TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH KHOA ĐIỆN ĐIỆN TỬ

**BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG**

\*\*\*\*\*\*\*\*\*\*\*\*



**Môn học: Thiết kế kết hợp HW/SW**

BÁO CÁO CUỐI KÌ

**Giáo viên hướng dẫn: TS. Phan Văn Ca**

|  |  |  |
| --- | --- | --- |
| **Nhóm sinh viên thực hiện: Nguyễn Thanh Bình** | **-** | **19119155** |
| **Nguyễn Quốc Vinh** | **-** | **19119235** |
| **Huỳnh Chí Cường** | **-** | **19119157** |

TP HỒ CHÍ MINH – 6/2022

# CHƯƠNG 1: CƠ SỠ LÝ THUYẾT

* 1. **GIỚI THIỆU VỀ ZYNQ**

Ngày nay, sự phát triển của công nghệ cho phép các công ty tiên tiến đóng gói các bộ phận khác nhau của hệ thống thành các thiết bị ngày càng nhỏ hơn, tất cả chỉ còn một con chip, bổ sung khái niệm System-on-Chip (SoC) cho thế giới điện tử. Trước khi phát minh ra Zynq, các bộ xử lý đã được ghép nối với FPGA khiến cho việc giao tiếp giữa Logic lập trình (PL) và Hệ thống xử lý (PS) trở nên phức tạp.

Ngày nay, Kiến trúc Zynq, là thế hệ mới nhất của các họ SoC có thể lập trình toàn bộ Xilix, kết hợp ARM Cortex-A9 lõi kép với truyền thống (FPGA). Giao diện giữa các yếu tố khác nhau trong kiến trúc Zynq dựa trên tiêu chuẩn Giao diện eXtensible (AXI) nâng cao, cung cấp các kết nối băng thông cao và độ trễ thấp.

# Dòng ZynQ-7000 SoC

ZynQ-7000 là SoC kết hợp kiến trúc FPGA với kiến trúc ARM cotex-A9 lõi kép tốc độ xử lý cao với các thành công vượt trội trong các ứng dụng thực tế, sản phẩm nghiên cứu và thương mại với giá thành tốt nhất cho hiệu năng tốt nhất, nó là lựa chọn hoàn hảo cho các ứng dụng nhúng vô cùng rộng rãi như xử lý video 4K2K Ultra-HDTV, y tế, giám sát, năng lượng, công nghiệp, không gian vũ trụ…



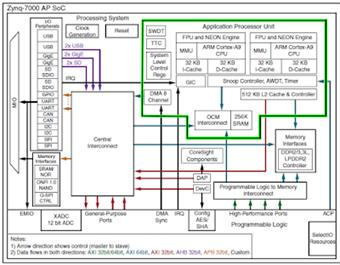
Hình 1.1. Zedboard ZynQ.

Họ vi mạch Zynq-7000 là hệ thống SoC có khả năng lập trình toàn diện (cả phần cứng và phần mềm) đầu tiên trên thế giới.

Trong mỗi một chip ZynQ-7000 đều đã tích hơp sẵn một hệ vi xử lý PS (Processing System) trên cơ sở bộ vi xử lý Cortex™-A9 lõi kép của hãng ARM và một kiến trúc logic khả trình PL (Programmable logic) sử dụng công nghệ 28nm của Xilinx. Một kiến trúc như vậy cung cấp cho người dùng một nền tảng phần cứng với hiệu năng xử lý và công suất tiêu thụ tương đương một thiết kế ASIC, trong khi vẫn đảm bảo tính linh hoạt trong thay đổi thiết kế phần cứng của FPGA, và tính dễ dàng trong việc lập trình lại sản phẩm của bộ vi xử lý.

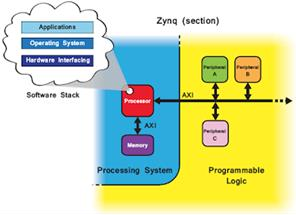
Cấu trúc này có thể được sử dụng theo một trong ba cách:

1. PS được sử dụng độc lập với PL.
2. Các lõi IP mềm có thể được tạo ra trong phần PL và kết nối để mở rộng chức năng của PS. Chúng ta có thể sử dụng tổ hợp PS+PL để tạo ra các hệ thống SoC phức tạp và hiệu năng cao.



*Hình 2: Sơ đồ khối chức năng Zynq-7000*

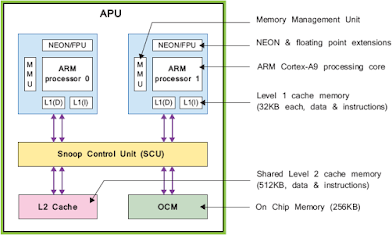
Tài nguyên logic trong PL có thể được thiết kế để hoạt động một cách độc lập với phần PS. Tuy nhiên PS hoặc JTAG phải được sử dụng để lập trình PL

Với các tài nguyên phần cứng sẵn có trên ZynQ, người sử dụng dễ dàng thực hiện phần cứng của hệ thống nhúng mong muốn trên ZynQ. (Hình 3)

*Hình 3: Ánh xạ phần cứng của một hệ thống SoC trên Zynq-7000*

Cho mục đích kiểm chứng mảng MUSRA chúng ta cần tạo ra một hệ thống SoC, trong đó phần điều khiển quá trình cấu hình cho MUSRA được thực hiện bởi bộ vi xử lý ARM Cortext-A9 trong khối PS, trong khi mảng MUSRA được thực hiện bởi các tài nguyên logic tái cấu hình trong khối PL.

* + 1. CÁC THÀNH PHẦN CHÍNH CỦA ZYNQ – 7000
  + Hệ vi xử lý PS:

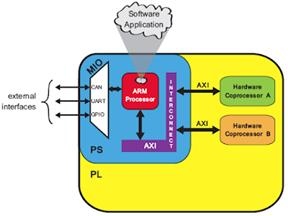


*Hình 4: Sơ đồ khối của đơn vị xử lý ứng dụng APU*

Tất cả các thiết bị ZynQ đều được tích hợp sẵn một hệ vi xử lý trên cơ sở bộ vi xử lý ARM Cortex-A9 lõi kép. Đây là một bộ xử lý được thực hiện “cứng” như một thành phần đã được tối ưu và chuyên dụng trên thiết bị ZynQ. Bên cạnh đó hệ vi xử lý cũng được trang bị nhiều tài nguyên tính toán bổ trợ khác tạo thành đơn vị xử lý ứng dụng APU (Application

Processing Unit) và nhiều giao diện ngoại vi, bộ nhớ cache, giao diện ghép nối bộ nhớ, mạng truyền thông, và mạch tạo tín hiệu đồng bộ clock.

Hình 4: mô tả APU bao gồm hai lõi vi xử lý ARM. APU cũng bao gồm một bộ nhớ cache L2 và bộ nhớ on-chip (On-Chip Memory: OCM) dùng chung cho cả hai lõi vi xử lý. Đơn vị điều khiển SCU (Snoop Control Unit) tạo thành cầu nối giữa các lõi xử lý và các bộ nhớ chia sẻ cũng như tới khối PL. SCU chịu trách nhiệm duy trì sự đồng bộ giữa bộ nhớ cache dữ liệu (L1(D)) và bộ nhớ cache L2, phân xử khi cả hai lõi xử lý ARM cùng muốn truy xuất tới cache L2.

Truyền thông giữa hệ PS với các thiết bị ngoại vi bên ngoài chip có thể thực hiện trực tiếp thông qua khối ghép nối MIO (Multiplexed Input/Output). Một cách khác để kết nối PS với các ngoại vi bên ngoài là thông qua khối EMIO (Extended MIO). Khi sử dụng EMIO, các kết nối giữa PS và thiết bị ngoại vi được thực hiện trung gian qua khối PL

*Hình 5: Giao diện ghép nối MIO giữa hệ PS và các linh kiện bên ngoài.*

Các giao diện ghép nối I/O được cung cấp bởi PS bao gồm:

 +**2 giao diện SPI** (Serial Peripheral Interface): cung cấp một phương thức truyền thông nối tiếp sử dụng 4 dây cho cả truyền và nhận.

 **+2 giao diện I2C:** cung cấp một phương thức truyền thông nối tiếp sử dụng 2 dây cho cả truyền và nhận.

 **+2 giao diện CAN (Controller Area Network):** chuẩn truyền thông nối tiếp được sử dụng phổ biến trong công nghiệp ô-tô.

 **+2 giao diện UART** (**Universal Asynchronous Receiver**) : giao diện truyền thông nối tiếp độ thấp thường được dùng để kết nối thiết bị đầu cuối với máy tính cho mục đích gỡ rối (debug).

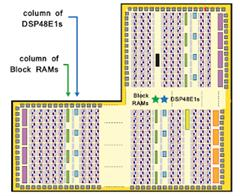
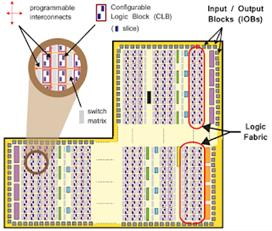
 +**4 giao diện GPIO** (gGeneral Purpose **Input/Output**) giao diện ghép nối song song 32-bit. 4 giao diện GPIO (gGeneral Purpose **Input/Output**)

 +**2 Giao diện SD:** giao diện dùng ghép nối với SDcard

 **+2 giao diện USB (Universal Serial Bus):** tương thích chuẩn USB 2.0

 **+2 Giao diện Ethernet:** hỗ trợ tốc độ kết nối 10Mbps, 100Mbps và 1Gbps

* Tài nguyên logic khả trình PL (Programmable Logic)



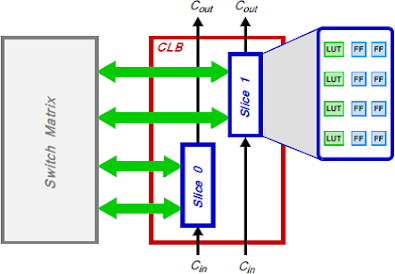
* + - 1. b.

*Hình 7*: *Cấu trúc của khối logic tái cấu hình PL*

Tài nguyên logic khả trình bao gồm các kết cấu logic có khả năng tái cấu hình và một số tài nguyên phần cứng chuyên dụng tương thích với chuẩn FPGA Artix-7 và Kintex-7 của Xilinx.

Các kết cấu logic khả tái cấu hình chủ yếu được chỉ ra trong Hình 7(a), bao gồm:

* + **Khối logic khả trình CLB (Configuration Logic Block):** CLB là một nhóm nhỏ các phần tử logic được sắp xếp thành một mảng 2 chiều trên PL. Các CLB được kết nối với nhau thông qua mạng định tuyến khả trình.



*Hình 8. Cấu trúc của CLB*

* + **Slice** là một thành phần cấu thành lên CLB. Trong kiến trúc của ZynQ, như chỉ ra trong Hình 8 mỗi Slice bao gồm 4 bảng LUT, 8 Flip-Flops và các tài nguyên logic khác.
  + **Bảng LUT (Lookup Table):** LUT là tài nguyên phần cứng cá khả năng thực hiện (i) các hàm logic 6 biến; (ii) các khối nhớ ROM (Read only Memory) nhỏ; (iii) các khối nhớ RAM (Random Access Memory); hay (iv) các thanh ghi dịch. Các LUTs có thể được kết hợp với nhau để thực hiện các hàm logic, bộ nhớ, thanh ghi dịch có kích thước lớn hơn.
  + **Flip-Flop (FF):** một phần tử mạch tuần tự thực hiện chức năng của một thanh ghi một bit với chức năng reset.
  + **Ma trận chuyển mạch (Switch Matrix)** – đặt cạnh mỗi CLB là một ma trận chuyển mạch cung cấp chức năng định tuyến linh hoạt cho việc tạo ra các kết nối (i) giữa các phần tử trong một CLB và đến các tài nguyên khác trong PL.
  + **Các khối vào/ra IOB (Input/Output Blocks)** – Các khối IOB cung cấp giao diện ghép nối giữa phần PL với các chân vật lý của chip ZynQ, cho phép các mạch điện

bên trong ZynQ thông tin với mạch điện ngoài chip. Mỗi IOB có thể điều khiển một tín hiệu vào hoặc ra 1 bit.

ZynQ cũng cung cấp 2 khối phần cứng chức năng đặc biệt như trong Hình 7(b) gồm:

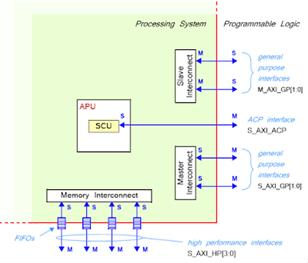
 +Block RAMs hướng tới các ứng dụng có yêu cầu bộ nhớ lớn

 +DSP48E1 cung cấp các phép tính số học tốc độ cao hướng tới các ứng dụng xử lý DSP.

* Giao diện ghép nối giữa PS và PL

Giao diện ghép nối giữa PS và PL cho phép người sử dụng có thể xây dựng các hệ thống phức hợp bằng cách ghép nối hệ thống vi xử lý PS với các lõi IP phần cứng chuyên dụng được thực thi trên PL.

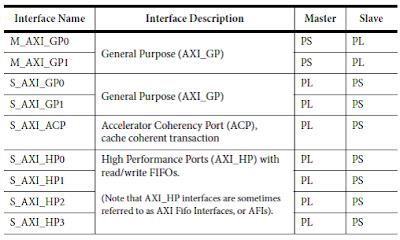
1. Giao diện ghép nối AXI



*Hình 9. Giao diện ghép nối AXI.*

Thành phần then chốt tạo nên giao diện ghép nối giữa PS và PL là chuẩn AXI. ZynQ sử dụng 9 giao diện ghép nối AXI cho mục địch kết nối PS với PL.

* + **Interconnect – liên kết nối:** là một chuyển mạch dùng để quản lý và điều hướng luồng thông tin giữa các giao diện AXI. Có một vài liên kết nối bên trong PS một vài trong số chúng được ghép nối trực tiếp với PL.
  + **Interface – giao diện ghép nối:** là một kết nối điểm - điểm dùng để truyền dữ liệu, địa chỉ và các tín hiệu bắt tay giữa các thiết bị chủ với các thiết bị khách bên trong hệ thống.

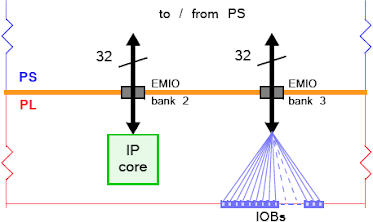


*Hình 10: Các giao diện ghép nối giữa PS và PL*

* + General Purpose AXI – giao diện AXI đa chức năng là một bus dữ liệu 32 bit phù hợp cho các ứng dụng truyền thông tốc độ thấp và trung bình giữa PS và PL.
  + Accelerator Cohenrency Port – một kết nối không đồng bộ giữa PL và SCU bên trong APU. Cổng này được sử dụng để đạt được tính kết hợp giữa bộ nhớ cache trong APU với PL. PL đóng vai trò thiết bị chủ của kết nối này.
  + High performance Ports – Bốn giao diện ghép nối AXI hiệu năng cao có chứa các bộ đệm FIFO để cung cấp các thao tác đọc viết theo kiểu “loạt”, hỗ trợ truyền thông tốc độ cao giữa PL và các bộ nhớ bên trong PS. Độ rộng dữ liệu là 32 hoặc 64 bit. PL đóng vai trò thiết bị chủ của cả bốn cổng này.

1. Giao diện EMIO

Một vài kết nối từ PS có thể được định tuyến thông qua PL tới các ngoại vi ngoài chip thông qua giao diện EMIO (Extended MIO). EMIO chỉ hỗ trợ một phần các giao diện ghép nối của MIO. Các kết nối của MIO được sắp xếp thành 2 bank 32 bit. Các giao diện ghép nối được định tuyến qua EMIO trong một vài trường hợp có thể kết nối trực tiếp với các chân bên ngoài của PL. Một lựa chọn sử dụng khác của EMIO là ghép nối PS với khối ngoại vi bên trong PL.



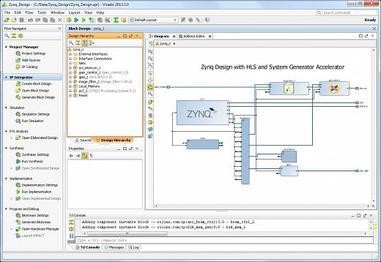
*Hình 11: Giao diện ghép nối EMIO giữa PS và PL.*

# PHẦN MỀM VIVADO CỦA XILINX

* + 1. **Giới thiệu về vivado**

Vivado Design Suite là bộ phần mềm do Xilinx sản xuất để tổng hợp và phân tích các thiết kế HDL, thay thế Xilinx ISE với các tính năng bổ sung cho hệ thống phát triển chip và tổng hợp cấp cao. Vivado đại diện cho việc viết lại và suy nghĩ lại toàn bộ dòng thiết kế (so với ISE), và đã được các nhà đánh giá mô tả là "được hình thành tốt, tích hợp chặt chẽ, nhanh, có thể mở rộng, có thể duy trì và trực quan".

Giống như các phiên bản sau của ISE, Vivado bao gồm ISIM giả lập logic tích hợp.Vivado cũng giới thiệu tổng hợp cấp cao, với một chuỗi công cụ chuyển đổi mã C thành logic lập trình.Vivado đã được mô tả là một "công cụ EDA toàn diện tiên tiến với tất cả các chuông và còi mới nhất về mô hình dữ liệu, tích hợp, thuật toán và hiệu suất".



Vivado Design Suite bao gồm Vivado GUI (giao diện đồ họa và quản lý dự án), IP Integrator (công cụ dùng để tích hợp các lõi IP vào hệ thống), Software Development Kit (bộ công cụ phát triển phần mềm), và một số tài nguyên khác (thư viện các lõi IP phần cứng, thư việc các trình điều khiển cho phát triển phần mềm), trình biên dịch và gỡ rối GNU hỗ trợ phát triển phần mềm ngôn ngữ C/C++ cho vi xử lý ARM Cortex-A9, tài liệu hướng dẫn sử dụng, ví dụ mẫu…). Chú ý là IP interator thay thế cho nhiều chức năng PlanAhead và XPS trong ISE Design Suite, trong khi Vivado logic Abalyzer thay thế cho chức năng ChipScope Analyzer của ISE.

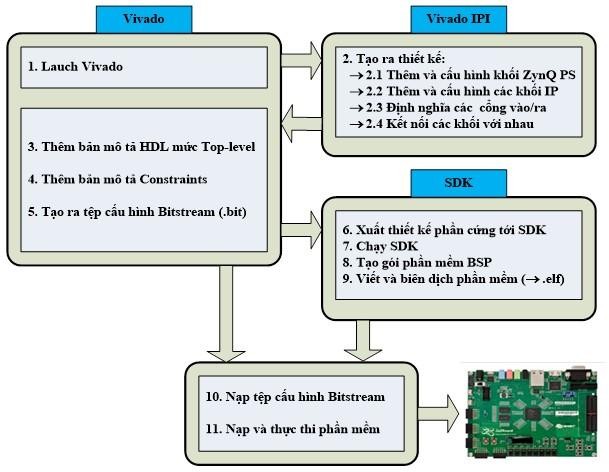
Vivado Design Suite cung cấp tập hợp các công cụ hỗ trợ cho thiết kế từ mức RTL, tổng hợp, thực thi và kiểm tra.

Công cụ tích hợp IP integration cho phép truy xuất tới các lõi IP đã được thiết kế sẵn bởi Xilinx (bao gồm cả các lõi vi xử lý). Bằng IP integration người sử dụng có thể dễ dàng tích hợp các lõi IP (như lõi xử lý ARM Cortex-A9, MicroBlaze, các lõi IP ngoại vi, lõi IP do người dùng tự thiết kế) để tạo thành một hệ thống vi xử lý nhúng hoàn chỉnh.

SDK là một môi trường phát triển tích hợp IDE bổ trợ cho IP Integrator trong việc phát triển và gỡ rối các chương trình phần mềm nhúng C/C++.

# Quy trình thiết kế cho ZynQ

Một trong những nhiệm vụ quan trọng của việc thiết kế SoC sử dụng ZynQ là việc tích hợp thêm các lõi IP vào hệ thống. Sau khi hệ thống phần cứng được tạo ra nó cần được xuất tới SDK để phát triển các phần mềm ứng dụng.



Hình :Quy trình thiết kế một hệ thống nhúng bằng bộ công cụ phần mềm Xilinx Vivado Suite.

Như hình trên chỉ ra lưu đồ các bước trong quy trình thiết kế một hệ thống nhúng SoC sử dụng chip ZynQ-7000 của Xilinx. Lưu ý rằng hệ vi xử lý PS có thể được sử dụng một cách độc lập với phần PL, do đó trong trường hợp này các bước 3, 4 và 5 trong lưu đồ là không cần thiết. Phần PL cũng có thể sử dụng độc lập với PS (khi đó các bước 2, 6-9 là không cần thiết), tuy nhiên PS và JTAG vẫn phải được sử dụng để lập trình cho PL. Công việc cụ thể của mỗi bước như sau:

*Bước 1*: Quá trình thiết kế và thực thi một hệ thống nhúng được bắt đầu bằng việc chạy phần mềm Vivado.

*Bước 2:* Trong Vivado, chọn Create Block Design để chạy IP Interator (tức IPI trong Hình 18).

* + - 1. Thêm lõi ZYNQ7 Processing System IP vào hệ thống. Kích đúp vào khối ZYNQ7 Processing System để thực hiện việc cấu hình cho hệ thống như: chọn các khối ngoại vi I/O chuyên dụng của PS, cấu hình bộ nhớ, cấu hình tốc độ xung nhịp, …
      2. Thêm các lõi IP từ thư viện có sẵn của Vivado hoặc thêm vào các lõi IP được tạo ra bởi người sử dụng.
      3. Định nghĩa các cổng vào/ra để cho phép hệ thống có thể giao tiếp với các khối ngoại vi bên ngoài.
      4. Kết nối các khối với nhau bằng việc kéo thả các tín hiệu/net từ cổng của một lõi IP này tới cổng thích hợp của IP khác. Việc kết nối cũng có thể được thực hiện một cách tự động bởi IP Integrator.

*Bước 3:* Tạo ra tệp mã nguồn HDL mức top-level mô tả thiết kế vừa tạo ra.

*Bước 4:* Đảm bảo rằng các ràng buộc (constraints) liên quan đến phần thiết kế trên PL được định nghĩa như yêu cầu của Vivado. Bước này được thực hiện thông qua việc tạo ra hoặc thêm vào một tệp XDC (Xilinx Design Contraints) tới project.

*Bước 5:* Tạo ra tệp bitstream cho cấu hình phần PL. Ở bước này một tệp định nghĩa phần cứng <system>. hdf và một tệp bitstream <system>.bit sẽ được tạo ra. Bitstream có thể được nạp vào chip FPGA từ môi trường Vivado hoặc trong SDK.

*Bước 6*: Sau khi phần cứng của hệ thống nhúng đã được tạo ra nó cần được xuất (export) tới SDK để tiêp tục phát triển phần mềm. Thực hiện xuất cấu trúc phần cứng của hệ thống (Export the hardware to SDK) để đảm bảo bảo phần cứng của thiết kế được tích hợp tự động với phần mềm.

*Bước 7:* Ngay sau khi phần cứng đã được xuất, chạy SDK (Launch SDK) từ Vivado để bắt đầu phát triển phần mềm.

*Bước 8:* Trong SDK, để phát triển các chương trình standalone (chương trình không chạy trên hệ điều hành) cần tạo ra một gói phần mềm hỗ trợ bo mạch BSP (Board Support Package) trên nền tảng phần cứng sẽ được sử dụng.

*Bước 9:* Viết chương trình ứng dụng và tiến hành biên dịcch chương trình để tạo tệp có phần mở rộng là .ELF – đây là tệp nhị phân có khả năng thực thi trên hệ thống. Tổ hợp của tệp bitstream và \*.ELF cung cấp một hệ SoC hoàn thiện bao gồm cả phần cứng và

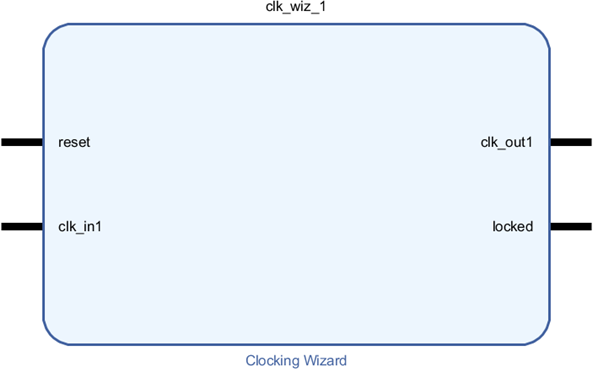
phần mềm trên ZedBoard.

*Bước 10*: Nạp tệp cấu hình .bit (Program FPGA) để thiết lập phần cứng trên bo mạch Zedboard.

*Bước 11:* Nạp và thực thi chương trình phần mềm .elf trên bo mạch Zedboard.

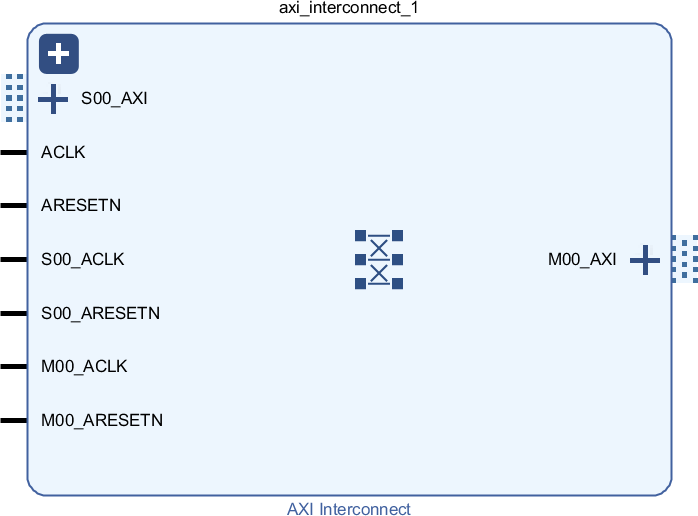
# CÁC KHỐI IPCORE ĐƯỢC SỬ DỤNG

* + 1. **Khối Clocking Wizards**



Khối này có chức năng tạo xung đồng hồ cho hệ thống

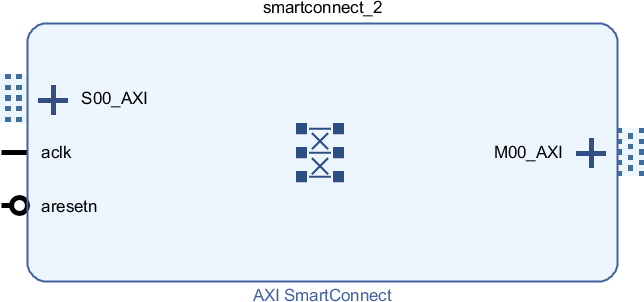
# Khối AXI Interconnect



IP sử dụng trong việc kết nối PL và PS trên nền tảng ZYNQ-7000. IP này kết nối một hoặc nhiều thiết bị chủ được ánh xạ bộ nhớ AXI với một hoặc nhiều thiết bị tớ được ánh xạ trên bộ nhớ.

Các giao diện AXI là giao diện AXI theo chuẩn AMBA, AXI4, AXI4-Lite. IP này chỉ dùng để ánh xạ bộ nhớ; nên vì thế giao diện AXI4-Stream – giao diện luồng dữ liệu phát trực tiếp tốc dộ cao sẽ không được áp dụng cho IP này. IP này có thể kết nối tối đa 16 thiết bị chủ với 16 thiết bị tớ.

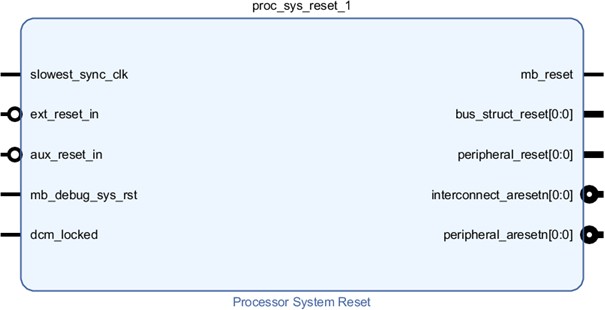
# Khối AXI SmartConnect



AXI SmartConnect là một khối IP phân cấp được thêm vào thiết kế khối tích hợp IP Vivado® trong Vivado Design Suite.

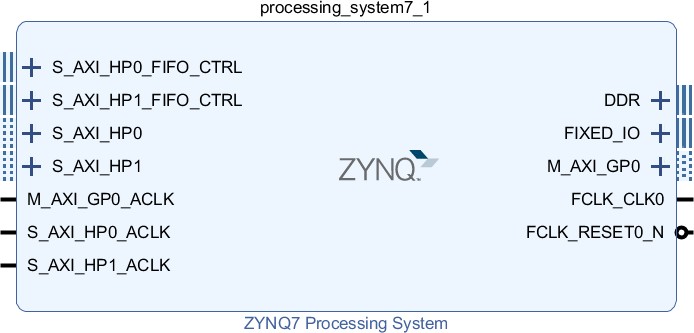
AXI SmartConnect là sự thay thế thả xuống cho lõi AXI Interconnect v2. AXI SmartConnect được tích hợp chặt chẽ hơn vào môi trường thiết kế Vivado để tự động định cấu hình và thích ứng với IP chủ và IP nô lệ được kết nối với sự can thiệp tối thiểu của người dùng.

# Khối Processor System Reset



Thiết lập lại hệ thống xử lý là một IP mềm cung cấp cơ chế xử lý các điều kiện đặt lại cho một hệ thống nhất định. Lõi xử lý nhiều điều kiện thiết lập lại ở đầu vào và tạo ra các thiết lập lại phù hợp ở đầu ra. Lõi này tạo ra các thiết lập lại dựa trên các điều kiện thiết lập lại bên ngoài hoặc bên trong.

# Khối ZYNQ7 Processing System

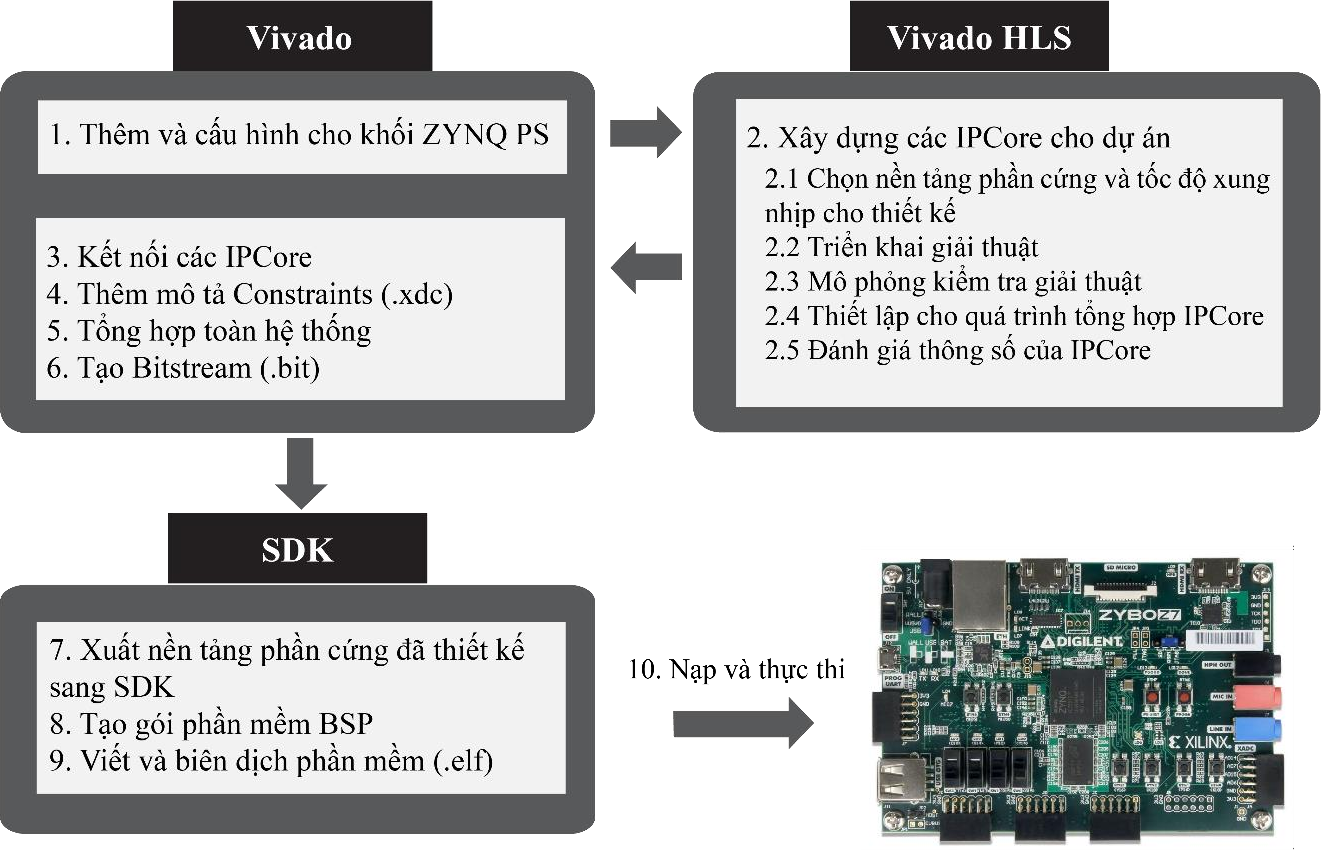


IP đại diện cho phần PS trên nền tảng ZYNQ-7000. Đây là khối thực thi giải thuật trên phần mềm.

Trong hệ thống IP này chiệu trahs nhiệm điều khiển toàn bộ hệ thống, IP có thể đống vai trò là một thiết bị chủ hoặc một thiết bị tớ tùy thuộc vào công việc cần thực hiện. Các tính năng cho IP này có thể được cấu hình một dễ dàng trong Vivado.

# CHƯƠNG 2: THỰC THI VÀ KẾT QUẢ TỪNG BƯỚC

* 1. **THỰC THI GIẢI THUẬT TRÊN NỀN TẢNG PHẦN CỨNG**



Hình **Error! No text of specified style in document.**.1. Quy trình thực thi thiết kế với phần cứng trên nền tảng ZYNQ-7000

Nội dung thực hiện của từng bước như sau:

* + - Bước 1: Khởi động Vivado và chọn nền tảng phần cứng cho thiết kế, dùng công cụ IP Interator để thêm lõi ZYNQ7 Processing System IP vào thiết kế, và tiến hành cấu hình cho lõi ZYNQ7, lõi này địa diện cho khối PS trong toàn bộ thiết kế.
    - Bước 2: Khởi động Vivado HLS và chọn nền tảng phần cứng cho thiết kế, đồng thời chọn tần số xung hoạt động cho IPCore cần xây dựng
    - Bước 3: Sau khi đã xây dựng các IPCore từ bước 2, ta tiến hành thêm các IP này vào “IP Catalog” của Vivado để có thể sử dụng chúng cho thiết kế hiện

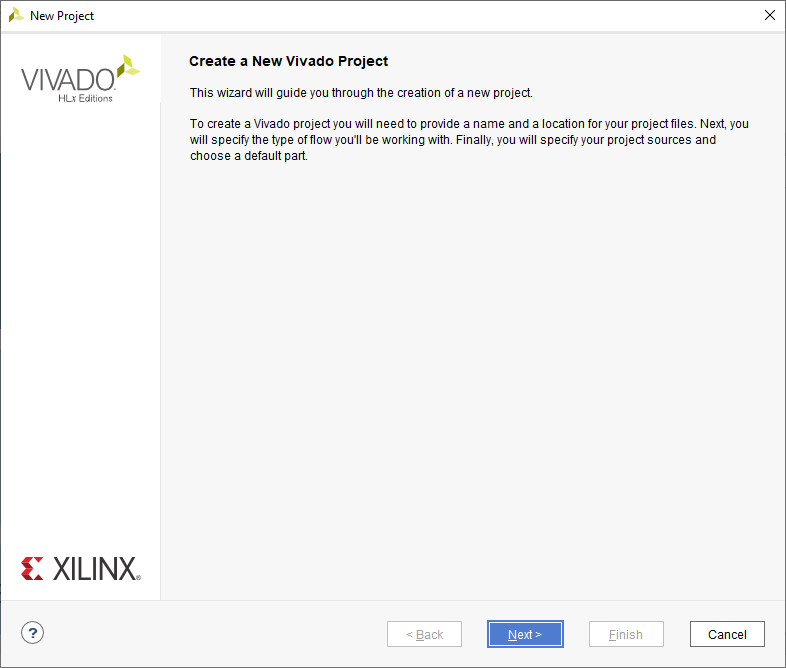
tại. Sử dụng IP Interator thêm các IP này vào thiết kế và tiến hành kết nối các IP lại với nhau.

* + - Bước 4: Thêm các tệp “Constraints” vào thiết kế, đây là các tệp bắc buộc cho một thiết kế, chứa các khai báo kết nối cho phần cứng và các định nghĩa khởi tạo xung đồng hồ, một thiết kế có thể bao gồm nhiều tệp constrainst, các tệp constraints đi kèm với các IPCore sẽ có mức độ ưu tiên hơn tệp constrainst ở thiết kế cấp cao nhất khi tổng hợp thiết kế.
    - Bước 5: Sau khi thêm các tệp constraints vào thiết kế, ta tiến hành kiểm tra kết nối và tổng hợp thiết kế.
    - Bước 6: Sau khi bước 5 hoàn thành tiến hành tạo tệp Bitstream (.bit) cho thiết kế, đây là tệp chứa các chuỗi nhị phân mô tả toàn bộ thiết kế bao gồm các kết nối, giá trị xung đồng hồ sử dụng, các I/O sử dụng trong hệ thống. Tệp này được xem nền tảng phần cứng cho thiết kế và dùng để tái cấu hình các khối CLB trên PL.
    - Bước 7: từ Vivado tiến hành khởi động SDK và bao gồm tệp Bitstream đã tạo ở bước 6.
    - Bước 8: Trong SDK, để phát triển các chương trình standalone (chương trình không chạy trên hệ điều hành) cần tạo ra một gói phần mềm hỗ trợ bo mạch BSP (Board Support Package) trên nền tảng phần cứng sẽ được sử dụng.
    - Bước 9: Viết chương trình ứng dụng và tiến hành biên dịcch chương trình để tạo tệp có phần mở rộng là .elf – đây là tệp nhị phân có khả năng thực thi trên hệ thống. Tổ hợp của tệp bitstream và \*.elf cung cấp một hệ SoC hoàn thiện bao gồm cả phần cứng và phần mềm.
    - Bước 10: từ các tệp đã tạo trên Vivado và SDK tiến hành tạo tệp BOOT.bin trên SDK, đây là tệp dùng để thực thi thiết kế trên bo mạch bằng cách sử dụng SDCard. Việc sử dụng SDCard ưu việt hơn khi nạp thẳng chương trình lên bo mạch, vì giải quyết được vấn đề dữ liệu của thiết kế đã thực thi bị mất đi khi mất điện

# QUÁ TRÌNH THỰC HIỆN TRÊN ỨNG DỤNG VIVADO

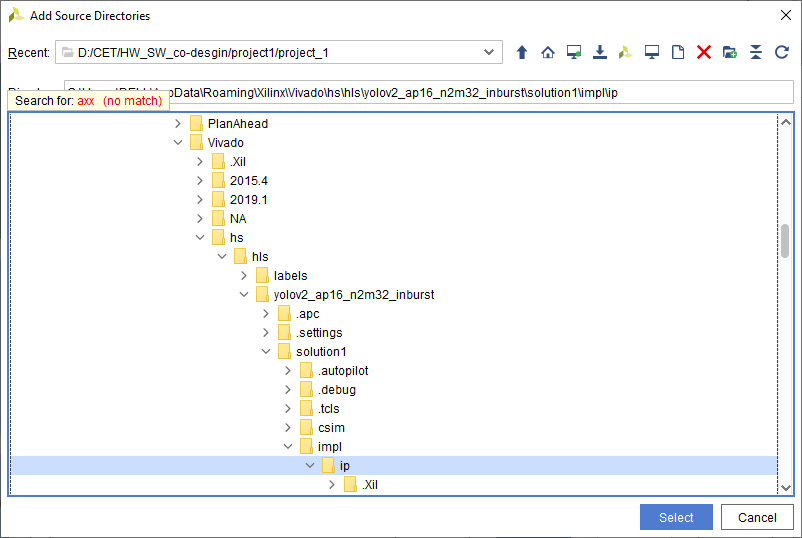
B1: Mở vivado và tạo project.

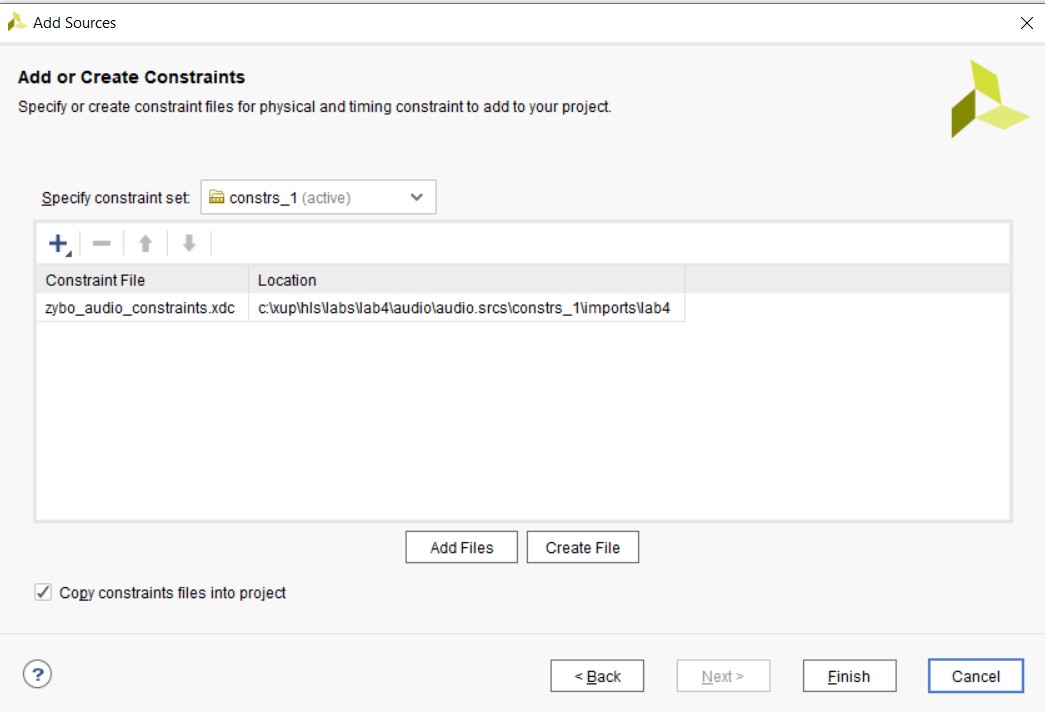
* + - Bước này ta chỉnh chọn ở RTL để viết theo verilog
    - Chọn thư mục lưu (lưu ý khi đặt tên không dấu)



B2: Cấu hình file nguồn cho project.

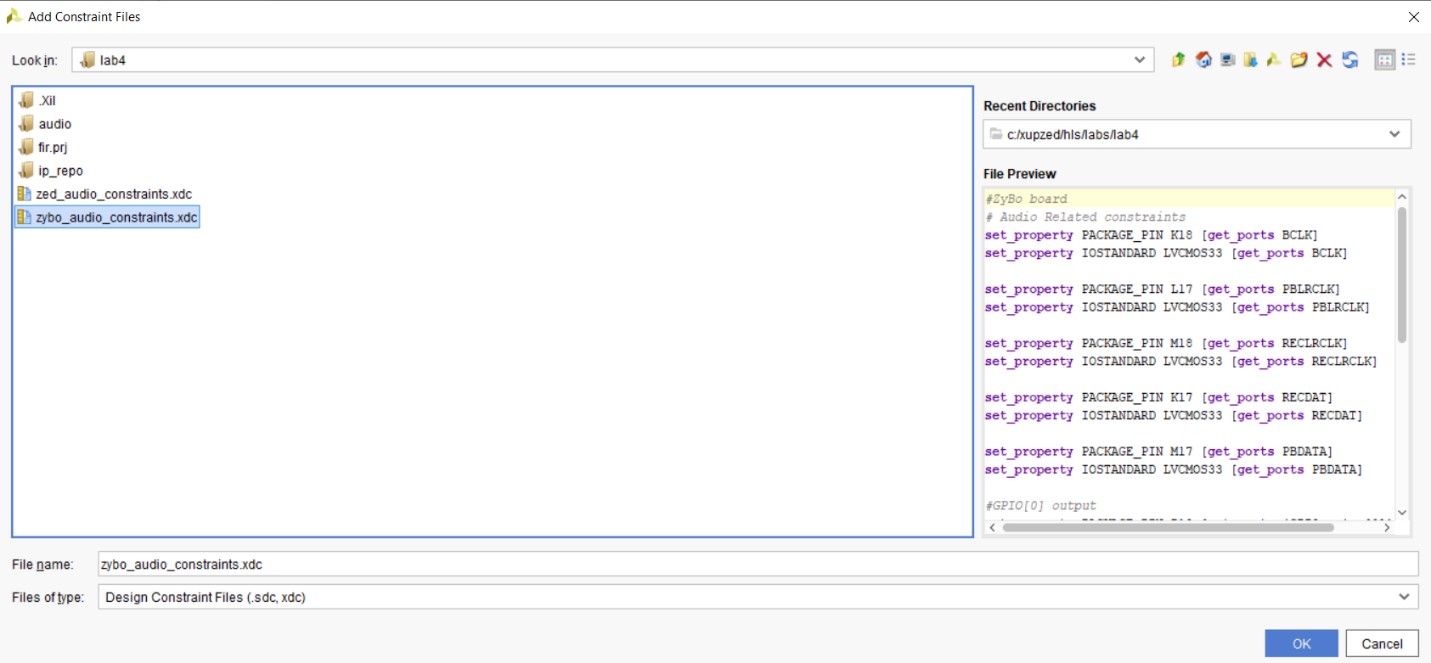
* + - Ta chọn thư mục ip chứa các IPCores của hệ thốngores của hệ thốngores của hệ thống

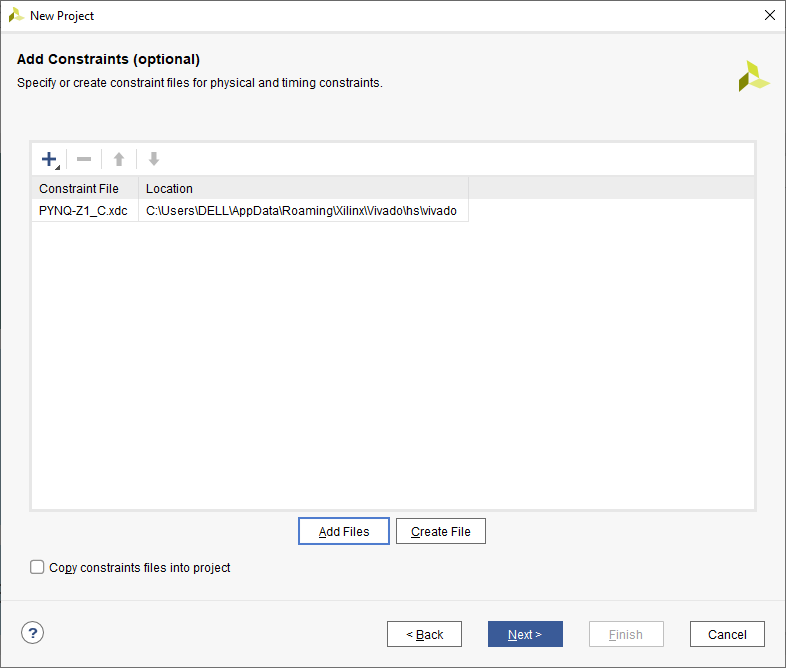




B3: Thêm file “constraints” vào project

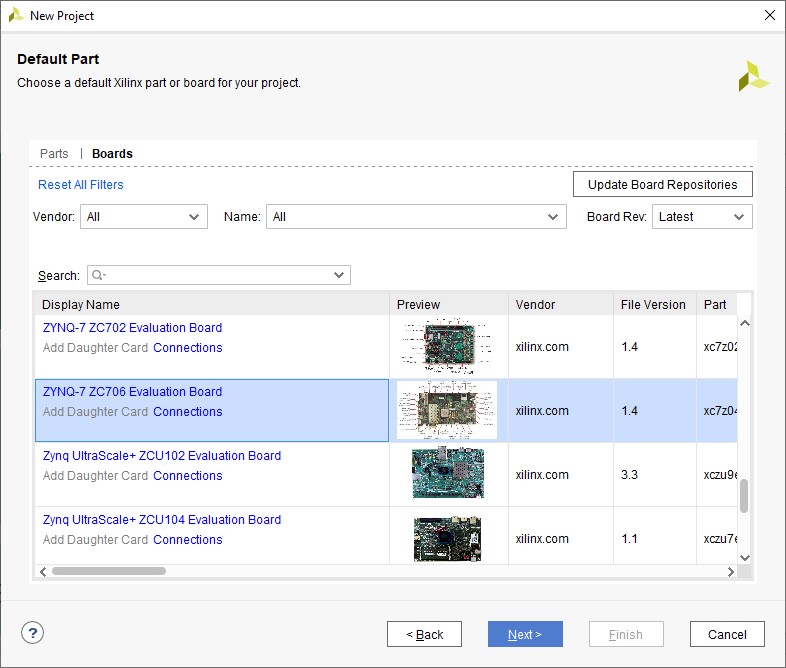
* + - Thêm các tệp “Constraints” vào thiết kế, đây là các tệp bắc buộc cho một thiết kế, chứa các khai báo kết nối cho phần cứng và các định nghĩa khởi tạo xung đồng hồ



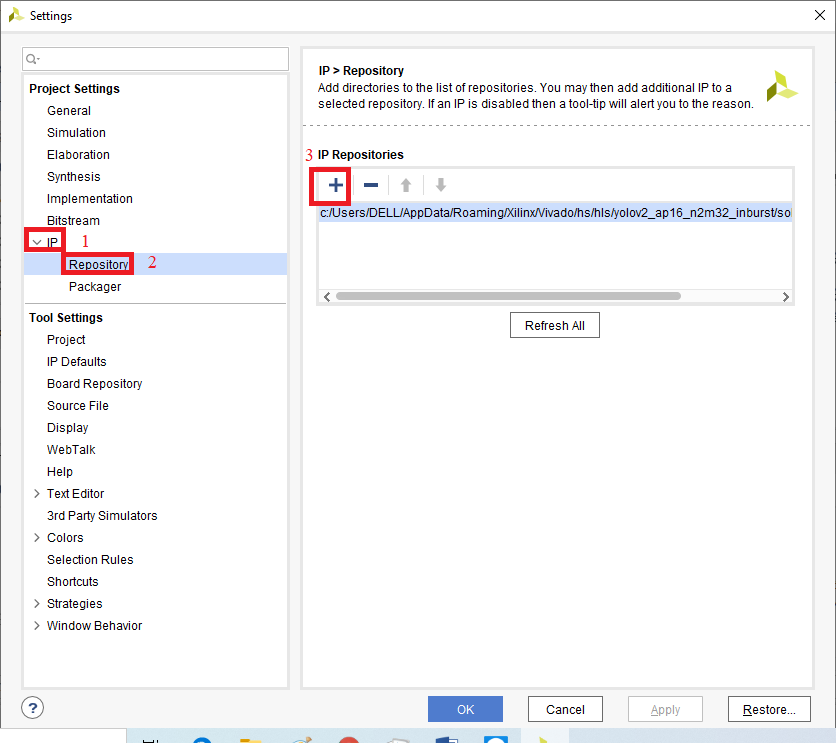


B4: Chọn board mình cần nạp

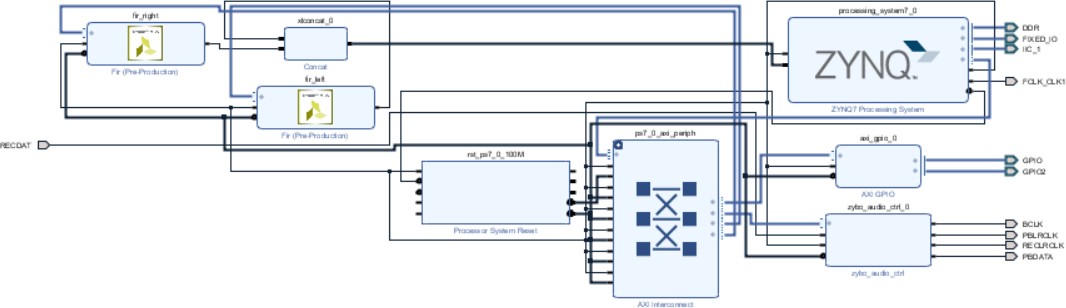
* + - Mình chọn board ZYNQ 7000 – ZC706 để phù hợp với kit sử dụng



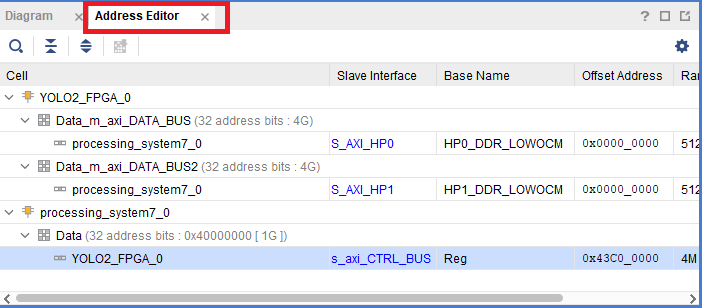
B5: Cấu hình IP để nhận IPCores

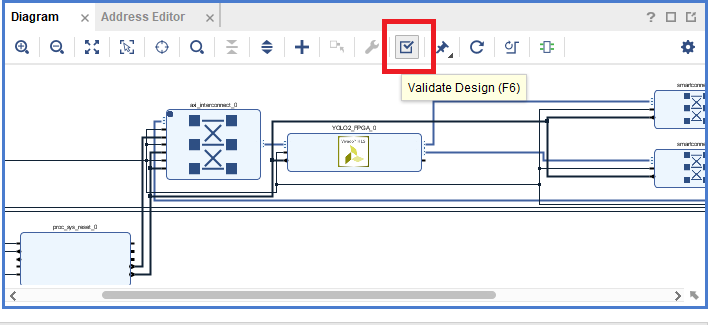


B6: “Create Block Design” và thêm vào các khối cần thiết

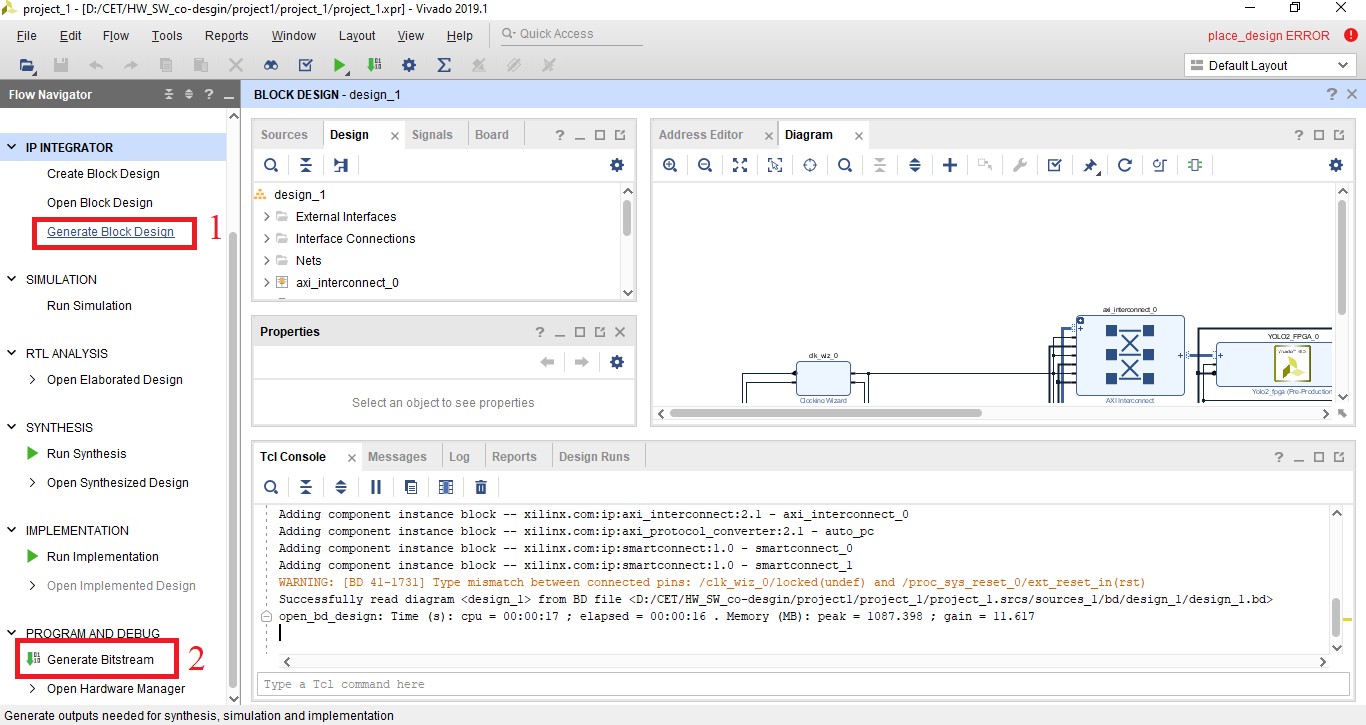


B7: Sau khi đã vẽ xong. Thêm đia chỉ IPCores vào và kiểm tra lỗi





B8: Tạo “block” và tiến hành xuất file “bistream”



B9: Tạo SDK để tiến hành nạp lên board

