BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC SAO ĐỎ

NGUYỄN VĂN LĂNG

THIẾT KẾ HỆ VI ĐIỀU KHIỂN LÕI MỀM MICROBLAZE 32 BIT TRÊN FPGA VÀ CÀI ĐẶT ỨNG DỤNG

LUẬN VĂN THẠC SĨ
CHUYÊN NGÀNH: KỸ THUẬT ĐIỆN TỬ

NGƯỜI HƯỚNG DẪN KHOA HỌC:
TS. HỒ KHÁNH LÂM

HẢI DƯƠNG – NĂM 2018

LỜI CAM ĐOAN

Tôi xin cam đoan kết quả đạt được trong luận văn là sản phẩm của riêng cá nhân, là kết quả của quá trình học tập và nghiên cứu khoa học độc lập. Trong toàn bộ nội dung của luận văn, những nội dung được trình bày hoặc là của cá nhân hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp. Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ luận văn nào khác.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Hải Dương, ngày 10 tháng 7 năm 2018

TÁC GIẢ

Nguyễn Văn Lăng

MỤC LỤC

LỜI CAM ĐOAN
DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT
DANH MỤC CÁC BẢNG
DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ
MỞ ĐẦU 1
CHƯƠNG 1: CÔNG NGHỆ FPGA 4
1.1. Phân loại các vi mạch tích hợp 4
1.1.1. Tổng quan phát triển các mạch tích hợp 4
1.1.2. Các mạch tích hợp ứng dụng chuyên biệt (ASIC) 6
1.1.3. Các thiết bị logic có thể lập trình được (PLD) 7
1.2. FPGA 7
1.2.1. Kiến trúc FPGA 7
1.2.2. Định tuyến trong FPGA 13
1.3. Phương pháp lập trình FPGA 15
1.3.1. Lập trình dựa vào bộ nhớ SRAM (Static Random Access Memory) 16
1.3.2. Lập trình dựa vào đốt cầu chì (anti-fuse) 16
1.4. So sánh FPGA với các công nghệ vi mạch tích hợp khác 16
1.4.1. FPGA và ASIC 16
1.4.2. FPGA và PLD 18
1.5. Công nghệ FPGA của một số nhà công nghệ 18
1.5.1. Xilinx FPGA 18
1.5.2. Altera FPGA 19
1.6. Kết luận chương 19
CHƯƠNG 2: THIẾT KẾ PHẦN CỨNG BẰNG VHDL 21
2.1. Ngôn ngữ mô tả phần cứng VHDL 21
2.1.1. Lịch sử của VHDL 21
2.1.2. Ứng dụng của VHDL 21
2.1.3. Đặc điểm của VHDL 22
2.1.3.1. Các mức trìu tượng trong thiết kế mạch tích hợp 24
2.1.3.2. Các tầng trìu tượng của thiết kế VHDL 25
2.1.3.3. Mô tả của các tâng trìu tượng trong thiết kế VHDL 26

2.2. Quá trình thiết kế phần cứng bằng VHDL 29
2.2.1. Các công đoạn thiết kế bằng VHDL 29
2.2.2. Thiết kế phần cứng trên Xilinx FPGA 30
2.2.2.1. Tính năng thiết kế 30
2.2.2.2. Tài liệu liên quan 31
2.2.3. Công cụ phần mềm thiết kế Xilinx ISE 32
2.2.3.1. Khởi động (Startup) 32
2.2.3.2. Trợ giúp (Help) 32
2.2.3.3. Tạo một Project mới 33
2.2.3.4. Bổ xung mã nguồn VHDL mới 35
2.2.3.5. Soạn thảo mã nguồn VHDL 38
2.2.3.6. Kiểm tra cú pháp 39
2.2.3.7. Gán chân tín hiệu 41
2.2.3.8. Synthesize, Translate, Map, và Place & Route 45
2.2.3.9. Synthesize, Translate, Map, và Place & Route 46
2.2.3.10. Chạy chương trình trên bảng Spartan-3E. 52
2.3. Kết luận chương 53
CHƯƠNG 3: THIẾT KẾ HỆ VI ĐIỀU KHIỂN LÕI MỀM MICROBLAZE 32-BIT VÀ CÀI ĐẶT ỨNG DỤNG THỬ NGHIỆM 54
3.1. Vi điều khiển Microblaze 32-bit 54
3.1.1. Kiến trúc của Microblaze 54
3.1.2. Các định dạng dữ liệu và tập lệnh của Microblaze 56
3.1.3. Hiệu năng của Microblaze 57
3.2. Thiết kế hệ nhúng đơn giản với Microblaze 57
3.2.1. Bảng phát triển trên FPGA Xilinx Starter-3E 500E 57
3.2.2. Lựa chọn cấu hình hệ nhúng với Microblaze 58
3.2.3. Các bước thiết kế và kết quả sử dụng Công cụ phần mềm Xilinx ISE14.1 59
3.3. Xây dựng và cài đặt các phần mềm ứng dụng 69
3.3.1. Phần mềm Hello.c và cài đặt thử nghiệm 69
3.3.2. Phần mềm kiểm tra bộ nhớ và cài đặt thử nghiệm 72
3.3.3. Phần mềm kiểm tra các giao tiếp ngoại vi và cài đặt thử nghiêm 73

3.4. Kết luận chương 74
KẾT LUẬN VÀ KIẾN NGHỊ 75
5. TÀI LIỆU THAM KHẢO 76

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

Từ viết tắt	Nghĩa tiếng anh
ABEL -	Advanced Boolean Equation Language
ADC -	Analog-to-Digital Converter
ALU -	Arithmetic Logic Unit
ASIC -	Application Specific Integrated Circuit
CLB -	Configurable Logic Block
DCM -	Digital Clock Management
DAC -	Digital-to-Analog Converter
DCR -	Device Control Register
DSP -	Digital Signal Peocessor
DXCL -	Data side Xillinx Cache Link Interface
EDK -	Embedded Development Kit
FPGA -	Field programmable Gate Array
FPU -	Floating Point Unit
IF -	Interface
IXCL -	Instruction side Xillinx Cache Link Interface
LMB -	Local Memory Bus
LUT -	Look-Up Table
MAC -	Multiply-accumulate circuits
NRE -	Non Recurring Engineering
OPB -	On-Chip Peripheral Bus
PLB -	Processor Local Bus
PC -	Personal Computer
PLD -	Programmable Logic Device
RISC -	Reduced Instruction Set Computer
SoC -	System on Chip
XPS -	Xilinx Platform Studio
XSD -	Xilinx Software Development
XCL -	Xillinx Cache Link
VHDL -	Very High Speed Hardware Desription Luaguage
VHSIC -	Very High Speed Integrated Circuits

DANH MỤC CÁC BẢNG

		Trang
Bảng 1.1	Họ Spartan-3 FPGA mật độ cổng cao	18
Bảng 2.1	Các cổng input/output của ENTITY top_level	42
Bảng 2.2	Các giá trị để đưa vào cửa sổ I/O Ports	44

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

		Trang
Hình 1.1	Phát triển số lượng nhân xử lý trên chip	4
Hình 1.2	Phân loại các mạch tích hợp	6
Hình 1.3	Mảnh PLD	7
Hình 1.4	Kiến trúc của FPGA dựa trên SRAM	9
Hình 1.5	Xilinx Virtex-5 FPGA LUT- cặp FF	11
Hình 1.6	Altera Stratix IV FPGA ALM	11
Hình 1.7	Các loại LUT của Stratix ALM	11
Hình 1.8	Định tuyến qua các hộp kết nối	13
Hình 1.9	Định tuyến qua các khối chuyển mạch	14
Hình 1.10	Định tuyến theo các ma trận chuyển mạch và các đường dây dài đơn trong Xilinx FPGA.	14
Hình 1.11	Kiến trúc định tuyến của Xilinx FPGA	15
Hình 1.12	Quá trình thiết kế trên FPGA và ASIC	17
Hình 2.1	Các tầng trìu tượng thiết kế mạch tích hợp	24
Hình 2.2	Các mức trìu tượng thiết kế VHDL	26
Hình 2.3	Các mô tả ở các mức trìu tượng	26
Hình 2.4	Ví dụ mô tả hành vi trong VHDL	27
Hình 2.5	Các tiến trình ở mức RTL	28
Hình 2.6	Quá trình thiết kế VHDL	29
Hình 2.7	Bảng phát triển Spartan-3E 500K/1600K	31
Hình 2.8	Cửa sổ khởi động ban đầu của Project	31
Hình 2.9	Thực đơn Help	32
Hình 2.10	New Project Wizard, Trang Create New Project	33
Hình 2.11	New Project Wizard, trang Project Settings	34
Hình 2.12	New Project Wizard, trang Project	34
Hình 2.13	Project – New Source…	35
Hình 2.14	New Source Project: Select Source Type:VHDL	35
Hình 2.15	New Source Project: Define Module	36
Hình 2.16	New Source Project, Summary	37
Hình 2.17	file nguồn mới top_level.vhd hiển thị trong tab	37
Hình 2.18	nội dung file top_level.vhd được hiển thị trong Project	38

Hình 2.19	nội dung file top_level.vhd hiển thị trong Project Navigator sau khi soạn	39
Hình 2.20	Project Navigator với mở rộng	40
Hình 2.21	Green tick next cho kiểm tra cú pháp	40
Hình 2.22	Ví dụ, trong đó lỗi đã xuất hiện dấu chéo đỏ ở chỗ kiểm tra lỗi	41
Hình 2.23	Một khoản của màn hình Project Navigator, với User	42
Hình 2.24	Hộp hội thoại yêu cầy tạo UCF file	43
Hình 2.25	Cửa sổ PlanAhead hiển thị lần đầu	43
Hình 2.26	Hiển thị cửa sổ I/O Ports riêng	43
Hình 2.27	Hiển thị cửa sổ I/O Ports mở rộng đến các cổng riêng	44
Hình 2.28	Hiển thị cửa sổ I/O Ports với các giá trị đã được điền	44
Hình 2.29	một khoản của màn hình Project Navigator, với mở ra	45
Hình 2.30	Một khoản của màn hình Project Navigator, với mở ra Implement Design, sau đó Translate, Map và Place & Router đã chạy	45
Hình 2.31	Một khoản của màn hình Project Navigator, với mở ra	46
Hình 2.32	Một khoản của màn hình Project Navigator, sau khi Generate Programming File chạy xong	46
Hình 2.33	Cửa sổ ban đầu của iMPACT	47
Hình 2.34	Cửa sổ iMPACT, sau khi click hai lần lên	47
Hình 2.35	Cửa sổ iMPACT, hiển thị chọn Initialize Chain	48
Hình 2.36	Cửa sổ iMPACT, gán các file cấu hình	48
Hình 2.37	Cửa sổ iMPACT, gán file cấu hình cho xc3e500e	49
Hình 2.38	Cửa sổ iMPACT, hộp hội thoại yêu cầu có gắn SPI hay BPI PROM hay không.	49
Hình 2.39	Cửa sổ iMPACT, bỏ qua xcf40**s**	50
Hình 2.40	Cửa sổ iMPACT, bỏ qua xc2c64a	50
Hình 2.41	Cửa sổ iMPACT, hộp hội thoại Device	51
Hình 2.42	cửa sổ iMPACT, hiển thị device chain	51
Hình 2.43	Cửa sổ iMPACT, các lựa chọn khi click vào xc3s500e	51
Hình 2.44	Cửa sổ **iMPACT**, sau khi tải thành công chương trình vào bảng Spartan-3E	52
Hình 2.45	Bảng Spartan-3E với chương trình đang chạy	53
Hình 3.1	Kiến trúc của Microblaze	54
Hình 3.2	Cửa sổ Xilinx Platform Studio 14.1	59
Hình 3.3	Tạo tên project Microblaze	59

Hình 3.4	Chọn I world like to create a new design -> next	60
Hình 3.5	Chọn bảng Xilinx Spartan-3E Starter Board	60
Hình 3.6	Chọn cấu hình 2 nhân cho Microblaze	61
Hình 3.7	Chọn đồng hồ và dung lượng nhớ trong cho từng nhân	61
Hình 3.8	Chọn cấu hình các thiết bị cho từng nhân	61
Hình 3.9	Chọn dung lượng 2KB cache cho từng nhân	62
Hình 3.10	Địa chỉ của Microblaze	62
Hình 3.11	Các giao tiếp bus của Microblaze	63
Hình 3.12	Tạo địa chỉ của hệ thống Microblaze	63
Hình 3.13	Các cổng của hệ thống Microblaze	64
Hình 3.14	Sơ đồ mạch của hệ thống Microblaze 2-core	64
Hình 3.15	Sơ đồ mạch của core Microblaze_0	65
Hình 3.16	Sơ đồ mạch của core Microblaze_1	65
Hình 3.17	Các giao tiếp mở rộng I/O của Microblaze	66
Hình 3.18	Tạo thành công Netlist của project Microblaze	66
Hình 3.19	Tạo thành công file cấu hình hệ thống Microblaze "system.bit"	67
Hình 3.20	Kiểm tra các kết nối thiết của PC bằng Device Manager	68
Hình 3.21	Chọn Export & Launch SDK	68
Hình 3.22	Các file cấu hình của thiết kế từ XPS đã được Export vào SDK	69
Hình 3.23	Thiết lập cổng COM-USB	69
Hình 3.24	Cổng COM-USB đã kết nối sau khi thiết lập	70
Hình 3.25	Chọn Program để nạp cấu hình lên FPGA	70
Hình 3.26	Tạo tên project: hello_world_0	70
Hình 3.27	Soạn file helloworld.c	71
Hình 3.28	Kết quả chạy helloworld trên FPGA (trả về PC)	71
Hình 3.29	Tạo ứng dụng memorytest	72
Hình 3.30	Chạy memorytest trên FPGA thành công	72
Hình 3.31	Tạo và biên dịch trình kiểm tra ngoại vi	73
Hình 3.32	Chạy thành công kiểm tra các ngoại vi	73

MỞ ĐẦU

1.1. Lý do chọn đề tài

Thiết kế các mạch điện tử số theo cách truyền thống trở nên khó khăn khi công nghệ vi mạch có mức tích hợp rất lớn(VLSI), phức tạp, và tốc độ cao. Trong năm 1980, Bộ quốc phòng Mỹ (DoD) đã tài trợ dự án chương trình VHSIC (Very high Speed Integrated Circuit) để tạo ra ngôn ngữ mô tả phần cứng chuẩn hóa. Năm 1983, DoD thiết lập các yêu cầu cho ngôn ngữ mô tả phần cứng VHSIC, gọi là VHDL (Very High speed integrated circuit hardware Description Language). Theo các nguyên tắc của IEEE, cứ 5 năm thì một chuẩn phải được đề xuất lại và được tiếp nhận. Theo đó, chuẩn VHDL 1076-1993 ra đời.

Kể từ năm 1980, các nhà công nghệ vi mạch tích hợp hàng đầu thế giới đã đẩy mạnh quá trình nghiên cứu về công nghệ vi mạch tích hợp mảng cổng lập trình được theo trường FPGA (field programmable Gate Array) và nhanh chóng cho ra các thế hệ FPGA với số lượng cổng và tốc độ ngày càng cao. FPGA được thiết kế đầu tiên bởi Ross Freeman, người sáng lập công ty Xilinx vào năm 1984. Các FPGA hiện nay, có số lượng cổng logic (logic gate) đủ lớn để có thể thiết kế thay thế cả một hệ thống bao gồm lõi CPU, Bộ điều khiển bộ nhớ (Memory Controller), các ngoại vi như SPI, Timer, I2C, GPIO, PWM, Video/Audio Controller…, tương đương với các hệ thống trên chip SoC (System on Chip) hiện đại. FPGA có thể được lập trình bằng các ngôn ngữ mô tả phần cứng HDL (Hardware Description Language) như VHDL, hay Verilog. HDL để tạo ra các thiết kế mạch số từ số lượng lớn cổng logic, và có thể cấu trúc lại mạch thiết kế khi đang sử dụng. Như vậy công đoạn thiết kế của FPGA đơn giản, chi phí giảm thiểu, rút ngắn thời gian đưa sản phẩm vào sử dụng. FPGA cũng rất phù hợp cho thiết kế thử nghiệm các hệ thống nhúng phức tạp và thông minh được ứng dụng trong nhiều lĩnh vực như tự động điều khiển, robot, điện tử dân dụng, viễn thông, các thiết bị di động, các phương tiện vận tải, thuật vi xử lý, các thiết bị thám mã, xử lý tín hiệu số, kiến trúc máy tính hiệu năng cao, v.v... Có thể thiết kế lõi mềm vi xử lý 32-bit kiến trúc RISC (Reduced Instruction Set Computer) đơn lõi hoặc lõi (Microblaze, ARM, Nios2,...), mạng trên chip NoC (network on chip), hay system on chip (SoC).

Những năm gần đây, đào tạo lập trình thiết kế hệ thống số bằng ngôn ngữ HDL đã được đưa vào giảng dạy trong nhiều trường đại học kỹ thuật ở nước ta. Các ngôn

ngữ VHDL và Verilog và FPGA trở nên hữu ích rất lớn trong trong đào tạo và nghiên cứu khoa học bậc nghiên cứu sinh, sau đại học, đại học và cả đào tạo nghề.

Do đó, học viên chọn đề tài "Thiết kế hệ vi điều khiển lõi mềm 32-bit trên FPGA và cài đặt ứng dụng".

1.2. Tính cấp thiết của đề tài

Lõi mềm vi xử lý, hay vi điều khiển 32-bit khác với chip vi mạch vi xử lý hay vi điều khiển 32-bit (lõi cứng):

- Lõi mềm có nghĩa là không cố định, có thể bằng lập trình cấu hình lại cấu trúc hay sửa đổi chức năng của vi xử lý (vi điều khiển) tùy biến phụ thuộc vào nhu cầu ứng dụng mong muốn, và có thể tạo ra các thiết kế linh hoạt cho các ứng dụng khác nhau.
- Lõi cứng, chỉ có thể sử dụng sẵn chip vi xử lý (vi điều khiển) do nhà công nghệ cung cấp, người dùng chỉ còn cách hiểu biết qua các mô tả kỹ thuật của nhà công nghệ (đặc tính kỹ thuật, tập lệnh) để thiết kế các hệ thống lớn hơn, và chỉ đáp ứng cố định cho một ứng dụng.

Do tính ứng dụng rộng rãi trong nhiều lĩnh vực, mà FPGA và các ngôn ngữ HDL trở nên cấp thiết trong đầu tư ứng dụng và đào tạo.

2. MỤC TIÊU VÀ PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Mục tiêu của đề tài

- Tìm hiểu một trong ngôn ngôn ngữ mô tả phần cứng là VHDL
- Tìm hiểu công nghệ FPGA
- Tìm hiểu một loại vi điều khiển lõi mềm 32-bit kiến trúc tập lệnh giảm thiểu (RISC)
- Thiết kế được vi xử lý lõi mềm bằng công cụ phần mềm thiết kế dựa vào HDL
- Xây dựng một ứng dụng phần mềm thử nghiệm hoạt động của vi điều khiển lõi mềm.

2.2. Nội dung nghiên cứu

- Công nghệ FPGA
- Ngôn ngữ lập trình VHDL
- Vi điều khiển Microblaze 32-bit kiến trúc tập lệnh rút gọn RISC
- Công cụ phần mềm phát triển Xilinx ISE 14.1 dùng cho thiết kế các hệ thống số trên Xilinx FPGA

- Các bước thiết kế vi điều khiển Microblaze 32-bit nhờ sử dụng ISE 14.1
- Xây dựng phần mềm ứng dụng trên ngôn ngữ C và cài đặt thử nghiệm hệ vi điều khiển đã cấu hình trên FPGA

2.3. Phương pháp luận và phương pháp nghiên cứu

2.3.1. Phương pháp luận

Dựa vào các phương pháp chuyên môn:

- Kỹ thuật điện tử và điện tử số
- Kỹ thuật vi xử lý
- Ngôn ngữ lập trình
- Thiết kế các hệ thống số bằng ngôn ngữ mô tả phần cứng

2.3.2. Phương pháp nghiên cứu

- Khảo sát và đánh giá các công trình nghiên cứu, các tài liệu kỹ thuật liên quan với đề tài
- Lựa chọn công nghệ FPGA của nhà công nghệ (Xilinx, Altera) cho đề tài
- Lựa chọn ngôn ngữ lập trình thiết kế hệ thống số (VHDL, Verilog) cho đề tài
- Trên cơ sở các mục tiêu của đề tài xây dựng kế hoạch thực hiện đề tài của luận văn, đánh giá kết quả thực hiện
- Dựa trên các yêu cầu và đánh giá của giáo viên hướng dẫn, thực hiện các chỉnh sửa, và hoàn chỉnh luận văn.

3. BỐ CỤC DỰ KIẾN CỦA LUẬN VĂN

Nội dung luận văn gồm các chương và phần chính sau:

Chương 1: Công nghệ FPGA

Chương 2: Thiết kế phần cứng bằng VHDL

Chương 3: Thiết kế hệ vi điều khiển lõi mềm Microblaze 32-bit và đặt ứng dụng thử nghiệm

Kết luận chung và hướng nghiên cứu.

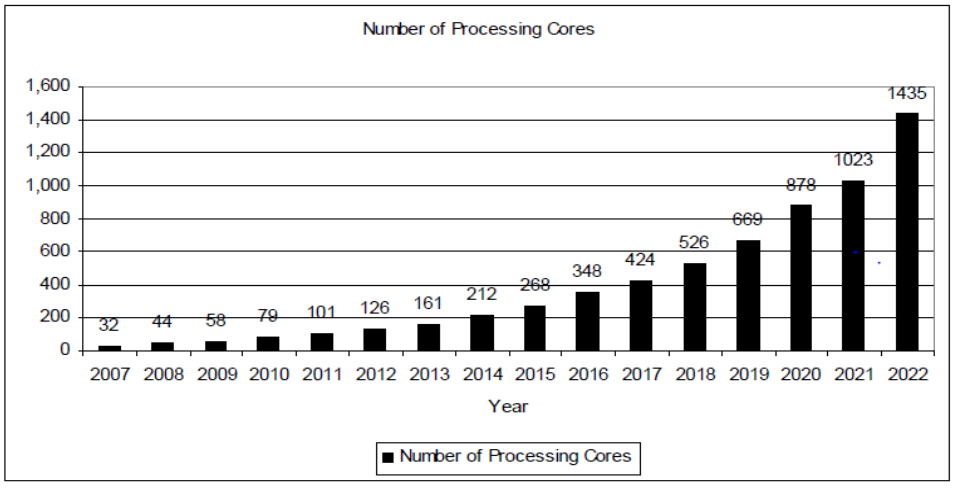
CHƯƠNG 1: CÔNG NGHỆ FPGA

1.1. PHÂN LOẠI CÁC VI MẠCH TỔ HỢP

1.1.1. Tổng quan phát triển các mạch tích hợp

Một mạch tích hợp IC (thường được gọi là vi mạch) là một mạch điện tử được sản xuất bằng sự khuếch tán theo khuôn mẫu của các thành phần nhỏ lên bề mặt của màng tinh thể silicon. Các vật liệu bổ xung lắng đọng và được định khuôn hình thành các liên kết giữa các thiết bị bán dẫn. Các mạch tích hợp được làm trên màng silicon tròn mỏng (vài trăm microns), mỗi màng silicon chứa hàng trăm mảnh IC nhỏ (IC die). Các transistors và dây nối được làm từ nhiều lớp (thường có từ 10 đến 15 lớp) chồng lên nhau. Mỗi một lớp tiếp theo có một mẫu được xác định nhờ dùng một mặt nạ tương tự như mảnh in chụp trong suốt. Sáu lớp đầu tiên (từ dưới lên) xác định các transistors. Các lớp trên còn lại xác định các dây nối bằng kim loại giữa transistors, gọi là các liên kết (interconnect) để tạo nên mạch tích hợp trên từng mảnh nhỏ (die).

Kể từ mạch tích hợp đầu tiên do Jack Kilby (Texas Instrumens) phát minh ngày 06/12/1959, sự phát triển của công nghệ từ cấp độ tích hợp nhỏ SSI (Small-Scale Integration) với một số ít bóng bán dẫn, đến cấp độ tích hợp siêu lớn ULSI (Ultra-Large-Scale Integration) với hàng triệu đến vài tỷ bóng bán dẫn trên một chip. Mật độ tích hợp các bóng bán dẫn trên vi mạch phù hợp với Định Luật Moore với một chu kỳ khoảng 18 tháng mật độ tích hợp các bóng bán dẫn trong diện tích 2.54 cm^2 trên chip tăng gấp đôi, và do đó mức độ tích hợp nhân xử lý trên chip đa xử lý CMP (chip multiprocessor) hay chip đa nhân (chip multicore) cũng tăng lên gấp 1.3 lần hàng năm (hình 1).



Hình 1.1: Phát triển số lượng nhân xử lý trên chip

Nguồn: ITRS,"The international technology roadmap for semiconductors: 2007".

Số lượng bóng bán dẫn (transistor) trong một chip vi mạch xác định cấp độ tích hợp của các vi mạch:

SSI (small-scale integration): có tới 100 transistors trong chip

MSI (medium-scale integration): từ 100 đến 3,000 transistors trong chip

LSI (large-scale integration): từ 3,000 đến 100,000 transistors trong chip

VLSI (very large-scale integration): từ 100,000 đến 1,000,000 transistors trong chip.

ULSI (ultra large-scale integration): từ vài triệu đến vài tỷ transistors trong chip

Năm 2012 Intel công bố chip đa nhân 62-core XeonPhi quá trình 22 nm với 5 tỷ transistors, nhưng Nvidia đã giữ "kỷ lục thế giới" với chip xử lý đồ họa GPU (graphics processing unit) chứa 7.08 tỷ transistors.

Có hai công nghệ lưỡng cực mà các vi mạch sử dụng: TTL (Transistor-Transistor Logic), CMOS (Complematary Metal-Oxide-Semiconductor).

Các vi mạch TTL được xây dựng từ các transistor nối lưỡng cực BJT (bipolar junction transistors) và các điện trở (resistors). Chúng tiêu thụ công suất nguồn nuôi cao (+5v), tản nhiệt lớn. một mạch cổng trong chip TTL tiêu thụ khoảng 10mW, vì vậy không thể cho cấp độ tích hợp cao được. Tuy nhiên các BJT lại có ưu điểm là cho tốc độ cao, hệ số khuếch đại lớn, và trở kháng ra thấp, nên chúng rất phù hợp cho chế tạo các mạch tương tự, như các bộ khuếch đại công suất lớn.

Các vi mạch CMOS sử dụng cả hai loại transistor hiệu ứng trường, FET (Field Effect Transistor) n và p (nMOSFET, pMOSFET), trong đó một loại transistor được dùng làm điện trở. Transistor FET có trở kháng vào lớn, và tại một thời điểm chỉ có một loại transistor ở trạng thái ON. Vì vậy ở trạng thái tĩnh, CMOS tiêu thụ điện năng rất thấp. Chúng chỉ tiêu thụ điện năng đáng kể khi các transistor chuyển trạng thái giữa ON và OFF. Một mạch cổng trong chip CMOS tiêu thụ thấp chỉ 10nW. Vì vậy, CMOS cho cấp độ tích hợp cao, và chùng được ứng dụng phổ biến để chế tạo các chip vi xử lý, vi điều khiển, xử lý tín hiệu số (DSP), các chip CPU cho các thiết bị mobile,…Trễ lan tuyền trong khoảng 25ns đến 50ns. Mức tiêu thụ điện năng của CMOS phụ thuộc nhiều vào tần số của nhịp đồng hồ. Tần số càng lớn thì điện năng tiêu thụ càng cao. nên cho phép xây dựng các mạch cổng tiêu thụ điện năng thấp. CMOS có khả năng miễn nhiễu cao. Công nghệ CMOS được sử dụng để chế tạo các chip vi xử lý, vi điều khiển, xử lý tín hiệu số, các bộ nhớ SRAM, các chip CPU cho các thiết bị mobile,…

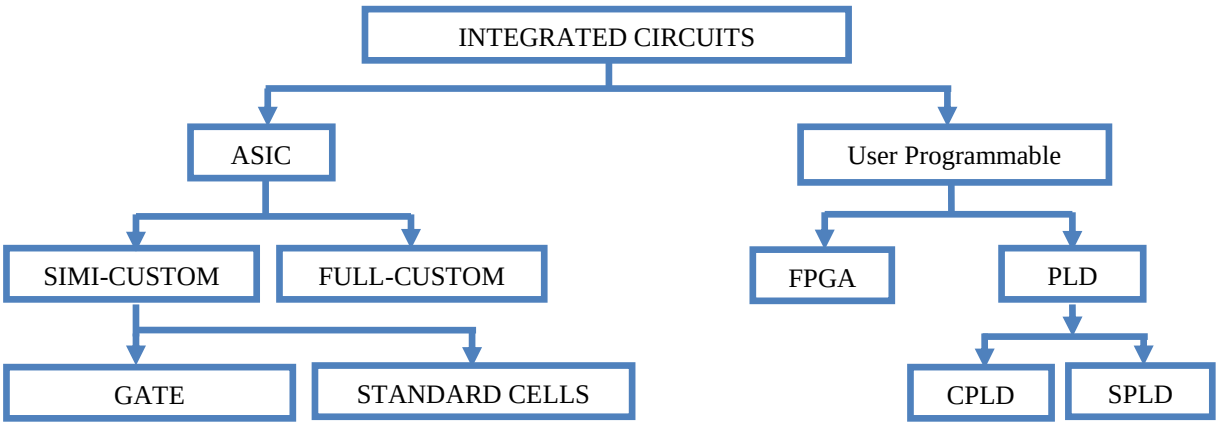
Công nghệ BiCMOS là sự kết hợp hai loại công nghệ lưỡng cực (bipolar) và CMOS để tận dụng ưu điểm của cả TTL và CMOS. Ví dụ, các mạch tích hợp sử dụng

công nghệ của BiCMOS là các bộ tạo dao động tần số radio, các chip vi xử lý Pentium, Pentium Pro, SuperSPARC.

Với các cấp độ tích hợp của các chip IC thì công nghệ đóng vỏ cũng thay đổi. Có nhiều kiểu đóng vỏ: các chip SSI có kiểu đóng vỏ gồm hai hàng chân, DIP (Dual-in-line package), có số lượng cân cắm (pin) không nhiều. Những các chip VLSI, ULSI thì kiểu DIP không phù hợp.

1.1.2. Các mạch tích hợp ứng dụng chuyên biệt (ASIC)

Các mạch tích hợp là các sản phẩm công nghiệp hàng loạt được chế tạo theo chủ quan của các nhà sản xuất. Mặc dù chúng thỏa mãn nhiều ứng dụng khác nhau và rất đa năng, nhưng đối với rất nhiều ứng dụng trong nhiều lĩnh vực khác nhau thì chúng không thể thỏa mãn được đầy đủ, hiệu năng không cao. Do đó, từ những năm đầu 1980 các nhà sản xuất chip đã thực hiện thiết kế chế tạo các mạch tích hợp có khả năng thiết kế cho các ứng dụng chuyên biệt, không phải cho mục đích sử dụng chung, những mạch này được gọi là ASIC (Application-Specific Integrated Circuit).



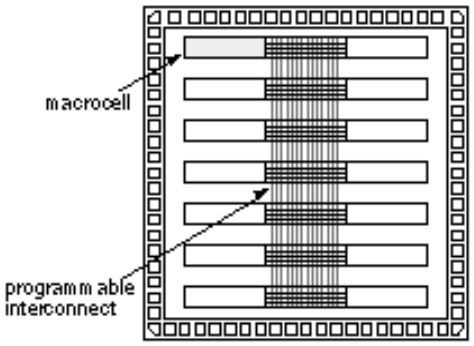
Hình 1.2: Phân loại các mạch tích hợp

Tuy nhiên, đối với nhiều người dùng cá nhân thì chi phí để trả cho thiết kế chế tạo (tại nhà máy) các ASIC là rất cao. Nên một số nhà cung cấp đã tạo ra khả năng cho người sử dụng cá nhân, đó là cung cấp các mạch tích hợp mà người sử dụng có thể tự

thiết kế và lập trình các vi mạch để tạo ra các thiết bị phù hợp nhất cho nhu cầu chuyên dụng của mình. Các mạch tích hợp đó được gọi là các mạch tích hợp có thể lập trình được cho người sử dụng (user programmable). Như vậy xét theo tính có thể lập trình được cho người sử dụng, các vi mạch có thể được chia thành hai loại chính, đó là ASIC và User Programmable (hình 1.2).

1.1.3. Các thiết bị logic có thể lập trình được (PLD)

Các thiết bị logic có thể lập trình được PLD (Programmable Logic Device) là các mạch tích hợp chuẩn sẵn có trong các cấu hình chuẩn từ các catalog và được bán với số lượng lớn cho nhiều người dùng. Tuy nhiên, các PLD có thể được cấu hình hoặc được lập trình để tạo một phần cho ứng dụng chuyên biệt, và như vậy chúng có thể được coi là thuộc họ các ASIC. Nhưng PLD sử dụng các công nghệ khác so với ASIC để lập trình.



Hình 1.3: Mảnh PLD.

Các macrocells chứa logic mảng có thể lập trình được kèm theo một flip-flop hoặc một mạch chốt. Các macrocells được kết nối sử dụng khối liên kết có thể lập trình được lớn

PLD là một dạng đặc biệt của mảng cổng, và nó có những đặc tính quan trọng như: các ô logic và các lớp mặt nạ không thể theo ứng dụng chuyên biệt, chuyển thiết kế nhanh chóng, một khối lớn của liên kết có thể lập trình được, ma trận các ô logic lớn (logic megacell) thường chứa logic mảng có thể lập trình được kèm theo một flip-flop hoặc mạch chốt. Có hai loại PLD: SPLD (Simple PLD), CPLD (Compled PLD).

1.2. FPGA

1.2.1. Kiến trúc FPGA

FPGA (Field-Programmable Gate Array) là vi mạch dùng cấu trúc mảng các phần tử logic mà người dùng có thể lập trình được. Chữ "Field" ở đây chỉ khả năng tái lập trình "bên ngoài" của người sử dụng, không phụ thuộc vào dây truyền sản xuất phức tạp của nhà máy bán dẫn). FPGA thiết kế đầu tiên bởi Ross Freeman, người sáng lập công ty Xilinx năm 1984. Công nghệ hiện nay của FPGA tich hợp số lượng lớn các phần tử logic cho phép thiết kế các hệ thống thiết bị trên chip phức tạp như: hệ thống trên chip (SoC), hệ thống nhúng (với vi xử lý và hệ điều hành nhúng), các hệ thống xử lý tín hiệu và điều khiển ứng dụng trong hàng không vũ trụ, quốc phòng, viễn thông,

công nghiệp sản xuất đồ điện tử gia dụng, các hệ thống máy tính tốc độ cao chuyên dụng (mặt mã, nhận dạng tiếng nói,..)., đặc biệt làm tiền thiết kế mẫu cho ASIC (ASIC prototyping).

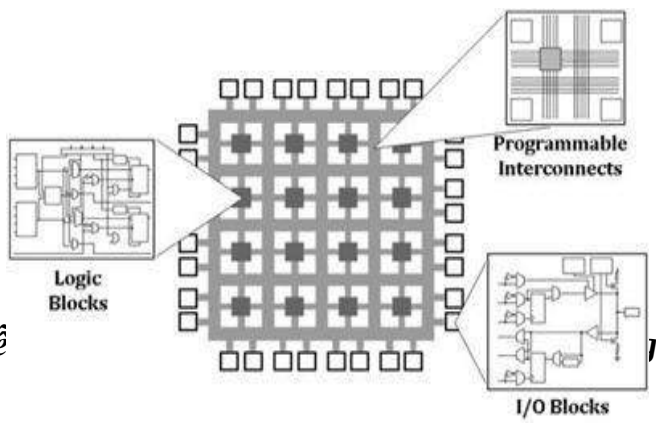
Có hai phương pháp lập trình FPGA: lập trình dựa trên SRAM (SRAM programming) và lập trình dựa trên anti-fuse (Anti-fuse programming). Do đó, có hai loại FPGA trên thị trường hiện nay: FPGA dựa trên SRAM, và FPGA dựa trên anti-fuse. Trong loại FPGA với lập trình dựa trên SRAM, Xilinx và Altera là hai nhà sản xuất hàng đầu xét theo số lượng người dùng. Cạnh tranh chính là AT&T. Đối với loại FPGA với lập trình dựa trên anti-fuse, Actel, Quicklogic, Cypress, và Xilinx là những nhà sản xuất cạnh tranh.

Lập trình dựa trên SRAM cần ít bit của SRAM cho từng phần tử lập trình. Sự ghi bit 0 làm tắt chuyển mạch (turns off a switch), trong khi sự ghi bit 1 bật chuyển mạch (turns on a switch). Đối với phương pháp thứ hai, khi lập trình, dòng lập trình tạo ra kết nối cầu chì anti-fuse (cầu trì bình thường có kết nối sẵn).

FPGA dựa trên SRAM: do các SRAM của FPGA có thể được ghi đọc như SRAM bình thường ngay cả khi chúng ở trong hệ thống, nên các FPGA có thể được lập trình lại nhiều lần. Tuy nhiên trễ định tuyến lớn trong các FPGA lại này. Loại FPGA này thường được sử dụng cho lập trình cấu hình các mẫu thử của các thiết kế phần cứng trên ASIC.

FPGA dựa vào anti-fuse: duy trì cố định nội dung lập trình ngay cả khi mất nguồn (non-volatile), và trễ định tuyến nhỏ. Tuy nhiên chúng yêu cầu một quá trình sản xuất phức tạp, và nếu đã lập trình xong một lần thì không thể thay đổi được nữa.

Kiến trúc chung của FPGA dựa trên SRAM cơ bản gồm có (hình 1.4): CLB (configurable logic block) - các khối logic có thể cấu hình được, IOB (Input/output block) - các khối vào ra có thể cấu hình được, Programmable interconnect hay routing - mạng liên kết có thể lập trình được, các khối RAM (Block RAM). Ngoài ra có thể còn có các mạch điều khiển các tín hiệu đồng hồ số (DCM – Digital Clock Management) phân phối cho từng khối logic và khối vào ra, các khối mạch logic bổ xung như các ALU, memory, các bộ ghép kênh (MUX), các thanh ghi dịch, các mạch giải mã.



Hình 1.4: Kiến trúc của FPGA dựa trên SRAM

CLBs (configurable Logic Blocks): các khối logic được sắp xếp theo ma trận dọc và ngang đều nhau, và chúng có thể cấu hình được. Các CLB là tài nguyên chính của FPGA. Mỗi CLB có các LUT (Look-up table). Ví dụ trong Xlinx FPGA của họ Spartan-3E, mỗi CLB có 8 LUT. Mỗi LUT cơ bản có 4 input và 1 output. LUT có thể được lập trình tạo ra các mạch tổ hợp logic, RAM phân tán, thanh ghi dịch,v.v…Trong hầu hết các FPGA, mỗi một CLB chứa một số các mảnh, mà mỗi mảnh lại chứa một số (thường là 2 hoặc 4) ô logic (logic cell) với một số thành phần nhớ (Flip-Flop) hoặc bộ ghép kênh (Multiplexer) nếu không dùng FF. Mỗi ô logic có thể được cấu hình để thực hiện các chức năng logic cơ bản (như AND, OR, NOT) trên các tín hiệu số nhờ sử dụng bảng LUT (look-up Table). Các CLB liên kết với nhau qua mạng liên kết có thể lập trình được (Programmable Interconnect hay routing).

Progammable Interconnect (hay Routing): mạng liên kết hay định tuyến được lập trình, là các mạng các đường dây nối dọc theo sắp xếp của các CLB với các chuyển mạch có thể lập trình tạo tại các nút giao tiếp các đường ngang và dọc để tạo các kết nối các khối logic. Tùy thuộc vào công nghệ FPGA của nhà sản xuất, mạng liên kết lập trình này có thể có các cấu trúc khác nhau (chúng ta sẽ xét ở sau đây). Lập trình định tuyến kết nối trên FPGA là một công đoạn phức tạp, nhưng được các công cụ thiết kế của các nhà sản xuất FPGA thực hiện tự động theo thiết kế của người dùng. *IOBs (Input/Output Blocks):* các khối vào/ra nằm bao xung quanh của miếng FPGA và nối với các chân tín hiệu vào/ra (I/O pin). Như vậy từng chân I/O của FPGA có thể được lập trình để đảm bảo các giao tiếp điện cần thiết cho kết nối FPGA với hệ thống mà nó là thành phần.

Block RAM: khối RAM, là RAM có dung lượng vài kilobits (trong Xilinx FPGA, block Ram 2-port), được nhúng ở vị các vị trí cố định trên FPGA để lưu trữ dữ liệu (không được sử dụng để thực hiện các chức năng logic khác).

Ngoài các thành phần trên, FPGA còn các logic nhỏ khác, như:

MAC (Multiply-accumulate circuits): các khối logic nhân tích lũy, để thực hiện các phép nhân và cộng hiệu quả.

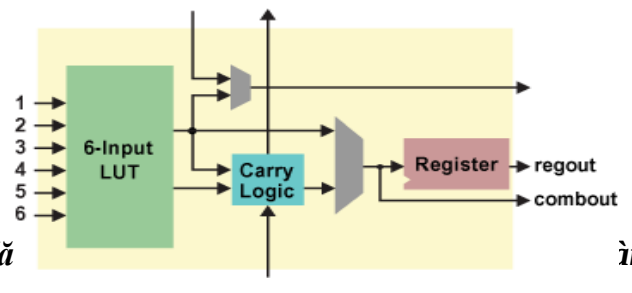
DCM (Digital Clock Manager) (có trong Xilinx FPGA): quản lý đồng hồ số. Trong Xilinx PGA thực hiện lặp khóa trễ (delay locked loop), tổng hợp tần số số (digital frequency synthesizer), dịch pha số (digital phase shifter), hoặc tải phổ số (fdigital spread spectrum). Các khối DCM được đặt xung quanh trên FPGA để cho EDK tool suite sử dụng).

Các khối thực hiện các chức năng đặc biệt: xử lý tín hiệu số và tương tự, ví dụ các bộ biến đổi tương tự-số ADC (Analog-to-Digital Converter) và các bộ biến đổi số-tương tự DAC (Digital-to-Analog Converter), cho phép FPGA vận hành như là một SoC. Một FPGA chứa từ 64 đến hàng chục ngàn khối logic và các flip-flop.

LUT giống như một RAM nhỏ, cũng được gọi là các bộ tạo chức năng, FG (Function generator), được sử dụng để thực hiện các chức năng logic nhờ cất giữ trạng thái logic ra đúng ở trong một vùng nhớ, mà trạng thái logic ra tương ứng với từng tổ hợp của các biến vào. LUT thường có 4 đầu vào có thể thực hiện bất kỳ chức năng logic 4-đầu vào.

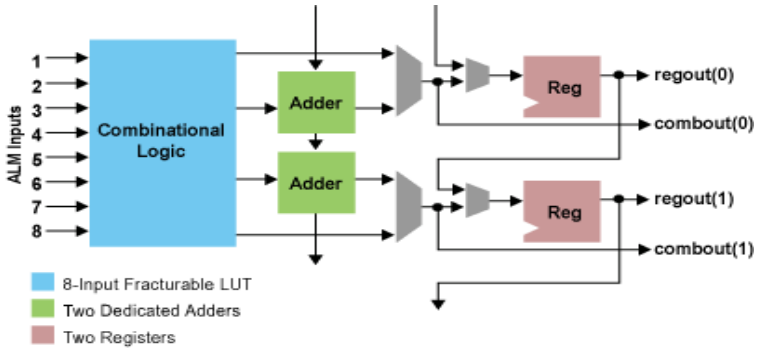
Các nhà sản xuất có xu hướng thiết kế các khối logic của FPGA thực hiện các chức năng lớn hơn để giảm liên kết cục bộ, đồng nghĩa với số lượng chân tín hiệu đầu vào của khối logic tăng lên, và nó cũng cho phép lập trình các khối logic linh hoạt hơn.

Ví dụ, Xilinx có kiến trúc Virtex-5 FPGA dựa trên cặp LUT 6-đầu vào với tổng số 64 bits của không gian lập trình và 6 đầu vào độc lập, và logic liên quan đảm bảo ưu việt trong sử dụng các tài nguyên so với các kiến trúc khác. Nó có thể thực hiện bất kỳ chức năng nào từ 6 đầu vào độc lập và các tổ hợp số của một hoặc hai chức năng nhỏ. LUT 6-đầu vào cũng bao gồm cả các bộ cộng (adder) với logic carry, các bộ dồn kênh (MUX), và flip-flop. Nó có thể được sử dụng bổ xung như là RAM 64-bit hay thanh ghi dịch 32 bit (hình 1.5).



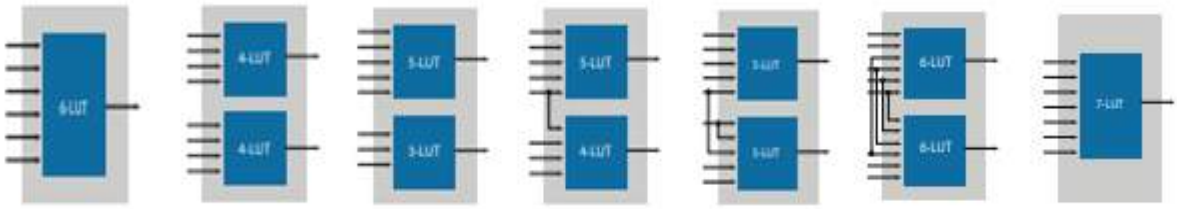
Hình 1.5: Xilinx Virtex-5 FPGA LUT- cặp FF

Kiến trúc của họ Altera Stratix FPGA đạt được hiệu năng cao nhờ đưa vào module logic thích ứng hiệu quả vùng - ALM (Adaptive logic Module). ALM gồm có logic tổ hợp, 2 thanh ghi, và 2 bộ cộng, như chỉ ra ở hình 1.6. logic tổ hợp có 8 đầu vào một bảng LUT (Lookup Table).



Hình 1.6: Altera Stratix IV FPGA ALM

Bảng LUT có thể được chia ra 2 ALUT (Adaptive LUT) với tổng số 64 bits của không gian lập trình và 8 đầu vào chia sẻ. Nó cũng có thể thực hiện bất kỳ chức năng nào của 6 đầu vào và các tổ hợp số của một hoặc hai chức năng nhỏ.. Họ Stratix của các FPGA cũng có hiệu quả trong định tuyến thông qua mạng liên kết MultiTrack™ . Các loại LUT của Stratix ALM (hình 1.7) cho phép đấu nối linh hoạt để tạo các các khối logic chức năng lớn hơn cho FPGA.



Hình 1.7: Các loại LUT của Stratix ALM

Các FPGA khác nhau có số lượng các ô logic, kích cỡ và số lượng các block RAM, các MAC khác nhau. Các FPGA sử dụng trong các hệ thống lai (hybrid system) thường có khoảng 100K-200K ô logic, 500KB của RAM bên trong và 100 MACs. Hệ

thống lai có thể sử dụng FPGA với 1000 khối I/O tương ứng với 1000 I/O pin để đảm bảo các giao tiếp với hệ thống chủ, cũng như với bộ nhớ cục bộ nối trực với FPGA.

Các FPGA thường được lập trình sau khi đã hàn gắn trên bảng mạch in, tương tự như các CPLD lớn. Nhưng dữ liệu cấu hình trong FPGA bị mất khi ngừng cấp nguồn (mất điện) giống như RAM trong máy tính vậy. Do đó, muỗi lần ngắt nguồn và bật lại thì ta phải nạp lại tệp cấu hình vào FPGA. Muốn lưu giữ lại cấu hình đã lập trình cho FPGA thì ta phải mắc thêm PROM hay EPROM ngoài. Bộ nhớ ngoài này có nhiệm vụ lưu tệp cấu hình ở dạng nhị phân (bitstream hay bit file) và tự động nạp dữ liệu cấu hình lại cho FPGA mỗi khi bật nguồn, như vậy dù có ngắt nguồn FPGA vẫn "không bị mất" dữ liệu. Các phiên bản EEPROM có thể có thể lập trình được trong hệ thống (hay trong mạch), thường thông qua giao tiếp JTAG. Tệp cấu hình chứa các thiết lập cho từng CLB, PSM, MAC, I/O và các thành phần có thể cấu hình khác của FPGA. Các FPGA được sử dụng trong các hệ thống máy tính lai có thể được lập trình lại vô số lần. Thời gian tải cấu hình mới thường chỉ chưa đến 1 giây. Một số FPGA hiện nay có khả năng trong khi đang hoạt động chuyển đến cấu hình mới đã được nạp trước vào thiết bị. Một số FPGA cũng cho phép cầu hình lại từng phần của thiết bị.

FPGA và CPLD có những điểm khác biệt đó là: FPGA bên trong dựa trên các bảng look-up (LUTs), trong khi các CPLD hình thành các chức năng logic bằng các nhiều mạch cổng (ví dụ tổng các tích); FPGA và CPLD đều cấu tạo từ các khối logic (các ô logic) là sự kết hợp của một khối logic và Flip-Flop. Nhưng, FPGA có số lượng lớn các khối logic (đến hang trăm ngàn) hơn nhiều so với CPLD; FPGA giống như RAM, phải nạp lại dữ liệu cấu hình mỗi khi bật nguồn. CPLD giống như EEPROM chỉ cần nạp một lần và không bị mất chức năng sau khi ngắt nguồn;

Do FPGA có số lượng rất lớn các khối logic nên có nhiều tài nguyên để thực hiện nhiều chức năng toán học chuyên dụng và phức tạp.Vì vậy các FPGA phù hợp cho các thiết kế phức tạp hơn so với CPLD. Nhìn chung các CPLD là sự lựa chọn tốt cho các ứng dụng tổ hợp, trong khi các FPGA phù hợp hơn cho các máy trạng thái lớn (như các vi xử lý).

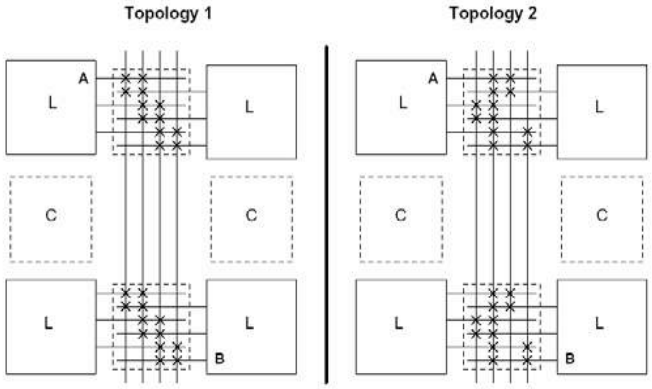
FPGA có các phần tử logic chạy theo dạng song song. Còn vi điều khiển dựa trên cấu trúc CPU thực thi theo mã lệnh theo dạng tuần tự.

FPGA dùng ngôn ngữ lập trình phần cứng (Verilog, VHDL) và lập trình trên FPGA gọi là lập trình phần cứng. Lập trình vi điều khiển là lập trình phần mềm phần cứng có sẵn.

1.2.2. Định tuyến trong FPGA

Định tuyến trong FPGA được thực hiện nhờ lập trình (hay cấu hình) cho các khối kết nối (Connection Block) hay các khối chuyển mạch (Switch Block). Các khối này nằm ở giao của các đường dây nối ngang và dọc giữa ma trận các khối logic (BLK).

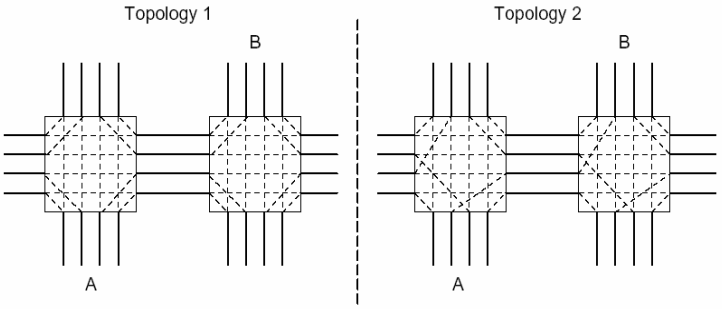
Các khối kết nối (Connection Block): các khối kết nối C nối các dây dẫn của kênh định tuyến với các chân tín hiệu của các CLB. Có hai đặc tính ảnh hưởng chính đến khả năng định tuyến của thiết kế: tính linh hoạt, *Fc*, là số dây dẫn mà từng tín hiệu của CLB có thể kết nối; và cấu hình, là mẫu của các chuyển mạch tạo lập kết nối (đặc biệt nếu giá trị *Fc* thấp).



Hình 1.8: Định tuyến qua các hộp kết nối

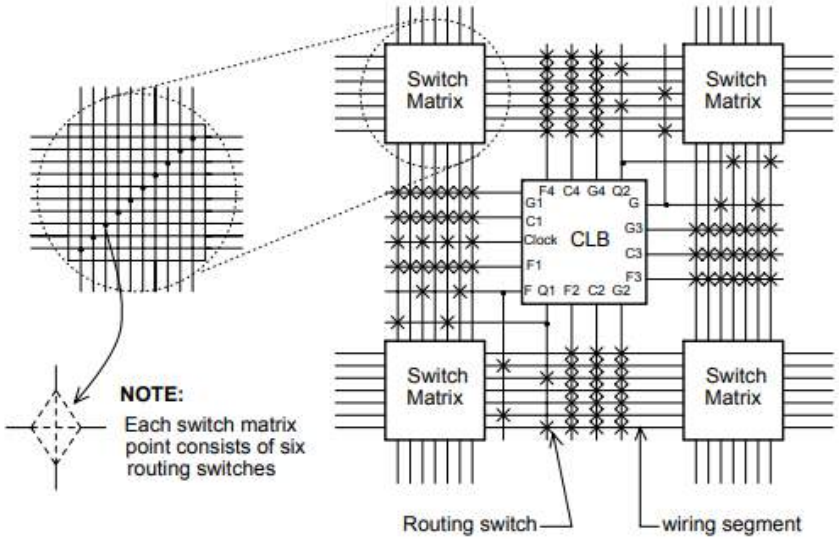
Ví dụ, trong hình 1.8, đối với hộp C với Fc=2, cấu hình 1 (topology 1) không thể nối chân A của một CLB với chân B của CLB khác, trong khí đó thì ở cấu hình 2 (Topology 2) là có thể.

Các khối chuyển mạch (Switch Block): các khối chuyển mạch S cho phép các dây dẫn chuyển mạch giữa các dây dọc và ngang . Tính linh hoạt, Fs, xác định số lượng các đoạn dây nối mà một đoạn dây nối đi vào trong khối S có thể kết nối. Cấu hình của các khối chuyển mạch S là rất quan trọng bởi vì có thể chọn hai cấu hình khác nhau có các khả năng định tuyến khác nhau với cùng một giá trị tính linh hoạt Fs. Ví dụ, hình 1.9 mô tả cấu hình 1 (topology 1) có thể nối chân tín hiệu A của một CLB với chân tín hiệu B của một CLB khác, trong khi đó cấu hình 2 (Topology 2) thì không thể.



Hình 1.9: Định tuyến qua các khối chuyển mạch

Một ma trận chuyển mạch có thể có đến 6 chuyển mạch định tuyến, nhờ đó, chúng cho phép linh hoạt kết nối định tuyến theo các đường dây dài đơn (Single-Length Line) (hình 1.10) cho trễ tín hiệu nhỏ.



Hình 1.10: Định tuyến theo các ma trận chuyển mạch và
các đường dây dài đơn trong Xilinx FPGA

Các đường dây nối trong Xilinx FPGA (hình 1.11) gồm:

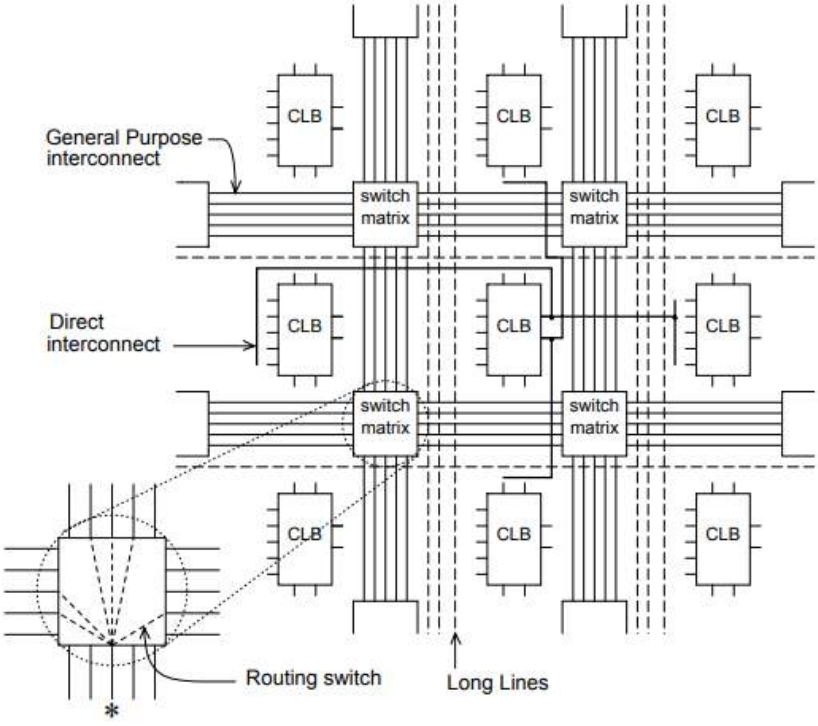
Các đường dây độ dài-đơn (Single-length lines): chúng được dùng cho các kết nối tương đói ngắn giữa các CLB và chúng trải rộng chỉ qua một CLB.

Các đường dây độ dài-gấp đôi (Double-length lines): chúng tương tự như các đường dây dài-đơn, ngoại trừ mỗi đường dây ở đây trải rộng qua hai CLB. Chúng cho các trễ định tuyến nhỏ đối với kết nối dài vừa phải.

Các đường dây dài (Long lines): chúng phù hợp cho các kết nối dài trải rộng một số CLB trong FPGA.

Hình 1.11 minh họa về định tuyến trong một Xilinx FPGA. Các liên kết của kênh định tuyến với khối logic (LB) được tạo ra thông qua khối kết nối, CB (Connection Block). Vì công nghệ SRAM được sử dụng để thực hiện các LUT, nên các phía kết nối là rộng. Khối logic được vây quay bởi các khối kết nối ở cả bốn phía kết nối. Các CB nối nối các chân tín hiệu (pin) của LB với các đoạn dây. Các chân tín hiệu của LB, mà chúng nối với các CB có thể sau đó nối với bất kỳ số lượng của các đoạn

dây thông qua các khối chuyển mạch, SB (Switch Block). Trong cấu hình này có bốn loại đoạn dây: các đoạn dây có mục đích chung (General purpose Interconnect): các dây lại này đi qua các chuyển mạch trong SB; liên kết trực tiếp (Direct Interconnect): kết nối các chân tín hiệu của LB với bốn khối kết nối xung quanh LB; đường dây dài (Long Line): là các dây nối thống nhất có hệ số trễ phân đầu ra cao; và các đường dây nhịp đồng hồ (Clock lines): dẫn tín hiệu nhịp đồng hồ đến tất cả các chip.



Hình 1.11: Kiến trúc định tuyến của Xilinx FPGA

1.3. Phương pháp lập trình FPGA

Có hai phương pháp lập trình FPGA: lập trình dựa trên SRAM (SRAM programming) và lập trình dựa trên anti-fuse (Anti-fuse programming). Do đó, có hai loại FPGA trên thị trường hiện nay: FPGA dựa trên SRAM, và FPGA dựa trên anti-fuse. Trong loại FPGA với lập trình dựa trên SRAM, Xilinx và Altera là hai nhà sản xuất hàng đầu xét theo số lượng người dùng. Cạnh tranh chính là AT&T. Đối với loại FPGA với lập trình dựa trên anti-fuse, Actel, Quicklogic, Cypress, và Xilinx là những nhà sản xuất cạnh tranh.

1.3.1. Lập trình dựa vào bộ nhớ SRAM (Static Random Access Memory)

Lập trình dựa trên SRAM cần ít bit của SRAM cho từng phần tử lập trình. Sự ghi bit 0 làm tắt chuyển mạch (turns off a switch), trong khi sự ghi bit 1 bật chuyển mạch

(turns on a switch). Đối với phương pháp thứ hai, khi lập trình, dòng lập trình tạo ra kết nối cầu chì anti-fuse (cầu trì bình thường có kết nối sẵn).

FPGA dựa trên SRAM: do các SRAM của FPGA có thể được ghi đọc như SRAM bình thường ngay cả khi chúng ở trong hệ thống, nên các FPGA có thể được lập trình lại nhiều lần. Tuy nhiên trễ định tuyến lớn trong các FPGA lại này. Loại FPGA này thường được sử dụng cho lập trình cấu hình các mẫu thử của các thiết kế phần cứng trên ASIC.

1.3.2. Lập trình dựa vào đốt cầu chì (anti-fuse)

FPGA dựa vào anti-fuse: duy trì cố định nội dung lập trình ngay cả khi mất nguồn (non-volatile), và trễ định tuyến nhỏ. Tuy nhiên chúng yêu cầu một quá trình sản xuất phức tạp, và nếu đã lập trình xong một lần thì không thể thay đổi được nữa.

1.4. SO SÁNH FPGA VỚI CÁC LOẠI VI MẠCH TÍCH HỢP KHÁC

1.4.1. FPGA và ASIC.

Công nghiệp FPGA chiếm 15% của công nghiệp ASIC về khối lượng và doanh thu.

Các ASIC có hiệu năng cao hơn, dung lượng cao hơn, nguồn nuôi thấp hơn, tích hợp các tín hiệu, và hiệu quả cao hơn so với các FPGA. Các thiết kế của FPGA tiêu thụ nhiều nguồn hơn so với các ASIC

Nếu sản xuất công nghiệp với số lượng lớn, thì ASIC cho chi phí trên một đơn vị thấp hơn các FPGA.

Nếu ta biết rằng có nhiều thiết bị và các máy tính cá nhân chạy ở tốc độ vài Gigaherz, thì các FPGA lại chạy với tốc độ thấp ở (vài trăm Megaherz). Trong khi đó các ASIC có thể tốc độ cao hơn FPGA nhiều.

FPGA hiệu quả cho các ứng dụng nhỏ bởi vì nó có chi phí thiết kế thấp. Nhưng nếu sản xuất công nghiệp với số lượng lớn, thì ASIC lại có giá rẻ hơn nhiều.

Quá trình thiết kế FPGA đơn giản và thời gian ngắn hơn so với quá trình thiết kế ASIC, bởi vì không cần phải sắp xếp linh kiện, không cần các mặt mạ hoặc các quá trình sau-cuối (back-end processes).

ASIC có thể có các thiết kế xử lý hỗn hợp các tín hiệu, hoặc chỉ là các thiết kế tương tự. Nhưng không thể thiết kế ASIC sử dụng các chip FPGA.

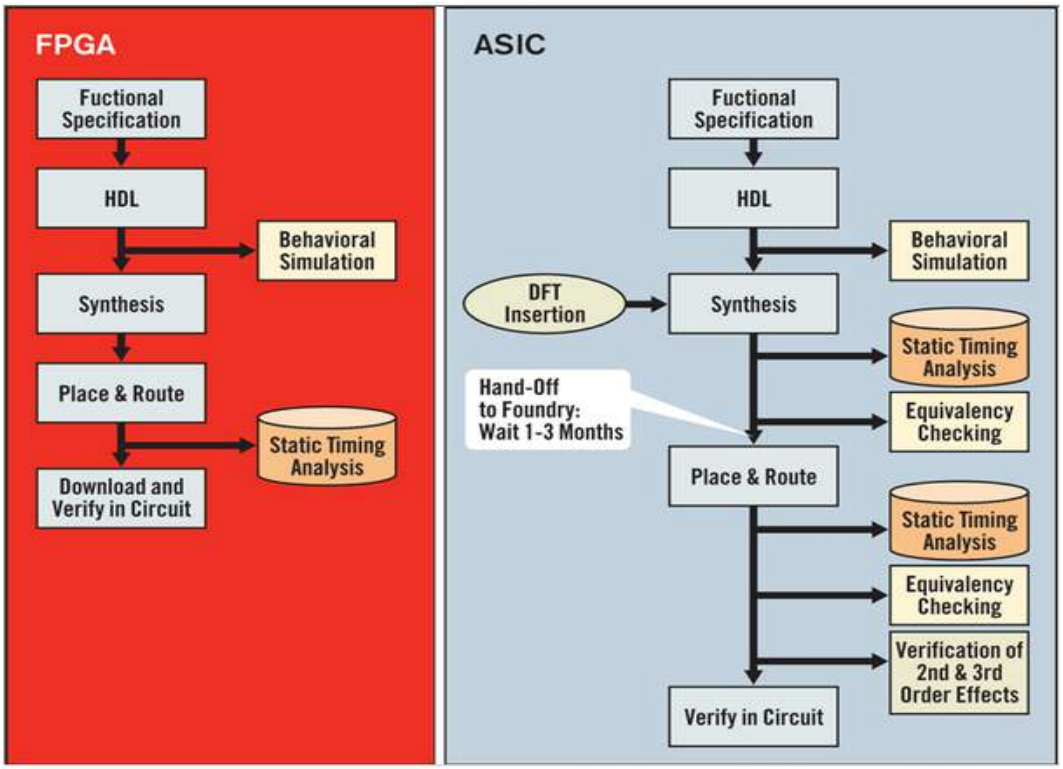
ASIC có thể có các thiết kế hoàn toàn cho các ứng dụng riêng và phức tạp, ví dụ vi xử lý hay bộ nhớ, nhưng FPGA thì không thể.

Bởi vì FPGA có thể được lập trình cấu hình lại vô số lần nên nó phù hợp cho các thiết kế mẫu cho ASIC. Như vậy với FPGA ta có thể tự thiết kế cả mẫu CPU theo mong muốn.

Các thiết kế ASIC phải tốn chi phí NRE (Non Recurring Engineering), đó là chi phí cho một lần nghiên cứu, thiết kế, và kiểm thử sản phẩm mới, trong khi đó thì các thiết kế FPGA lại không cần. Quá trình thiết kế ASIC kéo dài hơn so với FPGA (hình 1.12).

Các công cụ được sử dụng cho thiết kế FPGA thường rẻ hơn so với các công cụ thiết kế của ASIC.

Một FPGA có thể được sử dụng cho các ứng dụng khác nhau, nhờ lập trình lại FPGA. Nhưng với ASIC thì không thể.



Hình 1.12: Quá trình thiết kế trên FPGA và ASIC

1.4.2. FPGA và PLD.

FPGA chứa hơn 100000 khối logic nhỏ trong khi CPLD chỉ chứa tối đa vài nghìn khối.

Về kiến trúc, các FPGA được xem như các thiết bị có mật độ cao (fine-grain devices), trong khi CPLD là các thiết bị mật độ thưa (coarse-grain devices).

Các FPGA được sử dụng cho các ứng dụng phức tạp, trong khi các CPLD phù hợp cho các ứng dụng đơn giản và ít linh hoạt so với FPGA.

Các FPGA gồm có các khối logic nhỏ, trong khi các CPLD được làm ra từ các khối logic lớn.

FPGA là chip logic số dựa vào RAM, trong khi CPLD là chip dựa vào EEPROM.

Các FPGA dựa vào các bảng Look-up (LUTs) bên trong, trong khi các CPLD lại hình thành các hàm logic nhờ các mạch cổng sea-of-gates.

Hầu hết các FPGA có các mạch logic mức cao, ví dụ, các bộ cộng, bộ nhân và các bộ nhớ nhúng, và các khối logic thực hiện các bộ giải mã hoặc các hàm toán học, trong khi đó CLPD thì không.

Các trễ (Delays) trong các CLPD lớn hơn so với các trễ trong các FPGA.

1.5. CÔNG NGHỆ FPGA CỦA MỘT SỐ NHÀ CÔNG NGHỆ

1.5.1. Xilinx FPGA

Có một số họ Spartan FPGA: Spartan-II, Spartan-IIE (tương tự như Virtex), Spartan-3 (mật độ cổng cao, đạt tới 5 triệu cổng hệ thống), Spartan-3E (tương tự như Virtex-4, và tối ưu cho chi phí của logic), Spartan-3A (tối ưu cho chi phí pin), Spartan-3AN (cao cấp với flash), Spartan-3ADSP (cao cấp cho xử lý tín hiệu).

Device	XC3S50	XC3S200	XC3S400	XC3S1000	XC3S1500	XC3S2000	XC3S4000	XC3S5000
System Gates	50K	200K	400K	1000K	1500K	2000K	4000K	5000K
Logic Cells	1,728	4,320	8,064	17,280	29,952	46,080	62,208	74,880
Dedicated Multipliers	4	12	16	24	32	40	96	104
Block RAM Blocks	4	12	16	24	32	40	96	104
Block RAM Bits	72K	216K	288K	432K	576K	720K	1,728K	1,872K
Distributed RAM Bits	12K	30K	56K	120K	208K	320K	432K	520K
DCMs	2	4	4	4	4	4	4	4
I/O Standards	24	24	24	24	24	24	24	24
Max Single Ended I/O	124	173	264	391	487	565	712	784

Bảng 1.1: Họ Spartan-3 FPGA mật độ cổng cao

ng mật độ cổng tương đối cao và đủ các đặc tính cho thiết kế phần cứng nhiều chức năng dựa trên KIT phát triển Spartan-3E và các công cụ phần mềm của Xilinx.

1.5.2. Altera FPGA

Mảng cổng lập trình trường (FPGA) là một thiết bị bán dẫn có thể được lập trình sau khi sản xuất. Thay vì bị giới hạn ở bất kỳ chức năng phần cứng định sẵn nào, FPGA cho phép bạn lập trình các tính năng và chức năng của sản phẩm, thích nghi với các tiêu chuẩn mới và cấu hình lại phần cứng cho các ứng dụng cụ thể ngay cả sau khi sản phẩm

được cài đặt trên thực địa. ". Bạn có thể sử dụng một FPGA để thực hiện bất kỳ chức năng logic nào mà một mạch tích hợp ứng dụng cụ thể (ASIC) có thể thực hiện, nhưng khả năng cập nhật chức năng sau khi vận chuyển mang đến lợi thế cho nhiều ứng dụng.

Không giống như các FPGA thế hệ trước sử dụng I / Os với logic lập trình và liên kết, các FPGA ngày nay bao gồm nhiều hỗn hợp khác nhau của SRAM có thể cấu hình, bộ thu phát tốc độ cao, I / O tốc độ cao, khối logic và định tuyến. Cụ thể, một FPGA chứa các thành phần logic lập trình được gọi là các phần tử logic (LEs) và một hệ thống phân cấp các kết nối có thể cấu hình lại cho phép các LE được kết nối vật lý. Bạn có thể cấu hình LE để thực hiện các hàm tổ hợp phức tạp, hoặc đơn thuần là các cổng logic đơn giản như AND và XOR. Trong hầu hết các FPGA, các khối logic cũng bao gồm các phần tử bộ nhớ, có thể là các flipflops đơn giản hoặc các khối bộ nhớ hoàn chỉnh hơn.

Khi các FPGA tiếp tục phát triển, các thiết bị đã trở nên tích hợp hơn. Các khối sở hữu trí tuệ (IP) cứng được xây dựng trong vải FPGA cung cấp các chức năng phong phú trong khi giảm năng lượng và chi phí và giải phóng tài nguyên logic để phân biệt sản phẩm. Các gia đình FPGA mới hơn đang được phát triển với các bộ xử lý nhúng cứng, biến các thiết bị thành các hệ thống trên một chip.

1.6. KẾT LUẬN CHƯƠNG

Nội dung chương 1 trình bày tổng quan về công nghệ FPGA, một công nghệ phù hợp cho thiết kế các thiết bị số nói chung và các hệ thống nhúng. Nhưng để thiết kế được các thiết bị số và các hệ nhúng cần phải có các hệ phát triển mà các nhà sản xuất chip FPGA thường đưa ra, ví dụ như Altera, Xilinx, v.v… Vì vậy, ở trên đã trình bày các đặc điểm của bảng phát triển Xilinx Spartan-3E Starter Kit, mà trên đó có gắn chip FPGA với 500K khối logic hệ thống đủ để thiết kế cả một chip vi xử lý mềm 32-bit kiến trúc RISC (tập lệnh giảm thiểu) hai nhân. Ngôn ngữ mô tả phần cứng VHDL chuyên dùng cho thiết kế hệ thống số và hệ nhúng trên FPGA cũng được trình bày về cơ bản cấu trúc chương trình với một số lệnh đặc thù để lấy thư viện đưa vào chương trình (mô đun thư viện - library), mô tả các cổng vào ra của mạch số (mô đun thực thể - entity), diễn giải chức năng thực hiện của mạch số (mô đun kiến trúc – architecture).

Ngoài ra để lập trình được bằng ngôn ngữ VHDL cần nghiên cứu thêm về các kiến thức cơ bản về kiểu dữ liệu, mảng, hằng số, các lệnh, các toán tử.

CHƯƠNG 2: THIẾT KẾ PHẦN CỨNG BẰNG VHDL

2.1. NGÔN NGỮ MÔ TẢ PHẦN CỨNG VHDL

2.1.1. Lịch sử của VHDL

VHDL là ngôn ngữ của các vi mạch tích hợp tốc độ rất cao VHSIC (Very High Speed Integrated Circuits) Hardware Description Language. Giữa những năm 1980, Bộ Quốc phòng Mỹ (U.S. Department of Defense) và IEEE đã tài trợ sự phát triển của ngôn ngữ mô tả phần cứng này với mục đích làm tài liệu mô tả hành vi của ASIC và phát triển các vi mạch tích hợp tốc độ rất cao. Sử dụng VHDL một hệ thống số có thể được mô tả và kiểm chứng trước khi thực hiện phần cứng. Một chương trình của VHDL

bắt chước hành vi của một hệ thống số vật lý. Nó cũng cho phép kết hợp các định nghĩa về thời gian (các trễ của mạch cổng) cũng như để mô tả một hệ thống như là liên kết của các thành phần khác nhau. Từ 1987 VHDL trở thành chuẩn 1076 của IEEE và là một trong những ngôn ngữ chuẩn của công nghiệp được sử dụng để mô phỏng và phân tích các mạch logic lớn, phức tạp cho FPGA và ASIC. VHDL-2008 là chuẩn gần đây nhất của IEEE. Một ngôn ngữ khác mô tả phần cứng khác được sử dụng rộng rãi là Verilog. Cả VHDL và Verilog là những ngôn ngữ mạnh cho phép mô tả và mô phỏng các hệ thống phức tạp. Ngoài ra còn có ngôn ngữ đẳng thức Boole nâng cao ABEL (Advanced Boolean Equation Language) chuyên dùng cho các thiết bị logic có thể lập trình được PLD. ABEL không mạnh bằng hai ngôn ngữ trên và cũng ít phổ biến hơn trong công nghiệp. Mặc dù cả ba ngôn ngữ này được coi là tương tự như các ngôn ngữ lập trình thuận tiện. Ngôn ngữ mô tả phần cứng vốn là song song, nghĩa là các lệnh tương ứng với các mạch cổng logic luôn thực hiện các hàm logic với các tín hiệu đầu vào đến song song.

2.1.2. Ứng dụng của VHDL

VHDL được phát triển để giải quyết các khó khăn trong việc phát triển, thay đổi và lập tài liệu cho các hệ thống số. VHDL là một ngôn ngữ độc lập không gắn với bất kỳ một phương pháp thiết kế, một bộ mô tả hay công nghệ phần cứng nào. Người thiết kế có thể tự do lựa chọn công nghệ, phương pháp thiết kế trong khi chỉ sử dụng một ngôn ngữ duy nhất.

VHDL được sử dụng để tạo mô hình mô tả bằng văn bản các mạch logic. Mô hình bằng văn bản như vậy là một phần của thiết kế logic và được xử lý nhờ một chương trình tổng hợp. Ta dùng chương trình mô phỏng để kiểm thử thiết kế logic nhờ sử dụng các mô hình mô phỏng để thể hiện các mạch logic giao tiếp với thiết kế. Tập hợp các mô hình mô phỏng này thường được gọi là **testbench**.

VHDL có các khả năng đưa vào và đưa ra tệp tin, và có thể được sử dụng như là một ngôn ngữ dùng chung cho xử lý văn bản, nhưng các tệp tin được sử dụng nhiều hơn nhờ testbench cho các tác nhân hay dữ liệu so sánh. Có một số chương trình dịch VHDL tạo ra các tệp nhị phân thực hiện được. Trong trường hợp này có thể sử dụng VHDL để viết **testbench** để so sánh tính năng của thiết kế, và sử dụng các tệp tin trên máy tính chủ để xác định các tác nhân, để tương tác với người sử dụng, và so sánh các kết quả với mẫu mong đợi.

VHDL được sử dụng chủ yếu cho phát triển các hệ thống ASIC. Đối với ASIC, các công cụ để tự động chuyển mã VHDL thành danh sách Netlist ở mức cổng đã được phát triển trước. Sự chuyển mã VHDL thành Netlist được gọi là tổng hợp và là một phần tích hợp của luồng thiết kế, và đây là sự sắp xếp của công nghệ của ASIC (ASIC technology mapping).

Đối với VHDL tồn tại một số vấn đề: trong bước đầu tiên, các đẳng thức Boole được lấy từ mô tả của VHDL, và không quan trọng, thiết kế đích là ASIC, FPGA, hay PLD. Nhưng bây giờ, mã Boole phải được sắp xếp vào các khối logic có thể lập trình được (CLB) của FPGA. Đây gọi là sự sắp xếp của CLB (CLB mapping). Điều này thực hiện khó hơn so với sự đặt mã Boole vào trong các ASIC. Một vấn đề quan trọng ảnh hưởng đến hiệu năng của thiết bị đó là sự định tuyến các liên kết (interconnection) của các CLB gặp phải sự nghẽn nút cổ trai trong các FPGA có mật độ lớn các tài nguyên định tuyến (CLB, IOB, block RAM, các đường dây nối,…).

Đối với các PLD, VHDL được sử dụng các SPLD có các cấu trúc nhỏ, nhưng khó khăn sử dụng để thiết các PLD tương đối phức tạp (CPLD).

Trong các nghiên cứu thiết kế lệnh cho các hệ thống cứng/mềm, một phần của hệ thống được quan tâm lớn, đó là thực hiện phần mềm trong phần cứng. Và VHDL là mục tiêu nguyên cứu cho các hệ thống cứng/mềm. Bản thân VHDL cũng được sử dụng như là một công cụ giao tiếp C chuyên dụng.

2.1.3. Đặc điểm của VHDL.

VHDL có các cấu trúc để xử lý song song trong các thiết kế phần cứng, nhưng những cấu trúc này (gọi là các quá trình, **processes**) khác về cú pháp với các cấu trúc dùng ngôn ngữ Ada (trong Ada gọi là tasks). Cũng giống như Ada, VHDL là ngôn ngữ kiểu mạnh và nó không phân biệt các từ chữ to và chữ nhỏ (**non-case sensitivity).**

Trong khoa học máy tính và lập trình máy tính, một hệ thống thuộc "kiểu mạnh" khi nó xác định một hoặc nhiều hạn chế về phép tính lấy như thế nào các giá trị thuộc các kiểu dữ liệu khác nhau có thể được trộn lẫn nhau. VHDL là "loại mạnh" vì nó đặt ra một số hạn chế về sự trộn lẫn được phép xuất hiện, ngăn chặn sự biên dịch hoặc chạy mã nguồn sử dụng dữ liệu, mà dữ liệu đó được xem xét là một dạng lỗi. Ví dụ, phép tính cộng không thể sử dụng một số nguyên với các giá trị chuỗi.

Để trực tiếp trình bày các phép tính thường có trong phần cứng, có nhiều đặc tính của VHDL như tập hợp mở rộng của các phép tính Boole gồm cả NAND và NOR. VHDL cũng cho phép các mảng được chỉ số theo chiều tăng hoặc giảm.

Trong quá trình thiết kế phần cứng, sẽ phải sản sinh ra sơ đồ ở mức chuyển giao thanh ghi RTL (Register-Transfer Level) của mạch yêu cầu. RTL là mức trìu tượng được sử dụng để mô tả vận hành của mạch số đồng bộ. Trong thiết kế RTL, hành vi của mạch được xác định trong các khái niệm luồng các tín hiệu (hoặc vận chuyển dữ liệu) giữa các thanh ghi của phần cứng, và các phép tính logic được thực hiện trên các tín hiệu. Sau khi sơ đồ ở mức RTL được tạo ra, nó có thể được so sánh nhờ sử dụng phần mềm mô phỏng, mà phần mềm này đưa ra các tín hiệu song song của các đầu vào và đầu ra của mạch sau khi tạo ra **testbench** thích hợp. Để tạo ra một **testbench** thích hợp cho mạch cụ thể hay mã VHDL, các đầu vào cần phải được xác định đúng. Khi mô hình VHDL được chuyển thành "các cổng và các đường dây" được sắp xếp trên các chip CPLD hay FPGA, thì đó là phần cứng thực tế đã được cấu hình. Điều này khác với các ngôn ngữ lập trình khác mà ở đó sau khi biên dịch, chương trình được thực hiện trên hệ thống vi xử lý để tạo ra các kết quả.

VHDL phân biệt với các ngôn ngữ khác bởi sự thực hiện các phép gán với hai loại cơ bản: tuần tự và tương tranh.

Các câu lệnh tuần tự: được thực hiện lần lượt tiếp theo nhau, giống như trong các ngôn ngữ lập trình. Các câu lệnh sau có thể không bỏ qua các ảnh hưởng của các câu lệnh trước. Thứ tự của phép gán cần phải được xem xét khi các câu lệnh tuần tự được sử dụng.

Các câu lệnh tương tranh: được tích cực liên tục. Thứ tự của các câu lệnh tương tranh không có liên quan. Chúng phù hợp để mô hình đặc tính song song của phần cứng.

VHDL có ba kỹ thuật mô hình hóa quan trọng, đó là: trìu tượng (abstraction), tính module (modularity), và phân lớp (Hiearchy).

Trìu tượng: cho phép mô tả các phần khác nhau của hệ thống với số lượng chi tiết khác nhau. Các module chỉ cần cho mô phỏng thì không cần phải được mô tả chi tiết như các module để tổng hợp.

Tính module: cho phép người thiết kế phân chia các khối chức năng lớn và ghi module cho từng phần.Như vậy, một hệ thống phức tạp có thể được chia thành các hệ thống nhỏ hơn và đơn giản.

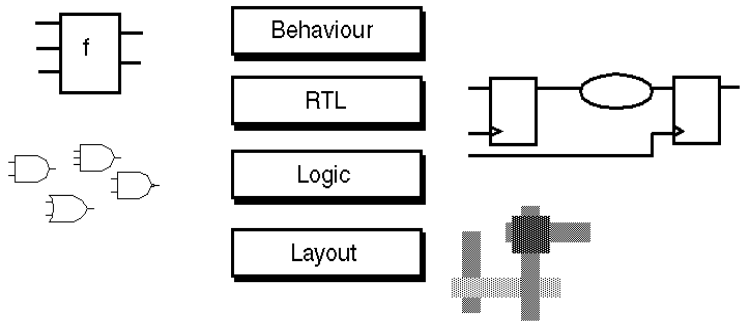
Phân lớp: cho phép người thiết kế xây dựng một thiết kế từ một số module, mà mỗi module này lại gồm một số module con nhỏ hơn. Mỗi một lớp có thể chứa các

module ở các mức trìu tượng khác nhau. Những module nhỏ của các mô hình này được thể hiện trong lớp tiếp theo thấp hơn.

2.1.3.1. Các mức trìu tượng trong thiết kế mạch tích hợp

Trìu tượng được xác định như là sự che dấu thông tin quá chi tiết. Thông tin được cho là không quan trọng cho xem xét ở mức hiện tại được loại ra khỏi mô tả. Các tầng trìu tượng được thể hiện bởi loại thông tin chung cho các mô hình của một lớp.

Một mô hình được gọi là ở mức trìu tượng nào đó nếu từng module có cùng một cấp độ trìu tượng. Nếu không phải như vậy thì mô hình sẽ là hỗn hợp của các lớp trìu tượng khác nhau. Có bốn tầng trìu tượng của thiết kế mạch tích hợp số được mô tả ở hình 2.1.



Hình 2.1: Các tầng trìu tượng thiết kế mạch tích hợ p

Tầng hành vi (Behaviour): mô tả các chức năng của một hệ thống (hệ thống làm gì, hay có hành vi như thế nào) chứ không phải là các thành phần và liên kết giữa chúng. Hệ thống là một hộp đen. Mô tả hành vi xác định quan hệ giữa các tín hiệu vào và các tín hiệu ra. Không có nhịp đồng hồ hệ thống và các chuyển tiếp tín hiệu được đồng bộ với thời gian chuyển mạch.Các mô tả ở mức này chỉ dùng cho mô phỏng, chứ không thể dùng cho tổng hợp được. Đây có thể là một biểu thức Boole hoặc là một mô tả trìu tượng phức tạp hơn như sự chuyển tải của thanh ghi (Register Transfer) hoặc mức thuật toán (Algorithmic level). Ví dụ, một mạch đơn giản cảnh báo các hành khách xe ca khi cửa xe mở hoặc dây đai an toàn chưa được cài khi chìa khóa điện được tra vào ổ khóa điện. Ở tầng hành vi, điều này có thể được biểu diễn như biểu thức:

Warning = Ignition_on AND (Door_open OR Seatbelt_off)

Tầng chuyển tải - thanh ghi, RTL (Register-Transfer Level): là tầng trìu tượng tiếp theo mô tả vận hành của mạch. Trong đó, mạch được chia ra logic tổ hợp và các thành phần nhớ. Các thành phần nhớ, FF (Flip-flop), các chốt (Latch) được điều khiển bởi nhịp đồng hồ hệ thống. Trong các thiết kế đồng bộ, các FF phải được dùng,

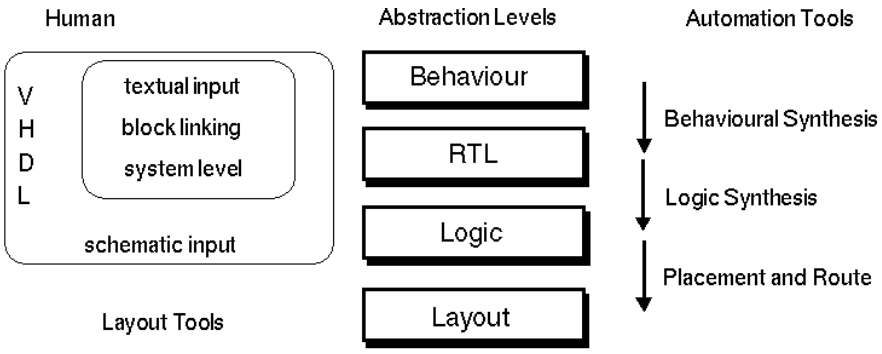
bởi chúng được điều khiển bởi sườn xung nhịp đồng hồ. Ngoại trừ các mạch chốt, vì các mạch chốt lại được điều khiển bởi mức điện thế của tín hiệu điều khiển. Trong thiết kế ở RTL, hành vi của mạch được xác định bởi các điều kiện của luồng các tín hiệu (hay chuyển tải dữ liệu) giữa các thanh ghi của phần cứng, và các vận hành logic được thực hiện trên các tín hiệu. Mô tả ở mức RTL là mô tả có thể tổng hợp được.

Tầng logic: trong mức logic, thiết kế được trình bầy như một danh sách Netlist các cổng (AND, OR, NOT,…), các thành phần nhớ, và có thể cả các trễ của từng thành phần.

Tầng sắp đặt (Layout): là tầng đáy của phân tầng trìu tượng. Ở tầng này, các ô của thiết kế cuối cùng được xếp đặt trên chip và các liên kết giữa chúng được định tuyến. Sau khi layout được so sánh, mạch sẵn sang cho quá trình sản xuất.

2.1.3.2. Các tầng trìu tượng của thiết kế VHDL

Thiết kế phần cứng nói chung bao gồm 4 tầng trìu tượng: hành vi, RTL,, Logic, và Layout. Nhưng với thiết kế thiết phần cứng bằng VHDL là chỉ áp dụng 3 tầng trìu tượng trên, đó là hành vi, RTL, và Logic. Mức Layout không phù hợp cho VHDL bởi vì nó được các công cụ Layout thực hiện tự động (hình 2.2).



Hình 2.2: Các mức trìu tượng thiết kế VHDL

Các mô tả trong các tầng hành vi và RTL được soạn thảo bởi trình soạn thảo văn bản. Bởi vì chúng chủ yếu là các câu lệnh.

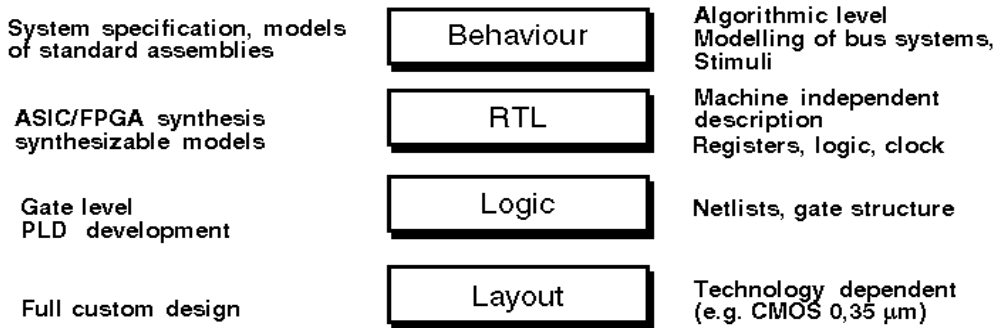
Ở tầng logic, thiết kế được trình bầy như một danh sách Netlist với các cổng (AND, OR, NOT,…) và các thành phần nhớ. Sơ đồ được thay đổi vì các mô tả của Netlist của VHDL có xu hướng trở nên ngày càng phức tạp hơn.

Có phần mềm hỗ trợ sự chuyển từ một tầng trìu tượng phía trên xuống tầng thấp hơn. Các công cụ phần mềm tổng hợp hỗ trợ tốt cho tổng hợp logic, chúng có thể tái tạo hành vi trong tầng logic.

2.1.3.3. Mô tả của các tầng trìu tượng trong thiết kế VHDL

Với thiết kế phần cứng gồm 4 tầng trìu tượng thì các mô tả ở từng tầng trìu tượng có sự khác nhau (hình 2.3).

a. Mô tả ở tầng hành vi:

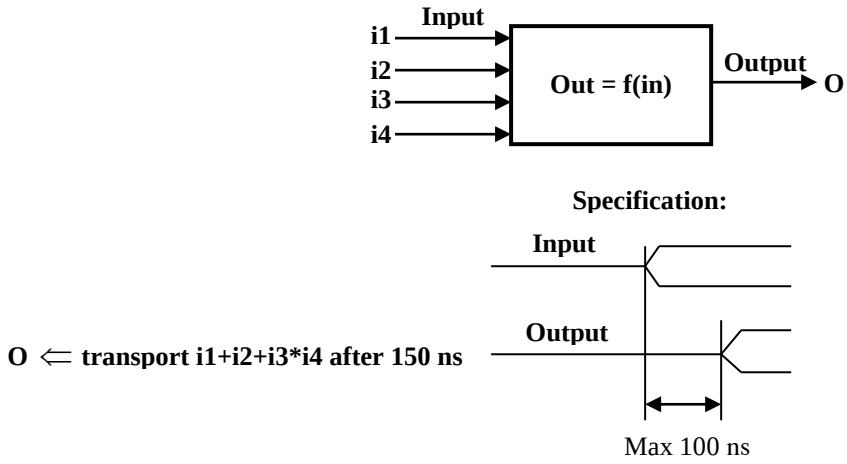


Hình 2.3: Các mô tả ở các mức trìu tượng

Tầng hành vi (Behaviour) mô tả hệ thống (là một hộp đen) bằng các thuật toán phức tạp (algorithmic level), các hệ thống bus (bus systems), và các tác nhân (Stimuli). Các thuật toán và các bus hệ thống được mô tả mà không cần quan tâm đến khả năng tổng hợp của mô tả. Các tác nhân (Stimuli) để mô phỏng các mô hình RTL được mô tả trong tầng hành vi. Ví dụ, tác nhân là các giá trị tín hiệu của các cổng vào của mô hình được mô tả trong **testbench**, đôi khi được gọi là giá trị **bench**. Người thiết kế phải tìm một tập hợp phù hợp các tác nhân vào không mẫu thuẫn với định nghĩa hệ thống (System specification). Các đáp ứng của mô hình phải được so sánh với các giá trị mong đợi, mà các giá trị này có thể được tạo ra với sự hỗ trợ của đồ thị song các tín hiệu, trong đó các giá trị của tín hiệu được mô phỏng.

Ví dụ, định nghĩa một một hàm đơn giản của một module được cho ở hình 2.4. Đầu ra O phụ thuộc vào 4 giá trị đầu vào i1, i2, i3, và i4. Cho rằng giá trị mới của đầu ra phải là ổn định ít nhất trong 150 ns sau khi các giá trị đầu vào thay đổi. Như vậy hàm này có thể được mô hình như là một đẳng thức toán, ví dụ, (i1+i2+i3*i4), cộng thêm trễ 150 ns tính từ lúc các giá trị đầu vào thay đổi cho đến khi đầu ra có gía trị kết quả mới của hàm tính toán. Trễ 150 ns là trường hợp xấu nhất cần thiết để tính giá trị mới cho đầu ra.

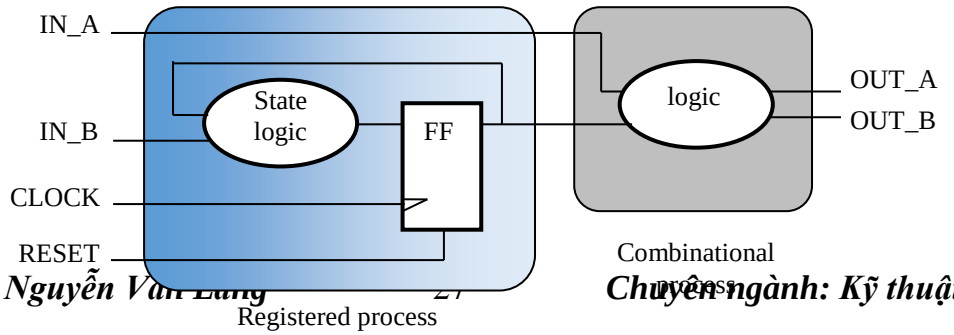
Mô hình hành vi là cách đơn giản để mô tả hành vi của một mạch, tương tự như các ngôn ngữ lập trình bình thông thường, như PASCAL hoặc C. Với mô tả đó, chỉ hành vi chức năng có thể được mô phỏng nhờ công cụ mô phỏng của VHDL.



Hình 2.4: ví dụ mô tả hành vi trong VHDL

b. Mô tả ở tầng RTL:

Hệ thống được mô tả trong các khái niệm của các thanh ghi, các thành phần logic tổ hợp, các thành phần nhớ, nhịp đồng hồ (Clock). Các thanh ghi (Registers) được nối với tín hiệu nhịp đồng hồ và đảm bảo cho hành vi đồng bộ. Trong VHDL hành vi chức năng được mô hình bởi các tiến trình, **processes.** Có hai kiểu của **process** ở các mô tả của mức RTL: process tổ hợp thuần tuý và process được định nhịp đồng hồ. Tất cả các process được định nhịp đồng hồ suy ra các Flip-Flop (FF) và có thể được mô tả bằng các khái niệm của cú pháp máy trạng thái (machine independent description). Bổ xung vào các tín hiệu của đầu vào và đầu ra của dữ liệu, các tín hiệu điều khiển như: nhịp đồng hồ của module (CLOCK) và xóa (RESET) cho các FF phải được xem xét trong mô hình hóa ở mức RTL. Khi áp dụng xóa không đồng bộ, đầu vào xóa được coi như một đầu vào dữ liệu bình thường. Mã VHDL ở lớp RTL cũng chứa một số loại thông tin cấu trúc bổ xung cho hành vi chức năng vì các thành phần nhớ và không nhớ tách biệt nhau. Định thời của các giá trị tín hiệu cũng được ra (ví dụ đồng bộ với tín hiệu nhịp đồng hồ).



Xung nhịp đồng hồ là nhãn phân biệt cho mô tả ở mức RTL. Tất cả các vận hành của mạch đều liên quan đến tín hiệu nhịp đồng hồ. Các mô phỏng ở mức RTL không cho thông tin về hành vi định thời thật sự, có nghĩa là không thể nói rằng tất cả các tín hiệu phải thiết lập các giá trị ổn định bên trong một chu kỳ nhịp đồng hồ hoặc không.

c. Mô tả ở tầng Logic (tầng cổng):

Tầng logic, hay còn gọi là tổng cổng hay Netlist, vì ở tầng này danh sách Netlist các cổng được tạo ra từ mô tả ở tầng RTL nhờ sự giúp đỡ của một công cụ tổng hợp và thư viện chứa thông tin về tất cả các cổng đã có sẵn và các thông số của chúng, như hệ số gộp đầu vào (Fan-in), hệ số tách đầu ra (Fan-out), các trễ, và thuyết minh cụ thể về các thành phần và các danh sách liên kết của chúng. Sơ đồ cấu trúc và thuyết minh của mạch số được mô tả ở đây.

Khi mô hình được mô tả ở tầng cổng, các trễ có thể được áp đặt vào các cổng để mô phỏng. Thông tin định thời là một phần của thư viện tổng hợp. Điều này cho phép đánh giá sơ bộ về hành vi định thời (đồ thị xung). Sự không chắc chắn nảy sinh từ trễ lan truyền dọc theo các đường dây tín hiệu chưa được quan tâm. Những trễ này có thể là phần đáng kể của toàn bộ trễ trong các thiết kế lớn.

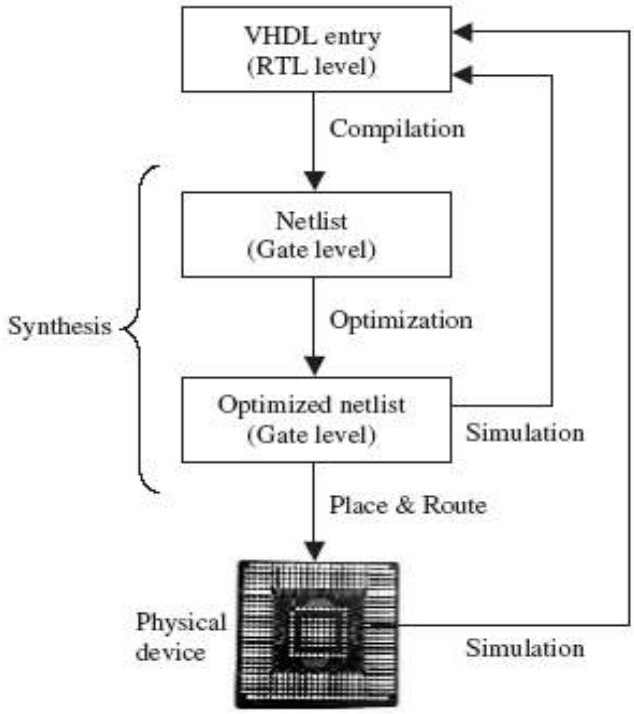
d. Mô tả ở tầng Layout:

Ở tầng sắp xếp (Layout), dựa vào Netlist, Layout của mạch được tạo ra. Các chiều dài của các dây nối có thể được đổi thành các trễ lan truyền mà chúng có thể được giải thích lại ở mức cổng. Điều này cho phép thông qua các mô phỏng định thời mà không cần đến phần mềm mô phỏng bổ xung. Nếu Layout đã hoàn thành, thì các độ dài của đường dây tín hiệu và các trễ lan truyền sẽ được biết. Thiết kế có thể được mô phỏng ở mức cổng với các giá trị trễ bổ xung và tiếp theo hành vi định thời của toàn bộ mạch có thể được đánh giá.

2.2. QÚA TRÌNH THIẾT KẾ PHẦN CỨNG BẰNG VHDL

2.2.1. Các công đoạn thiết kế bằng VHDL

Các ứng dụng của VHDL là chế tạo các mạch hoặc các hệ thống trên các chip vi mạch tích hợp có thể lập trình được (FPGA, PLD).



Hình 2.6: Quá trình thiết kế VHDL

Chế tạo ra phần cứng bằng HDL được chia thành ba giai đoạn như sau (hình 2.6):

Giai đoạn 1: xây dựng thiết kế bằng VHDL (VHDL entry)

Dựa vào các mô tả hành vi của phần cứng (tầng hành vi), chúng ta bắt đầu thiết kế bằng viết mã VHDL. Mã VHDL này sẽ được lưu vào file có đuôi là **.vhd** và có tên cùng với tên của ENTITY. Mã VHDL sẽ được mô tả ở tầng chuyển thanh ghi (RTL).

Giai đoạn 2: Tạo danh sách mạng (netlist)

Bước 1: Biên dịch. Quá trình biên dịch sẽ chuyển mã VHDL vào danh sách mạng (netlist) ở mức logic (hay mức cổng).

Bước 2: Tối ưu. Quá trình tối ưu được thực hiện trên netlist ở mức cổng về tốc độ và phạm vi. Trong giai đoạn này, thiết kế có thể được mô phỏng để kiểm tra và phát hiện những lỗi xảy ra trong quá trình thiết kế VHDL.

Giai đoạn 3: Tối ưu thiết kế (Optimized netlist)

Giai đoạn 3 thực hiện tối ưu thiết kế. Trong giai đoạn này cần phải thực hiện mô phỏng (simulation) thông thường sử dụng phương pháp mô phỏng định thời các tín hiệu vào ra của thiết kế. Kết quả mô phỏng tốt cho phép thực hiện sắp xếp (layout) vật lý cho chip PLD/FPGA: thực hiện sắp đặt và định tuyến (Place & Route) hoặc tạo ra mặt nạ cho thiết bị vật lý (physical device) trên ASIC.

Giai đoạn 2 và 3 là giai đoạn tổng hợp (Synthesis) ở mức logic. Thiết kế được kiểm thử (testbench) cả hai giai đoạn này nhờ các công cụ mô phỏng. Kết quả mô phỏng được thể hiện ở dạng đồ thị xung với các thông số thời gian, các giá trị kiểm thử, hoàn chỉnh và tối ưu thiết kế.

2.2.2. Thiết kế phần cứng trên Xilinx FPGA

Các bước lập trình FPGA bao gồm:

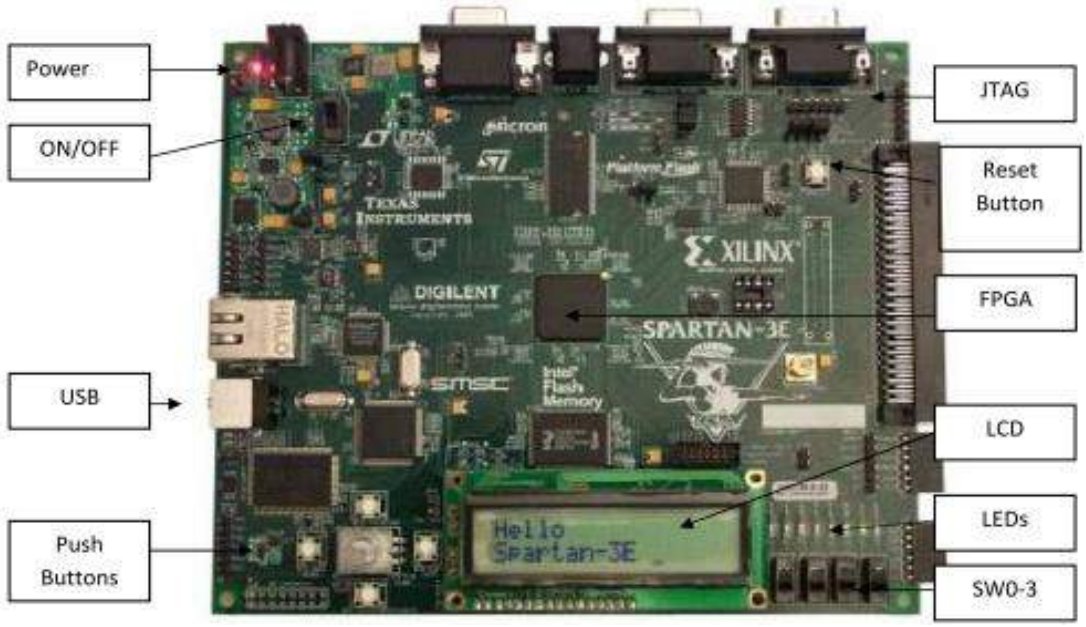
Viết một chương trình nhỏ bằng ngôn ngữ VHDL thực hiện một mạch logic tổ hợp đơn giản.

Nối các đầu vào và đầu ra của chương trình với các bộ chuyển mạch, các nút (buttons) và các đèn LED trên bảng Spartan-3E.

Tải chương trình vào bảng Sparrtan-3E nhờ sử dụng phần mềm Project Navigator.

2.2.2.1. Tính năng thiết kế

Tính năng thiết kế thực hiện được tóm tắt trong bảng 2.7. Bốn LEDs trên bảng sang phụ thuộc vào các tổ hợp khác nhau của các nút chuyển mạch (SW0-3) và các nút bấm (Push Buttons) trên bảng (hình 2.7).



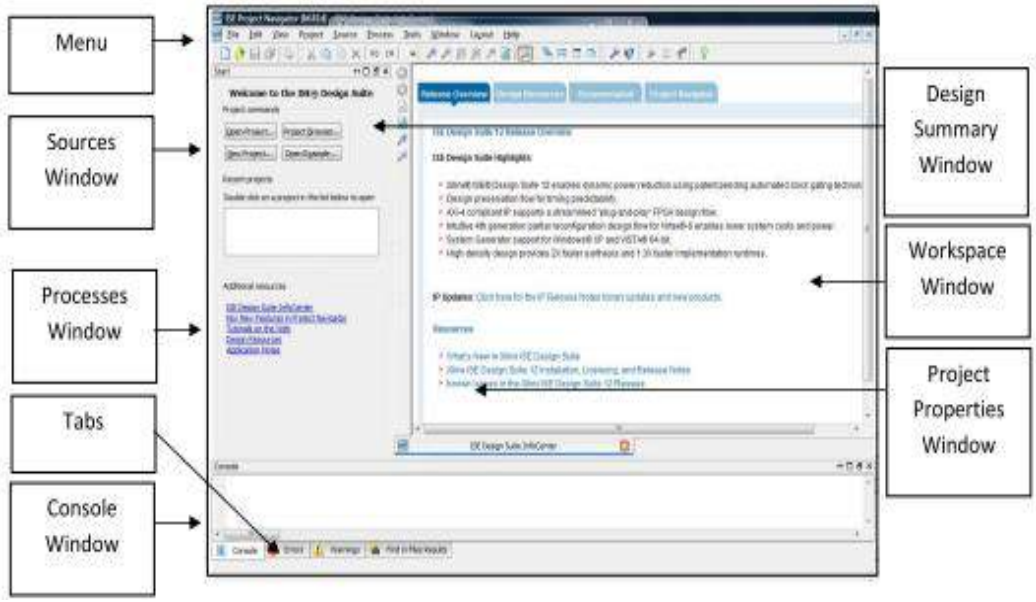
Hình 2.7: Bảng phát triển Spartan-3E 500K/1600K

2.2.2.2. Tài liệu liên quan

Trước khi bước vào thực hành lập trình FPGA, cần phải tải tài liệu hướng dẫn sử dụng bảng phát triển Spartan-3E FPGA Development board User Guide. Tài liệu này mô tả các chân tín hiệu (pins) của chip FPGA và các thiết lập cần thiết để nối các chân tín hiệu với các thiết bị vào/ra khác nhau trên bảng.

Những trang bị sau đây cần phải có để làm việc với công việc lập trình FPGA:

Phần mềm Xilinx ISE Project Navigator (các phiên bản 12.4 hoặc cao hơn). Phần mềm này có thể tải miễn phí từ website, http://www.xilinx.com/.



Hình 2.8: Cửa sổ khởi động ban đầu của Project

Bộ KIT Spartan-3E bao gồm: bảng phát triển Spartan-3E (hình 2.7), cáp nguồn, cáp JTAG và cáp USB để nối với PC.

2.2.3. Công cụ phần mềm thiết kế Xilinx ISE

2.2.3.1. Khởi động (Startup)

Click vào biểu tượng **Xilinx ISE Design Suite 14.1** trên màn hình Desktop, hoặc Khởi động phần mềm Project Navigator bằng cách:

Start->All programs->Xilinx ISE Design Suite 14.1->ISE Design Tools->Project Navigator

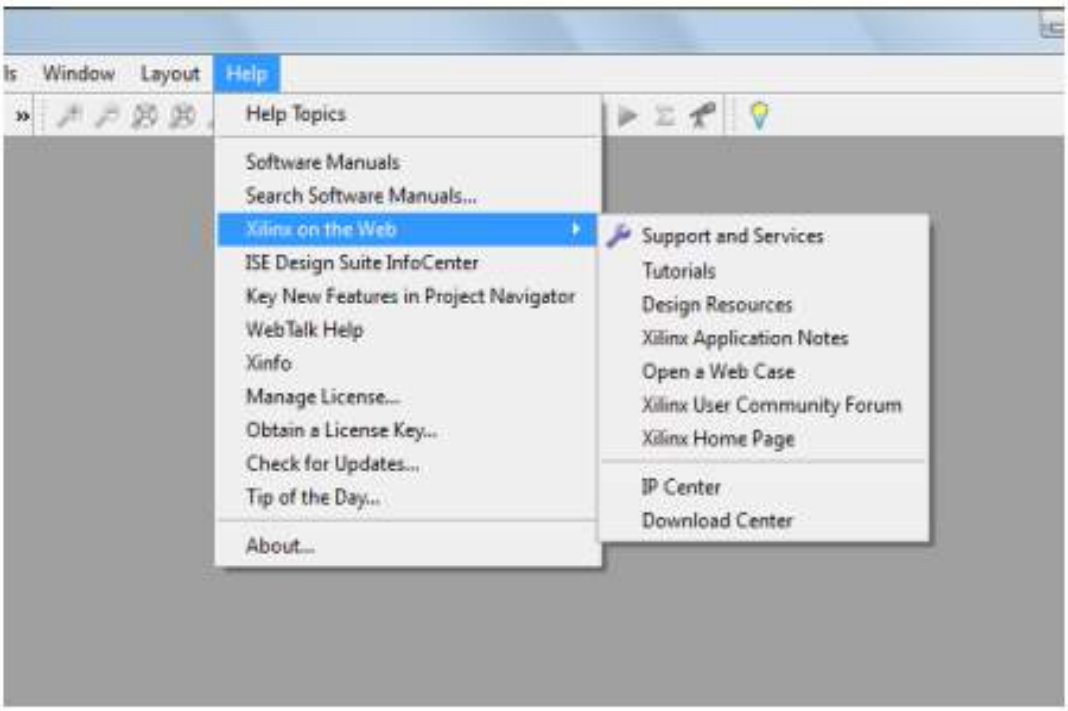
Xuất hiện cửa sổ **ISE Project Navigator**

2.2.3.2. Trợ giúp (Help)

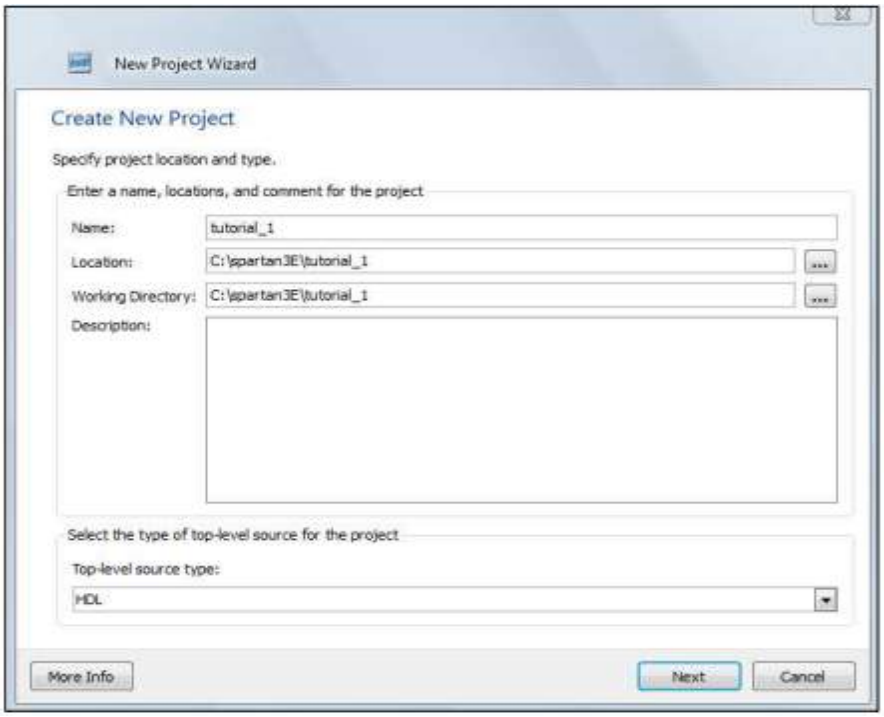
Như chỉ ra ở hình 2.9, trợ giúp có thể được truy nhập thông qua thực đơn **Help** trên Project Navigator.

Tải phần mềm, tài liệu và các hội thảo (forums) có thể được truy nhập ở http://www.xilinx.com/support/.

Các ví dụ thiết kế sử dụng bộ KIT Spartan-3E có thể tài từ http://www.xilinx.com/products/boards/s3estarter/reference_designs.htm.



Hình 2.9: Thực đơn **Help**



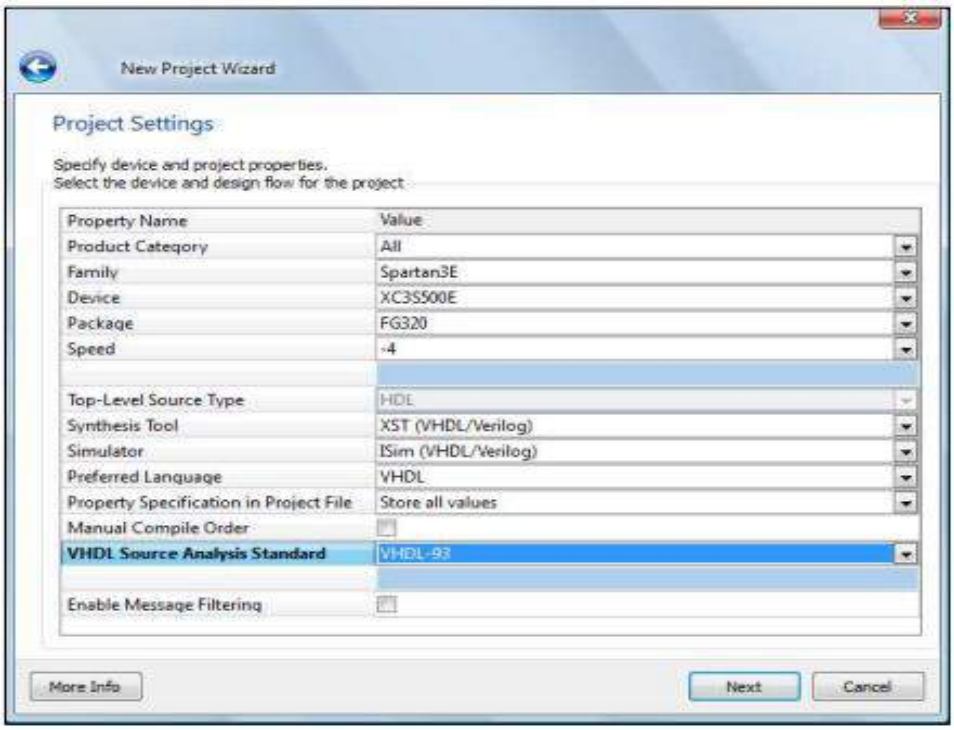
Hình 2.10**: New Project Wizard**, Trang **Create New Project**

2.2.3.3. Tạo một Project mới

- Chọn **File->New Project**. Sẽ xuất hiện cửa sổ **New Project Wizard**.
- Gõ tutorial_1 trong trường **Name:**
- Chọn một vùng thích hợp **Location:** và **Working Diecrectory:** cho project mới
- Xem xét **Top-level source type**: được chọn như **HDL**.

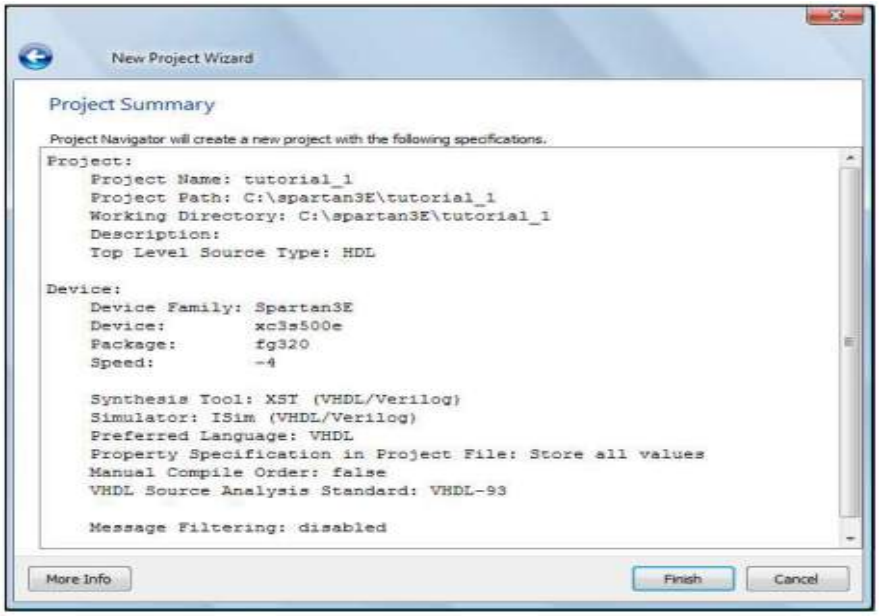
Các thuộc tính phải được thiết lập như chỉ ra ở hình 2.10. Click **Next** để chuyển tới trang **Project Settings**.

- Chọn và điền vào trong các thuộc tính như sau:
- Property Name: **Value**
- Product Category: **All**
- Family: **Spartan3E**
- Device: **XC3S500E**
- Package: **FG320**
- Speed Grade: **-4**
- Top-Level Source Type: **HDL**
- Synthesis Tool: **XST (VHDL/Verilog)**
- Simulator: **ISIM (VHDL/Verilog)**
- Preferred Language: **VHDL**
- Property Specification in Project File: **Store All Values**
- Manual Compile Order: **unchecked**
- VHDL Source Analysis Standard: **VHDL-93**
- Enable Message Filtering: **unchecked.**



Hình 2.11: **New Project Wizard**, trang **Project Settings**

- Click **next** để chuyển đến trang **Project Summary** (hình 2.11).

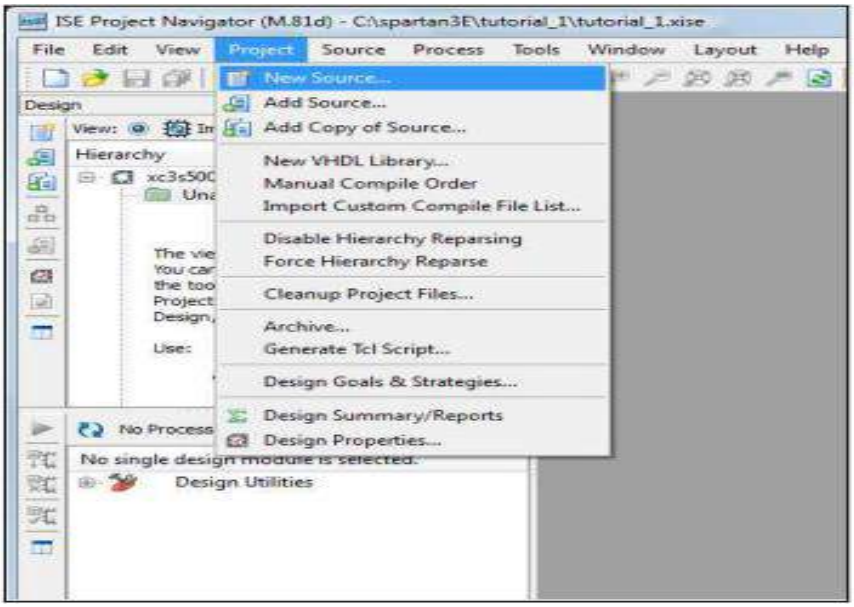


Hình 2.12: **New Project Wizard**, trang **Project**

- Click **Finish** để rời khỏi **New Project Wizard** trở về cửa sổ **ISE Project Navigator**

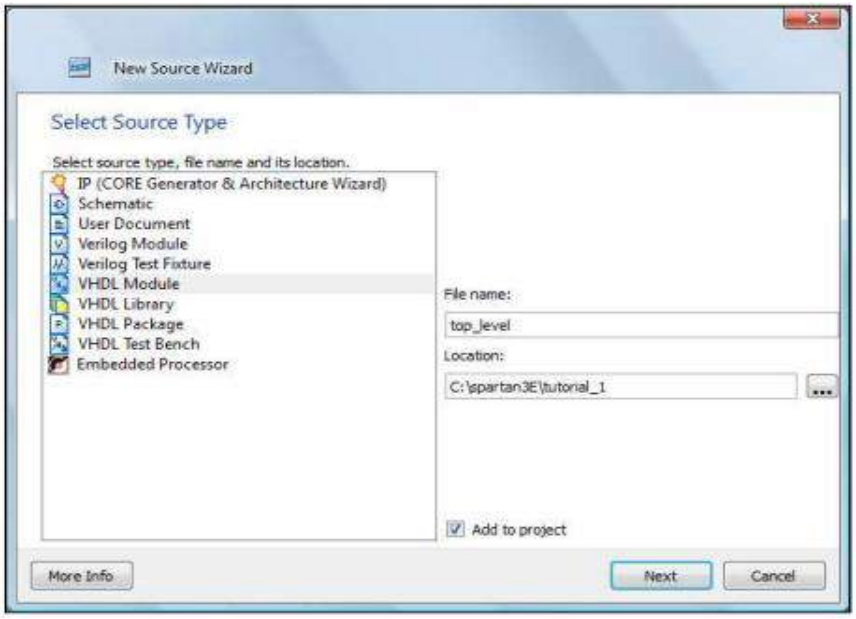
2.2.3.4. Bổ xung mã nguồn VHDL mới

Trên cửa sổ **ISE Project Navigator** (hình 2.13):



Hình 2.13: **Project – New Source…**

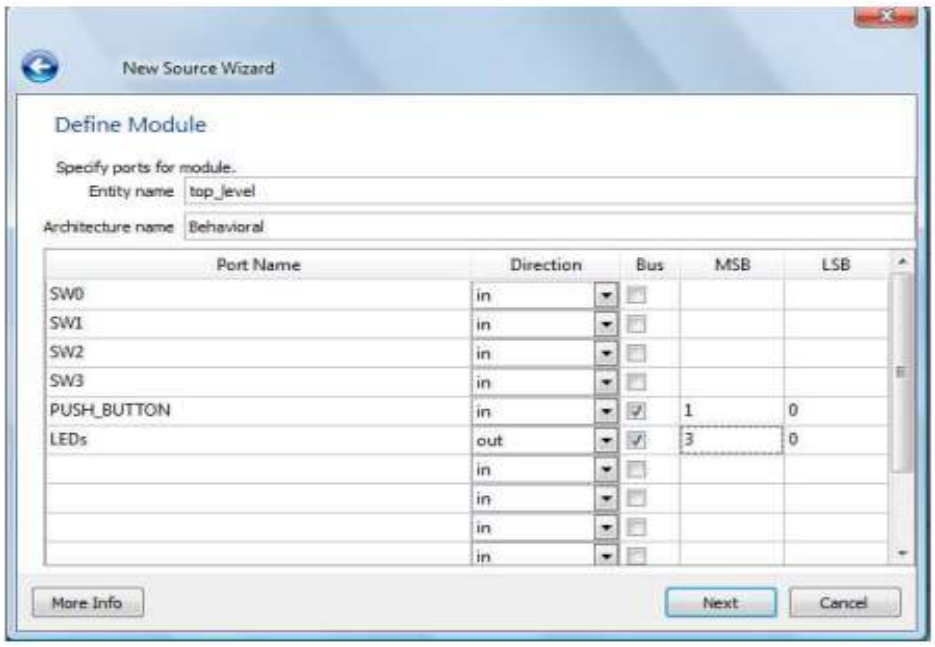
- **Project->** một click **New Source ->** sẽ xuất hiện cửa sổ **New Source Wizard**
- Trên cửa sổ **New Source Wizard** (hình 2.14): Select Source Type: chọn kiểu nguồn: **VHDL Module.**



Hình 2.14: **New Source Project: Select Source Type**:**VHDL**

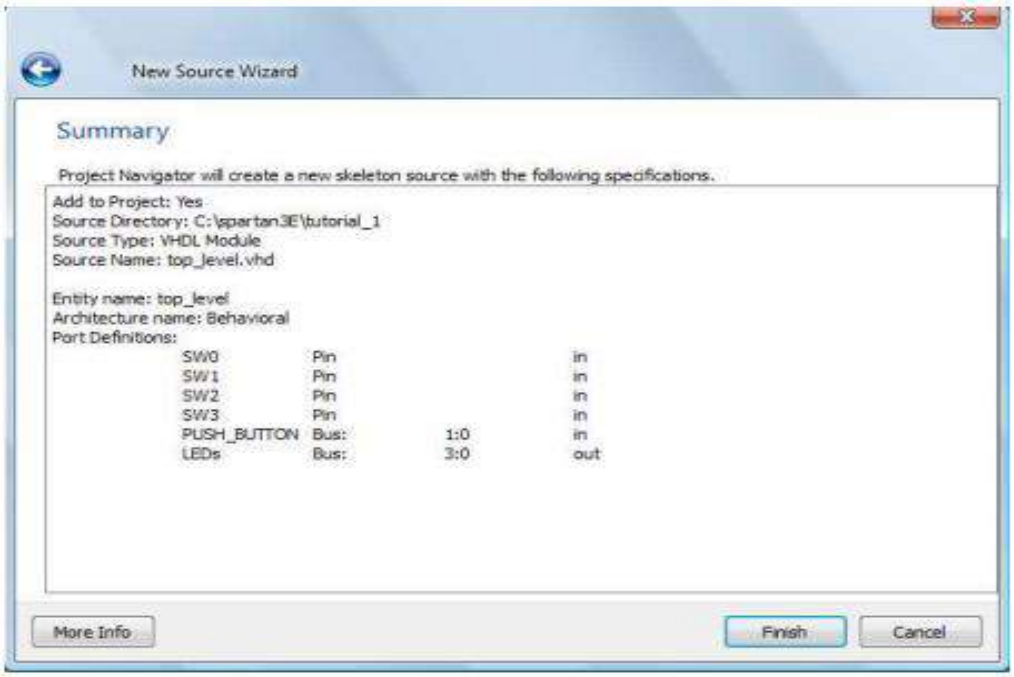
- Đưa vào tên file **top_level**, và đưa vào vùng của file (location: D:\Startan3E\tutorial_1).
- **Add to project box** phải được checked.
- Click **Next** để chuyển đến cửa sổ **New Source Wizard**.
- Trên cửa sổ **New Source Project: Define Module** (hình 2.15):

Xác định các cổng (inputs và outputs của thiết kế) nhờ đưa vào thông tin cho **Port name** như sau:



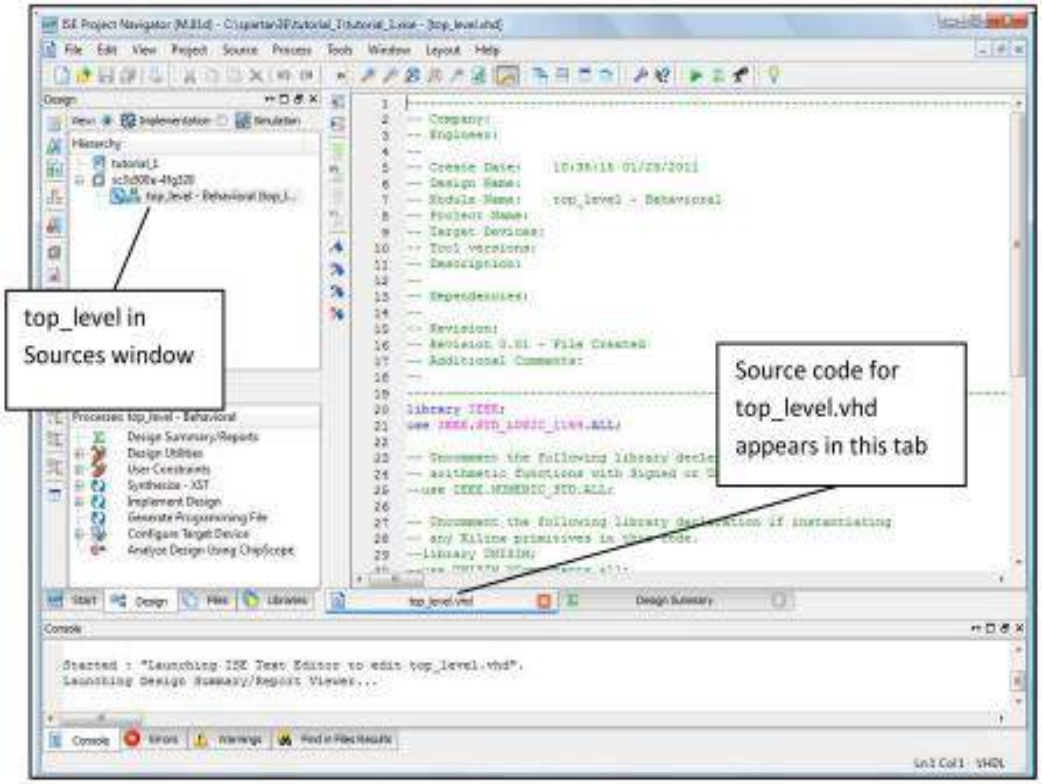
Hình 2.15:
New Source Project: Define Module

- SW0-3 là các bit vào: IN, và sẽ kết nối với các nút chuyển mạch trên bảng Spartan-3E.
- PUSH_BUTTON là đầu vào: IN, gồm 2 bit, và được kết nối với hai nút chuyển mạch trên Spartan-3E. Vì đây là đầu vào nhiều bit, nên Bus check box được checked, MSB (bit lớn nhất) đượt thiết lập bằng '1', và LSB (bit nhỏ nhất) được thiết laajo bằng '0'.
- Các đèn LEDs là đầu ra: OUT, gồm 4 bit, và sẽ được kết nối với 4 LEDs trên Spartan-3E. Vì đây là đầu ra nhiều bit, nên Bus check box được checked, MSB được thiết lập bằng 3, và LSB được thiết lập bằng 0.
- Click **Next** để chuyển đến trang **Summary** (hình 2.16).



Hình 2.16: **New Source Project**, **Summary**

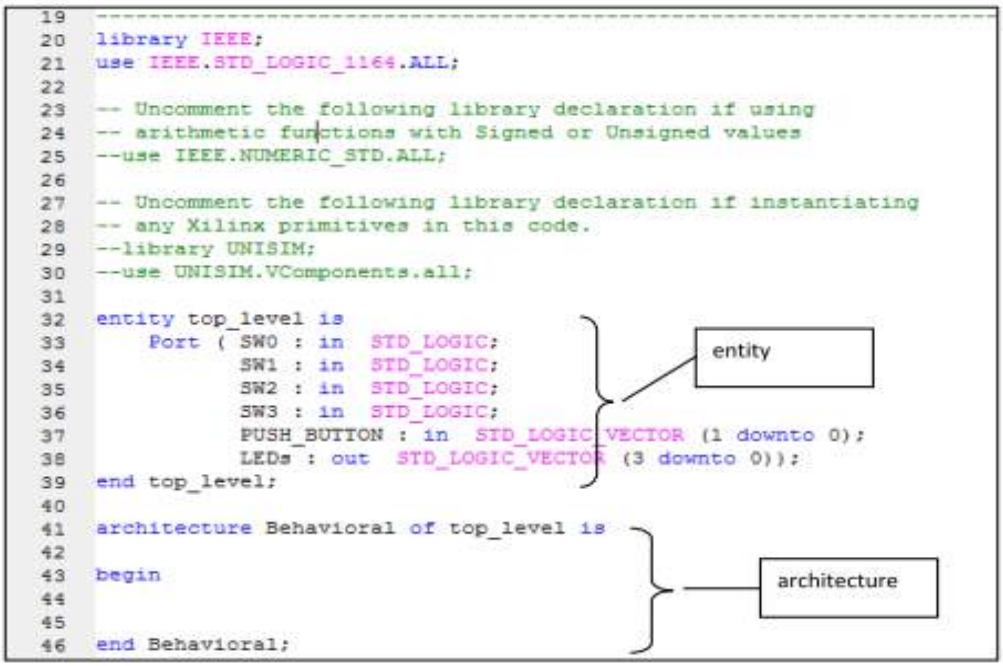
- Click **Finish** để ra khỏi **New Source Wizard** và trở về Như chỉ ra ở hình 3.11, **top_level** sẽ xuất hiện trong cửa sổ Sources. Click hai lần trên top_level trong của sổ Sources sẽ hiển thị file**, top_level.vhd** trong tab.



Hình 2.17: file nguồn mới **top_level.vhd** hiển thị trong tab

2.2.3.5. Soạn thảo mã nguồn VHDL

Trong mục 2.2.3.4, file nguồn VHDL mới top_level.vhd đã được tạo ra. Đó thực chất là một file văn bản ASCII có thể được soạn thảo bởi một công cụ soạn thảo văn bản. Tuy nhiên, thuận tiện để soạn file ta sử dụng phần mềm Project Navigator. File top_level.vhd có nội dung như cho ở hình 2.18.



Hình 2.18: nội dung file top_level.vhd được hiển thị trong Project

Trong hình 2.18, có thể nhận thấy rằng các mã VHDL đều có mẫu để cho dễ đọc. Các dòng bình luận (bắt đầu bằng --) có mầu xanh lá cây. Các từ khóa có mầu xanh da trời, các kiểu của VHDL có mầu đỏ.

Mã trong hình 2.18 chứa ENTITY và ARCHITECTURE. ENTITY xác định các đầu vào và đầu ra của khối phần cứng. Các chuyển mạch SW0, SW1, SW2, SW3 là các bit của đầu vào của kiểu STD_LOGIC. PUSH_BUTTON là hai bit dài, và nó thuộc kiểu STD_LOGIC_VECTOR. Các LEDs của đầu ra là 4 bit dài, và cũng thuộc kiểu STD_LOGIC_VECTOR.

Phần ARCHITECTURE chứa mã mà phần cứng thực hiện. Nó có thể được coi rằng ban đầu nó là rỗng. Nếu mã đã được tải vào FPGA, thì nó sẽ không làm gì cả. Vì vậy ta cần phải bổ xung mã giữa các câu lệnh BEGIN và END trong khối ARCHTECTURE.

Các thay đổi mã được làm như sau:

1. Đưa vào mã dưới đây giữa các câu lệnh BEGIN và END trong khối ARCHITECTURE. Điều này thực hiện tính năng đã liệt kê trong bảng 2.7.

   ```
   LEDs(0)<=SW0 or SW1;
   LEDs(1)<=SW2 or SW3;
   LEDs(2)<=(SW0 or SW1) and (SW2 or SW3);
   LEDs(3)<=PUSH_BUTTON(0) or PUSH_BUTTON(1);
   ```

2. Lưu file nhờ File->Save trên thực đơn chính. Sau khi soạn thảo, file nguồn top_level.vhd sẽ xuất hiện như chỉ ra ở hình 2.18.

```
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity top_level is
33      Port ( SW0 : in  STD_LOGIC;
34             SW1 : in  STD_LOGIC;
35             SW2 : in  STD_LOGIC;
36             SW3 : in  STD_LOGIC;
37             PUSH_BUTTON : in  STD_LOGIC_VECTOR (1 downto 0);
38             LEDs : out  STD_LOGIC_VECTOR (3 downto 0));
39  end top_level;
40
41  architecture Behavioral of top_level is
42
43  begin
44    LEDs(0) <= SW0 or SW1;
45    LEDs(1) <= SW1 or SW2;
46    LEDs(2) <= (SW0 or SW1) and (SW2 or SW3);
47    LEDs(3) <= PUSH_BUTTON(0) or PUSH_BUTTON(1);
48  end Behavioral;
```

Hình 2.19: nội dung file top level.vhd hiển thị trong Project Navigator sau khi soạn

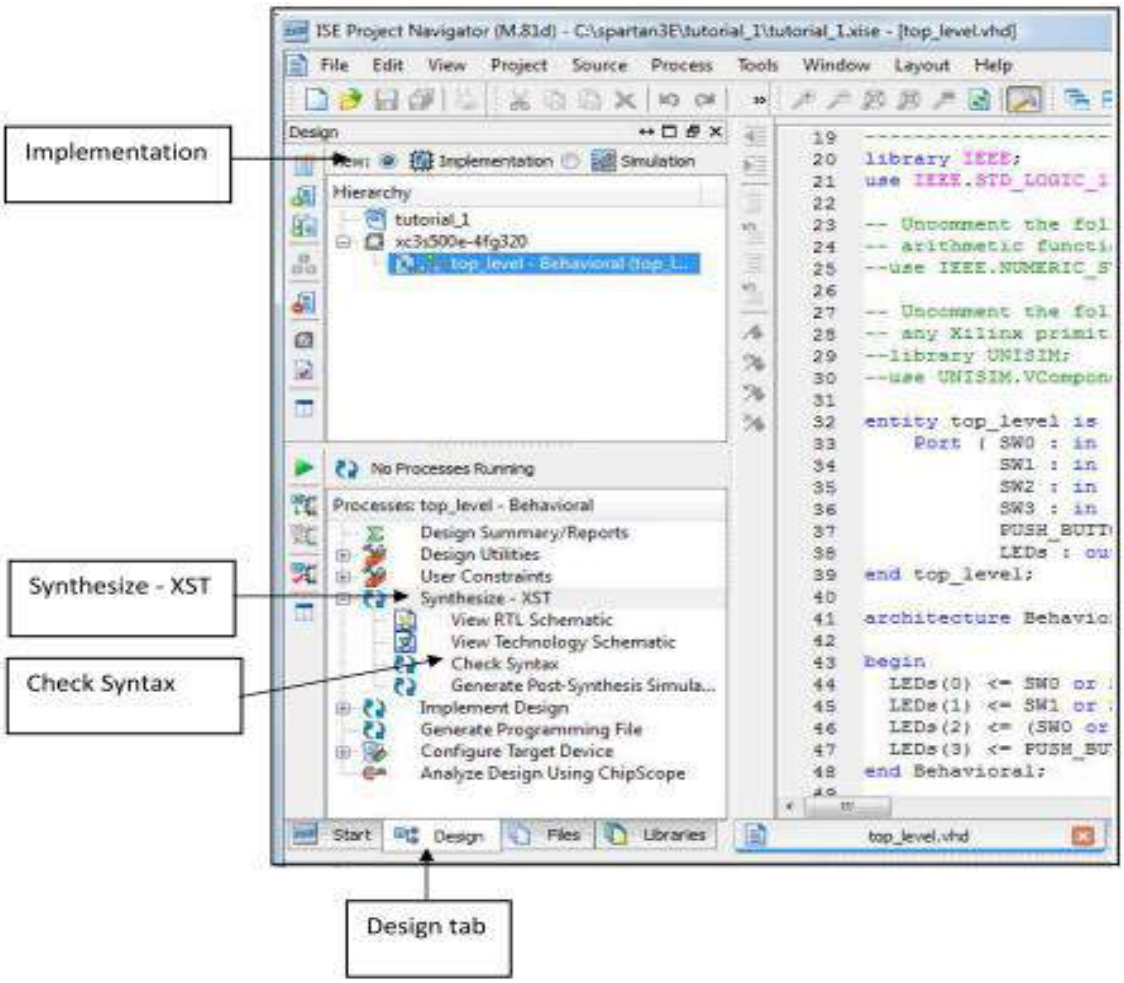
2.2.3.6. Kiểm tra cú pháp

Bước tiếp theo là kiểm tra cú pháp, để kiểm tra mã VHDL có đưa vào đúng hay không. Có các bước sau đây tham chiếu đến màn hình của Project Navigator cho ở hình 2.20.

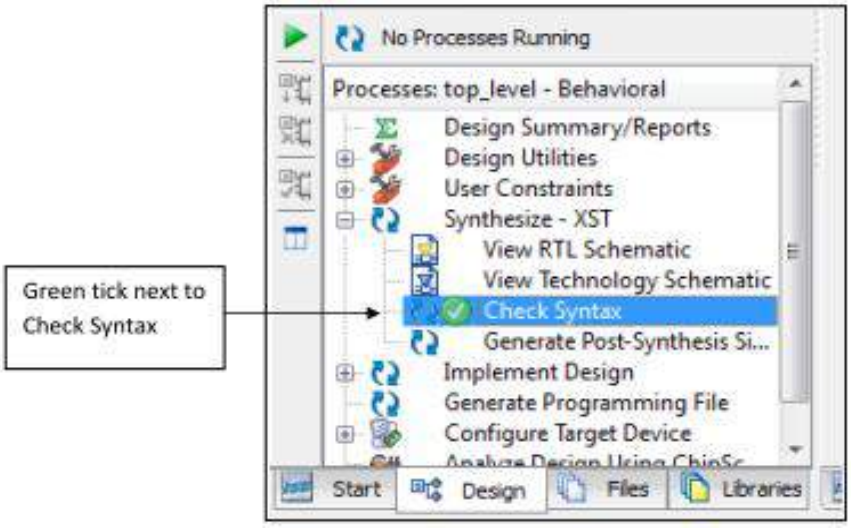
1. Click **Implementation** trên cửa sổ Design ở góc trên bên trái của cửa sổ Design

2. Xem xét Design tab đã được chọn.

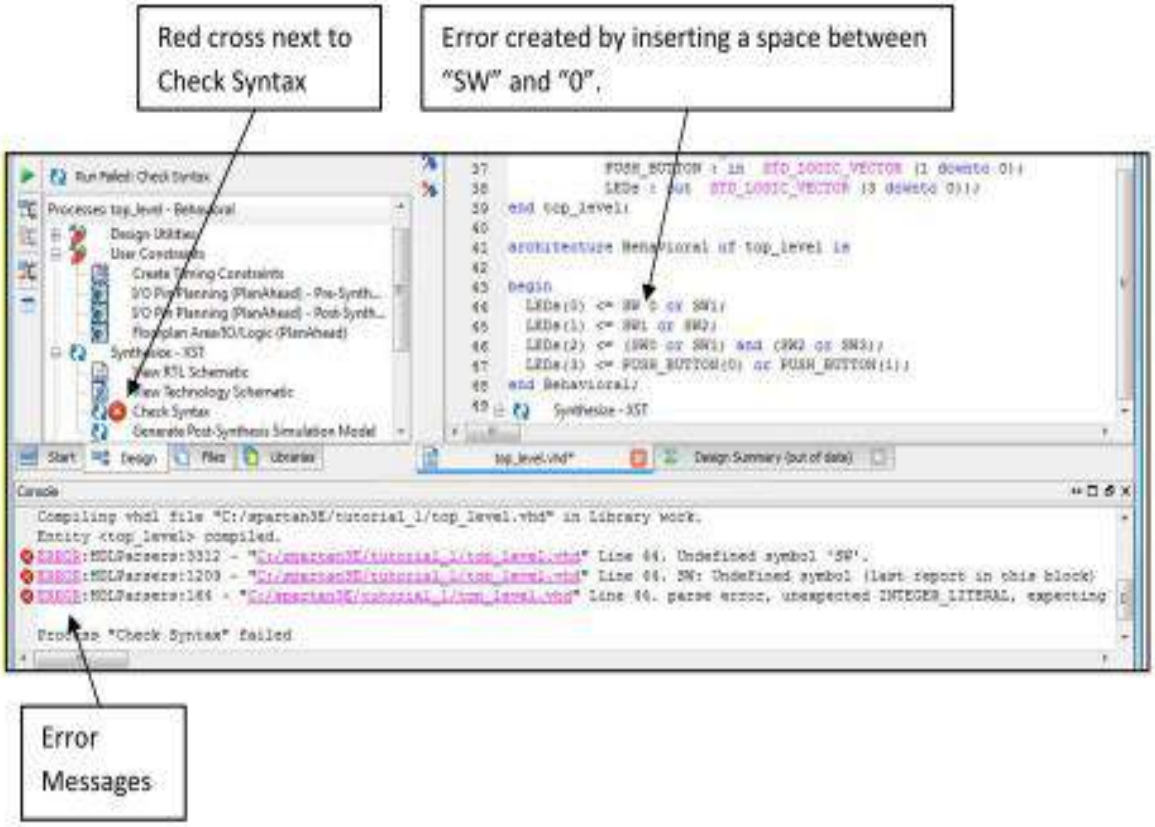
3. Click trên '+' next to **Synthesize-XST**. Điều này sẽ mở rộng ra các khoản hiển thị khác nhau, gồm cả **Check Syntax.** Click **Check Syntax** để chạy kiểm tra cú pháp. Nếu cú pháp mã không sai, sẽ có checked mầu xanh lá cây bên cạnh Check Syntax.



Hình 2.20: Project Navigator với mở rộng



Hình 2.21: Green tick next cho kiểm tra cú pháp



Hình 2.22: Ví dụ, trong đó lỗi đã xuất hiện dấu chéo đỏ ở chỗ kiểm tra lỗi

2.2.3.7. Gán chân tín hiệu

Xét lại mã VHDL cho thực thể top_level:

```
ENTITY top_level IS
        PORT (SW0: in STD_LOGIC;
                SW1:  in STD_LOGIC;
                SW2:  in STD_LOGIC;
                SW3:  in STD_LOGIC;
                PUSH_BUTTON: in STD_LOGIC_VECTOR (1 downto 0);
                LEDs: out STD_LOGIC_VECTOR (3 downto 0)); END top_level;
```

Ta muốn nối các đầu vào và các đầu ra của ENTITY top_level với các chuyển mạch, các nút ấn buttons và các LEDs trên bảng Spartan-3E. Ví dụ, ta muốn nhận các đầu vào SW0, SW1, SW2, và SW3 từ 4 nút chuyển mạch. Những đầu vào này nối với các chân tín hiệu L13, L14, H18 và N17 của chip FPGA. Tương tự, ta muốn nhận các đầu vào PUSH_BUTTON(0) và PUSH_BUTTON(1) từ hai nút bấm (Push Button) trên bảng Spartan-3E. Trong trường hợp này, ta sẽ sử dụng các Buttons bắc và đông, mà chúng được kết nối với các chân (pins) V4 và H13 của FPGA. Cuối cùng, ta sẽ nối các đầu ra LEDs(0), LEDs(1), LEDs(2), và LEDs(3) với 4 đèn LEDs trên bảng Spartan-3E.

Trong trường hợp này ta sẽ sử dụng 4 đèn LEDs bên phải tương ứng với các chân tín hiệu F12, E12, E11 và F11.

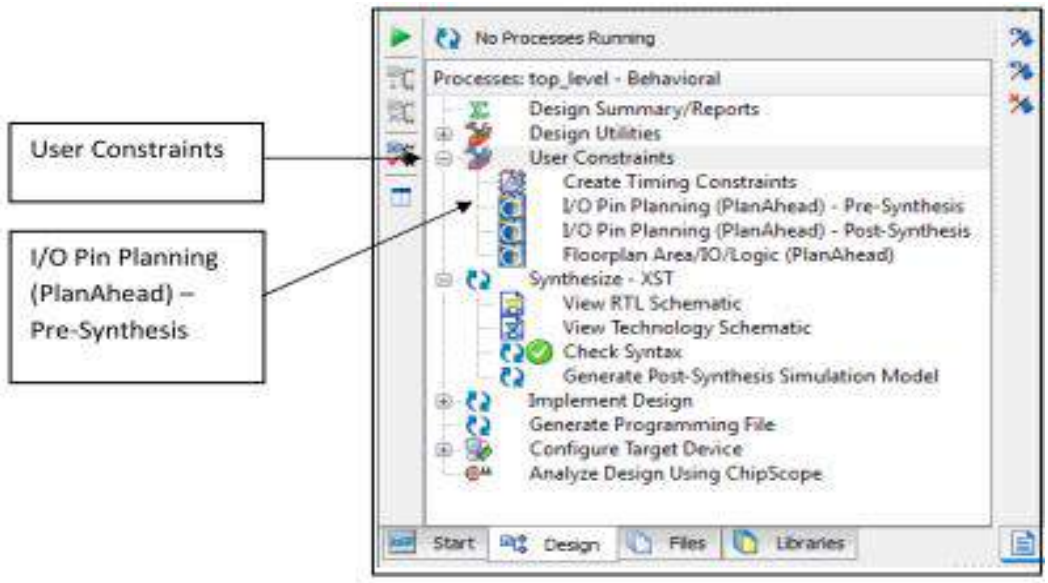
Đối với từng cổng vào và ra của ENTITY top_level, bảng 2.7 liệt kê tên của thiết bị trên bảng Spartan-3E mà ta muốn nối với cổng.

Port name	Spartan-3E board device name	Description	FPGA pin
SW0	SW0	Slider switch	L13
SW1	SW1	Slider switch	L14
SW2	SW2	Slider switch	H18
SW3	SW3	Slider switch	N17
PUSH_BUTTON(0)	BTN_North	Push button	V4
PUSH_BUTTON(1)	BTN_East	Push button	H13
LEDs(0)	LD0	LED	F12
LEDs(1)	LD1	LED	E12
LEDs(2)	LD2	LED	E11
LEDs(3)	LD3	LED	F11

Bảng 2.1: Các cổng input/output của ENTITY top_level

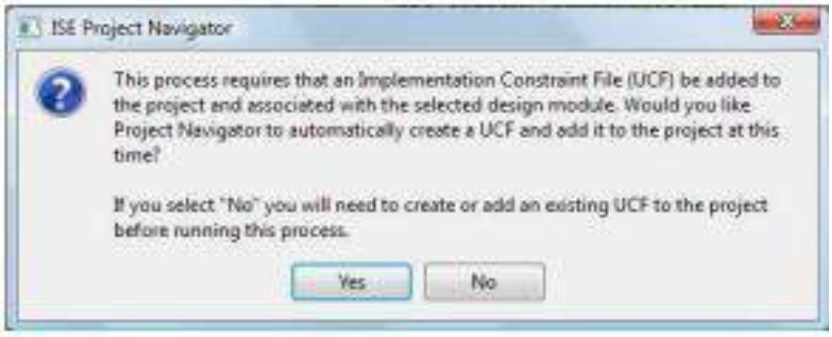
Các bước sau đây được sử dụng để nối các đầu vào và đầu ra với các chuyển mạch, buttons và LEDs trên bảng Spartan-3E:

1. Một Clich vào '+' **User Constrains**. Sẽ hiển thị mở rộng ra các mục, kể cả **I/O Pin Planning (PlanAhead) – Pre-Synthesis** (hình 2.23).



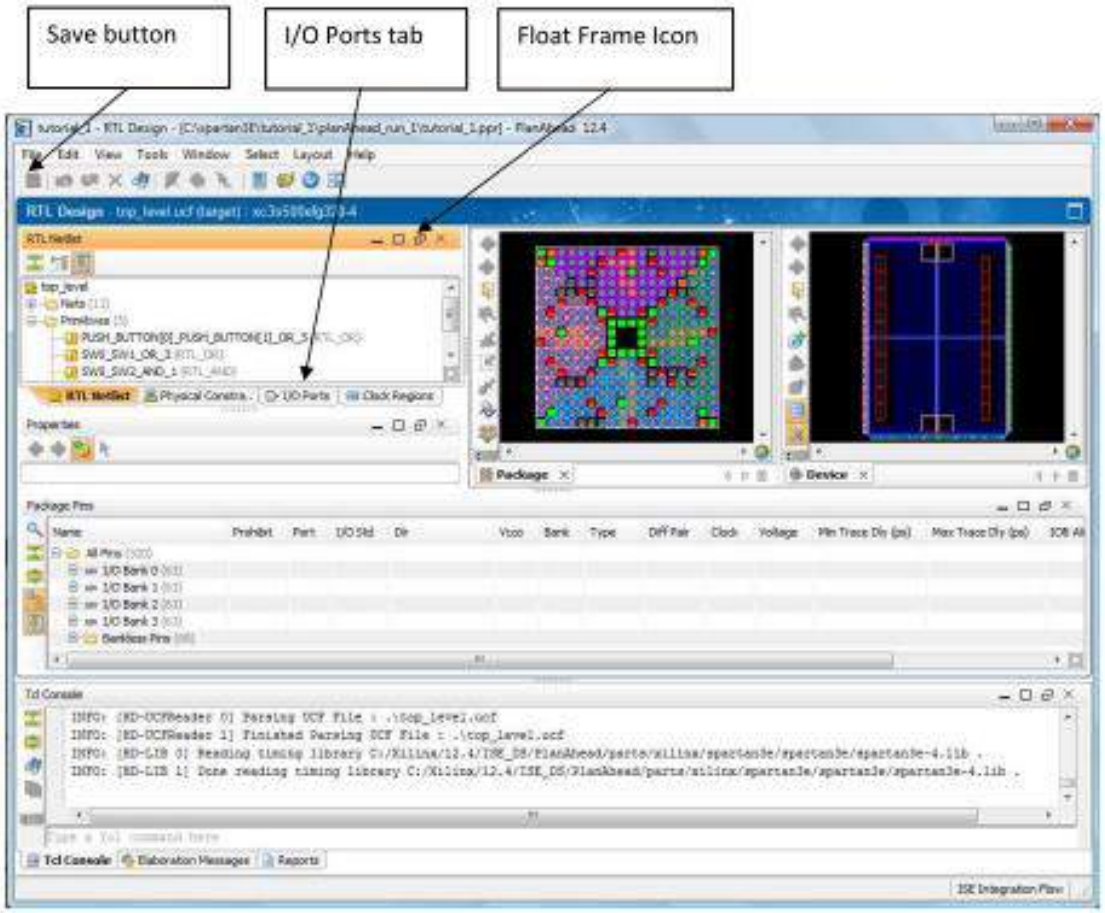
Hình 2.23: Một khoản của màn hình **Project Navigator**, với **User**

2. Hai Click **I/O Pin Planning (PlanAhead) – Pre-Synthesis**. Cửa sổ của hình 4.18 sẽ xuất hiện, yêu cầu OK để tạo UCF file. Click **Yes**.

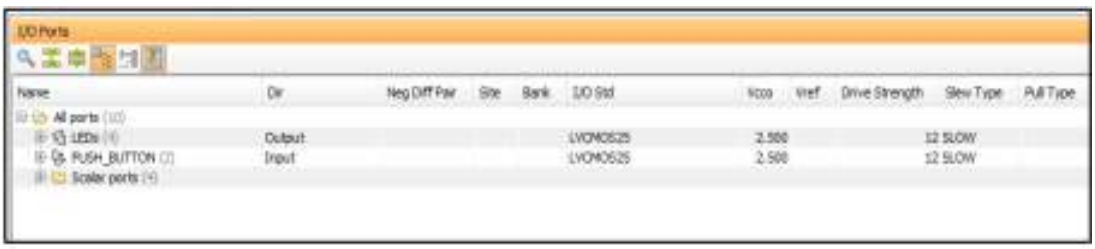


Hình 2.24: Hộp hội thoại yêu cầy tạo UCF file

3. Click **Yes** ở hộp hội thoại (hình 2.24), cửa sổ **PlanAhead** xuất hiện (hình 2.25). Click vào **I/O Ports** tab, và sau đó vào **float frame icon**. Sẽ hiển thị **I/O Ports** trong một cửa sổ riêng (hình 2.26).

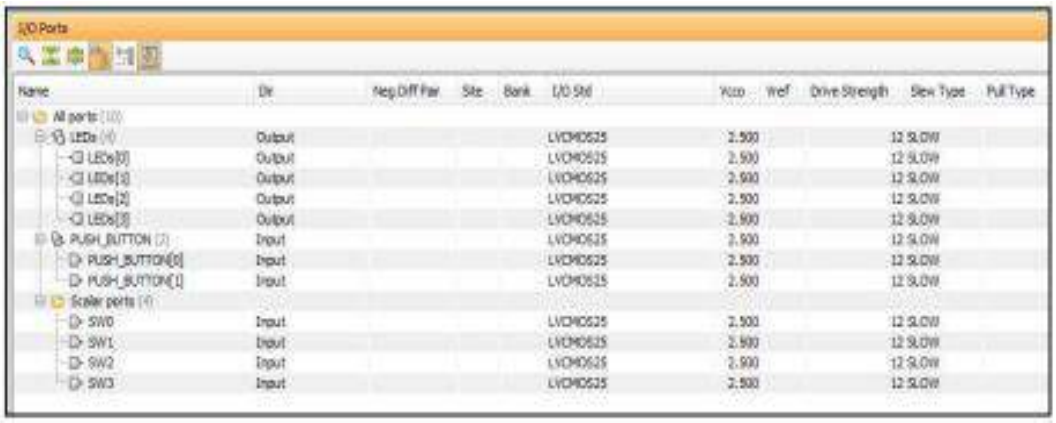


Hình 2.25: cửa sổ **PlanAhead** hiển thị lần đầu



Hình 2.26: Hiển thị cửa sổ **I/O Ports** riêng

4. Click vào '+' tiếp đến LEDs(4), PUSH_BUTTON(2) và Scalar ports ở hình 2.26. Sẽ hiển thị tất cả các inputs/outputs riêng biệt như chỉ ra ở hình 2.27

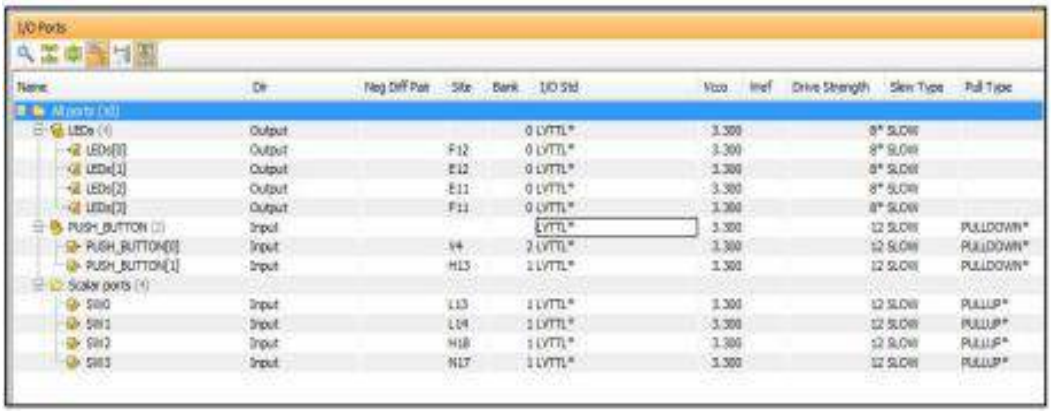


Hình 2.27: Hiển thị cửa sổ **I/O Ports** mở rộng đến các cổng riêng

5. Đưa vào các cột **Site, I/O Std, Drive Strength, Slew Type** và **Pull Type**, bằng các giá trị đã cho ở bảng 2.2. Các cột **Bank** và **Ccco** sẽ được điền tự động khi các cột **Site** và **I/O Std** được điền giá trị vào.

Port	Site	I/O Std	Drive Strength	Slew Type	Pull Type
LEDs[0]	F12	LVTTL	8	SLOW	
LEDs[1]	E12	LVTTL	8	SLOW	
LEDs[2]	E11	LVTTL	8	SLOW	
LEDs[3]	F11	LVTTL	8	SLOW	
PUSH_BUTTON[0]	V4	LVTTL			PULLDOWN
PUSH_BUTTON[1]	H13	LVTTL			PULLDOWN
SW0	L13	LVTTL			PULLUP
SW1	L14	LVTTL			PULLUP
SW2	H18	LVTTL			PULLUP
SW3	N17	LVTTL			PULLUP

Bảng 2.2: Các giá trị để đưa vào cửa sổ I/O Ports



Hình 2.28: Hiển thị cửa sổ **I/O Ports** với các giá trị đã được điền

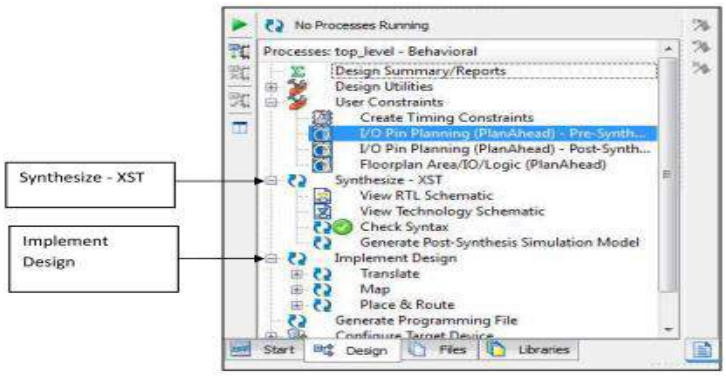
6. Click Save trên cửa sổ **PlanAhead** (vị trí của nút Save chỉ ra ở hình 2.25), tên file là top_level.ucf để lưu các chân tín hiệu đã được đưa vào. Cửa sổ **PlanAhead** có thể được đóng lại ở trang này.

2.2.3.8. Synthesize, Translate, Map, và Place & Route

Giai đoạn tiếp theo ta phải thực hiện là các bước: tổng hợp (Synthesize), chuyển đổi (Translate), Sắp xếp (Map), và đặt vi trí & tuyến (Place & Route). Các bước này được thực hiện nhờ Project Navigator, và được mô tả ngắn gọn như sau:

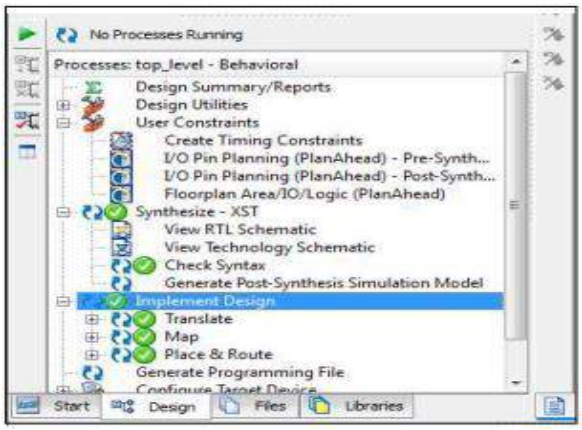
- Synthesize: tạo các danh sách netlists cho từng file nguồn.
- Translate: gộp nhiều file vào một netlist.
- Map: thiết kế được sắp xếp vào các mảnh và các khối I/O.
- Place & Route: thiế kế được đặt vào chip và kết nối các thành phần.

1. Như chỉ ra ở hình 2.29, click vào '+' đến Implement Design. Sẽ mở rộng đến Translate, Map và Place & Route



Hình 2.29: một khoản của màn hình **Project Navigator**, với mở ra

2. Hai Click vào **Implement Design**. Sẽ gây ra **Synthesize-XST** chạy lần đầu. Sau đó, **Translate, Map và Place & Route** sẽ chạy lần lượt. Khi mỗi một bước thực hiện xong, một dấu kiểm mầu xanh lá cây sẽ xuất hiện bên cạnh. Sau khi tất cả giai đoạn hoàn thành, màn hình **Project Navigator** sẽ xuất hiện như cho ở hình 2.30.

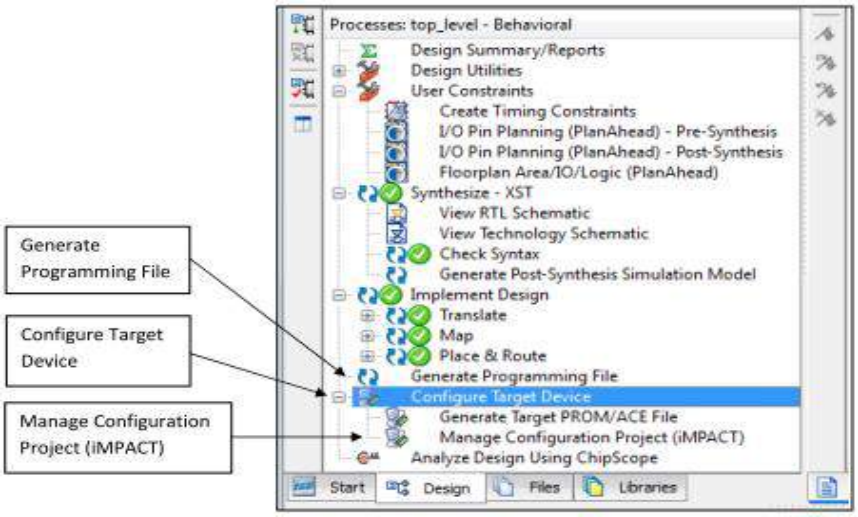


Hình 2.30: một khoản của màn hình **Project Navigator**, với mở ra **Implement Design,** sau đó **Translate, Map** và **Place & Router** đã chạy

2.2.3.9. Tải thiết kế lên bảng Spartan-3E starter

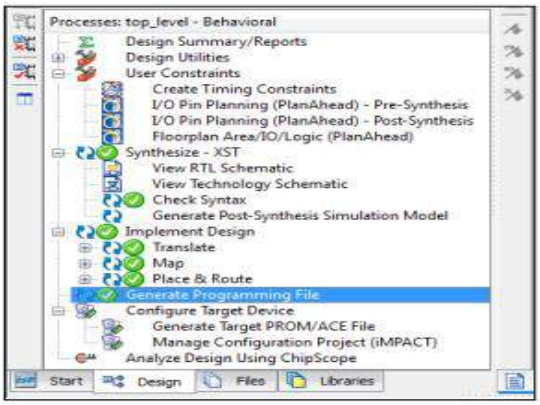
Các bước tiếp theo là tạo file chương trình, và tải nó vào bảng Spartan-3E nhờ sử dụng iMPACT.

1. Một Click vào '+' ở **Configure Target Device** (hình 2.31), sẽ mở rộng đến lựa chọn **Manage Configuration Project (iMPACT)**.



Hình 2.31: Một khoản của màn hình **Project Navigator**, với mở ra

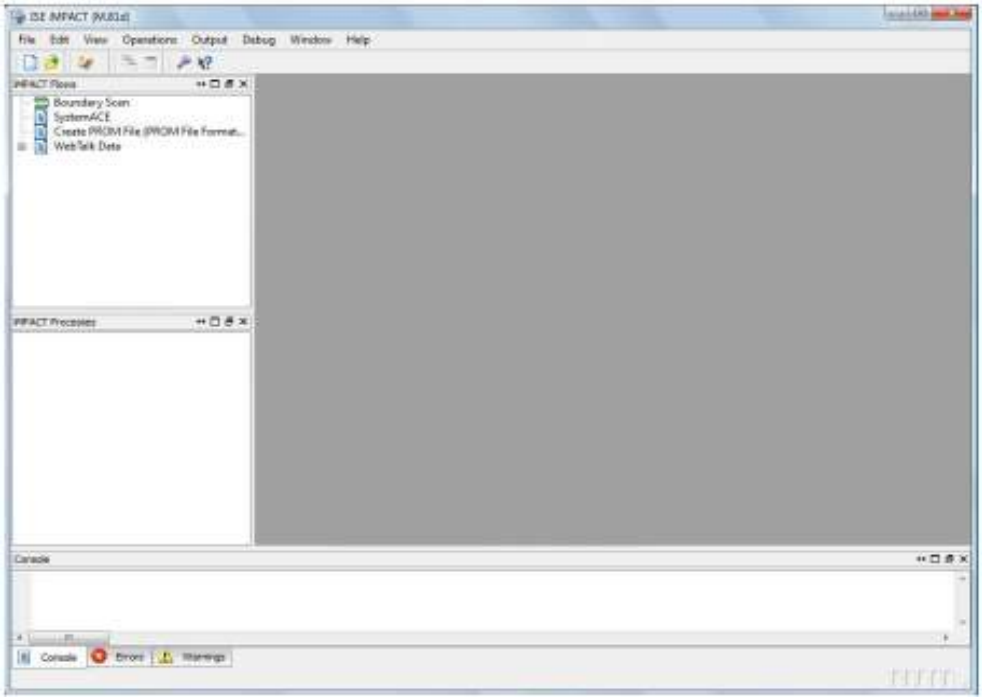
2. Hai Click vào **Generate Programming File**. Khi chạy xong file này, một dấu kiểm mầu xanh lá cây xuất hiện bên cạnh **Generate Programming File** như chỉ ra ở hình 2.32.



Hình 2.32: Một khoản của màn hình **Project Navigator**,
sau khi **Generate Programming File** chạy xong

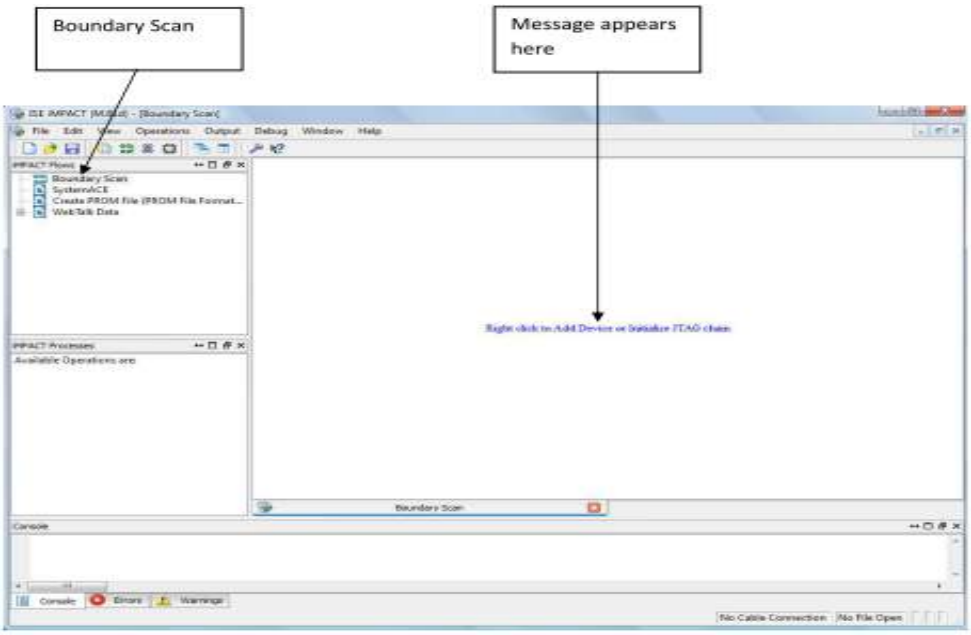
3. Nối cáp nguồn và cáp USB với bảng Spartan-3E. Cắm cáp USB từ bảng Spartan-3E vào PC, và bật nguồn cho bảng Spartan-3E.

4. Clich hai lần lên **Manager Configuration Project (iMPACT).** Cửa sổ iMPACT sẽ hiển thị như hình 2.33.



Hình 2.33: Cửa sổ ban đầu của **iMPACT**

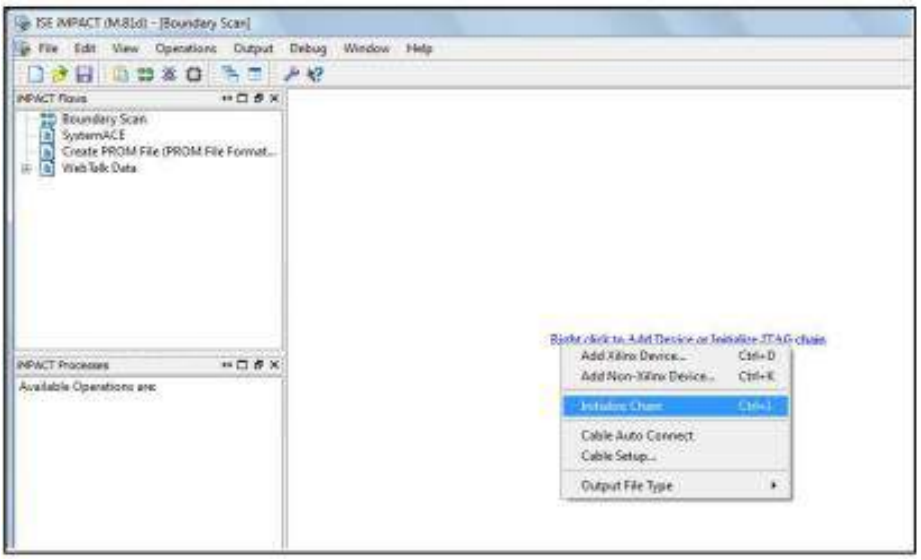
5. Click hai lần vào Boudary Scan như chỉ ra ở hình 2.34. Thông báo "Right click to Add Device or Initialize JTAG Chain" sẽ phải xuất hiện phía bên phải màn hình.



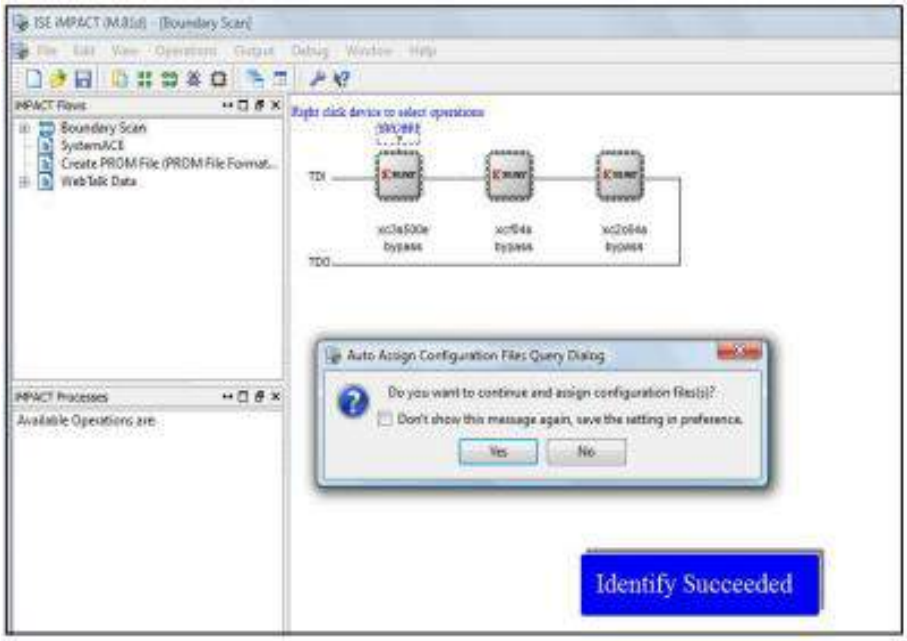
Hình 2.34: Cửa sổ **iMPACT**, sau khi click hai lần lên

6. Click chuột phải lên "Right click to Add Device or Initialize JTAG Chain", và chọn **Initialise Chain**, như hiển thị ở hình 2.35.

7. Sau một lúc, một hình ảnh của "chain" sẽ xuất hiện, theo với một thong báo **Identify Succeeded** trong hộp mầu blue (hình 2.36). Chip đầu tiên, **xc3s500e**, là chip FPGA mà ta muốn lập trình. Hai chip khác, **xcf40**s và **xc2c64a** trên bảng sẽ bỏ qua. Hộp hội thoại yêu cầu "Do you to continue and select configuration file (s)?" sẽ xuất hiện như chỉ ra ở hình 2.36).

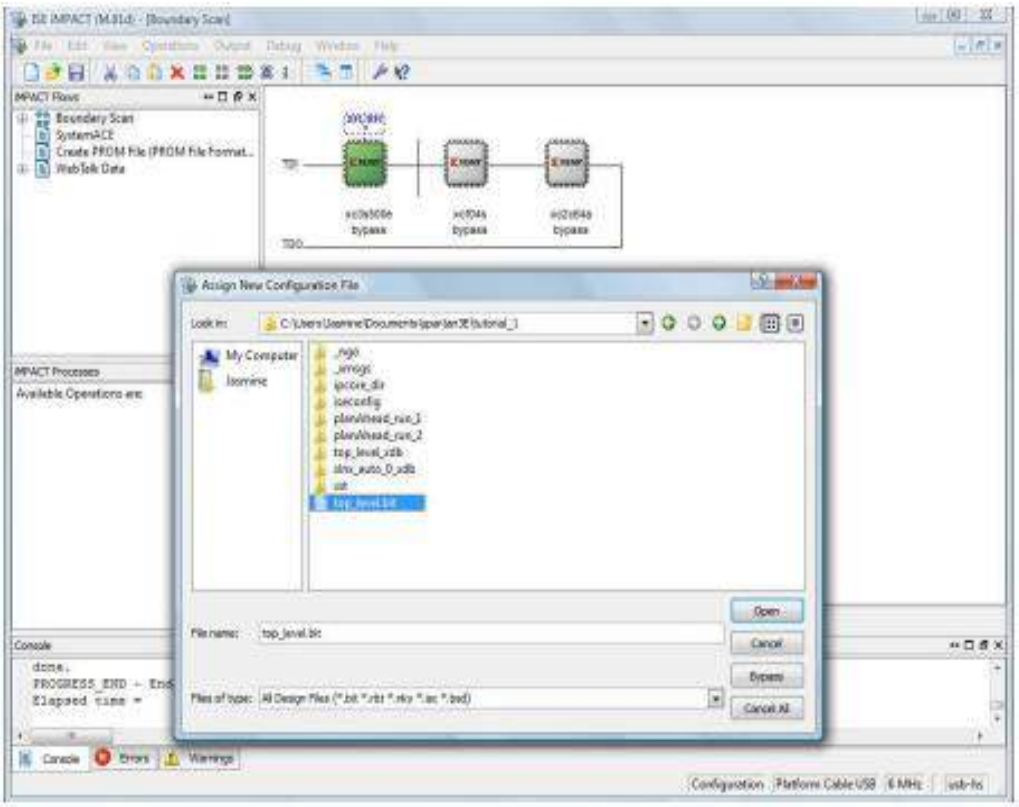


Hình 2.35: Cửa sổ **iMPACT**, hiển thị chọn **Initialize Chain**

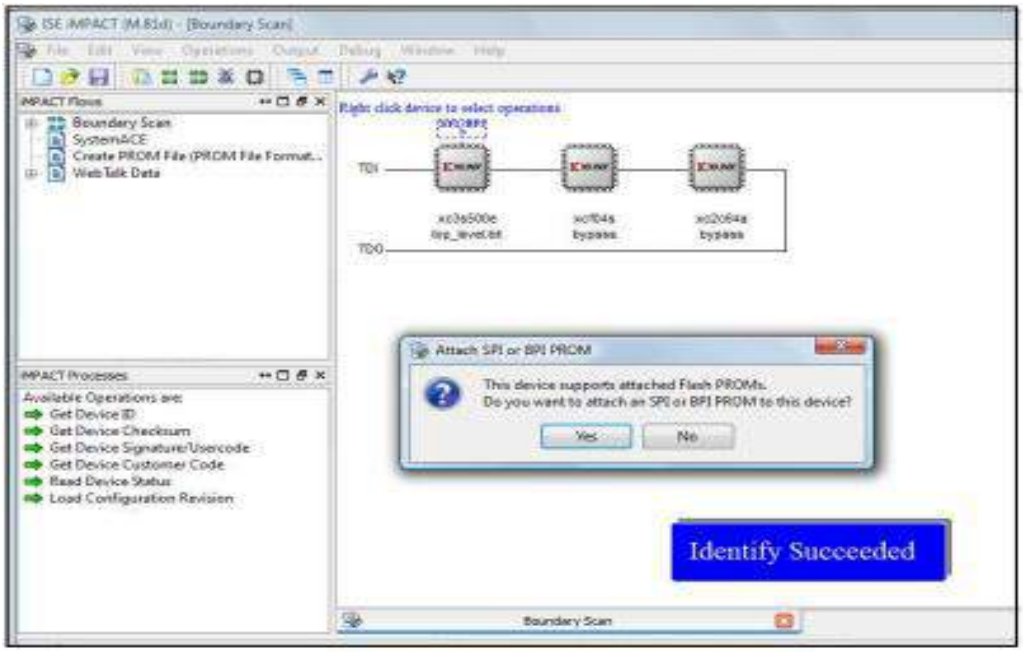


Hình 2.36: cửa sổ **iMPACT**, gán các file cấu hình

8. Cửa sổ **Assign New Configuration File** sẽ xuất hiện (hình 2.37). chọn file "top_level.bit", và click **Open**. Điều này liên kết file top_level.bit với **xc3s500e**.



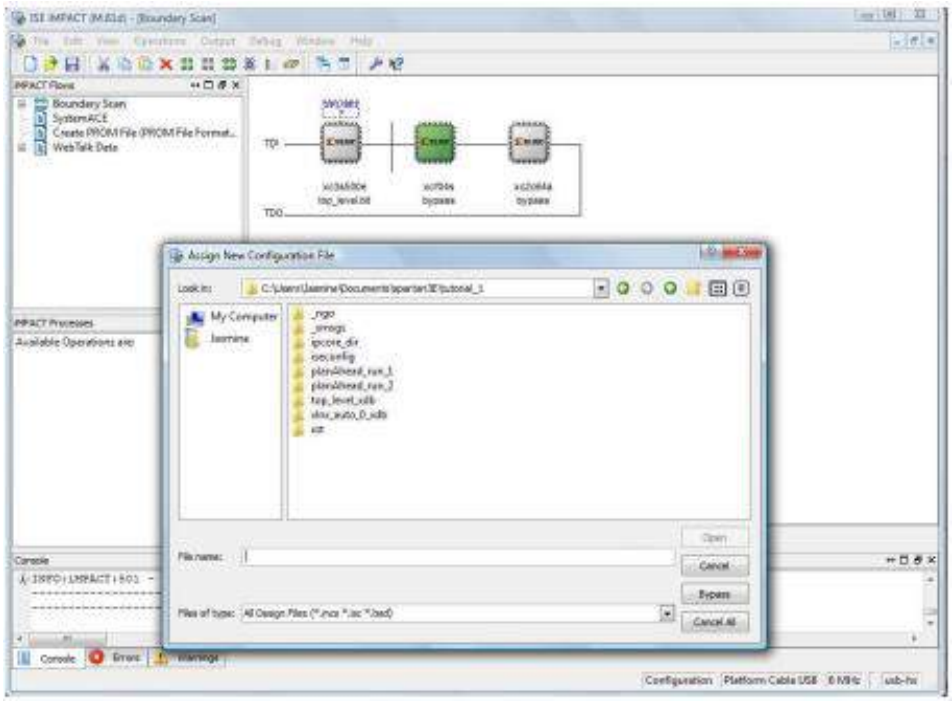
Hình 2.37: Cửa sổ **iMPACT**, gán file cấu hình cho **xc3e500e**



Hình 2.38: Cửa sổ **iMPACT**, hộp hội thoại yêu cầu có gắn SPI hay BPI PROM hay không.

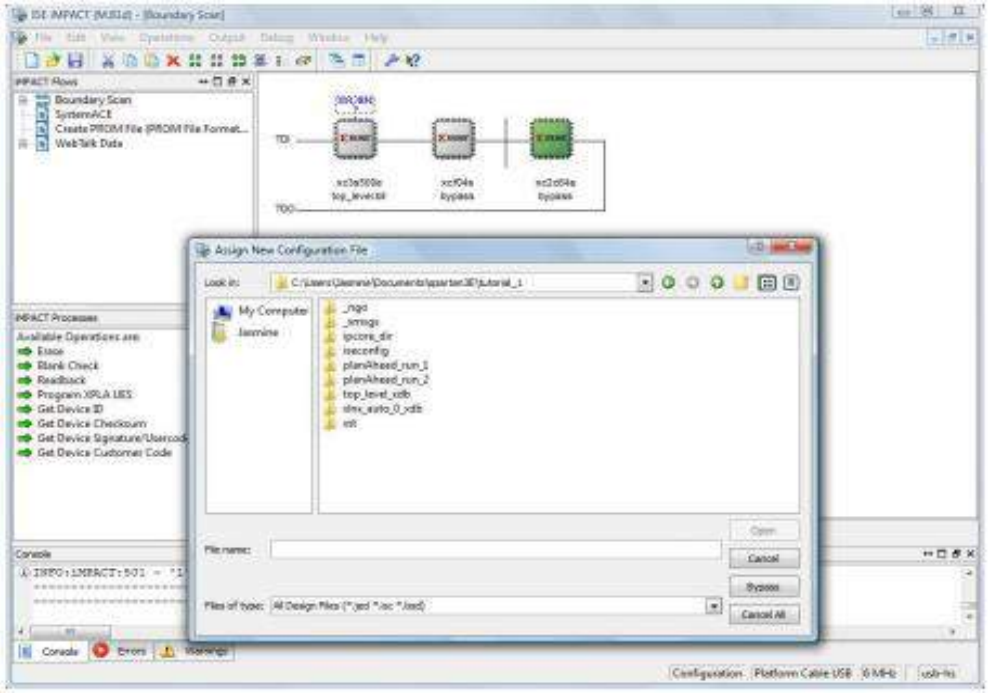
9. Một thông báo " This device support attached flash PROMs. Do you want to attach an SPI or BPI PROM this device?" sẽ xuất hiện (hình 2.38). Điều này chưa cần cho thiết kế này. Click **No**.

10. Cử sổ **Assign New Configuration File** sẽ xuất hiện trở lại (hình 2.39). Trong trường hợp này, Click **Bypass**. Điều này đảm bảo bỏ qua **xcf04s.**



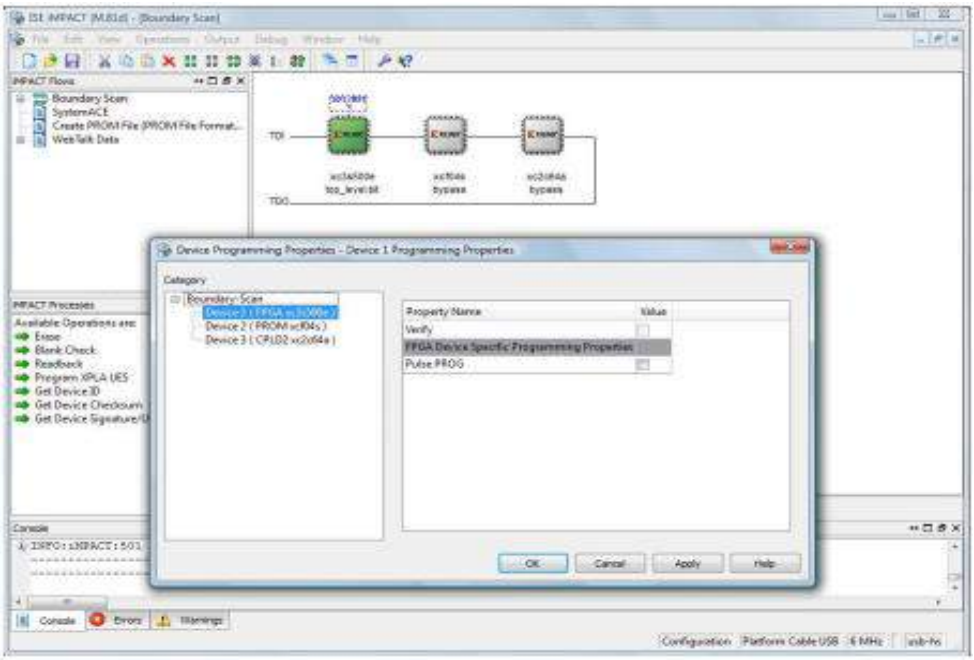
Hình 2.39: Cửa sổ **iMPACT**, bỏ qua **xcf40s**.

11. Cửa sổ **Assign New Configuration File** sẽ xuất hiện lần nữa (hình 2.40). Lại Click **Bypass**. Điều này đảm bảo bỏ qua **xc2c64a**.



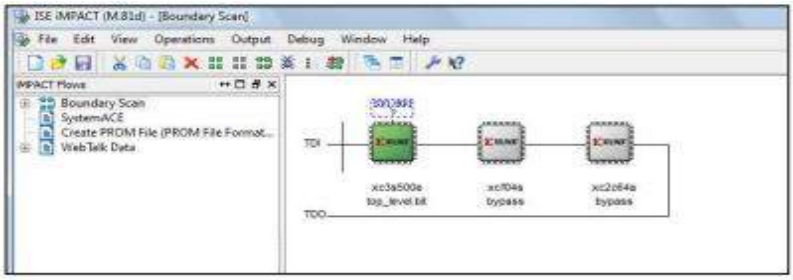
Hình 2.40: Cửa sổ **iMPACT**, bỏ qua **xc2c64a**

12. Cửa sổ **Device Programming Properties – Device 1 Programming Properties** sẽ xuất hiện (hình 2.41). Click **OK**.

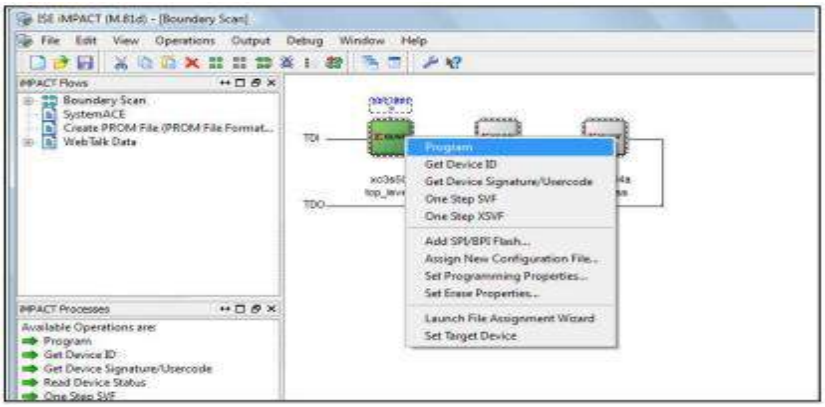


Hình 2.41: Cửa sổ **iMPACT**, hộp hội thoại **Device**

13. Cửa sổ **iMPACT** sẽ xuất hiện như ở hình 2.42. Click vào chip **xc3e500e** (hình 2.43) và chọn **Program**.

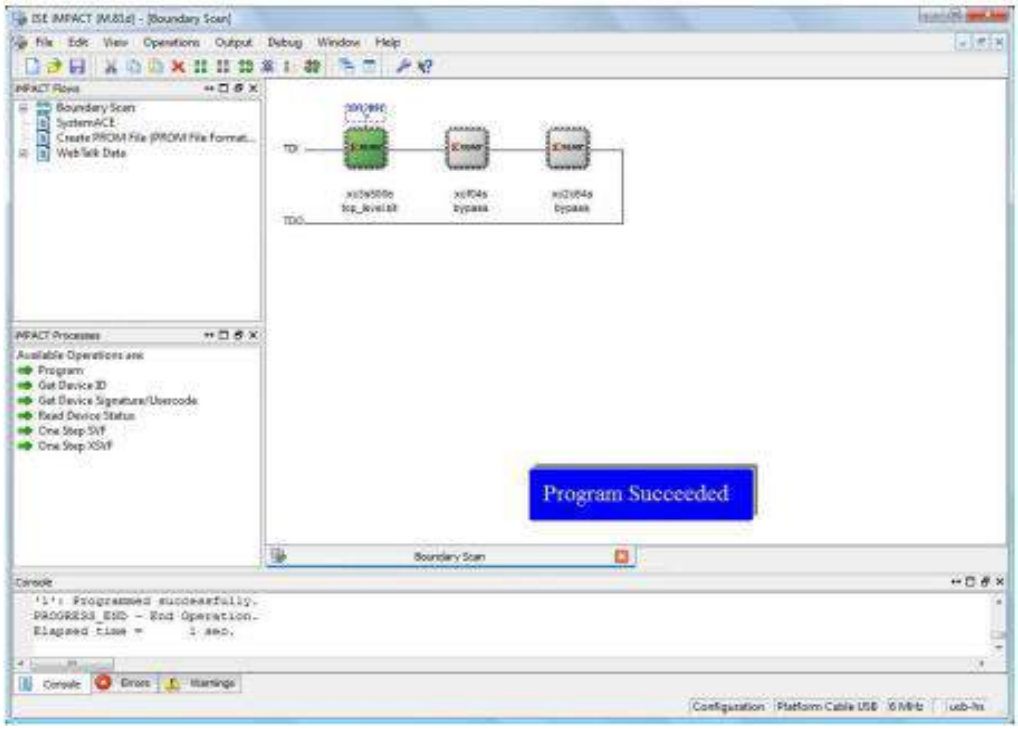


Hình 2.42: cửa sổ **iMPACT**, hiển thị device chain



Hình 2.43: Cửa sổ **iMPACT**, các lựa chọn khi click vào **xc3s500e**

14. Chương trình bây giờ sẽ được tải vào bảng Spartan-3E. Sau khi tải xong, một thông báo sẽ xuất hiện "Program Succeeded" (hình 2.44). Ta có thể thực hiện chạy từng thao tác của chương trình bằng cách click vào từng Available Operations are: Program, Get Device ID, Get Device Signature/Usercode, Read Device Status, One Step SVF, One Step XSVF của cửa sổ **iMPACT Procceses.** Mỗi Operation thành công sẽ có thong báo thành công (Program Successfully, ReadIdcode Successfuly).

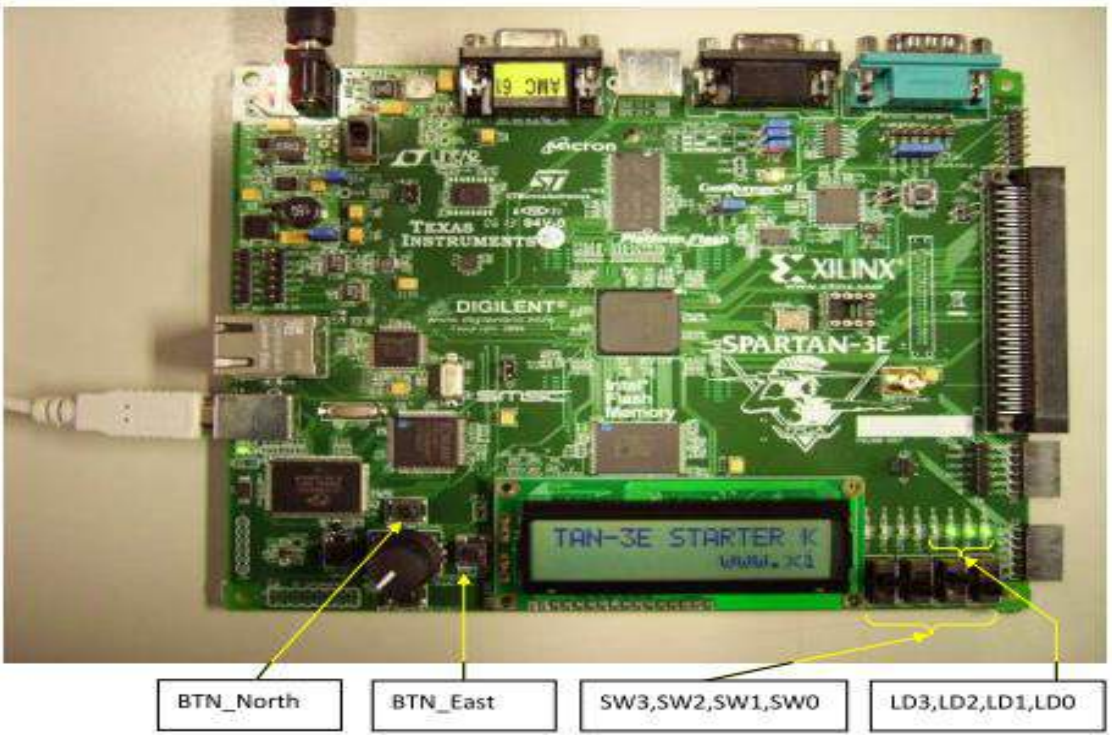


Hình 2.44: Cửa sổ **iMPACT**, sau khi tải thành công chương trình vào bảng Spartan-3E

2.2.3.10. Chạy chương trình trên bảng Spartan-3E.

Bảng Spartan-3E starter kit sau khi tải chương trình chỉ ra ở hình 2.45. Hiển thị trên LCD có thể khác như chỉ ra ở hình này trong quá trình chạy chương trình.

Bây giờ chương trình có thể được kiểm tra trên bảng Spartan-3E starter kit. Nếu chương trình hoạt động tốt thì contact xoay (ở giữa các nút BTN) không có tác dụng. Với các nút chuyển mạch SW0-3, vị trí bật lên trên là vị trí bật (on). LED LD0 phải sáng khi hoặc là SW0 hoặc SW1 ở ON. LED LD1 sẽ phải sáng khi hoặc SW2 hoặc SW3 ở ON. Cả 3 LEDs: LD0. LD1 và LD2 sẽ cùng sáng khi (SW0 hoặc SW1) và (SW2 hoặc SW3) là ON. Cuối cùng LED LD3 sẽ phải sáng khi nút BTN_East hoặc nút BTN_North được ấn xuống (các nút này tự nhả)..



Hình 2.45: Bảng Spartan-3E với chương trình đang chạy

2.3. KẾT LUẬN CHƯƠNG

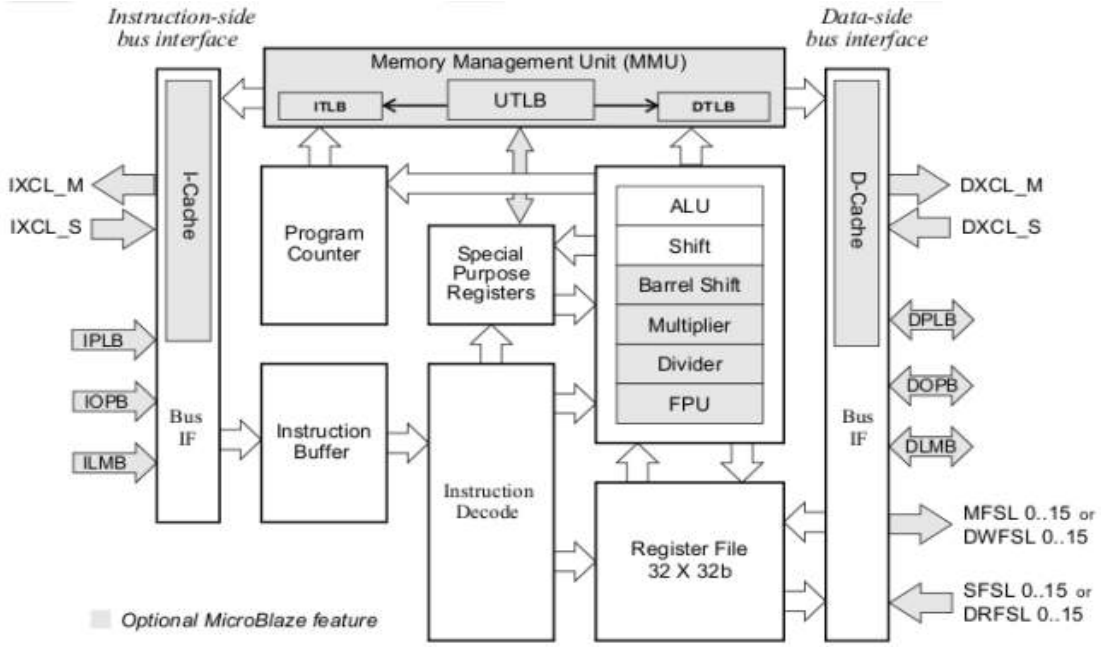
Chương 2 trình bày các nội dung:

- Ngôn ngữ mô tả phần cứng VHDL: Lịch sử VHDL; ứng dụng VHDL; đặc điểm của VHDL.
- Các mức trìu tượng trong thiết kế mạch tích hợp; các tầng trìu tượng của thiết kế VHDL; mô tả các tầng trìu tượng trong thiết kế VHDL.
- Quá trình thiết kế phần cứng bằng VHDL
- Các công đoạn thiết kế bằng VHDL; thiết kế phần cứng trên Xilinx FPGA; các tính năng thiết kế; tài liệu liên quan; công cụ phần mềm thiết kế Xilinx ISE và ví dụ một thiết kế đơn giản với các kết nối 4 nút chuyển mạch (SW0, SW1, SW2, SW3), nút nhấn nhả với contact xoay (Button), và 8 đèn led hiển thị (led0, led1, led2, led3, led4, led5, led6, led7) để chứng minh các bước thiết kế phần cứng bằng VHDL đã đạt được yêu cầu bằng hoạt động của bảng Xilinx Spartan-3E starter board.

CHƯƠNG 3: THIẾT KẾ HỆ VI ĐIỀU KHIỂN LÕI MỀM MICROBLAZE 32-BIT VÀ CÀI ĐẶT ỨNG DỤNG THỬ NGHIỆM

3.1. VI ĐIỀU KHIỂN MICROBLAZE 32-BIT

3.1.1. Kiến trúc của Microblaze



Hình 3.1: Kiến trúc của Microblaze

Các bộ vi xử lý có sẵn dùng cho dòng **FPGA** (Field Programmable Gate Arrays) của Xillinx sử dụng với các công cụ phần mềm có trong phần mềm EDK (Embedded Development kit) được phân thành hai loại: Bộ vi xử lý mềm Microblaze. Bộ vi xử lý cứng đã được nhúng sẵn Power PC. Microblaze là bộ vi xử lý được dùng hầu hết trong FPGA các dòng như Spartan-II, Spartan-III, Virtex của hãng Xillinx. Microblaze là một vi điều khiển ảo, nó tồn tại dưới dạng phần mềm đã được phát triển của hãng Xillinx, nhà thiết kế có thể thiết lập các thông số để sử dụng đối với vi điều khiển này (UART, các cổng vào ra ngoại vi, ..) thông qua phần mềm EDK. MicroBlaze là bộ xử lý mềm nhúng 32-bit của Xillinx. Tập lệnh rút gọn RISC (Reduced Instruction Set Computer), với các bus riêng biệt để truy xuất dữ liệu và lệnh từ bộ nhớ on-chip và bộ nhớ ngoài tại cùng một thời điểm.

Kiến trúc của MicroBlaze có các đặc điểm sau: Từ lệnh 32 bit với 3 toán hạng và 2 chế độ định địa chỉ. Đường bus 32 bit địa chỉ. Một khối ghi dịch. Hai cấp độ ngắt. Khối ALU (Arithmetic Logic Unit): gồm các bộ cộng/trừ, ghi dịch/logic, nhân. Trong đó có các thành phần sau:

- **Bus IF:** (Bus interface) đường bus giao tiếp
- **Instruction Buffer**: bộ đệm lệnh
- **Instruction** Decode: bộ giải mã lệnh
- **Program Counter**: bộ đếm chương trình
- **Add/Sub**: khối cộng/trừ
- **Shift/Logical**: khối ghi dịch/lôgic
- **Multiply**: khối nhân
- **Register File**: tập thanh ghi dữ liệu gồm 32 thanh ghi 32-bit
- **32-bit barrel shifter** (lựa chọn)

MicroBlaze cung cấp 3 giao tiếp bộ nhớ:

- **LMB**: Local Memory Bus
- **PLB**: Processor Local Bus
- **OPB** (On-Chip Peripheral Bus) và **XCL** (Xillinx Cache Link)

Microblaze sử dụng LMB bus cho kết nối nhanh với on-chip memory (FPGA BRAM), trong đó, chia ra **ILMB**: Instruction interface, Local Memory Bus: giao tiếp lệnh theo chuẩn bus LMB, chỉ dùng cho giao tiếp BRAM, và **DLMB**: Data interface, Local Memory Bus: giao tiếp dữ liệu theo chuẩn bus LMB, chỉ dùng cho giao tiếp BRAM. Các giao tiếp với bộ nhớ cache cũng được chia ra bus kết nối với cache lệnh và cache kết nối dữ liệu, trong đó **IXLC**: Instruction side Xillinx Cache Link Interface (cặp liên kết đơn FSL chủ/tớ), và **DXCL**: Data side Xillinx Cache Link Interface (cặp liên kết đơn FSL chủ/tớ). Bus vào/ra đầu tiên của Microblaze là AXI interconnect, là bus giao dịch system-memory với khả năng chủ/tớ (master/slave). Các giao tiếp đồng xử lý được hỗ trợ bởi các kết nối AXI4-Stream connections. Các giao tiếp với ngoại vi được thực hiện qua các bus OPB theo chuẩn IBM CoreConnect, chuẩn này cung cấp ba loại bus giành cho kết nối liên tục đa lõi, các thư viện macro và logic người dùng: Bus lõi xử lý nội PLB (Processor Local Bus), Bus ngoại vi on-chip OPB (On-chip Peripheral Bus), Bus thanh ghi điều khiển thiết bị DCR (Device Control Register). Với OPB có chia ra: **IOPB**: Instruction interface, và **DOPB**: Data interface, On-chip Peripheral Bus: giao tiếp dữ liệu theo chuẩn bus OPB. Toàn bộ các ngoại vi qua giao tiếp OPB như sau:

- Watchdog Timer/Timebase
- General purpose Timer/Counters
- Interrupt Controller

- SRAM Controlller
- Flash Memory Controller
- ZBT Memory Controlller
- BRAM Controller
- DDR Controller
- SDRAM Controller
- UART Lite
- General purpose I/O
- SPI
- I2C (evaluation version)
- UART 16450/550 (evaluation version)
- Ethernet 10/100 MAC (evaluation version)

Ngoài ra, nhà thiết kế có thể xác định và bổ xung các ngoại vi cho các chức năng yêu cầu, như một giao tiếp cho thiết kế bên trong FPGA.

3.1.2. Các định dạng dữ liệu và tập lệnh của Microblaze

Tập lệnh rút gọn RISC của Microblaze sử dụng hai định dạng lệnh 32-bit:

Định dạng loại A: được các lệnh register-register sử dụng. Nó gồm có trường opcode, một thanh ghi toán hạng đích và hai thanh ghi nguồn.

Định dạng loại B: được các lệnh register-immediate sử dụng. Nó gồm có trường opcode, một thanh ghi đích và một thanh ghi nguồn, và một giá trị nguồn trực tiếp 16-bit.

Microblaze sử dụng kiến trúc Load/Store của RISC, trong đó, có thể truy cập đến bộ nhớ theo ba kích thước dữ liệu: Byte (8 bits), Halfword (16 bits), và Word (32 bits)

Hạt nhân của kiến trúc RISC của Microblaze dựa vào tập thanh ghi 32-bit LUT RAM với các lệnh truy cập riêng bộ nhớ dữ liệu và bộ nhớ lệnh. Microblaze hỗ trợ kết nối các khối RAM (BRAM) cả trong chip và ngoài chip.

Cũng như các vi xử lý kiến trúc RISC khác, Microblaze có kiến trúc đường ống lệnh (pipeline architecture) với ba giai đoạn thực hiện lệnh: đọc lệnh IF (Instruction Fetch), giải mã lệnh ID (Instruction Decode), và thực hiện lệnh IE (Instruction Execution). Các quá trình đẩy dữ liệu (data forwarding), trễ chờ trong đường ống (pipeline stall) và thực hiện các rẽ nhánh (branches) được thực hiện tự động trong phần cứng.

Lõi xử lý mềm Microblaze được đưa vào Kit phát triển hệ nhúng của Xilinx EDK (Embedded Development Kit) và được cung cấp theo thỏa thuận bản quyền của phí nhà cung cấp Xilinx Core. Bản quyền này cho phép người dùng sử dụng Microblaze trong các thiết kế hạn chế trên Xilinx FPGA.

3.1.3. Hiệu năng của Microblaze

Hiệu năng của Microblaze phụ thuộc vào cấu hình của bộ xử lý và kiến trúc của FPGA và cấp độ tốc độ. Cấu hình của Microblaze phụ thuộc vào cấp độ tích hợp của FPGA. Với FPGA trên Xilinx Spartan-3E starter board gồm khoảng 500000 Logic Elements, có thể cấu hình 2 nhân xử lý 32-bit. Với những chip FPGA có tích hợp nhiều LE hơn, ta có đạt được cấu hình đến 8 nhân xử lý. Trong những loại Xilinx FPGA cho thiết kế SoC hay NoC, ta có thể tạo ra mạng kết nối 9 đơn vị xử lý 32-bit. Khi đó có thể đạt được hiệu năng của các hệ nhúng FPGA hiệu năng cao. Đã có những thiết kế siêu máy tính với cấu trúc các nút xử lý gồm có các bộ đồng xử lý là các khối kết hợp FPGA và vi xử lý [7].

3.2. THIẾT KẾ HỆ NHÚNG ĐƠN GIẢN VỚI MICROBLAZE

3.2.1. Bảng phát triển trên FPGA Xilinx Starter-3E 500E

Bảng phát triển Xilinx Spartan-3E 500E starter cho ở hình 2.45 có những thành phần như sau [8]:

- Xilinx XC3S500E Spartan-3E FPGA
  - 232 user-I/O pins
  - Đóng vỏ 320-pin FBGA
  - Trên 10,000 logic cells
- Xilinx 4 Mbit Platform Flash configuration PROM
- Xilinx 64-macrocell XC2C64A CoolRunner™ CPLD
- DDR SDRAM 64 MByte (512 Mbit) x16 data interface, 100+ MHz
- Parallel NOR Flash (Intel StrataFlash)16 MByte (128 Mbit)
  - Bộ nhớ cấu hình (FPGA configuration storage)
  - MicroBlaze code storage/shadowing
- SPI serial Flash (STMicro)16 Mbits
  - FPGA configuration storage
  - MicroBlaze code shadowing

- Hiển thị 2-line, 16-character LCD
- Cổng nối PS/2 mouse hoặc keyboard
- Cổng hiển thị VGA
- 10/100 Ethernet PHY (Ethernet MAC trong FPGA)
- Hai cổng 9-pin RS-232 (DTE- và DCE)
- Cổng gao tiếp USB để lập trình FPGA/CPLD
- 50 MHz clock oscillator
- Bảo vệ SHA-1 1-wire serial EEPROM cho sao chép bitstream
- Đầu nối mở rộng Hirose FX2
- Ba đầu nối mở rộng Digilent 6-pin
- 4-output, SPI-based Digital-to-Analog Converter (DAC)
- 2-input, SPI-based Analog-to-Digital Converter (ADC) với tiền khuếch khả trình
- Cổng ChipScope™ SoftTouch cho gỡ rối
- Bộ mã hóa vòng với push-button shaft
- Tám discrete LEDs
- Bốn khối chuyển mạch mảnh (slide switches)
- Bốn push-button switches
- SMA clock input
- 8-pin DIP socket cho auxiliary clock oscillator

3.2.2. Lựa chọn cấu hình hệ nhúng với Microblaze

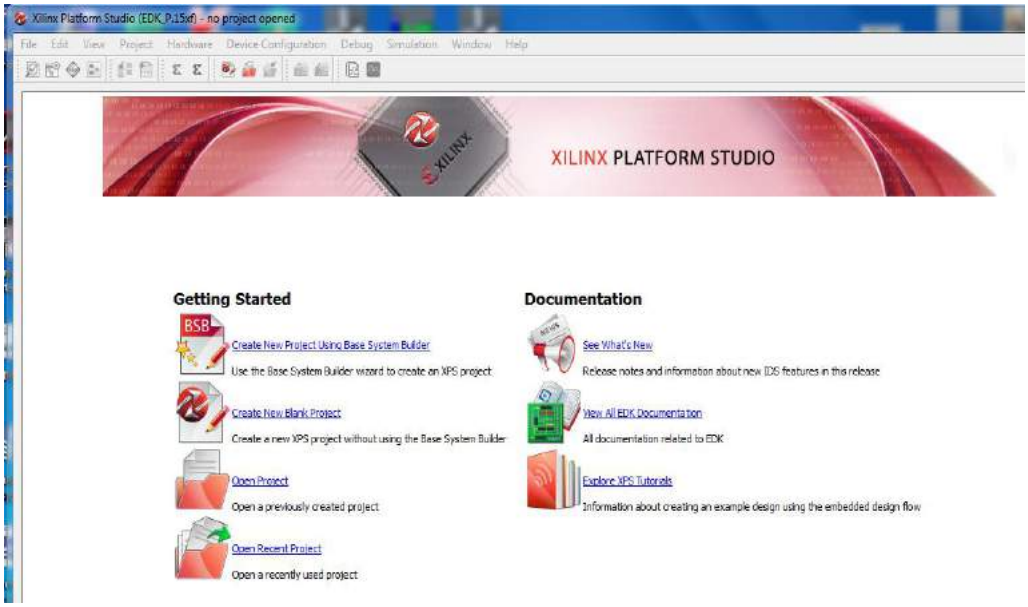
Trên bảng Xilinx Spartan-3E 500E có thể cấu hình hệ nhúng với một nhân, hoặc hai nhân Microblaze với bộ nhớ SRAM, DRAM, flash memory, với timer, các kết nối với các thiết bị: bốn switch (SW0, SW1, SW2, SW3), bốn nút bấm (button), tám led (led0 - led7), hai cổng COM RS232 (DTE, DCE), cổng ethernet, LCD.

Lựa chọn thiết kế hai nhân xử lý Microblaze 32-bit. Vì các cổng nối với các thiết bị không chia sẻ cho hai nhân xử lý, nghĩa là một cổng vào/ra chỉ được kết nối với một nhân xử lý.

3.2.3. Các bước thiết kế sử dụng Công cụ phần mềm Xilinx ISE14.1

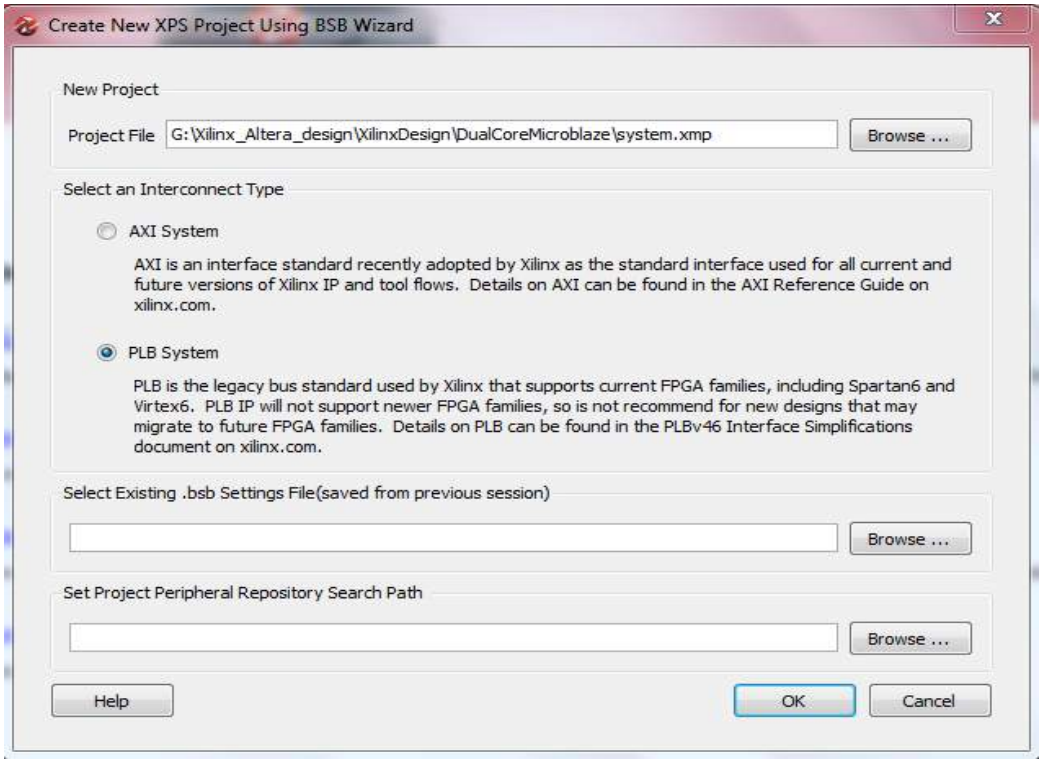
Sử dụng phần mềm XPS: chọn biểu tượng XPS trên Desktop chạy chương trình tạo hệ nhúng Microblaze.

1) Trên cửa sổ Xilinx Platform Studio, chọn Create New Project Using Base System Builder



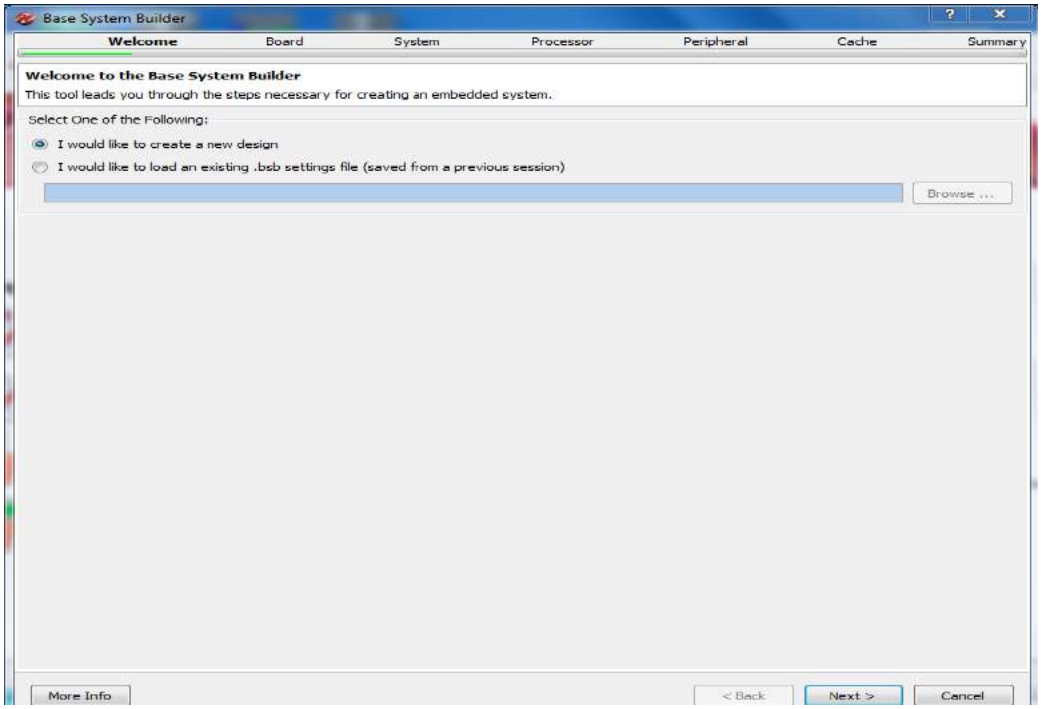
Hình 3.2: Cửa sổ Xilinx Platform Studio 14.1

2) Chọn thư mục, tạo tên project Microblaze



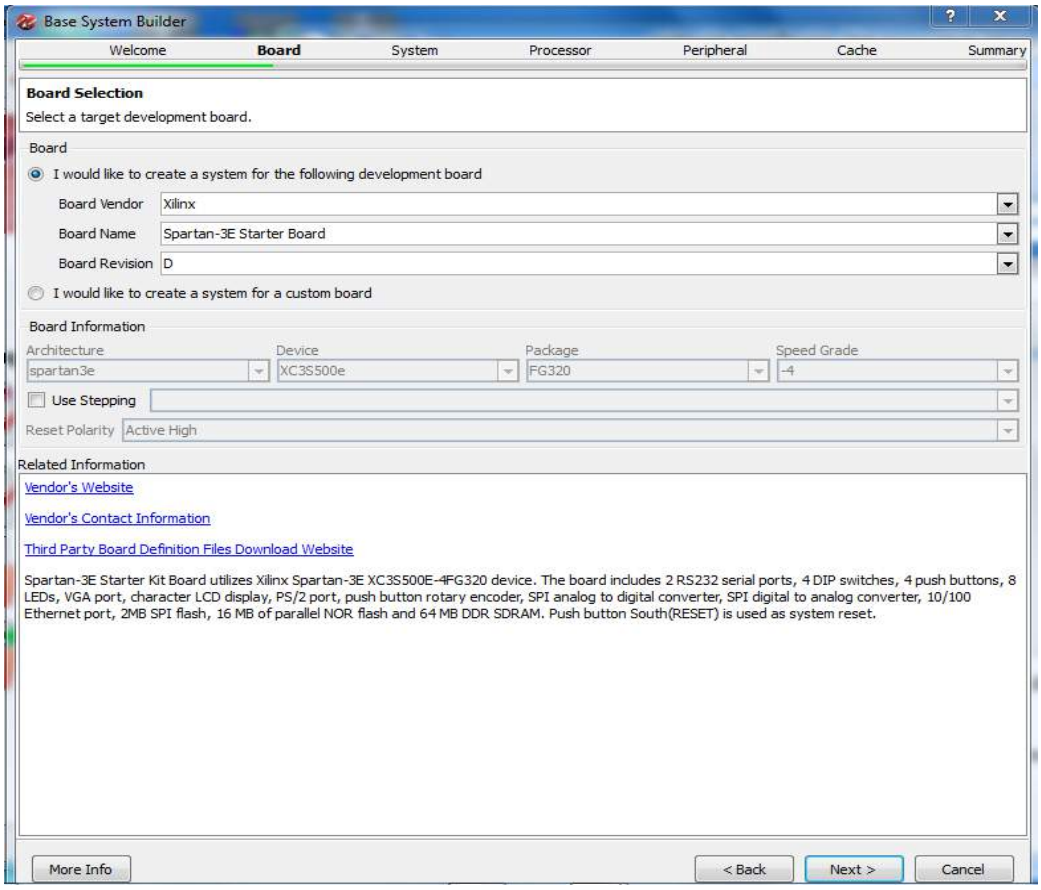
Hình 3.3: Tạo tên project Microblaze

3) Chọn tạo project



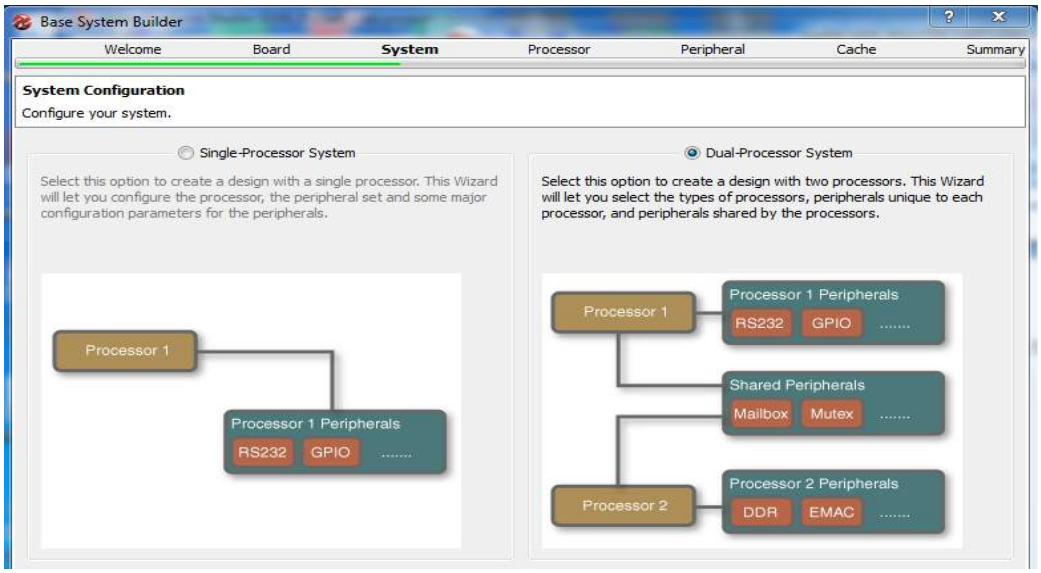
Hình 3.4: Chọn I world like to create a new design -> next

4) Chọn bảng phát triển Xilinx spartan-3E Stater board



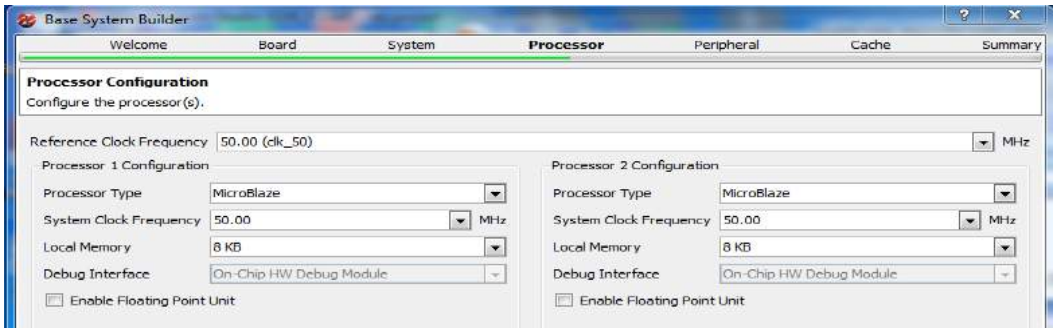
Hình 3.5: Chọn bảng Xilinx Spartan-3E Starter Board

5) Chọn cấu hình 2 nhân cho Microblaze



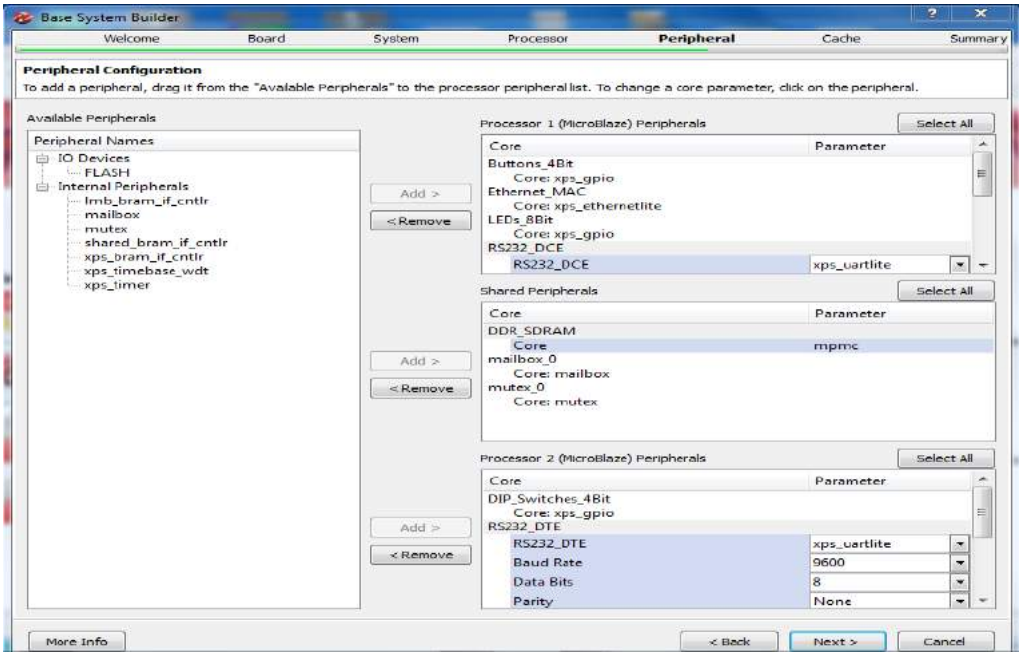
Hình 3.6: Chọn cấu hình 2 nhân cho Microblaze

6) Chọn đồng hồ và dung lượng bộ nhớ trong (BRAM) cho từng nhân



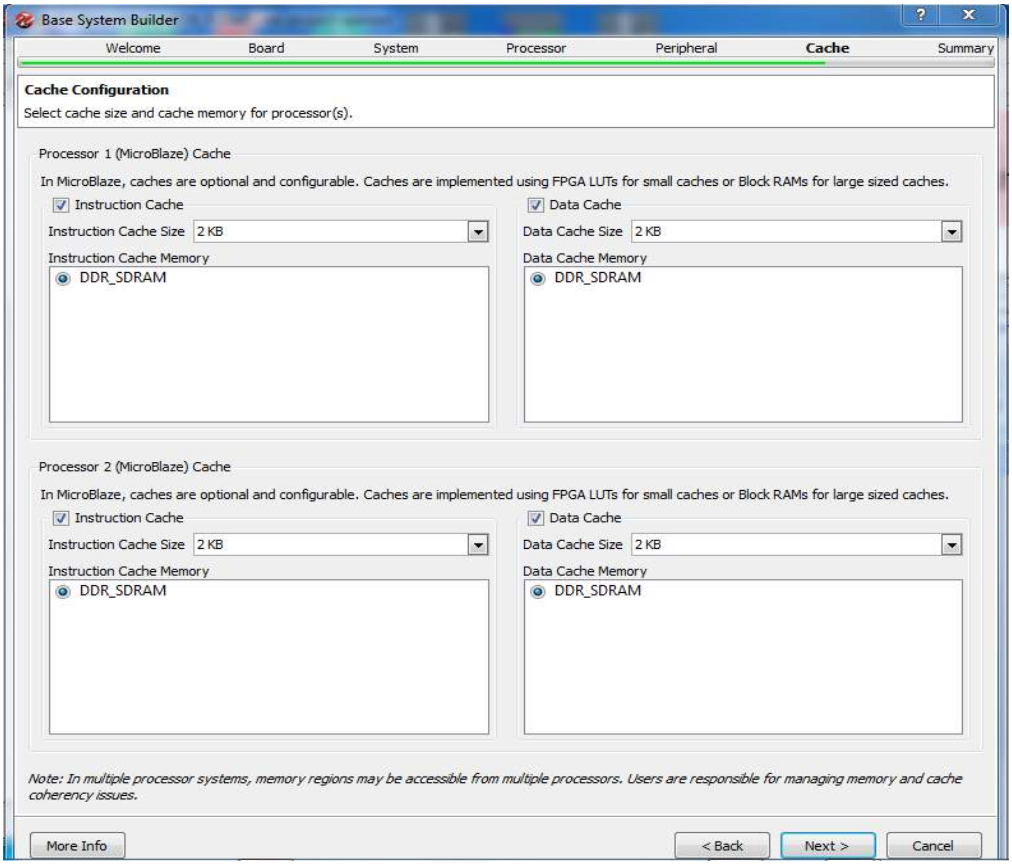
Hình 3.7: Chọn đồng hồ và dung lượng nhớ trong cho từng nhân

7) Chọn cấu hình các thiết bị cho từng nhân



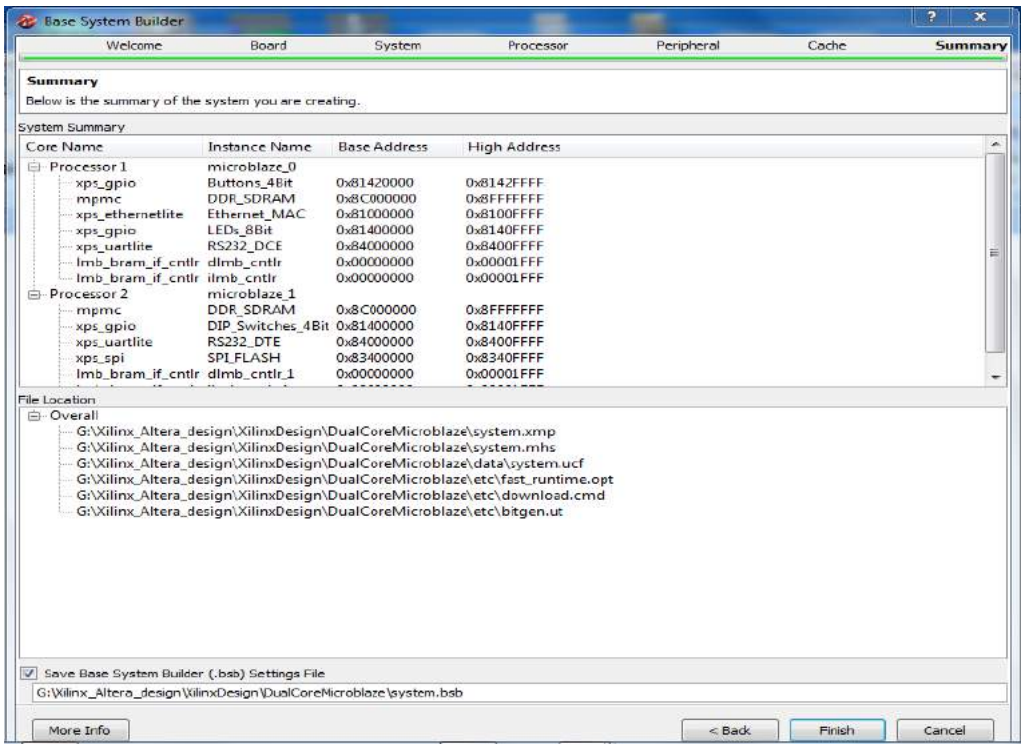
Hình 3.8: Chọn cấu hình các thiết bị cho từng nhân

8) Chọn dung lượng các cache cho từng nhân



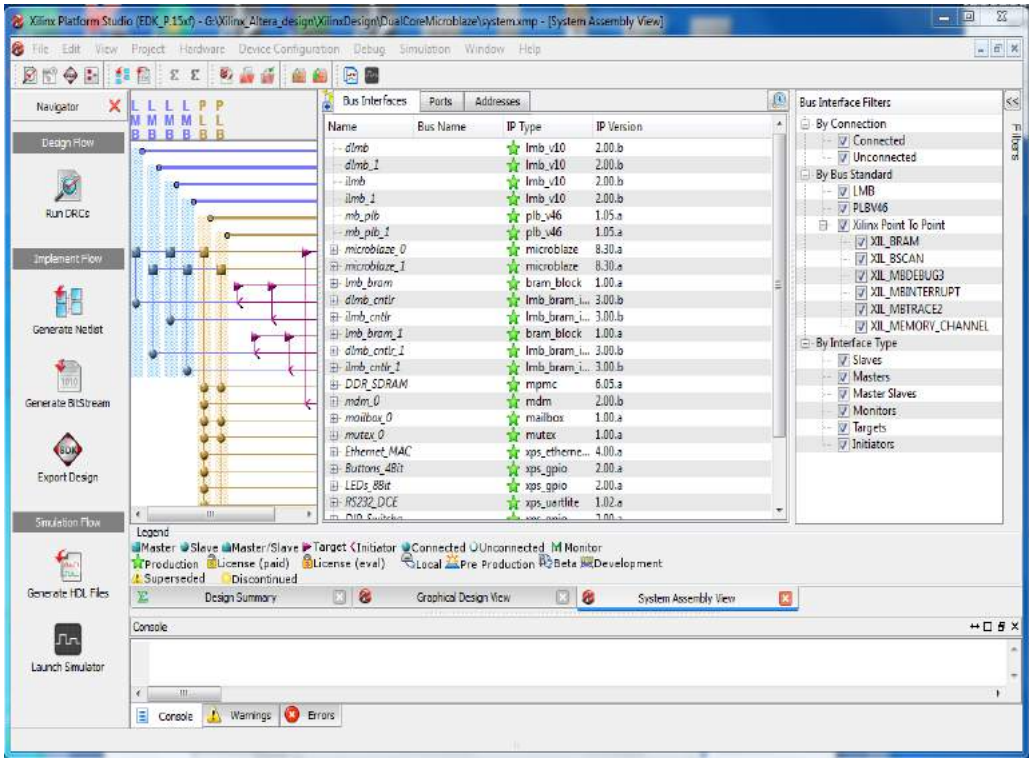
Hình 3.9: Chọn dung lượng 2KB cache cho từng nhân

9) Hiển thị thiết lập các địa chỉ của microblaze



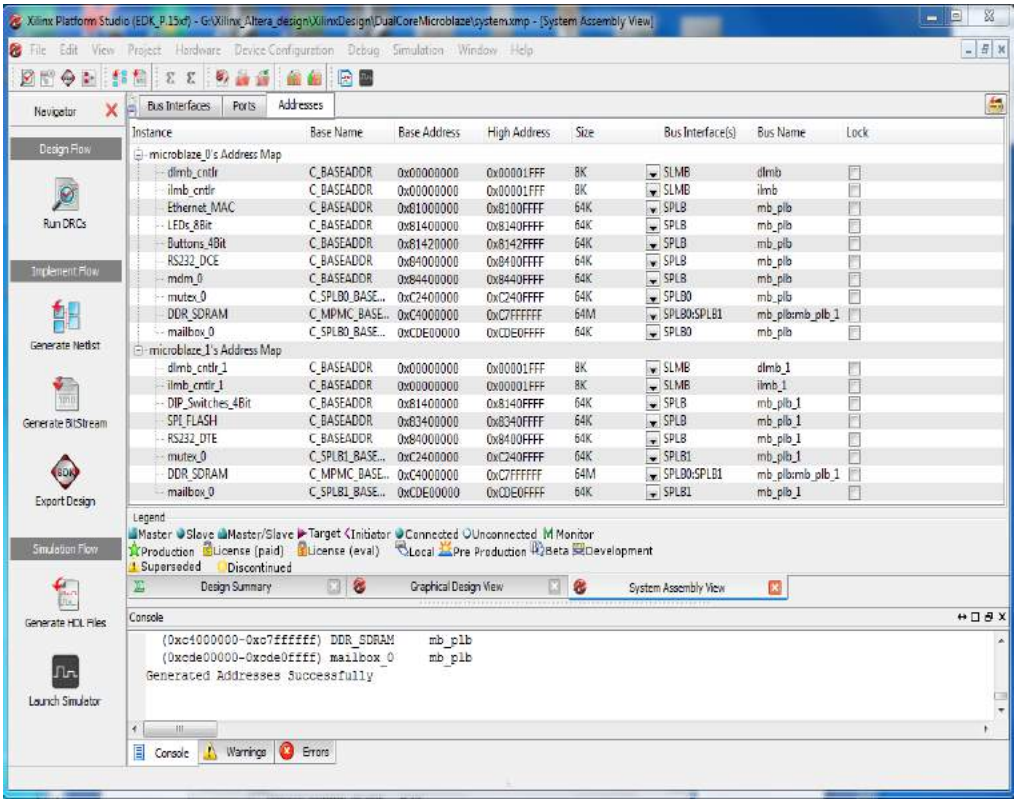
Hình 3.10: Địa chỉ của Microblaze

10) Các giao tiếp bus của Microblaze



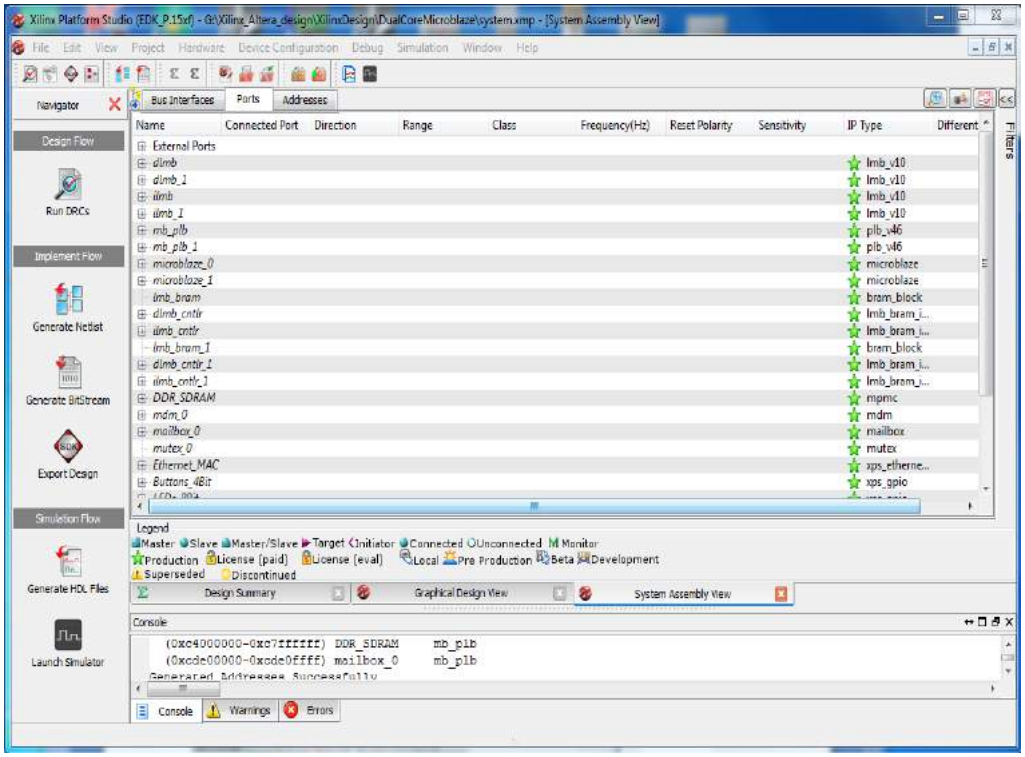
Hình 3.11: Các giao tiếp bus của Microblaze

11) Tạo địa chỉ cho hệ thống Microblaze



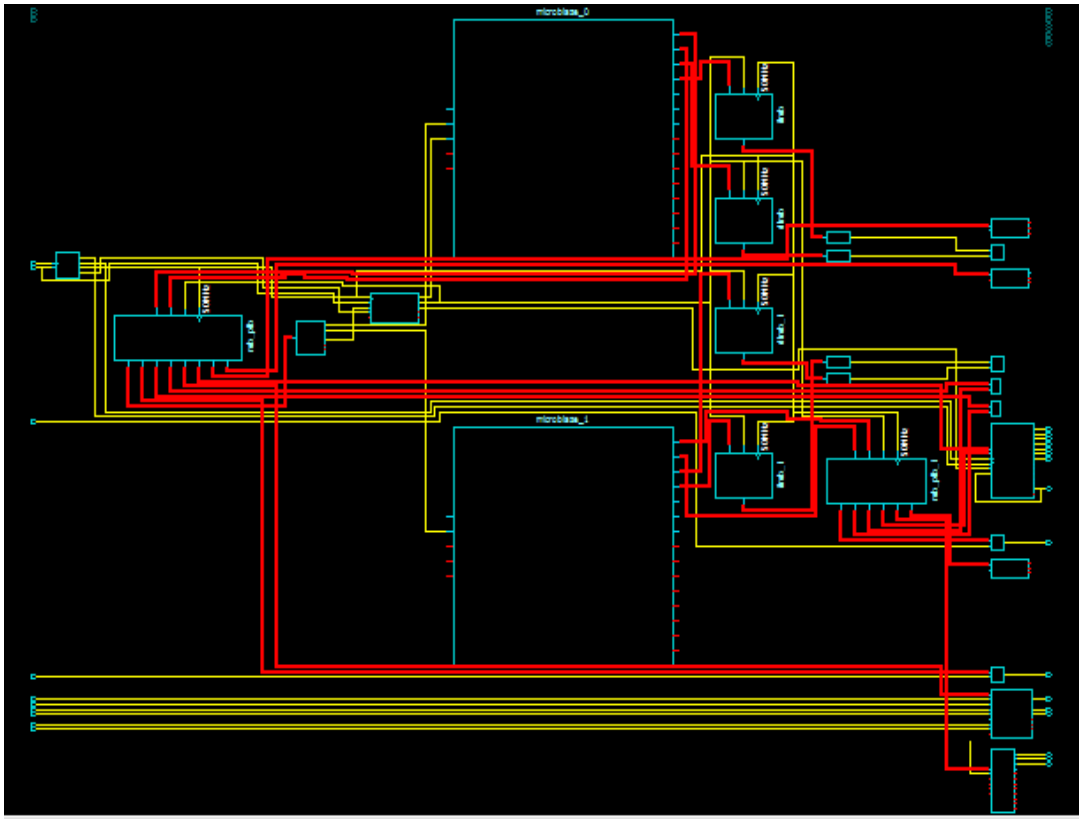
Hình 3.12: Tạo địa chỉ của hệ thống Microblaze

12) Sơ đồ mạch của hệ thống 2 nhân Microblaze



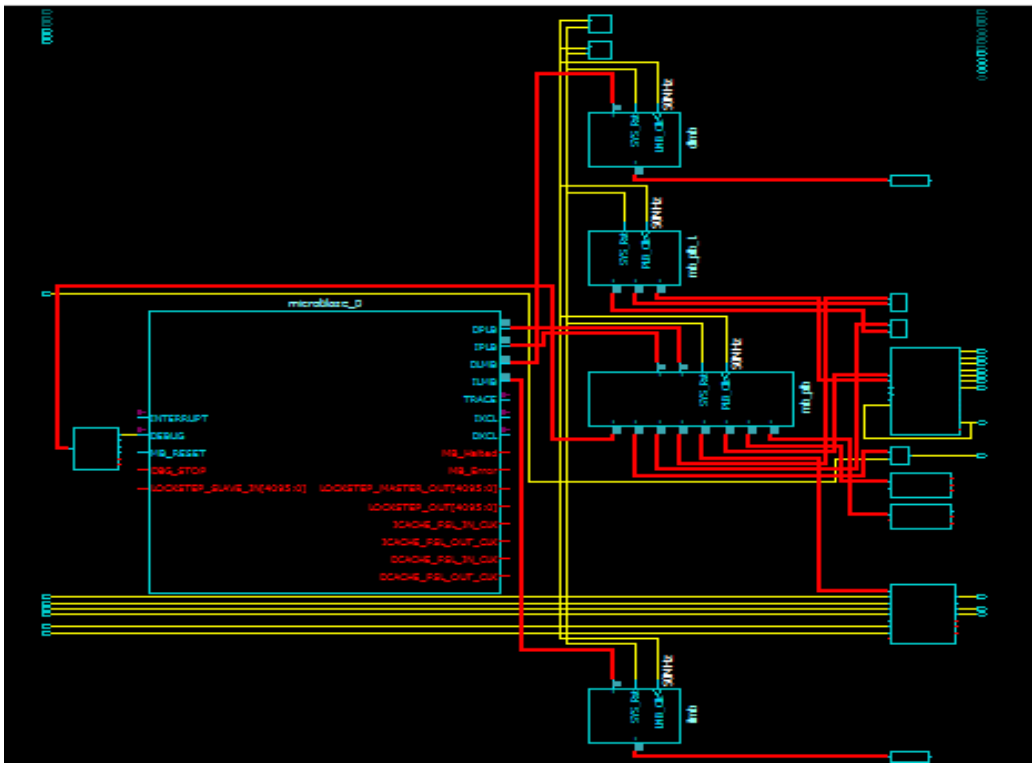
Hình 3.13: Các cổng của hệ thống Microblaze

13) Sơ đồ mạch của hệ thống Microblaze 2-core



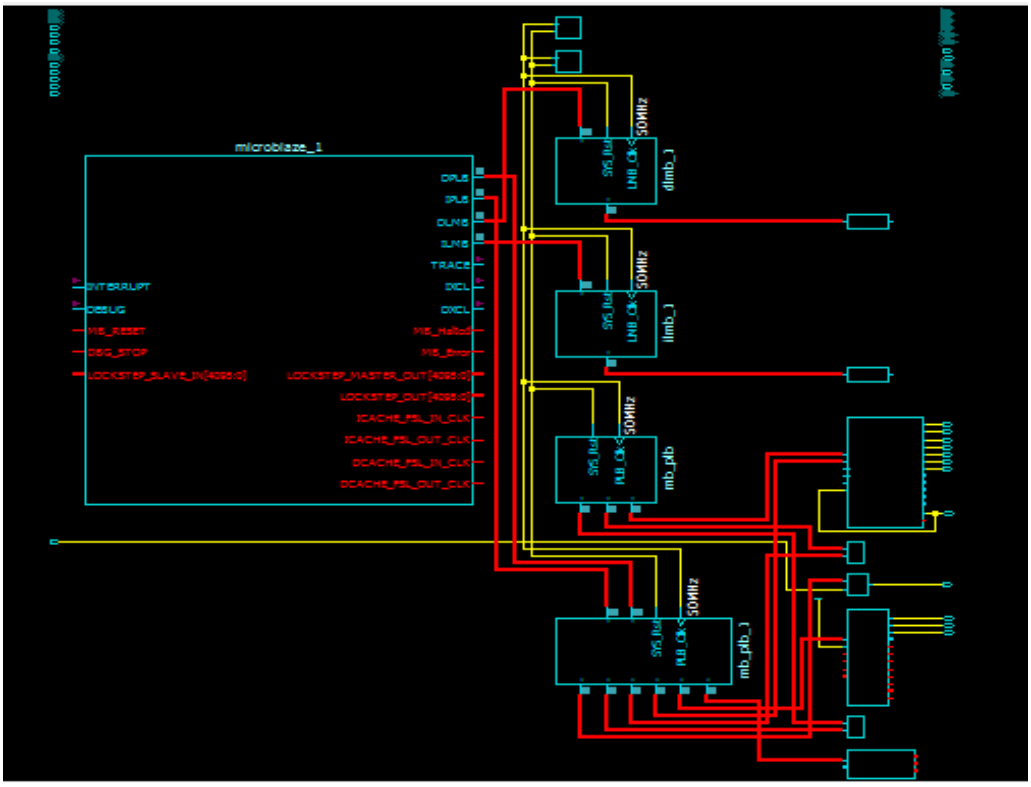
Hình 3.14 Sơ đồ mạch của hệ thống Microblaze 2-core

14) Sơ đồ mạch nối các thiết bị của nhân 0 (Microblaze_0)



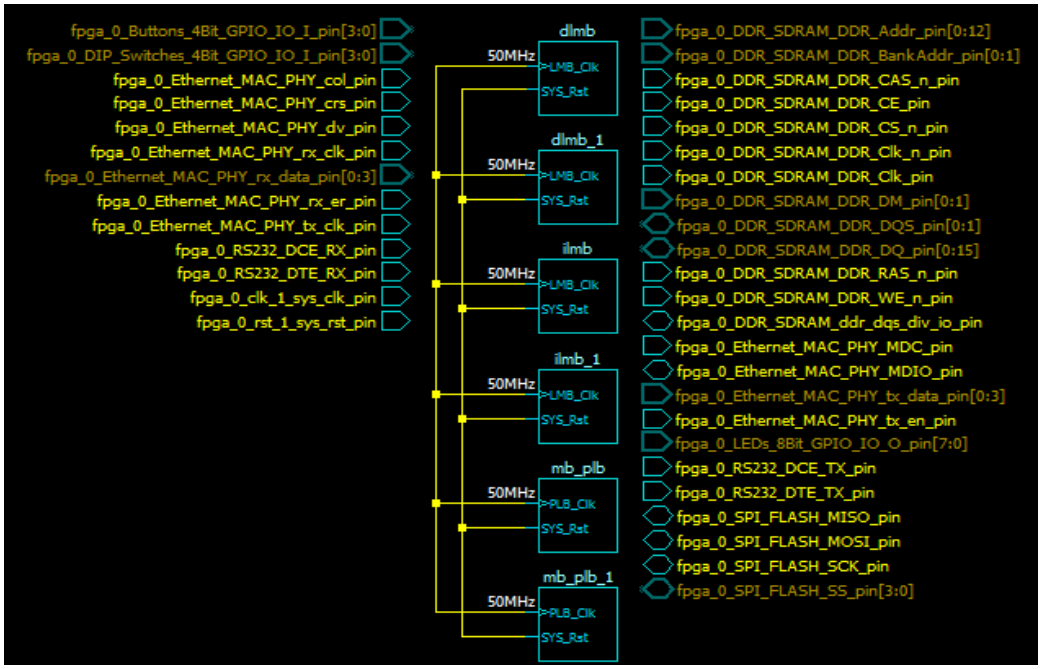
Hình 3.15: Sơ đồ mạch của core Microblaze_0

15) Sơ đồ mạch của core Microblaze_1



Hình 3.16: Sơ đồ mạch của core Microblaze_1

16) Các giao tiếp mở rộng I/O của hệ thống Microblaze

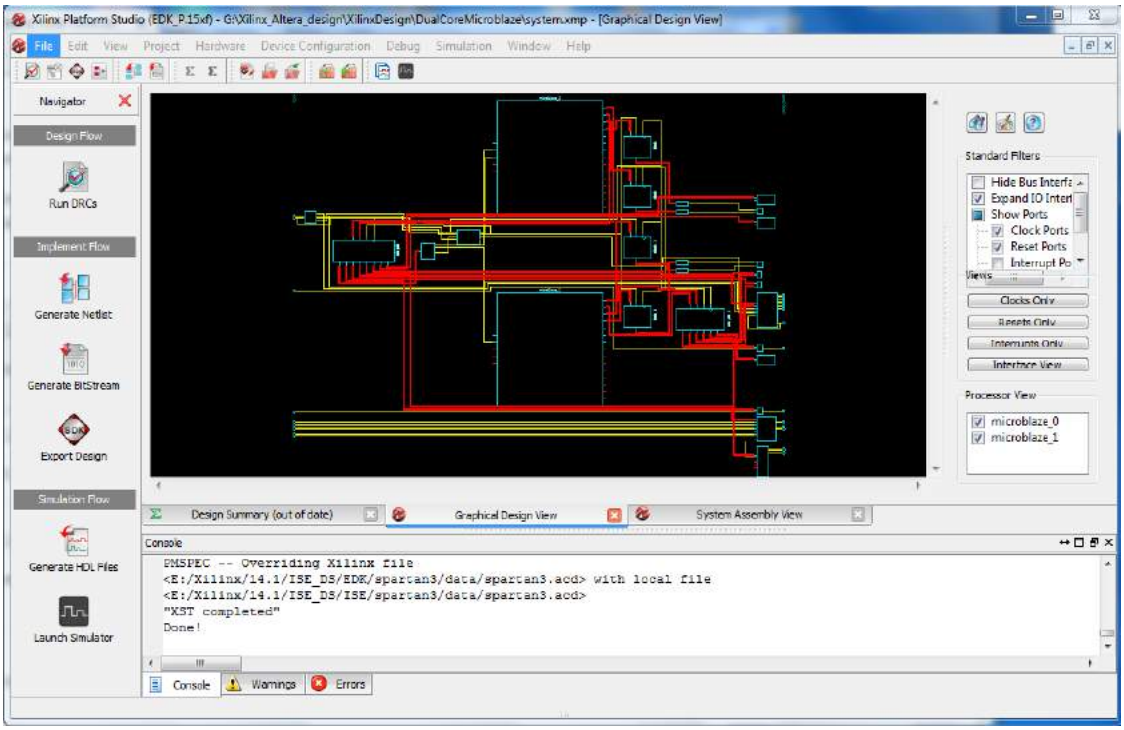


Hình 3.17: Các giao tiếp mở rộng I/O của Microblaze

Trên cửa sổ XPS, ta có thể lựa chọn các phần cứng khác nhau của hệ thống Microblaze để kiểm tra thiết kế. Sau khi đã kiểm tra, in ấn thiết kế, lưu thiết kế bằng:

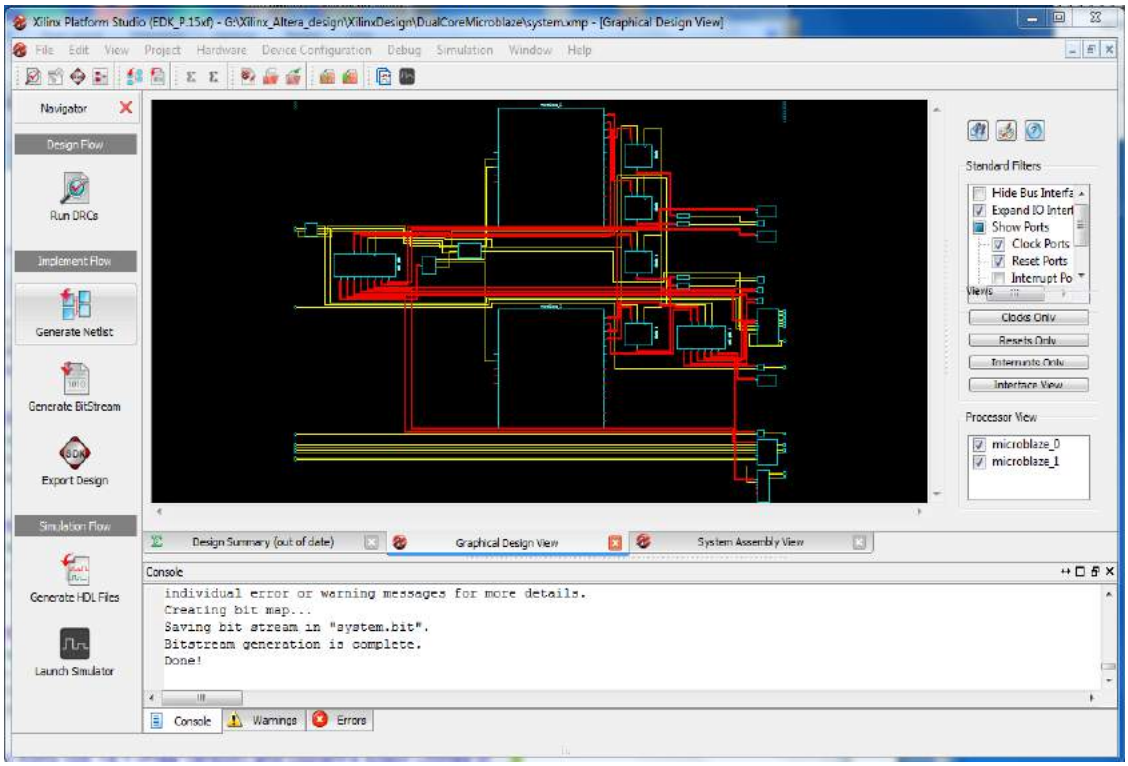
file -> save project.

18) Chọn Hardware -> Generate Netlist để tạo danh sách của thiết kế Netlist (thời gian tạo Netlist khoảng vài phút, tùy thuộc vào cấu hình máy tính).



Hình 3.18: Tạo thành công Netlist của project Microblaze

19) Chọn Hardware -> Generate Bitstream tạo file cấu hình (system.bit) của hệ thống Microblaze (thời gian tạo Bitstream file khoảng vài phút)



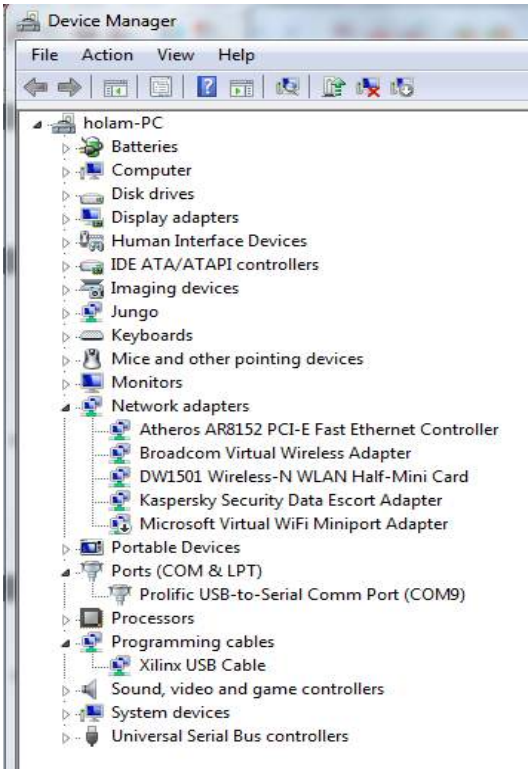
Hình 3.19: Tạo thành công file cấu hình hệ thống Microblaze "system.bit"

Thiết kế được tạo ra trong thư mục:

Xilinx_Altera_design\XilinxDesign\DualCoreMicroblaze\implementation\:

netlist	7/10/2018 11:22 AM	MASM Listing	4 KB
system.bgn	7/10/2018 11:29 AM	BGN File	8 KB
system.bit	7/10/2018 11:29 AM	BIT File	278 KB
system.bld	7/10/2018 11:22 AM	BLD File	36 KB
system.bmm	7/10/2018 10:34 AM	BMM File	1 KB
system.drc	7/10/2018 11:28 AM	DRC File	4 KB
system.ncd	7/10/2018 11:28 AM	NCD File	2,925 KB
system.ngc	7/10/2018 11:16 AM	NGC File	3,011 KB
system.ngc_xst.xrpt	7/10/2018 11:16 AM	XRPT File	12 KB
system.ngd	7/10/2018 11:22 AM	NGD File	11,720 KB
system.pad	7/10/2018 11:28 AM	PAD File	18 KB
system	7/10/2018 11:28 AM	PAR File	66 KB

20) Nối Xilinx Spartan-3E starter Board với PC bằng cáp Xilinx USB, cáp chuyển đổi USB-RS232: Xilinx Board DCE <--> USB PC. Thiết lập tốc độ cho cổng này (COM9) tốc độ 9600bps giống như đã cấu hình cho DCE của bảng (cho Microblaze_0)



Hình 3.20: Kiểm tra các kết nối thiết của PC bằng Device Manager

21) Trên XPS, chọn Project -> Export Hardware Design to SDK ...: chuyển các file cấu hình của project đến SDK để nạp lên bảng FPGA và xây dựng ứng dụng.

Hình 3.21: Chọn Export & Launch SDK

Ta có thể lựa chọn Exoprt Only để chuyển các file cấu hình của thiết kế hệ Microblaze đến SDK lưu, sau đó có thể SDK và thực hiện nạp file cấu hình lên bảng FPGA. Cũng có thể chọn Export & lauch SDK, khi đó sau khi export ta thực hiện chạy ngay SDK.

3.3. XÂY DỰNG VÀ CÀI ĐẶT PHẦN MỀM ỨNG DỤNG

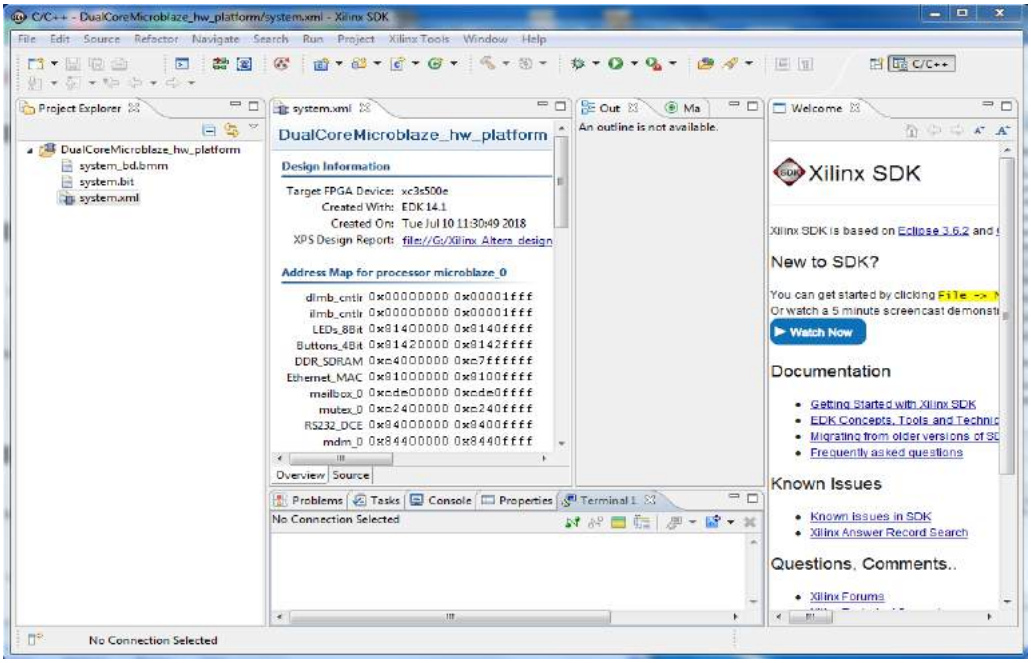
3.3.1. Phần mềm Helloworld.c và cài đặt thử nghiệm

1) Chọn Export & Lauch SDK

2) Chọn vùng làm việc cho project

Xilinx_Altera_design\XilinxDesign\Microblaze_on_sp3e_dualCore_noFPU_FLASH_ISE14.1\Workspace\

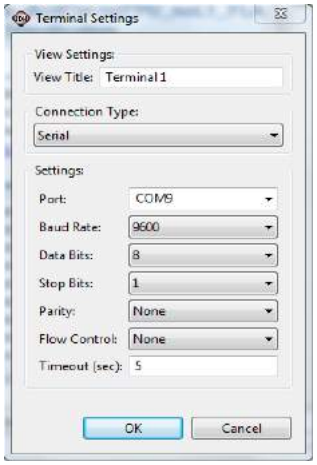
3) Chạy SDK



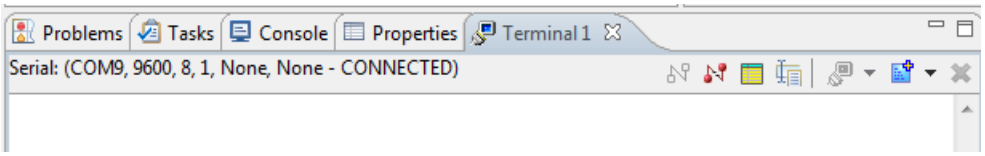
Hình 3.22: Các file cấu hình của thiết kế từ XPS đã được Export vào SDK

4) Thiết lập cổng COM-USB

Định dạng dữ liệu, tốc độ bit phải phù hợp với cấu hình phần cứng đã được thiết lập trong XPS.



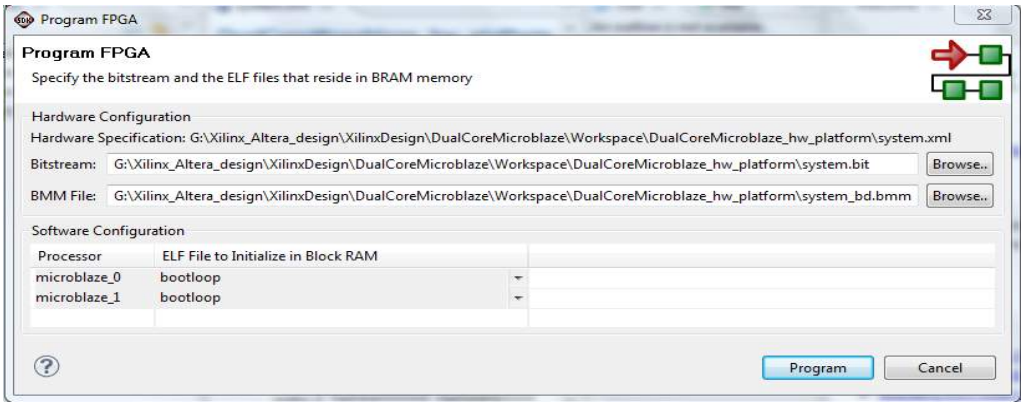
Hình 3.23: Thiết lập cổng COM-USB



Hình 3.24: Cổng COM-USB đã kết nối sau khi thiết lập

5) Nạp file cấu hình "system.bit" của hệ thống Microblaze lên FPGA

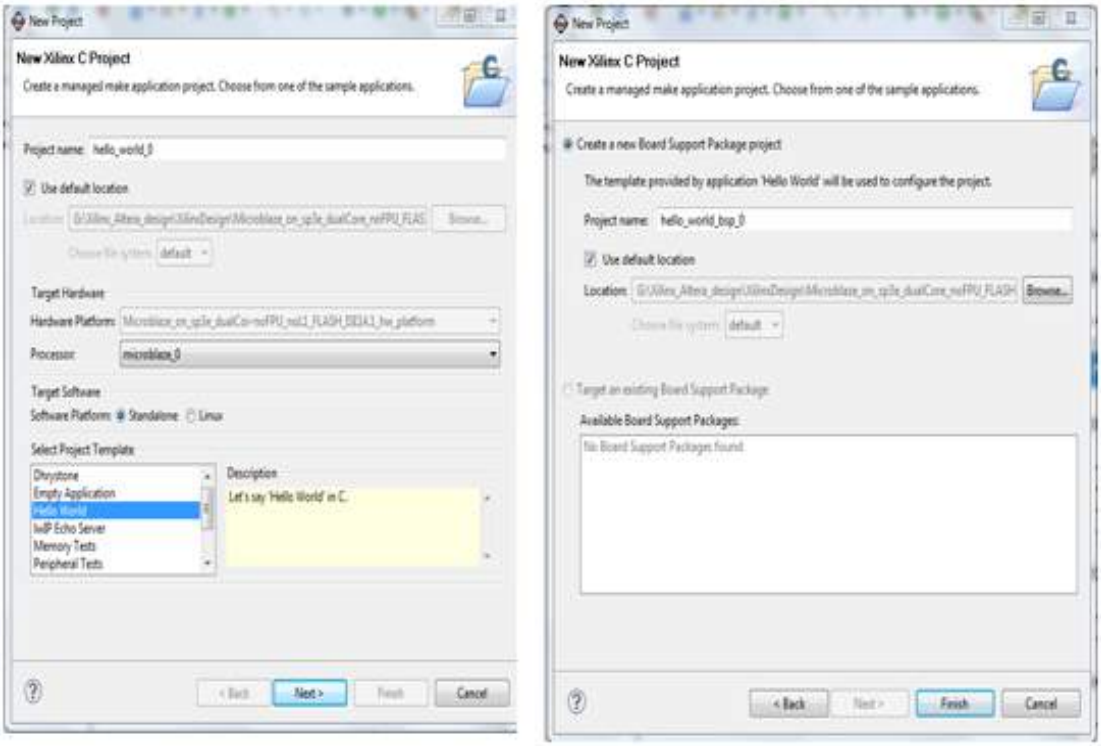
Chọn: Xilinx Tools -> Program FPGA"



Hình 3.25: Chọn Program để nạp cấu hình lên FPGA

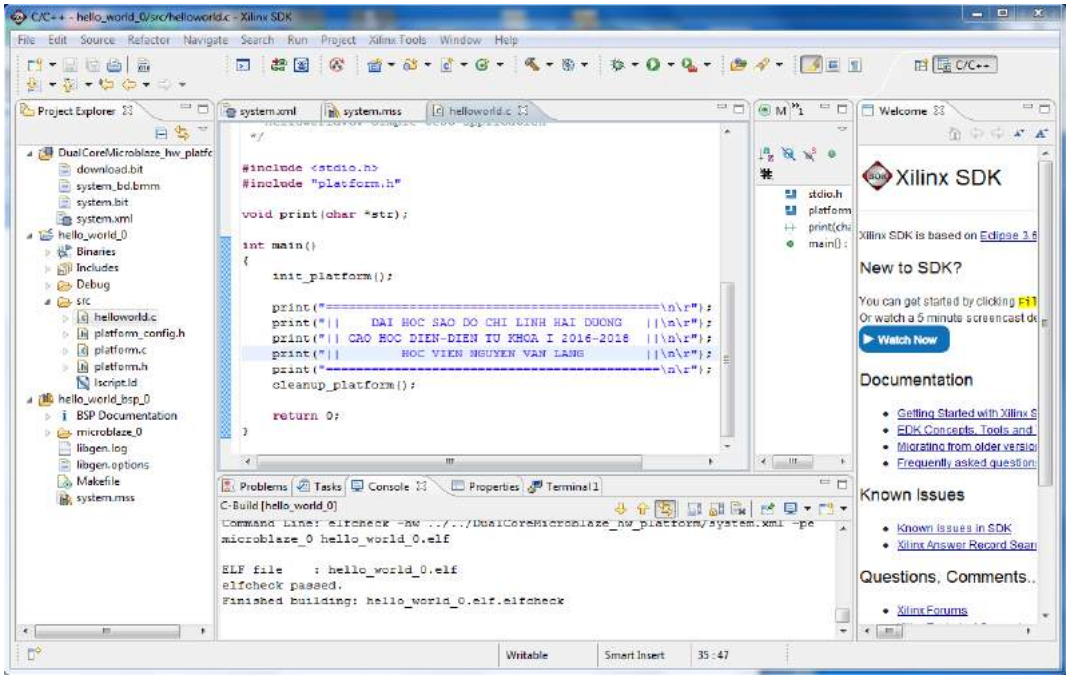
6) Tạo phần mềm helloworld.c

Chọn file -> New -> Xilinx C Project



Hình 3.26: Tạo tên project: hello_world_0

7) Soạn file helloworld.c



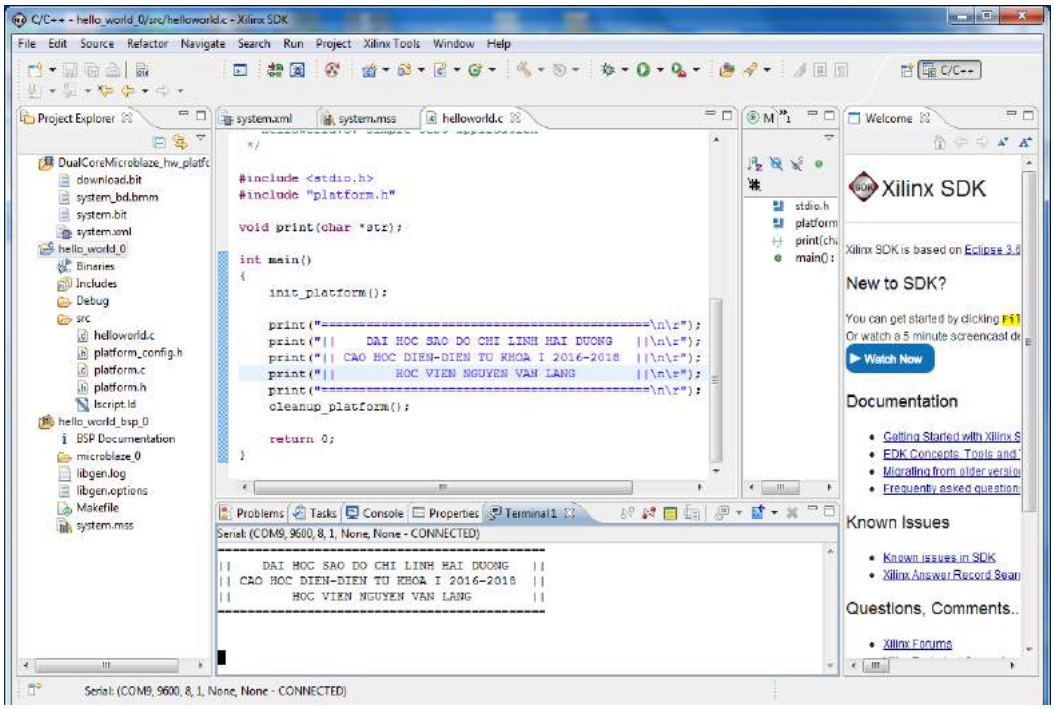
Hình 3.27: Soạn file helloworld.c

8) Chọn hello_world_0 -> Build Project: biên dịch, tạo file nhị phân

9) Chạy ứng dụng helloword:

Chọn hello_world_0 -> Run As -> Launch on Hardware

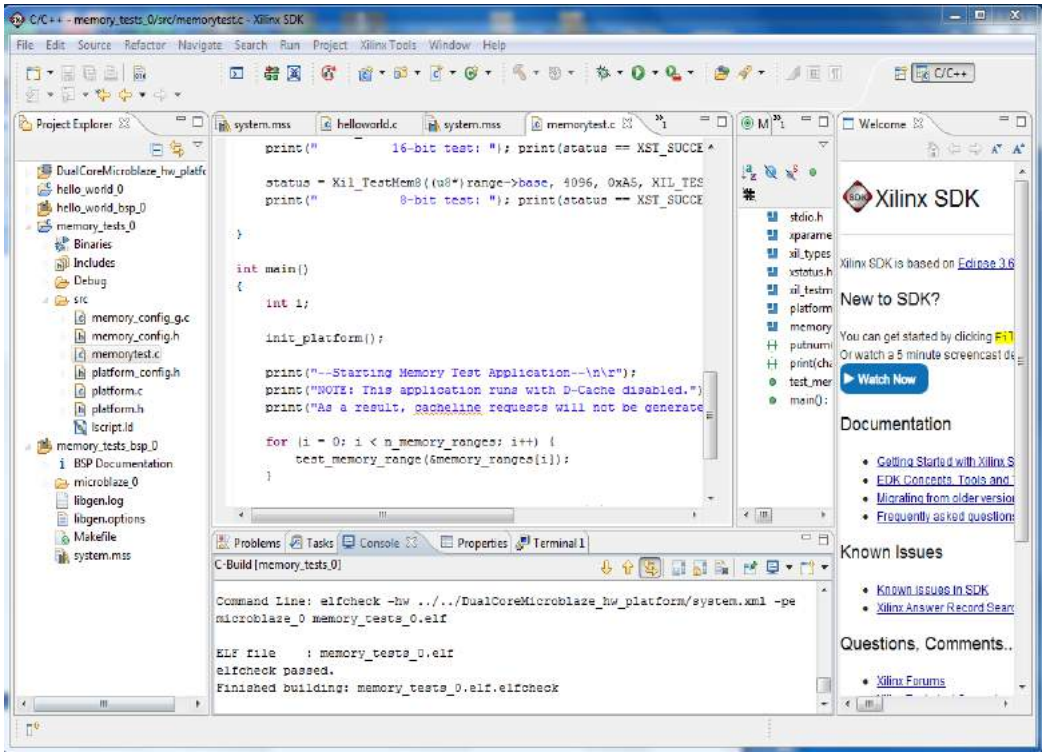
Chương trình được nạp lên bảng Spartan-3e Starter, và chạy, cho đáp ứng kết quả lên PC qua cáp terminal COM-USB (Lúc này Bảng Xilinx là host, PC là Terminal).



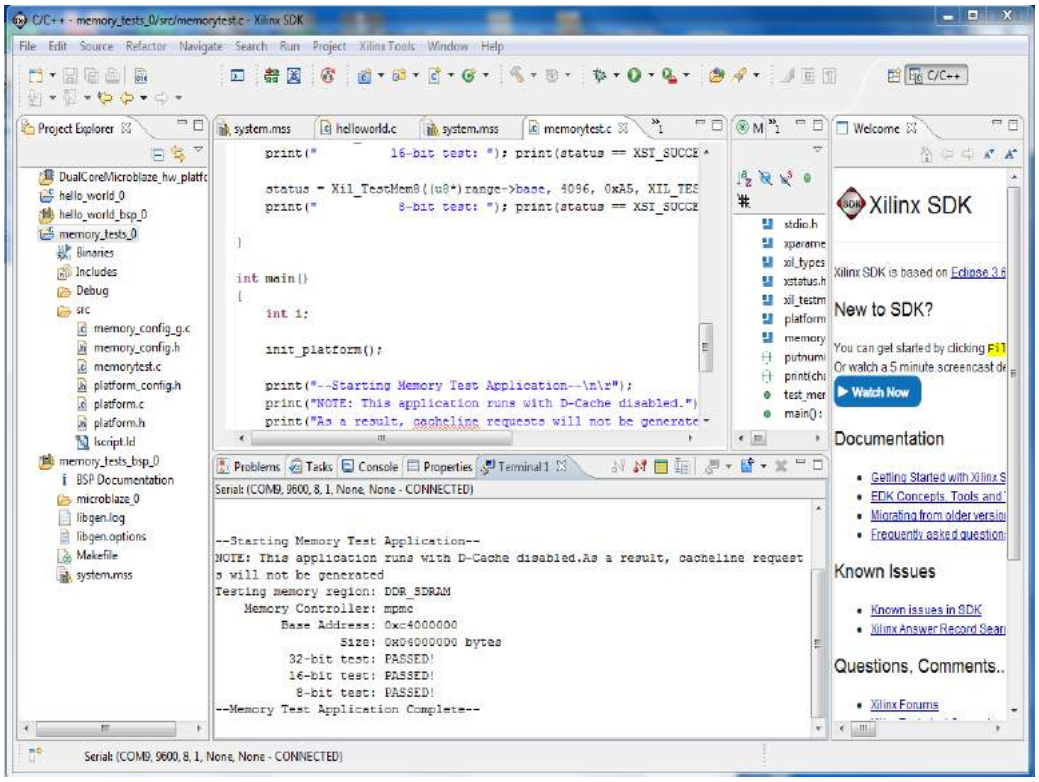
Hình 3.28: Kết quả chạy helloworld trên FPGA (trả về PC)

3.3.2. Phần mềm kiểm tra bộ nhớ và cài đặt thử nghiệm

Các bước tạo và chạy ứng dụng kiểm tra bộ nhớ cũng tương tự như cho ứng dụng helloworld.



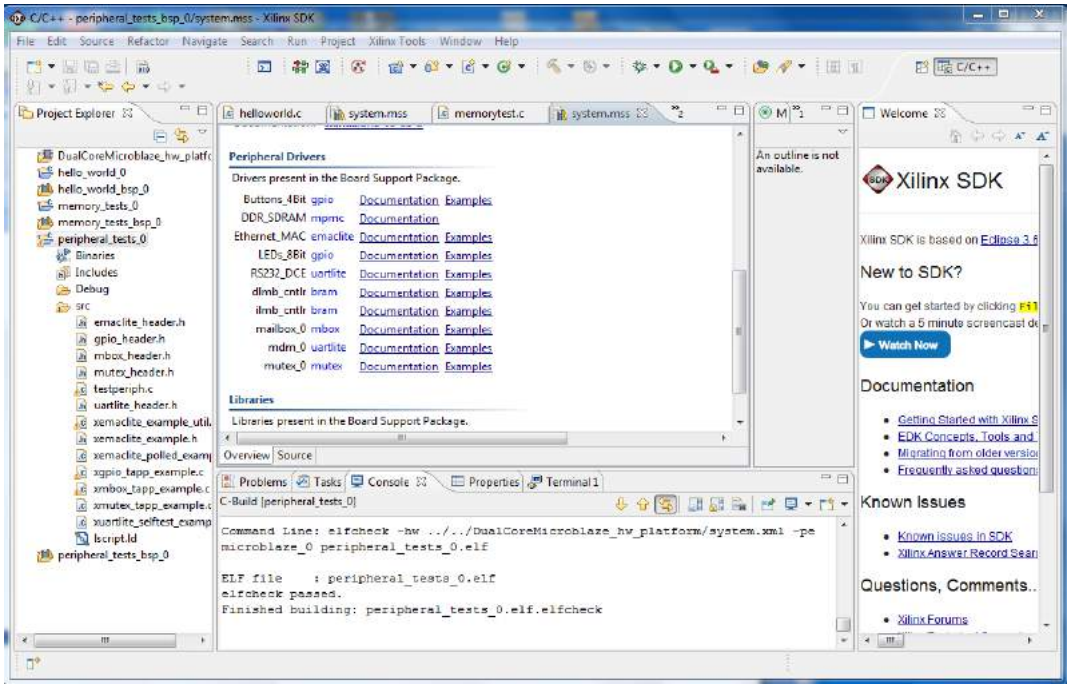
Hình 3.29: Tạo ứng dụng memorytest



Hình 3.30: Chạy memorytest trên FPGA thành công

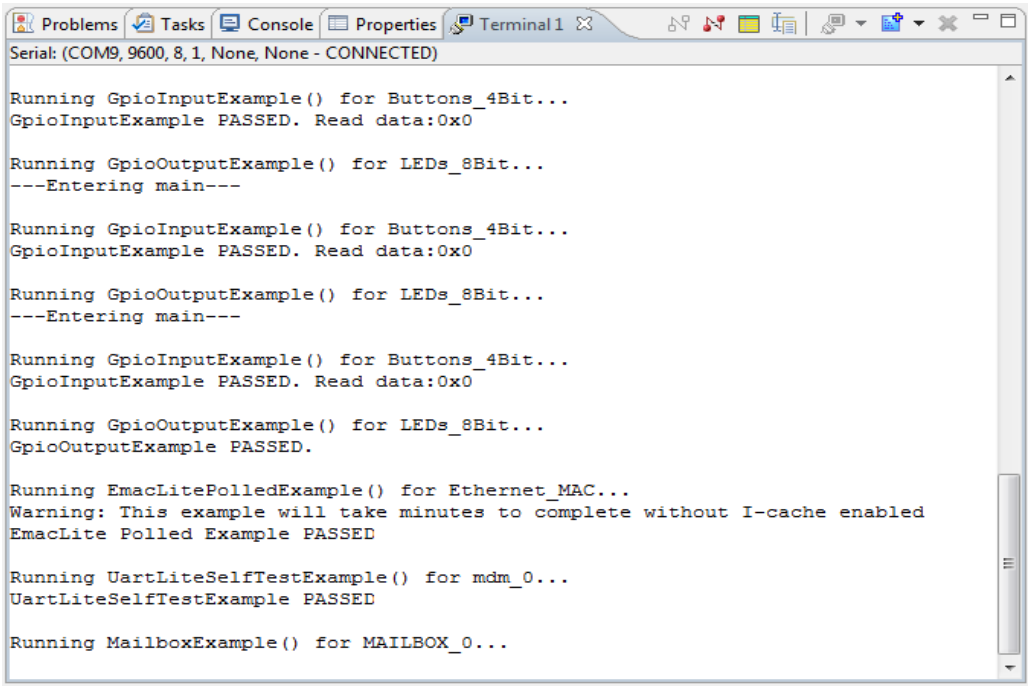
3.3.3. Phần mềm kiểm tra các giao tiếp ngoại vi và cài đặt thử nghiệm

Tương tự như các ứng dụng trên, thực hiện tạo, biên dịch, và chạy chương trình kiểm tra các giao tiếp ngoại vi.



Hình 3.31: Tạo và biên dịch trình kiểm tra ngoại vi

Trong quá trình kiểm tra các ngoại vi: leds, ethernet, buttons, uart, ... kéo dài vài phút



Hình 3.32: Chạy thành công kiểm tra các ngoại vi

3.4. KẾT LUẬN CHƯƠNG

Nội dung chương trình đã trình bày toàn bộ quá trình tạo một hệ vi xử lý 2 nhân Microblaze 32-bit, cài đặt cấu hình lên FPGA, và xây dựng các ứng dụng chạy trên Microblaze thành công. Điều này chứng tỏ cấu hình thiết kế mà học viên lựa chọn là phù hợp. Do chip FPGA của bảng Xilinx Spartan-3e starter board có chip FPGA mật độ không cao (500000 LEs), nên cấu hình thiết kế cho Microblaze 2 nhân không lớn: không có FPU (đơn vị xử lý số dấu phẩy động), dung lượng bộ nhớ DRAM nhỏ (64MB), cache chỉ 2KB, nhưng có kết nối với các ngoại vi: leds, switches, buttons, ethernet, COM DTE, DCE, đủ để xây dựng các ứng dụng liên quan đến các thiết bị này.

KẾT LUẬN VÀ KIẾN NGHỊ

1. KẾT LUẬN

- Đóng góp của luận văn là đưa ra các bước thiết kế một hệ nhúng với vi điều khiển trên FPGA nhờ sử dụng ngôn ngữ lập trình mô tả phần cứng VHDL
- Tạo ra một hệ nhúng trên FPGA với CPU là vi điều khiển lõi mềm. Từ đó, có thể thiết kế nhiều hệ thống số khác nhau cho các ứng dụng khác nhau mong muốn
- Với hiểu biết về VHDL (verilog), FPGA, và phương pháp thiết kế, có thể triển khai các nghiên cứu khoa học nâng cao sử dụng FPGA.
- Nội dung luận văn đã thực hiện được các mục tiêu của đề tài theo đề cương nghiên cứu.

2. KIẾN NGHỊ

FPGA đã và đang chứng tỏ khả năng ứng dụng cho nghiên cứu phát triển, cho đào tạo. Vì vậy, tìm hiểu FPGA, các phương pháp thiết kế các hệ thống trên FPGA là rất cần thiết. Với thời gian có hạn, học viên chỉ nêu được những vấn đề cơ bản trong thiết kế vi xử lý Microblaze và cài đặt các ứng dụng nhỏ. Các phát triển cho các ứ ng dụng xử lý tín hiệu, điều khiển trong công nghiệp, điện, tự động hóa, cảm biến môi trường, v.v... hứa hẹn cho các nghiên cứu sau này có ứng dụng FPGA.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Hồ Khánh Lâm, "Giáo trình Lập trình VHDL thiết kế hệ thống số trên FPGA". NXB KHKT, 1015.

Tiếng Anh

[2] Prabhakar R, Thirumal Murugan J and Praveenkumar B," Efficient Method for Controlling Electric Power by Automated Monitoring System using FPGA". International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 3 Issue 11, November-2014.

[3] HAYASHI Toshifumi et al." Application of FPGA to nuclear power plant I&C systems". Nuclear Safety and Simulation, Vol. 3, Number 1, March 2012. Received date: March 13, 2012 (Revised date: April 12, 2012).

[4] Lauri Lötjönen," Field-Programmable Gate Arrays in Nuclear Power Plant Safety Automation". School of Electrical Engineering. Thesis submitted for examination for the degree of Master of Science in Technology.

[5] IAEA Nuclear Energy Series No. NP-T-3.17," Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants".

[6] Usama Bin Rehan et al.,"Power Line Control and Monitoring Using FPGA". 2017 2nd International Electrical Engineering Conference (IEEC 2017) May 19th- 20th, 2017 at IEP Centre, Karachi, Pakistan.

[7] Prasanna Sundararajan," High Performance Computing Using FPGAs".WP375 (v1.0) September 10, 2010. www.xilinx.com/support/white_papers/ /wp375_HPC_Using_FPGAs.pdf.

[8] "Spartan-3E FPGA Starter Kit Board User Guide".UG230 (v1.2) January 20, 2011. www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf.