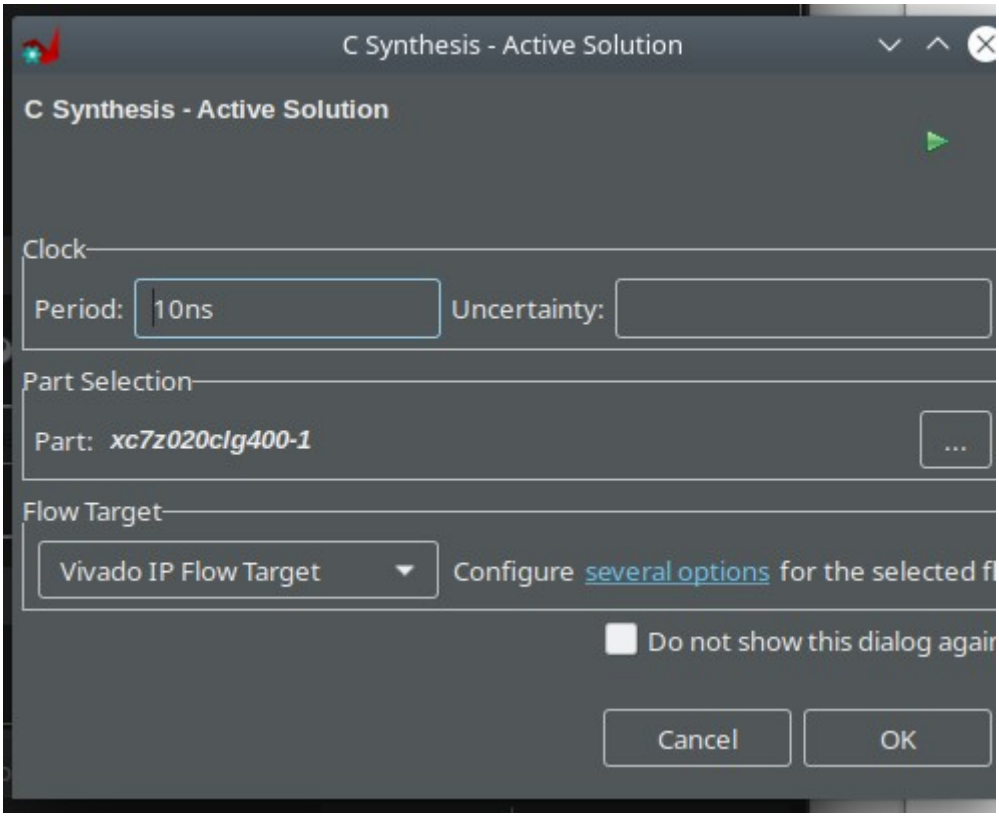
8. Buatlah isikan code CPP tersebut dengan code berikut.

```
#include <stdio.h>
#include <stdint.h>
#include <math.h>
typedef int64_t dout4_t;

void mul_axi(char *a, char *b, dout4_t *c)
{
#pragma HLS INTERFACE s_axilite port=a bundle=BUS_A
#pragma HLS INTERFACE s_axilite port=b bundle=BUS_A
#pragma HLS INTERFACE s_axilite port=c bundle=BUS_A
#pragma HLS INTERFACE s_axilite port=return bundle=BUS_A

  *c += *a * *b;
}
```

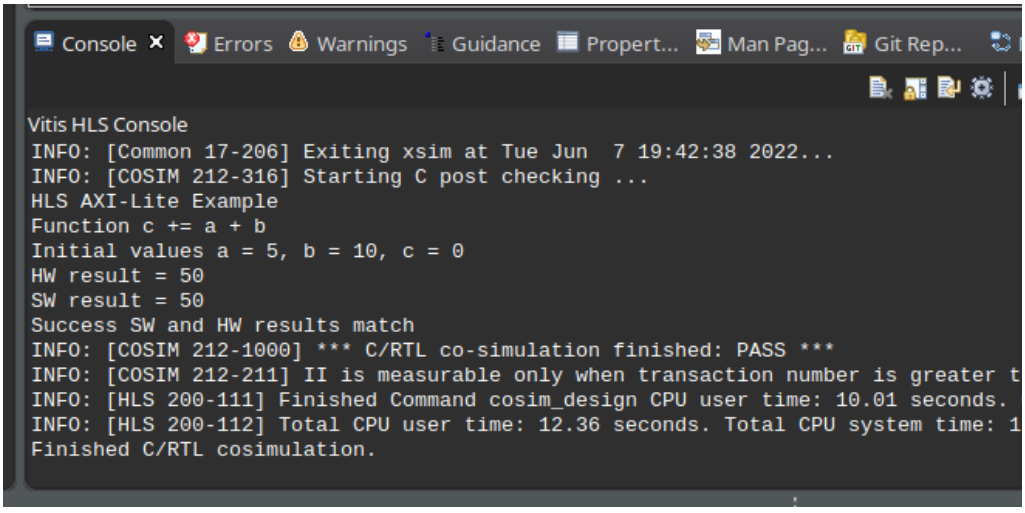
9. Untuk testbenchnya anda bisa menambahkannya dengan mengeklik Project → New Test Bench File. Kemudian masuk ke direktori multiplier_axi, lalu beri nama file mul_axi_tb.cpp, isinya adalah sebagai berikut.

```
#include <stdio.h>
#include <stdint.h>
#include <math.h>

void mul_axi(char *a, char *b, int64_t *c);

int main()
{

  char a;
  char b;
  int64_t c;
  char d;
  char sw_result;
  printf("HLS AXI-Lite Example\n");
  printf("Function c += a + b\n");
  printf("Initial values a = 5, b = 10, c = 0\n");

  a = 5;
  b = 10;
  c = 0;
  d = 0;

  mul_axi(&a,&b,&c);
  d += a * b;

  printf("HW result = %d\n",c);
  printf("SW result = %d\n",d);

  if(d == c){
    printf("Success SW and HW results match\n");
    return 0;
  }
  else{
    printf("ERROR SW and HW results mismatch\n");
    return 1;
  }
}
```

10. Kemudian build dengan klick berikut Solution → Run C Synthesis → Active Solution.
11. Setting periode seperti berikut, berikut settingan hardware yang saya gunakan menggunakan xc7z200clg400-1 dengan periode 10ns. Lalu Click OK.

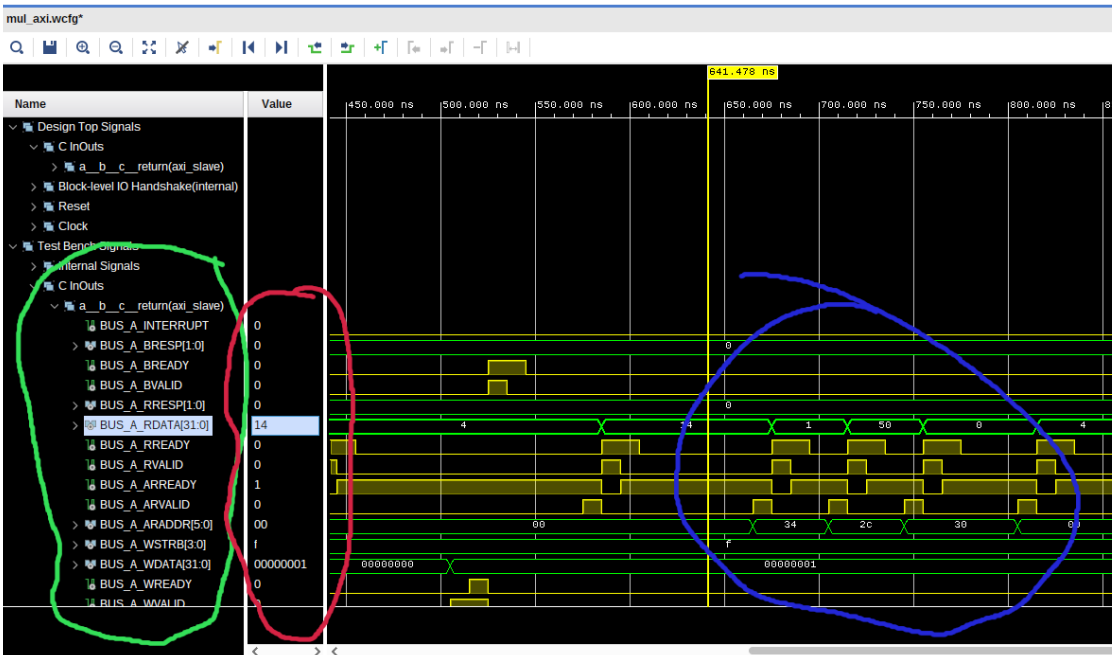


12. Setelah itu simulasikan dengan Click berikut Solution → Run C/RTL Cosimulation → Rubah Dump Trace dialog ke mode All → OK
13. Maka di Console akan muncul sebagai berikut.



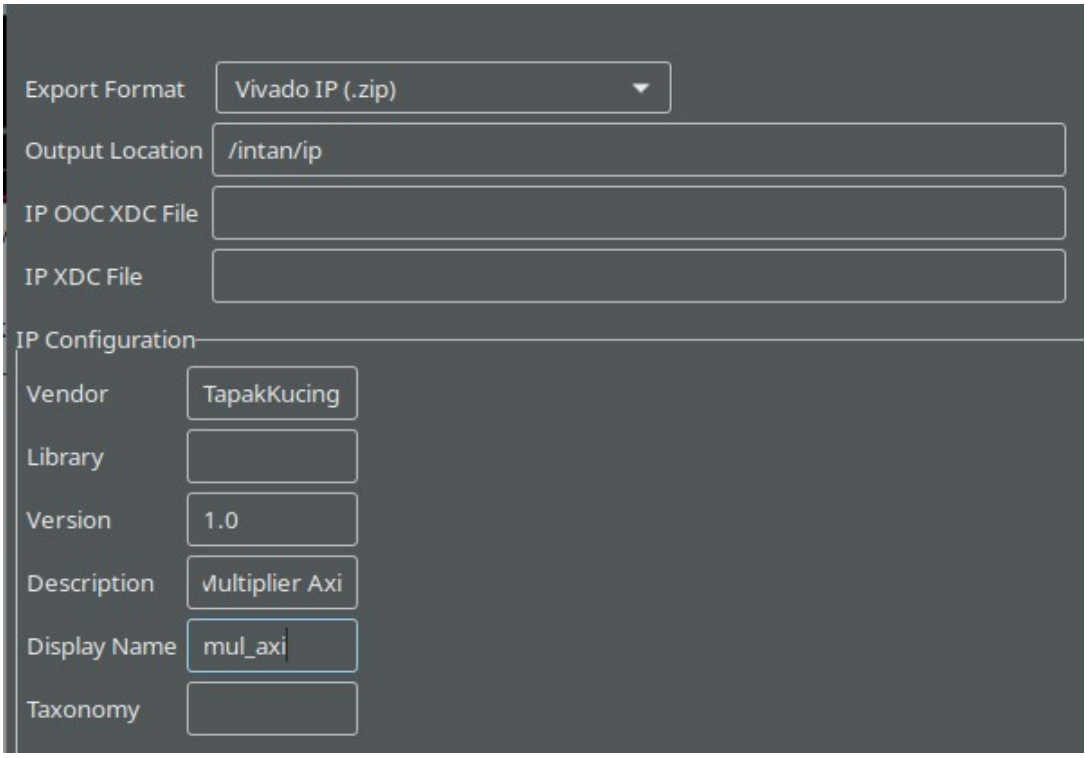
14. Dari hasil diatas kita mengetahui bahwa hasil dari software dan hardware sama. Terkadang kasus error muncul ketika terjadi overflow semisal kita mendeklarasikan perkalian 8 bit dengan 8 bit yang menhasilkan lebih dari 8 bit nilai output, untuk itu perhatikan bahwa nilai output dari sistem yang kita desain sudah mencukupi untuk menampung nilai pada jangkauan tertentu.

15. Untuk memberikan gambaran tentang timing diagram kita bisa menggunakan tools simulation dari vivado yang terpasang di Vitis HLS. Gunakan prosedur berikut untuk menampilkan waveform dari hardware dan testbench yang kita desain. Click Solution → Open Wave Viewer . . . lalu akan muncul seperti berikut.



16. Gambar diatas adalah waveform dari simulasi yang dilakukan melalui Vitis High Level Syntesis.

17. Kemudian kita coba export RTL code tadi kedalam bentuk IP dengan prosedur berikut. Solution→ Export RTL. Lalu muncul dialog seperti dibawah ini, lakukan masukkan nilai nilai yang diperlukan hingga kemudian export ke RTL.



18. Kemudian Click OK

19. Buka Direktory dengan file manager lalu extract export.zip

Appendix C : Vivado IP Export

Untuk memprogram kedalam hardware kita bisa menggunakan langsung dengan vivado namun kita bisa juga mengkombinasikannya dengan menggunakan Vitis IDE, Vitis IDE berbeda dengan Vitis HLS, Vitis HLS merupakan IDE untuk mendesain hardware yang nanti akan dikonversi ke dalam bentuk RTL, sedangkan Vitis IDE merupakan alat yang digunakan untuk memprogram kode C ke dalam hardware. Untuk memulainya kita buka Vivado,

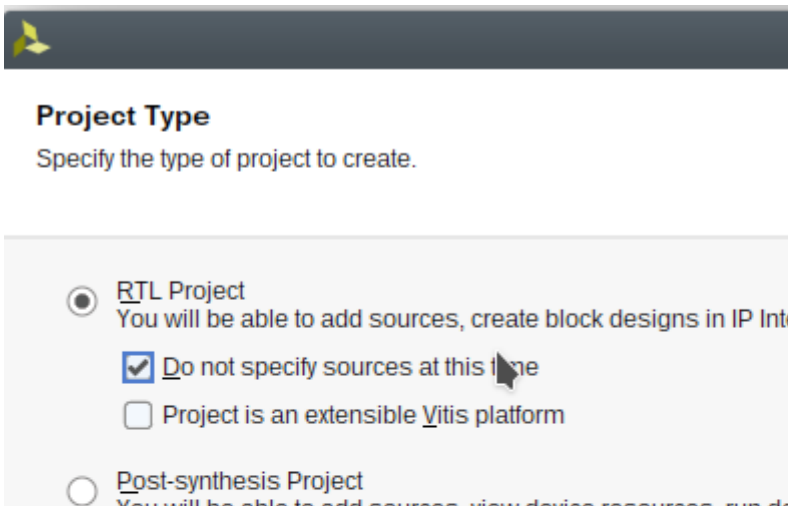
Start → Vivado,

kemudian create project File → Project → New,

pada Window Create Vivado Project klick Next.

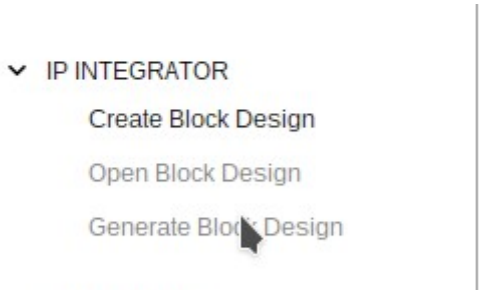
Masukkan Nama Project misal mul_axi_hw dan setting direktori path nya.

Kemudian pilih RTLProject

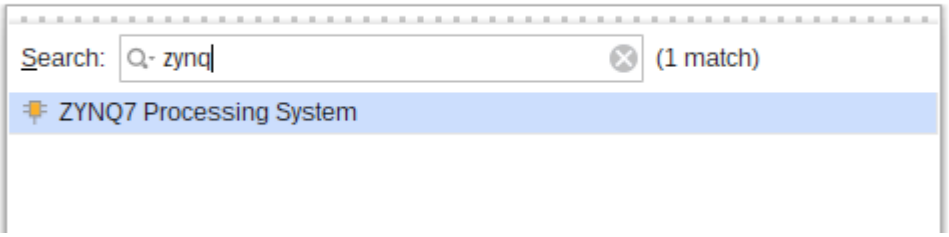


Kemudian pilih Next. Kemudian cari part xc7z020clg400-1 kemudian klick next. Lalu Finish.

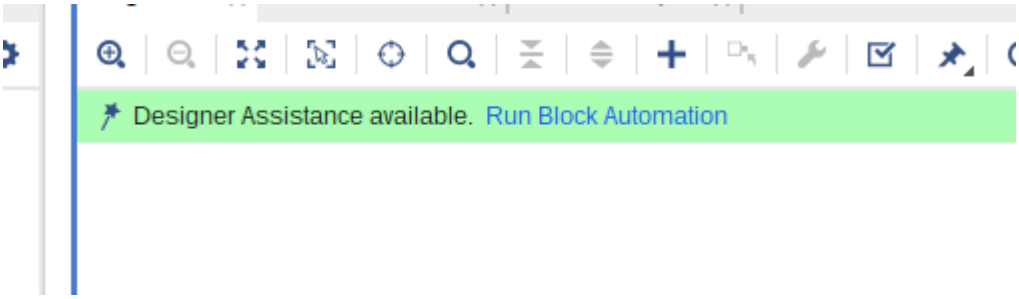
Pada Project Manager bagian IP integrator pilih Create Block Design



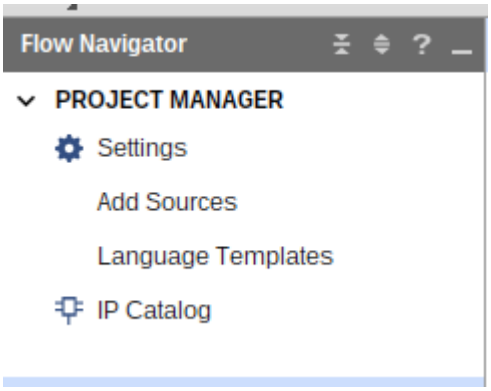
Kemudian tambahkan IP dengan menekan tombol Ctrl+i lalu pilih zynq



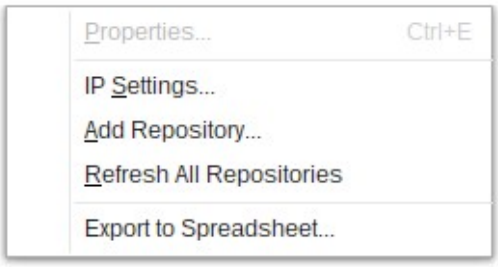
Kemudian click run block automation.



Setelah itu tambahkan repository yang telah kita buat seperti pada Appendix B dengan menggunakan cara berikut, pada Flow Navigator → Project Manager → Klik IP Catalog.

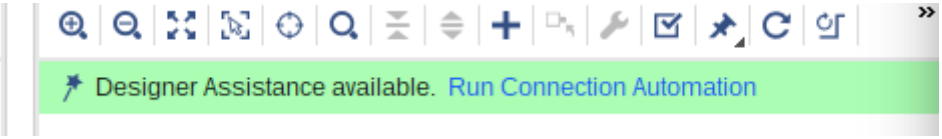


Kemudian Kembali ke bagian IP Catalog dan klik kanan di area tersebut lalu tambahkan IP

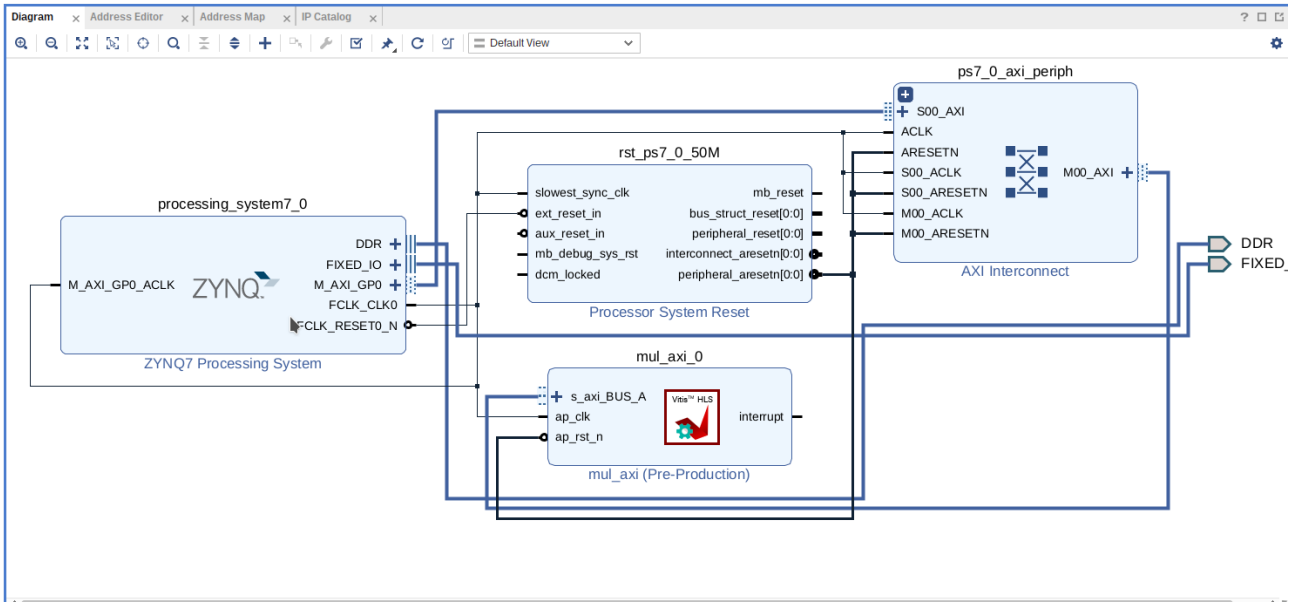


Lalu tambahkan Repository dengan pilih Add Repository. Pilih direktory yang diextract export.zip dari Appendix B.

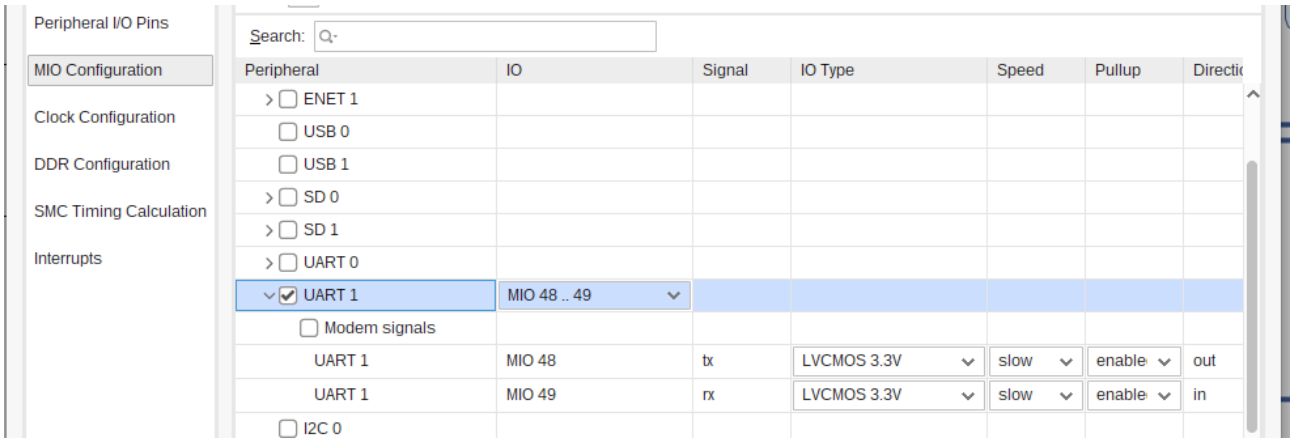
Sekarang IP sudah tersedia di Vivado. Kemudian kembali ke diagram terus add IP (Ctrl+i), Kemudian run Connection Automation.



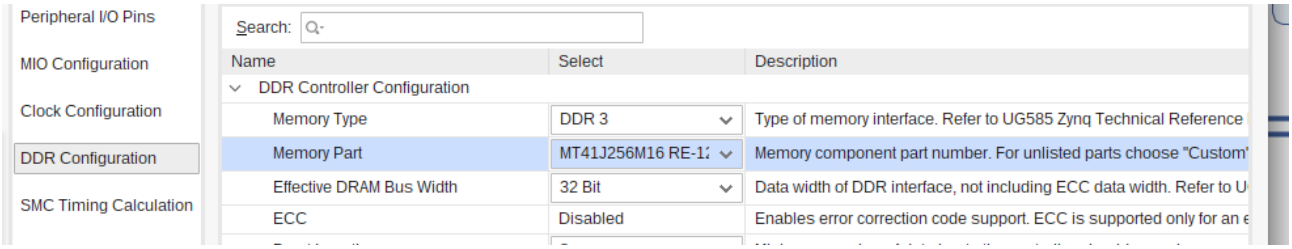
Setelah di klik Run Connection Automation maka akan muncul kotak dialog lalu klik OK (biarkan saja secara deafult merka saling menghubungkan). Maka akan muncul seperti gambar berikut pada gambar diagram.



Klik 2 kali pada bagian block Zynq Processing System, lalu akan muncul configurasi dialog untuk setting Zynq Ada dua bagian yang perlu diset yaitu RAM dan Uart1 sebagai console log.



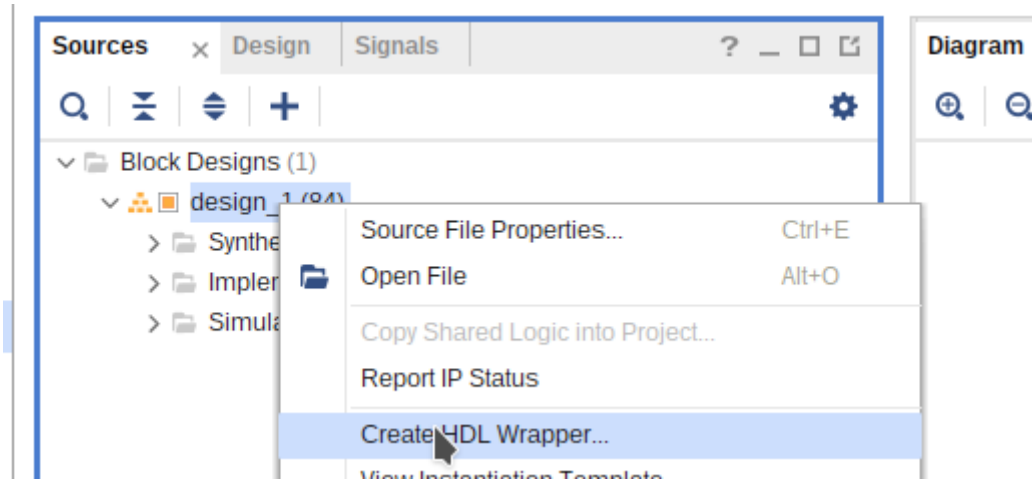
Pada bagian MIO configuration set UART 1 dengan memberikan dia tanda check.



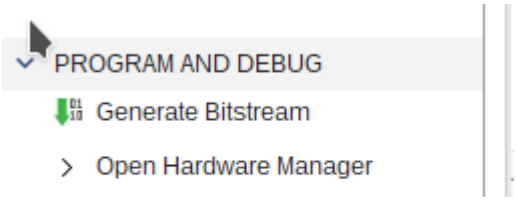
Pada bagian DDR Configuration → DDR Controller Configuration → Memory Part set kedalam nilai MT41J256M16 RE-125.

Kemudian Generate Block Design dengan mengeklick bagian Flow Navigator → IP Integrator → Generate Block Design, lalu akan muncul dialog kemudian klick Generate.

Setelah itu generate buka bagian tab source, klik kanan pada bagian design terus klik Create HDL Wrapper . . .



Lalu akan muncul dialog lalu let Vivado manage wrapper and auto update. Setelah jadi maka akan muncul code verilog pada bagian design, setting top pada code wrapper tersebut (karena hanya satu verilog code maka tidak perlu diset top biasanya).



Setelah itu click Generate Bitstream pada Flow Navigator → Program and Program Debug. Nanti muncul kotak dialog untuk melakukan Synthesis dan Implementation, untuk kasus ini anda silahkan tinggal klik ok. Namun ketika ada dialog open Implementation Design anda tinggal klik cancel saja, karena kita akan langsung mengeksport bitstream tersebut untuk bisa diprogram dengan Vitis IDE.

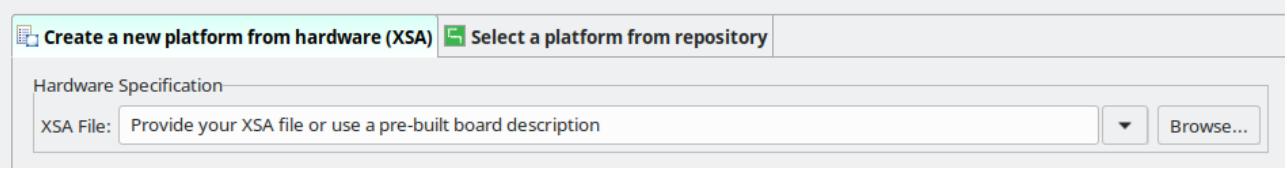
Untuk Export ke Bitstream ke Vitis IDE gunakan cara berikut :

Klik File → Export → Export Hardware kemudian akan muncul dialog yang berjudul Export Hardware Platform → Klik Next, pilih Include Bitstream → Next, selanjutnya kita akan memilih direktori output untuk xsa file kemudian klik next, kemudian Finish.

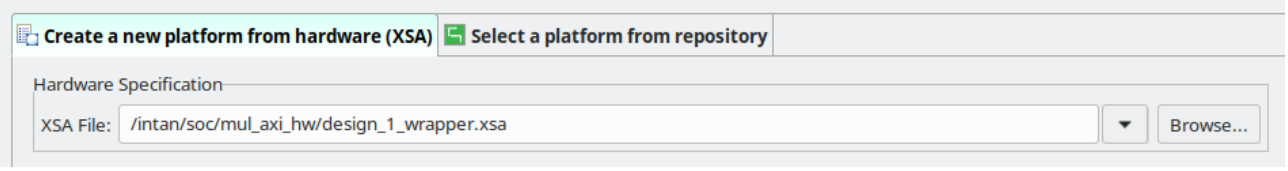
Appendix D : Vitis IDE Test IP

Biasanya Vitis IDE dijadikan alat untuk membuat program yang akan diletekakan untuk processor arm namun kita juga bisa mengganakannya sebagai progam untuk mengetes hardware dari FPGA yang kita buat. Vitis IDE berbeda dengan HLS hal ini perlu diperhatikan.

Start Menu → Xilinx Vitis 2021.1 → set Workspace lalu akan muncul workspace untuk bekerja anda. Pertama kita harus membuat platform hardware dengan vitis, File → New → Platform Project, kemudian berinama platform project anda, saya memberi nama mul_axi_plat kemudian pilih next.

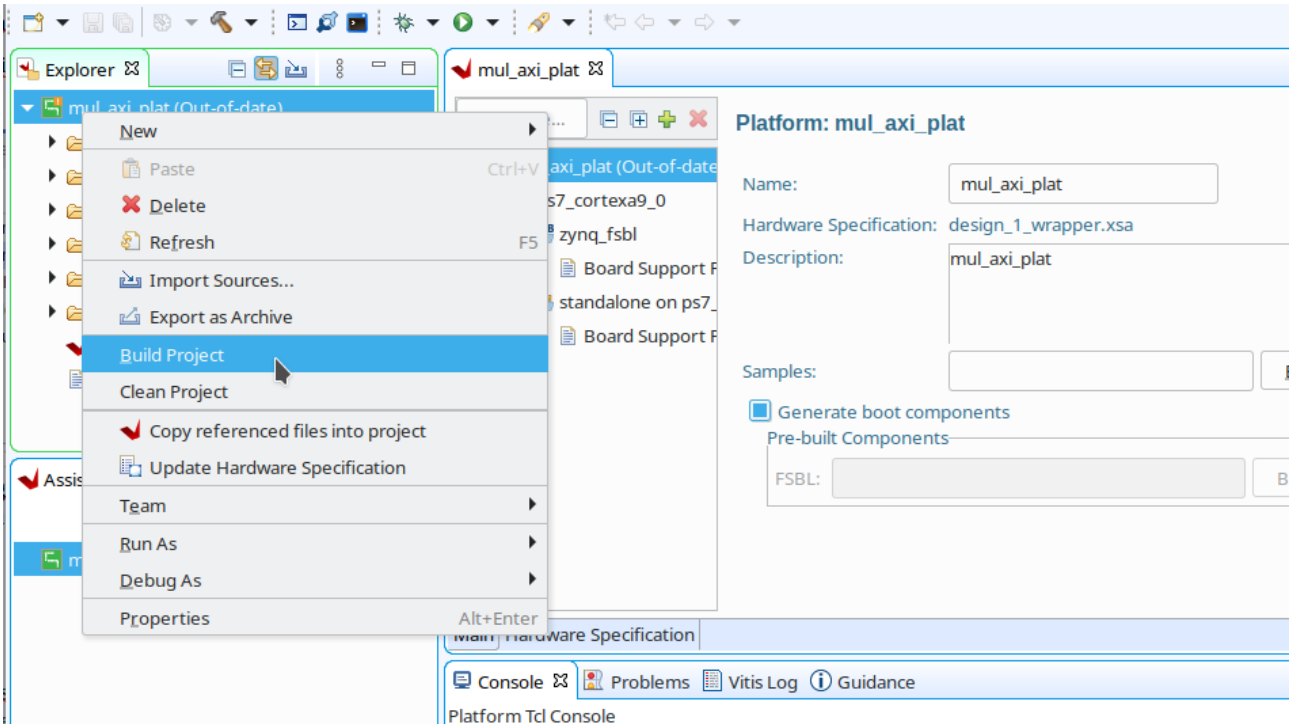


Kemudian pada bagian Hardware Specification pilih file XSA yang telah dibuat dalam Appendix C dengan mengeklik tombol Browse . . . ,



Setalah itu pilih tombol Finish.

Kemudian Build platform yang sudah terbuat

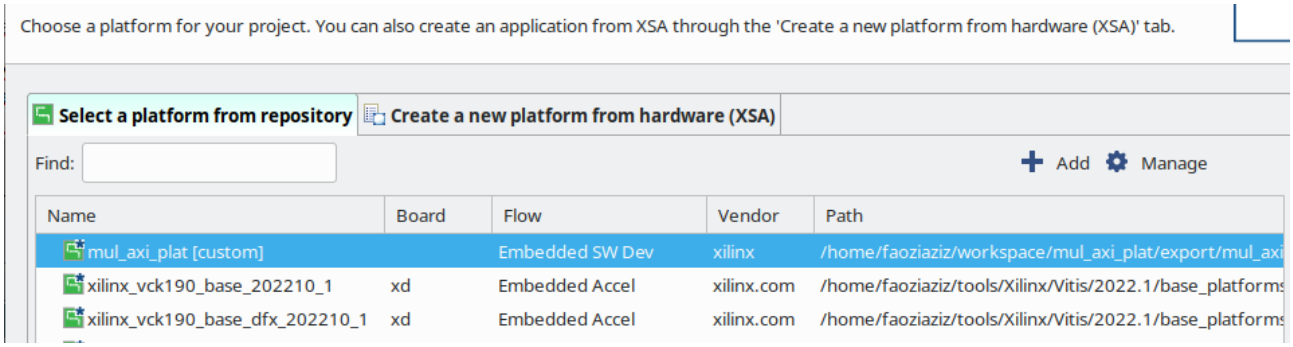


Klik kanan pada mul_axi_plat lalu klik kiri pada build project.

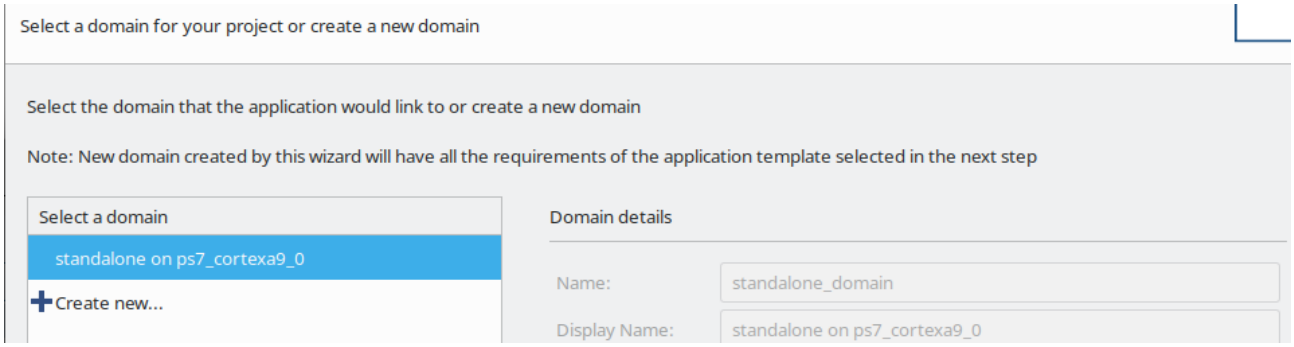
Setelah menunggu beberapa saat maka dan build berhasil, selanjutnya anda perlu membuat project aplikasi yang akan digunakan untuk mengetest FPGA yang telah anda desain sebelumnya.

Untuk membuat aplikasi project, klik File → New → Application Project

Klick Next pada Dialog Pertama → pilih mul_axi_plat → Next

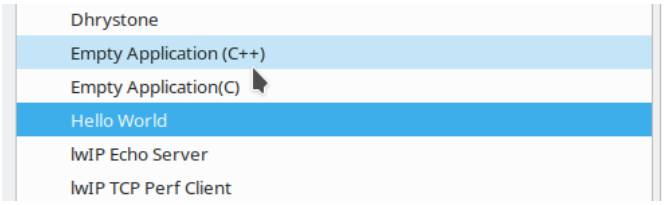


Masukkan nama application project misal mul_axi_app → Next → Pilih Standalone → Next



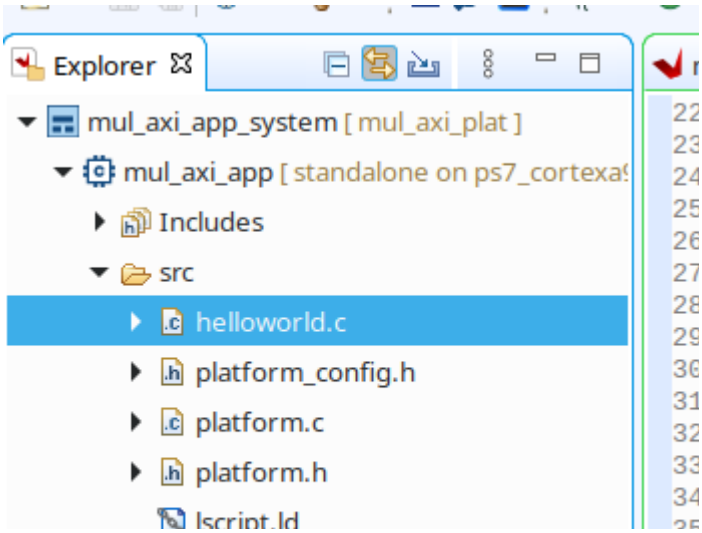
Secara default biasanya anda tinggal klik next pada dialog ini.

Kemudian pilih Hello World Templates

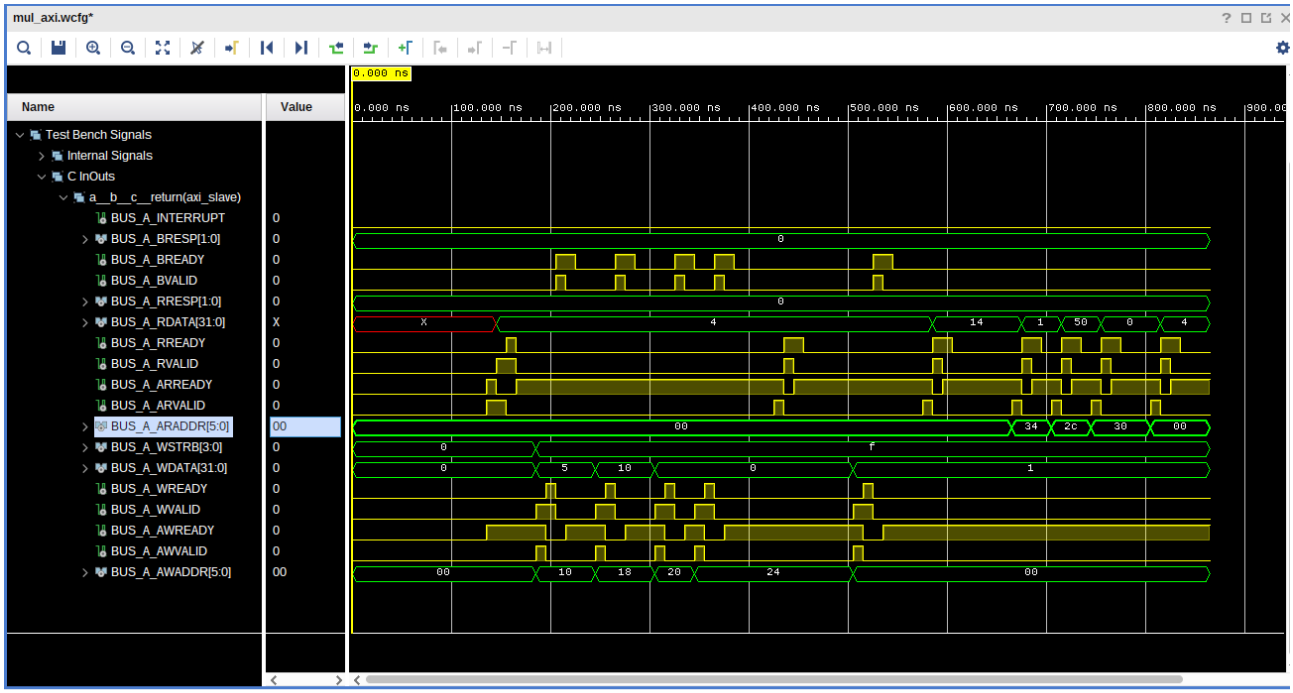


Pilih Hello World → Finish.

Kemudian buka File helloworld.c lalu edit filenya.



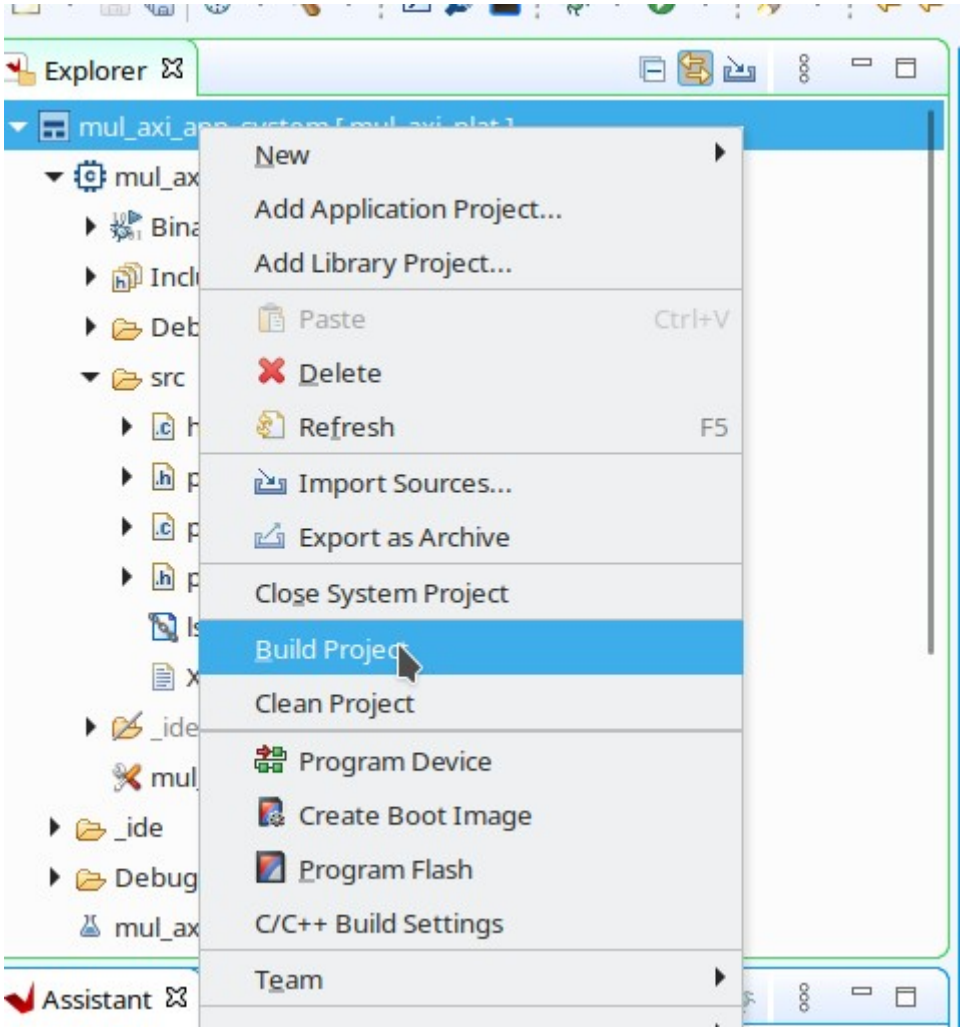
Anda bisa mengedit file di Explorer tab kemudian cari src → helloworld.c kemudian klick 2 kali untuk membuka file.



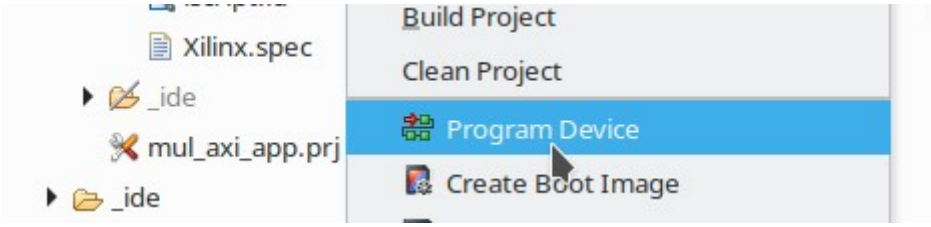
Sebelum kita melakukan coding dengan Vitis IDE untuk mengetest hardware yang kita desain dengan FPGA kita bisa menggunakan waveform dari hasil test Vitis HLS untuk menentukan alamat register yang akan kita berikan perintah untuk mendapatkan alamat pengisian AXI. Dari gambar terlihat bahwa alamat untuk inputan A beralamatkan Base address + 0x10 dan alamat B beralamat Baseaddress + 0x18, dan untuk exekusi program kita hanya perlu memberikan nilai 1 pada baseaddress + 0x0. Kita bisa membaca hasil perkalian di alamat C yaitu di baseaddress + 0x2c. Untuk kode implementasinya adalah sebagai berikut

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "xil_printf.h"
#include "xil_io.h"
#define MULTI_BASE 0x40000000
int main(){
	/* menulis input a */
	Xil_Out8(MULTI_BASE+0x10, 5);
	/* menulis input b*/
	Xil_Out8(MULTI_BASE+0x18, 10);
	/* execute the program on processor */
	Xil_Out8(MULTI_BASE+0x24, 0);
	Xil_Out8(MULTI_BASE, 1);
	/*read from a and b */
	xil_printf("a and b and 34 = %d and %d and %d,",
		Xil_In8(MULTI_BASE+0x10),
		Xil_In8(MULTI_BASE+0x18),
		Xil_In8(MULTI_BASE+0x34));

	/* read output c */
	xil_printf("Hasil Perkalian: %d,\r\n", Xil_In16(MULTI_BASE+0x2C));
	return 0;
}
```

Dari kode diatas kita akan menggunakan address 0x24 juga karena dari waveform sepertinya itu juga bagian interupt. Kemudian build program dan launch as hardware, pastikan device telah terhubung dan driver telah terinstall.



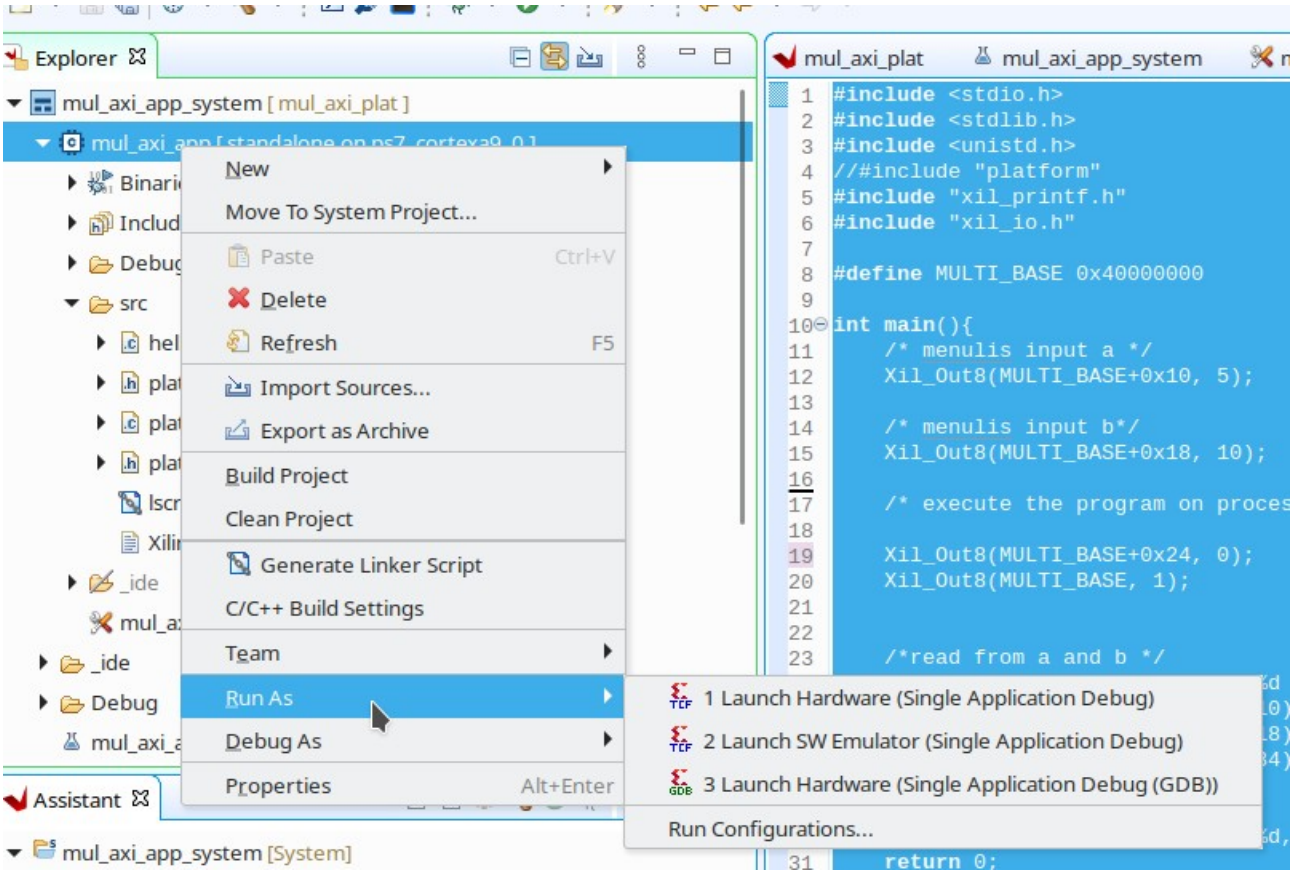
Pertama build project, kemudian program Device.



Setelah build project selesai kemudian program Device, Device akan terprogram tapi kita perlu launch program dengan processor untuk melakukan simulasi. Kita bisa menggunakan putty atau minicom untuk bisa mengakses port uart1 dari board Zynq yang kita miliki. Karena saya menggunakan minicom saya menggunakan command berikut

minicom -D */dev/ttyUSB1 -b 115200*

Setelah itu launch hardware seperti gambar berikut.



Setelah dilaunch perhatikan uart1 port yang diamati dengan minicom, maka hasilnya akan seperti berikut

```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyUSB1, 12:50:07

Press CTRL-A Z for help on special keys

a and b and 34 = 5 and 10 and 1,Hasil Perkalian: 50,
```