

Thiết kế hệ thống nhúng dùng SoPC

Các từ viết tắt:

PC: Personal computer

IC: Integrated circuit

RTOS: Real-time operating system

ROM: Read-only memory

RAM: Random access memory

MIPS: Millions of instructions per second

LUT: Look up table

LE: Logic element

LAB: Logic array block

SoPC: System on a Programmable Chip

EDA: Electronic design automation.

DRAM: Dynamic Random Access Memory

SDRAM: SDR Synchronous DRAM

DDR SDRAM: Double data rate synchronous dynamic random-access memory

CAS: Column Access Strobe

RAS: Row address strobe

RISC: Reduced instruction set computer

MMU (Memory management unit)

MPU (Memory protection unit)

ISR (Interrupt Service Routine)

Chương 1 Các khái niệm cơ bản

I. Tổng quan về hệ thống nhúng

I.1 Khái niệm về hệ thống nhúng

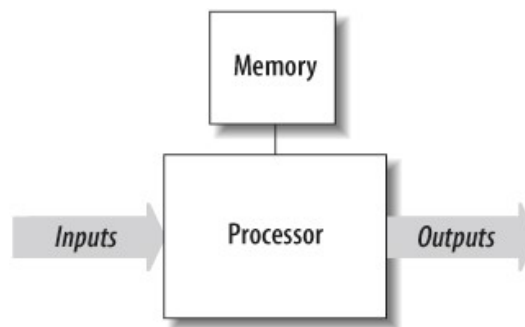
Một hệ thống nhúng bao gồm phần cứng và phần mềm và có thể có thêm một số thành phần khác (về điện tử, cơ khí ...) được thiết kế để thực hiện một chức năng chuyên dụng nào đó. Thông thì một hệ thống nhúng là một thành phần của một hệ thống lớn.

Cần phải thấy rõ là máy tính cá nhân (personal computer - PC) không phải là một hệ thống nhúng vì chúng được thiết kế hướng tới các ứng dụng đa năng. Trong nhiều trường hợp, PC được dùng để giao tiếp với các hệ thống nhúng chẳng như chứa các môi trường phát triển tích hợp để thiết kế và phát triển các hệ thống nhúng (gọi là IDE).

Về mặt lý thuyết, một chức năng chuyên dụng được thực hiện bởi một hệ thống nhúng nào đó có thể được thực hiện bằng một mạch tích hợp chuyên dụng tương ứng mà không cần một bộ xử lý (và phần mềm của nó). Tuy nhiên, một hệ thống nhúng thường có một bộ xử lý và phần mềm sẽ giúp cho chúng có sự linh hoạt và hiệu quả cần thiết.

I.2 Các thành phần của một hệ thống nhúng

Về cơ bản, một hệ thống nhúng có các thành phần chung được trình bày như Hình 1.1.



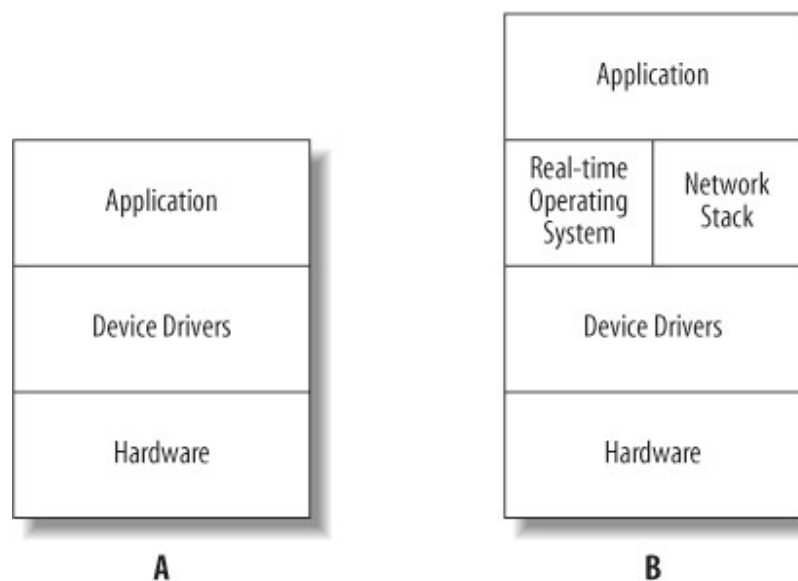
Hình 1.1: Các thành phần chung của một hệ thống nhúng

Bộ nhớ bao gồm bộ nhớ chỉ đọc (ROM – Read Only Memory) thường được dùng để chứa chương trình khởi động và các dữ liệu cố định và bộ nhớ truy xuất ngẫu nhiên (RAM – Random Access Memory) dùng để chứa chương trình ứng dụng và các kết quả tính toán trung gian. Những hệ thống không cần nhiều bộ nhớ thì bộ nhớ sẽ tích hợp chung với bộ xử lý vào trong

một vi mạch, nếu một hệ thống cần bộ nhớ thì bộ nhớ sẽ là các vi mạch nhớ riêng biệt với bộ xử lý.

Ngoài ra, các hệ thống nhúng còn có các giao tiếp vào/ ra. Các giao tiếp ngõ vào thường là các cảm biến, các tín hiệu truyền thông, các nút nhấn ... và các giao tiếp ngõ ra thường là các bộ hiển thị, các tín hiệu truyền thông, hoặc các tín hiệu dùng để điều khiển các thành phần khác...

Hình 1.2 trình bày mô hình phần mềm được thực hiện trong một hệ thống nhúng nói chung.



Hình 1.2: Mô hình phần mềm trong một hệ thống nhúng

Các Driver thiết bị là các mô đun phần mềm nhúng chứa các hàm điều khiển các thiết bị phần cứng tương ứng, điều này giúp chương trình ứng dụng có thể truy xuất đến phần cứng mà không cần quan tâm đến cấu trúc phần cứng bên dưới.

Trong một hệ thống nhúng có nhiều chức năng hơn (và phức tạp hơn) thì sẽ cần thêm một số thành phần nữa chẳng hạn như hệ điều hành thời gian thực (Real - time operation system - RTOS) và các thư viện hỗ trợ giao tiếp.

I.3 Các yếu tố ảnh hưởng đến lựa chọn trong thiết kế một hệ thống nhúng

Bên cạnh giá thành của sản phẩm, các yếu tố ảnh hưởng đến thiết kế thường bao gồm:

Khả năng xử lý:

Một trong những thông số thể hiện tốc độ xử lý của một bộ xử lý là triệu lệnh trên giây (Millions of instructions per second – MIPS). Thông thường, nếu MIPS lớn thì tốc độ xử lý nhanh hơn nếu xét ở cùng một ngữ cảnh (kiến trúc phần cứng, kiến trúc tập lệnh, kích thước thanh ghi...). Hiện nay, các máy tính đa năng thường có các bộ xử lý lớn (32 -bit, 64-bit) nhưng các hệ thống nhúng vẫn có thể dùng các bộ xử lý có số bit ít hơn (chẳng hạn như 8 -bit, 16-bit).

Bộ nhớ

Một hệ thống nhúng cần bộ nhớ (ROM và RAM) cần để lưu trữ mã lệnh và dữ liệu cần cho ứng dụng. Người thiết kế cần ước lượng dung lượng bộ nhớ cần thiết cho phần mềm dự kiến được phát triển. Cũng cần lưu ý là dung lượng bộ nhớ cần dùng có thể ảnh hưởng đến việc lựa chọn bộ xử lý tương ứng. Bộ xử lý có các tín hiệu địa chỉ n – bit thì dung lượng bộ nhớ tối đa là 2^n ô nhớ.

Số lượng đơn vị sản phẩm

Giá thành của sản phẩm và giá thành để tạo ra sản phẩm được quyết định chủ yếu bởi số lượng đơn vị sản phẩm được sản xuất và được bán ra.

Công suất tiêu thụ

Là lượng công suất tiêu thụ trong quá trình hoạt động. Điều này cực kỳ quan trọng trong các hệ thống nhúng, nhất là các thiết bị cầm tay sử dụng Pin. Một đơn vị thường dùng để so sánh là mW/MIPS (Miliwatts/MIPS). Giá trị này càng lớn thì công suất tiêu thụ càng lớn. Việc hướng đến công suất tiêu thụ thấp còn dẫn đến các đặc trưng khác của các thiết bị cầm tay như ít tỏa nhiệt, kích thước Pin nhỏ, nhẹ, kích thước nhỏ và thiết kế cơ khí đơn giản.

Chi phí phát triển sản phẩm

Bao gồm chi phí thiết kế phần mềm và phần cứng.

Thời gian tồn tại của sản phẩm

Thời gian tồn tại của sản phẩm quyết định tất cả các khía cạnh của thiết kế, từ việc lựa chọn các thành phần phần cứng, thời gian phát triển hệ thống và sản xuất ra sản phẩm cho đến giá thành.

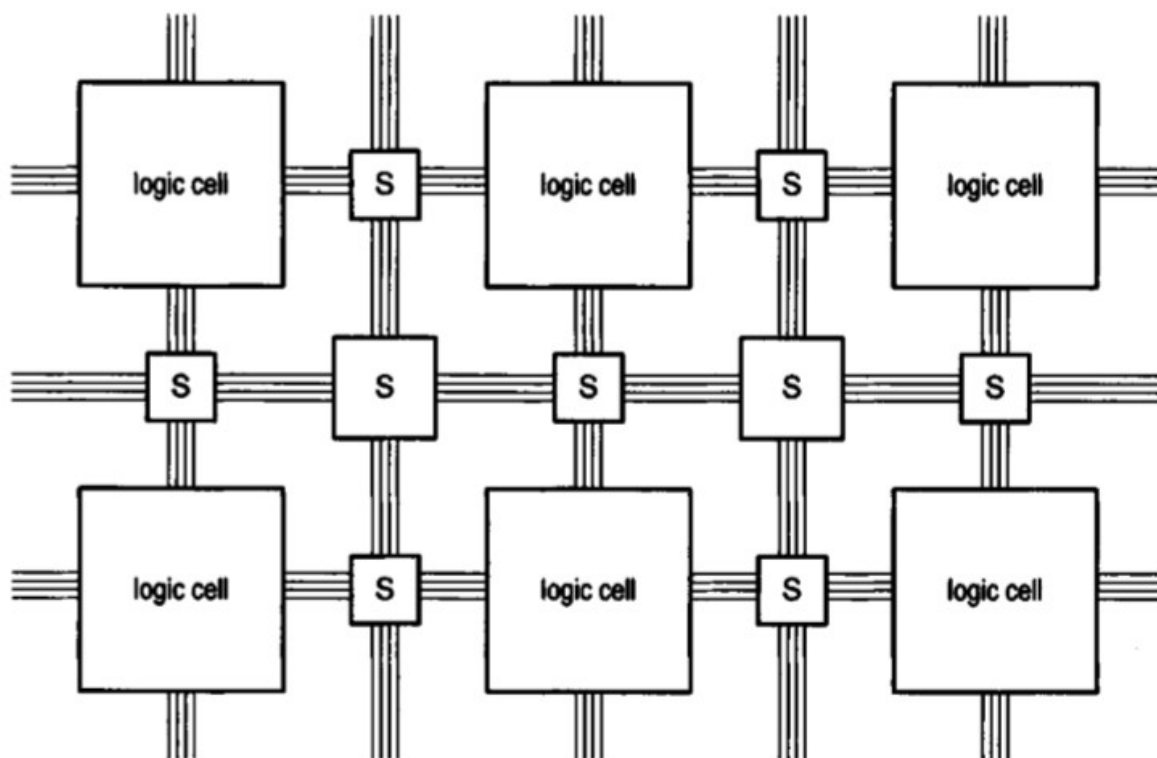
Độ tin cậy

Có những loại sản phẩm không cần phải đảm bảo hoạt động đúng 100% thời gian (ví dụ như đồ chơi của trẻ em...) nhưng có những sản phẩm đòi hỏi độ chính xác tuyệt đối (ví dụ như các loại khóa cửa ...)

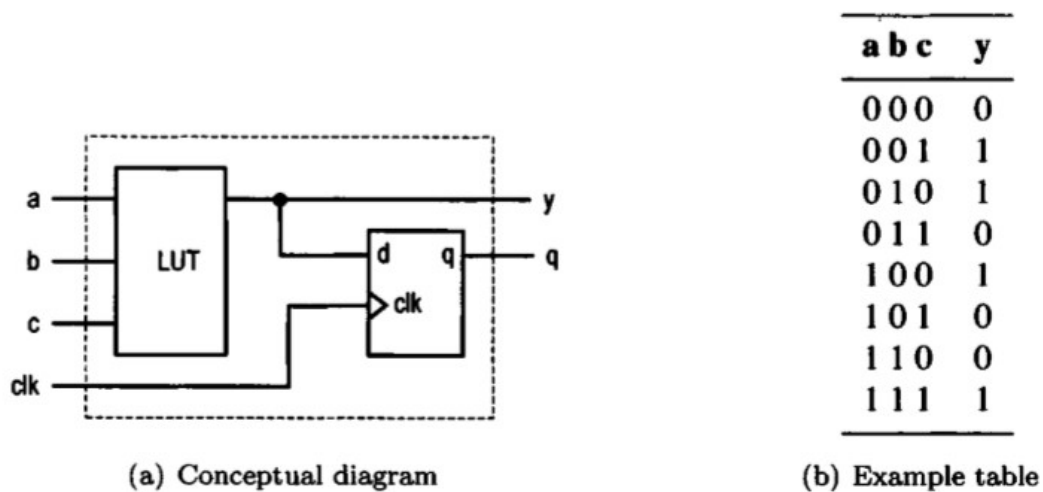
II. FPGA và quy trình thiết kế dùng phần mềm EDA (Electronic design automation).

II.1.1 Khái niệm về FPGA

FPGA là một linh kiện logic khả trình chứa mảng hai chiều của các tế bào logic đa năng (có thể cấu hình thực hiện các chức năng logic khác nhau) và các chuyển mạch có thể lập trình được. Mô hình đơn giản của linh kiện FPGA được minh họa ở hình 1,3. Tế bào logic có thể được lập trình để thực hiện một chức năng logic đơn giản và bộ chuyển mạch có thể lập trình để tạo ra kết nối giữa các tế bào logic. Một chức năng logic bất kỳ có thể được thực hiện bằng cách cấu hình các tế bào logic (thực hiện hàm logic) và thiết lập các chuyển mạch để tạo các kết nối giữa các tế bào logic. Sau khi thiết kế và tổng hợp mạch xong (bằng phần mềm thiết kế trên máy tính), chúng ta có thể dùng cáp nạp để tải xuống FPGA để cấu hình các tế bào logic và các chuyển mạch để tạo ra mạch logic mong muốn cần thiết kế. Quá trình này có thể thực hiện tại chỗ (in the field) chứ không cần thực hiện tại nhà máy (Fab). Chính vì vậy có cụm từ “field programmable” trong từ FPGA.



Hình 1.3: Cấu trúc mạng tính khái niệm của một FPGA



Hình 1.4: Minh họa thiết kế logic dựa trên LUT

Một tế bào logic thường có chứa một mạch tổ hợp nhỏ có thể cấu hình được, thường là một bảng tra cứu (LUT) và một flip - flop D (DFF) như được trình bày ở hình 1.4a. Một LUT n

ngõ vào có thể được xem như là các tín hiệu địa chỉ của một bộ nhớ và như vậy LUT sẽ hình thành một bộ nhớ gồm 2^n ô nhớ, mỗi ô nhớ chứa 1-bit. Chúng ta có thể dùng LUT để thực hiện hàm logic tổ hợp của một mạch có n ngõ vào bất kỳ với giá trị kết quả được lưu trữ ở nội dung ô nhớ tương ứng. Hình 1.4 b minh họa cho việc dùng LUT để thực hiện chức năng “ $y = a \text{ XOR } b \text{ XOR } c$ ”. Ngõ ra của LUT có thể được dùng trực tiếp (trong trường hợp chúng ta thiết kế mạch tổ hợp) hoặc được lưu trữ vào DFF (khi chúng ta thiết kế mạch tuần tự).

Hầu hết các linh kiện FPGA còn chứa các “macro cell” hoặc “macro block” là các khối bộ nhớ, các bộ nhân, các mạch quản lý đồng hồ (clock) ... được thiết kế sẵn. Các FPGA tiên tiến còn có chứa các bộ xử lý bên trong (như ARM chẳng hạn).

II.1.2 Giới thiệu về FPGA họ Cyclone IV của hãng Intel.

Họ linh kiện Cyclone IV FPGA của hãng Intel hướng đến các sản phẩm công suất thấp và giá rẻ. Họ Cyclone IV chia làm hai loại: CycloneIV E có công suất thấp, giá rẻ và đa năng và Cyclone IV GX tích hợp thêm các giao tiếp thu nhận dữ liệu đạt tốc độ 3.125 Gbps. Bảng 1,1 và bảng 1.2 trình bày các họ FPGA này.

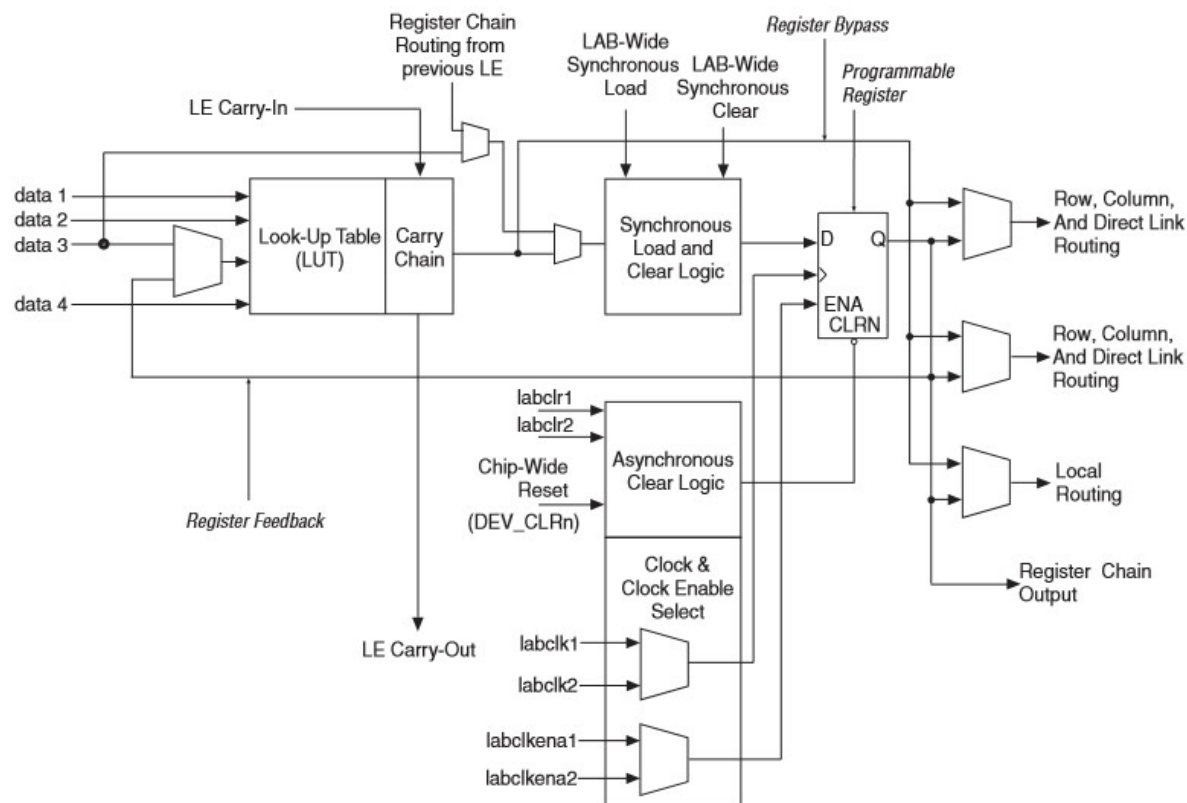
Resources	EP4CE6	EP4CE10	EP4CE15	EP4CE22	EP4CE30	EP4CE40	EP4CE55	EP4CE75	EP4CE115
Logic elements (LEs)	6,272	10,320	15,408	22,320	28,848	39,600	55,856	75,408	114,480
Embedded memory (Kbits)	270	414	504	594	594	1,134	2,340	2,745	3,888
Embedded 18 × 18 multipliers	15	23	56	66	66	116	154	200	266
General-purpose PLLs	2	2	4	4	4	4	4	4	4
Global Clock Networks	10	10	20	20	20	20	20	20	20
User I/O Banks	8	8	8	8	8	8	8	8	8
Maximum user I/O	179	179	343	153	532	532	374	426	528

Bảng 1.1 Họ linh kiện Cyclone IV E

Resources	EP4CGX15	EP4CGX22	EP4CGX30 (1)	EP4CGX30 (2)	EP4CGX50	EP4CGX75	EP4CGX110	EP4CGX150
Logic elements (LEs)	14,400	21,280	29,440	29,440	49,888	73,920	109,424	149,760
Embedded memory (Kbits)	540	756	1,080	1,080	2,502	4,158	5,490	6,480
Embedded 18 × 18 multipliers	0	40	80	80	140	198	280	360
General-purpose PLLs (GPLLs)	1	2	2	4 (4)	4 (4)	4 (4)	4 (4)	4 (4)
Multi-purpose PLLs (MPLLs)	2 (3)	2 (3)	2 (3)	2 (3)	4 (3)	4 (3)	4 (3)	4 (3)
Global clock networks	20	20	20	30	30	30	30	30
High-speed transceivers (7)	2	4	4	4	8	8	8	8
Transceiver maximum data rate (Gbps)	2.5	2.5	2.5	3.125	3.125	3.125	3.125	3.125
PCIe (PIPE) hard IP blocks	1	1	1	1	1	1	1	1
User I/O banks	9 (5)	9 (5)	9 (5)	11 (6)	11 (6)	11 (6)	11 (6)	11 (6)
Maximum user I/O	72	150	150	290	310	310	475	475

Bảng 1.2: Họ linh kiện Cyclone IV GX

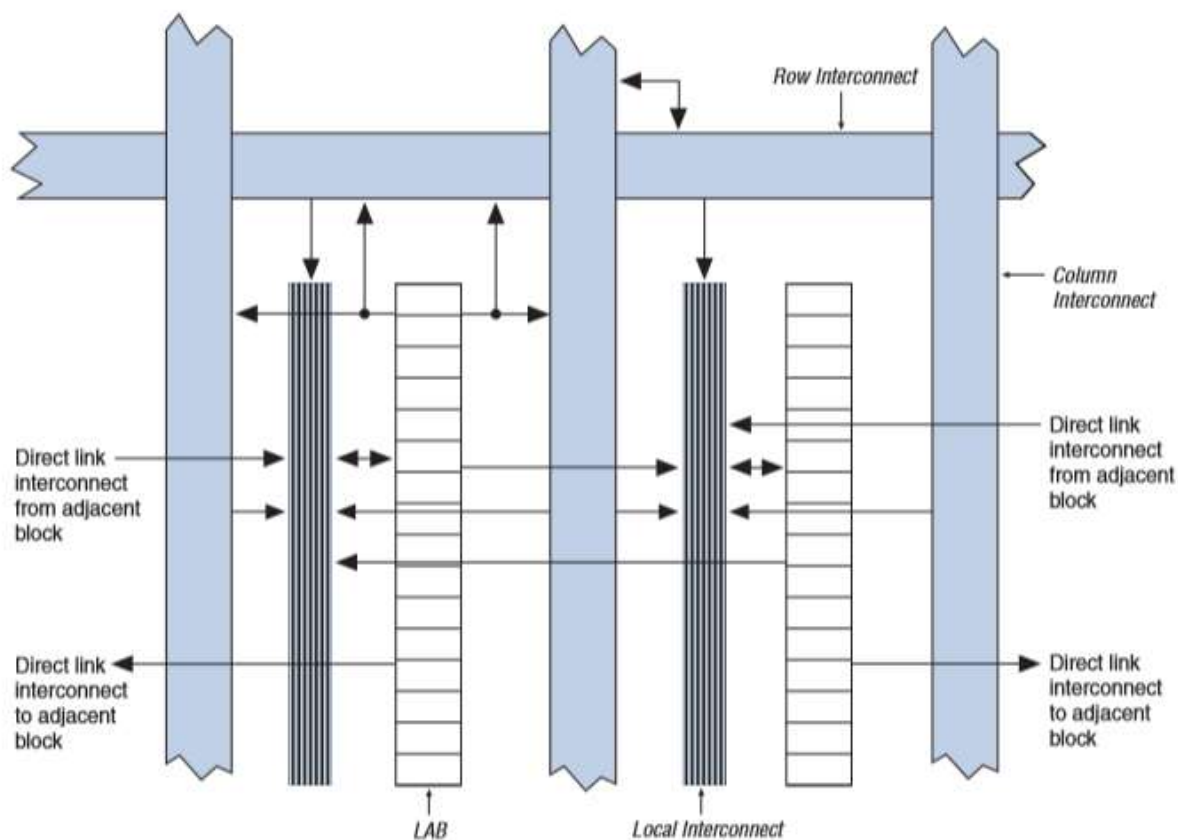
Hình 1.5 trình bày cấu trúc của một phần tử Logic lập trình được (Logic Element – LE) bên trong một Cyclone IV FPGA. Chúng ta có thể cấu hình các thanh ghi lập trình được của LE để thực hiện các loại Flip Flop (như DFF, TFF, JK FF hoặc SR FF) trong đó, mỗi thanh ghi có đầy đủ các ngõ vào (Data, Clock, Clock Enable, Clear ...). Các tín hiệu đồng hồ (clock) và xóa (clear) có thể được điều khiển từ các tín hiệu mạng đồng hồ toàn cục (global clock network), từ các chân I/O đa năng hoặc bất kỳ các tín hiệu logic bên trong nó. Chức năng của LUT đã được nói ở trước.



Hình 1.5: Cấu trúc của LE của Cyclone IV

Hình 1.6 trình bày cấu trúc của một khối mảng Logic (Logic array blocks - LABs). Mỗi LAB có những tính chất sau:

- Có 16 LE bên trong
- Các tín hiệu điều khiển LAB
- Các tín hiệu ghép nối giữa các chuỗi LE
- Các tín hiệu ghép nối các chuỗi thanh ghi
- Các tín hiệu ghép nối cục bộ bên trong LAB.



Hình 1.6: Cấu trúc LAB của Cyclone IV

II.2 Giới thiệu board mạch phát triển DE2

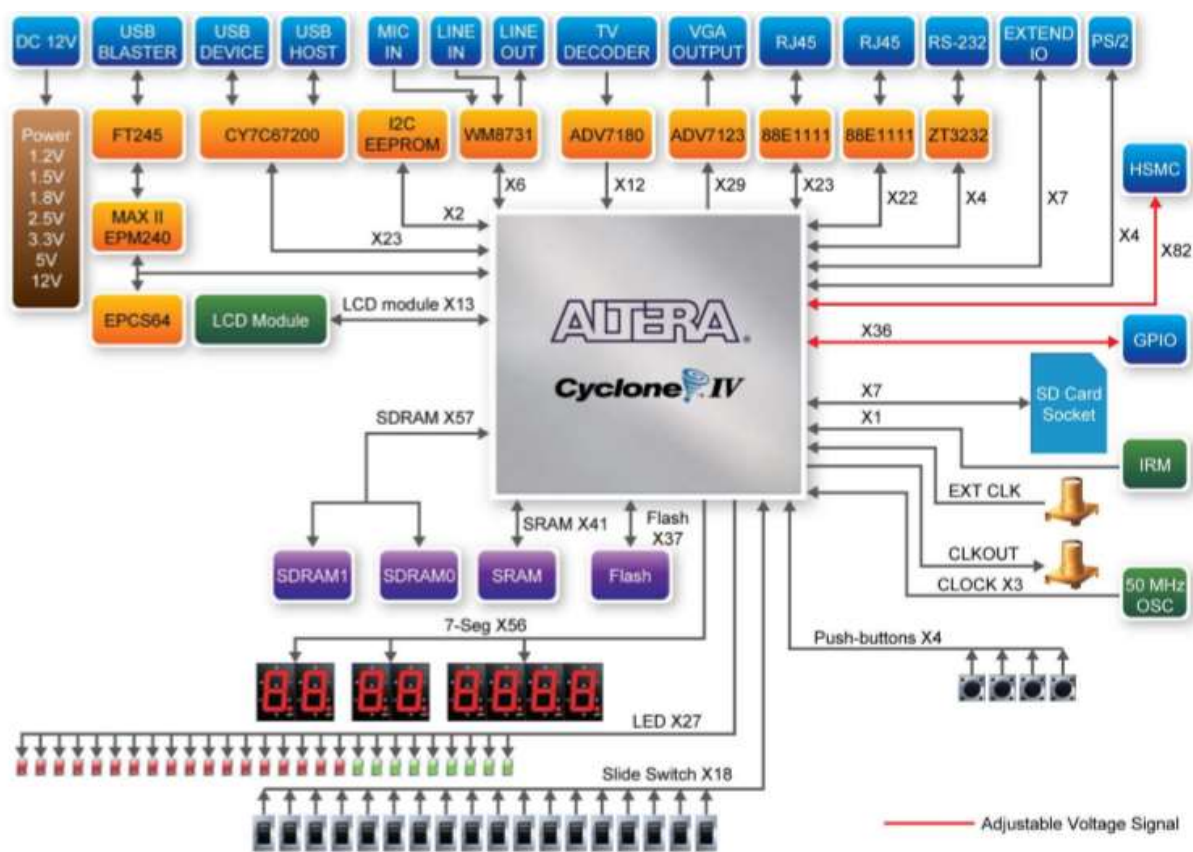
Board mạch phát triển DE2-115 được thiết kế nhằm cho phép người dùng có thể thực hiện trên diện rộng các thiết kế khác nhau từ các mạch đơn giản đến các ứng dụng phức tạp (như đa phương tiện).

Phân cứng được cung cấp trên board DE2 – 115:

- Linh kiện chính: Cyclone® IV 4CE115
- Linh kiện cấu hình nối tiếp (cho FPGA): EPCS64
- Bộ lập trình trên board: JTAG và AS (Active Serial)
- RAM tĩnh: 2MB
- RAM động: 2x64MB
- Flash memory: 8MB
- Khe cắm SD Card

- 4 nút nhấn
- 18 công tắc
- 18 LED màu đỏ
- 9 LED màu xanh
- Bộ dao động 50 MHz cho Clock
- Bộ mã hóa/giải mã âm thanh 24-bit (audio CODEC) với các đầu nối
- Bộ chuyển đổi từ tín hiệu số sang tương tự cho chuẩn hiển thị VGA DAC và đầu nối theo chuẩn VGA.
- Bộ giải mã tín hiệu TV (theo chuẩn (NTSC/PAL/SECAM) và đầu nối.
- Hai bộ giao tiếp chuẩn Ethernet tốc độ 2 Gigabit và đầu nối theo chuẩn RJ45.
- Bộ giao tiếp theo chuẩn USB loại A và loại B
- Giao tiếp thu nhận dữ liệu theo chuẩn RS-232 và đầu nối 9 chân
- Đầu nối PS/2 dùng cho bàn phím và chuột
- Bộ thu tín hiệu hồng ngoại (IR)
- Hai đầu nối SMA dùng để giao tiếp với Clock bên ngoài
- Một đầu nối giao tiếp 40 chân có Diode bảo vệ
- Một đầu nối giao tiếp tốc độ cao theo chuẩn HSMC(One High Speed Mezzanine Card - HSMC)
- Module LCD 16x2

Ngoài ra, người dùng cần làm quen với phần mềm thiết kế Quartus để có thể sử dụng và phát triển các ứng dụng trên DE2 – 115, Hình 1.7 trình bày sơ đồ khối của board mạch DE2 – 115.



Hình 1.7: Cấu trúc Bỏad mạch DE2 - 115

II.3 Quy trình thiết kế với FPGA

Quá trình thiết kế một ứng dụng trên FPGA luôn cần sự hỗ trợ dưới sự hỗ trợ của các phần mềm trên máy tính (ví dụ như phần mềm Quarus của hãng Intel). Hình 1.8 trình bày quy trình thiết kế tổng quát.

Design entry: Là thiết kế được xác định bởi người dùng ở dạng sơ đồ thiết kế hoặc ngôn ngữ mô tả phần cứng (Verilog HDL hoặc VHDL).

Synthesis (tổng hợp mạch): Là quá trình tạo ra mạch logic từ thiết kế của người dùng. Sythesis bao gồm 3 bước:

- *Tạo ra Netlist:* Bao gồm quá trình kiểm tra lỗi và thông báo lỗi từ thiết kế của người dùng. Sau khi các lỗi đã được chỉnh sửa quá trình này sẽ tạo ra một Nestlist (các biểu thức Logic để mô tả mạch bao gồm các thành phần như các bộ cộng, flip - flop, máy trạng thái...).

- *Tối ưu mức cổng*: Quá trình này sử dụng thông tin từ Netlist để tạo ra một mạch tương đương nhưng tốt hơn tùy theo mục tiêu tối ưu hướng đến (chẳng hạn như tối ưu tài nguyên thiết kế hay tốc độ thực thi hay cả hai).
- *Ánh xạ thiết kế vào công nghệ tương ứng*: Ở bước này, các thành phần trong Netlist được hiện thực bằng các tài nguyên có sẵn trong vi mạch mà thiết kế sẽ dùng (chẳng hạn như LE, bộ nhớ nhúng ... bên trong vi mạch FPGA).

Mô phỏng chức năng của thiết kế: Ở bước này, với sự hỗ trợ của phần mềm thiết kế (ví dụ như Modelsim) người dùng có thể thực hiện kiểm tra thiết kế về mặt chức năng mà chưa quan tâm đến thời gian trễ trong mạch bằng cách áp vào Netlist (đã được tổng hợp) các mẫu dữ liệu kiểm tra và nhận được dữ liệu ngõ ra (là kết quả xử lý của mạch thiết kế). So sánh dữ liệu ngõ ra cho bởi mạch thiết kế từ phần mềm và kết quả mong đợi, người dùng có thể biết được thiết kế của mình có đúng như mong muốn về mặt chức năng hay không.

Thiết kế vật lý: Bao gồm ba giai đoạn là bố trí, nối dây, và phân tích thời gian tĩnh.

Giai đoạn bố trí (placement): Giai đoạn này sẽ lựa chọn vị trí trên linh kiện dùng cho thiết kế (ví dụ như linh kiện Cyclone IV) cho mỗi khối Logic trong Netlist đã tạo ra ở trước.

Giai đoạn nối dây (routing): Sau khi những vị trí dành cho các khối Logic trong mạch được xác định trên Chip, giai đoạn nối dây sẽ thực hiện việc kết nối các khối với nhau bằng cách dùng các dây nối tồn tại bên trong trong Chip.

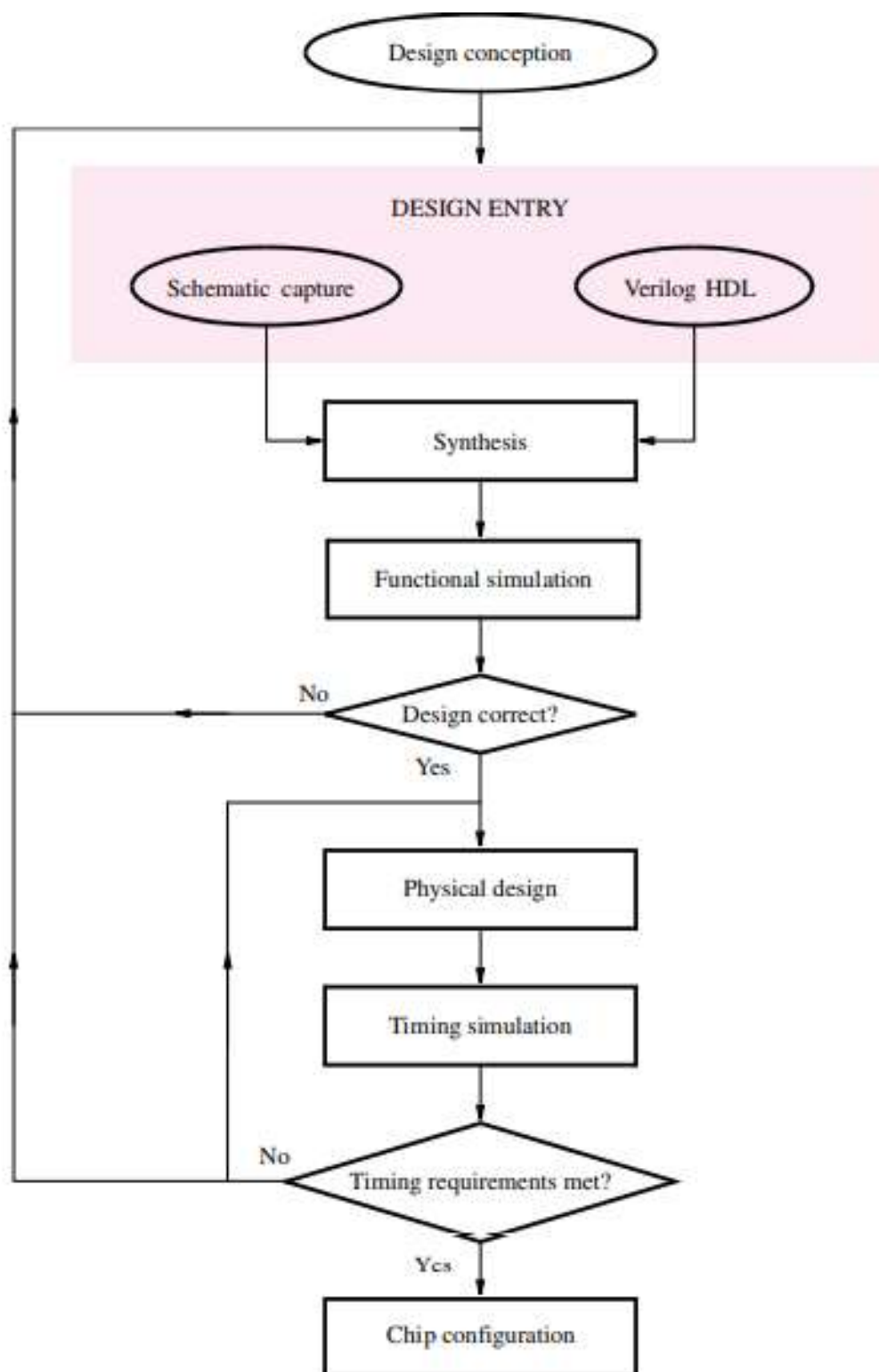
Giai đoạn phân tích thời gian tĩnh: Sau giai đoạn nối dây, phần mềm thiết kế sẽ tính toán thời gian trễ giữa các khối và các dây nối trong Chip. Phần mềm phân tích thời gian tĩnh sẽ tính toán các thông tin thời gian trễ và tạo ra một bảng chứa thông tin xác định năng suất hoạt động của mạch. Thường có bốn tham số quan trọng được cung cấp là F_{\max} : Xác định tần số hoạt động tối đa của Clock của mạch, được xác định bởi đường đi trễ nhất giữa hai Flip Flop bất kỳ trong mạch, t_{su} : Thời gian thiết lập (Setup time) là thời gian tín hiệu cần giữ ổn định trước khi Clock tác động, t_{co} : là thời gian trễ của tín hiệu tính từ khi Clock tác động để tín hiệu xuất hiện ở ngõ ra của Flip Flop cho đến khi tín hiệu xuất hiện ở chân của linh kiện, t_h : Thời gian giữ tín hiệu (hold time) là thời gian tối thiểu cần duy trì giá trị của tín hiệu ổn định sau khi Clock tác động. Bảng 1.3 Trình bày kết quả phân tích thời gian tĩnh cho một thiết kế cụ thể.

Parameter	Actual	Required	Slack	From	To
F_{max}	261.1 MHz	200 MHz	1.17 ns	AddSub	Overflow
t_{su}	2.356 ns	10.0 ns	7.644 ns	b_0	$breg_0$
t_{co}	6.772 ns	10.0 ns	3.228 ns	$zreg_0$	z_0
t_h	0.240 ns	N/A	N/A	b_1	$breg_1$

Bảng 1.3. Kết quả phân tích môi trường cho bởi phần mềm Quartus

Mô phỏng thời gian của thiết kế: Đây là giai đoạn cuối cùng trong quá trình thiết kế dùng CAD. Cả hai hành vi về thời gian và chức năng của mạch cùng được mô phỏng. Các công cụ CAD sẽ ước tính thời gian trễ của tất cả các phần tử Logic, dây nối và các nguồn tài nguyên khác tồn tại trong linh kiện được dùng để thiết kế mạch.

Sau khi kết quả mô phỏng đáp ứng được các yêu cầu thì thiết kế sẽ được nạp xuống FPGA để cấu hình FPGA thực hiện chức năng Logic mong muốn.



Hình 1.8: Quy trình phát triển một thiết kế cho FPGA

III. Thiết kế hệ thống nhúng dùng SoPC builder

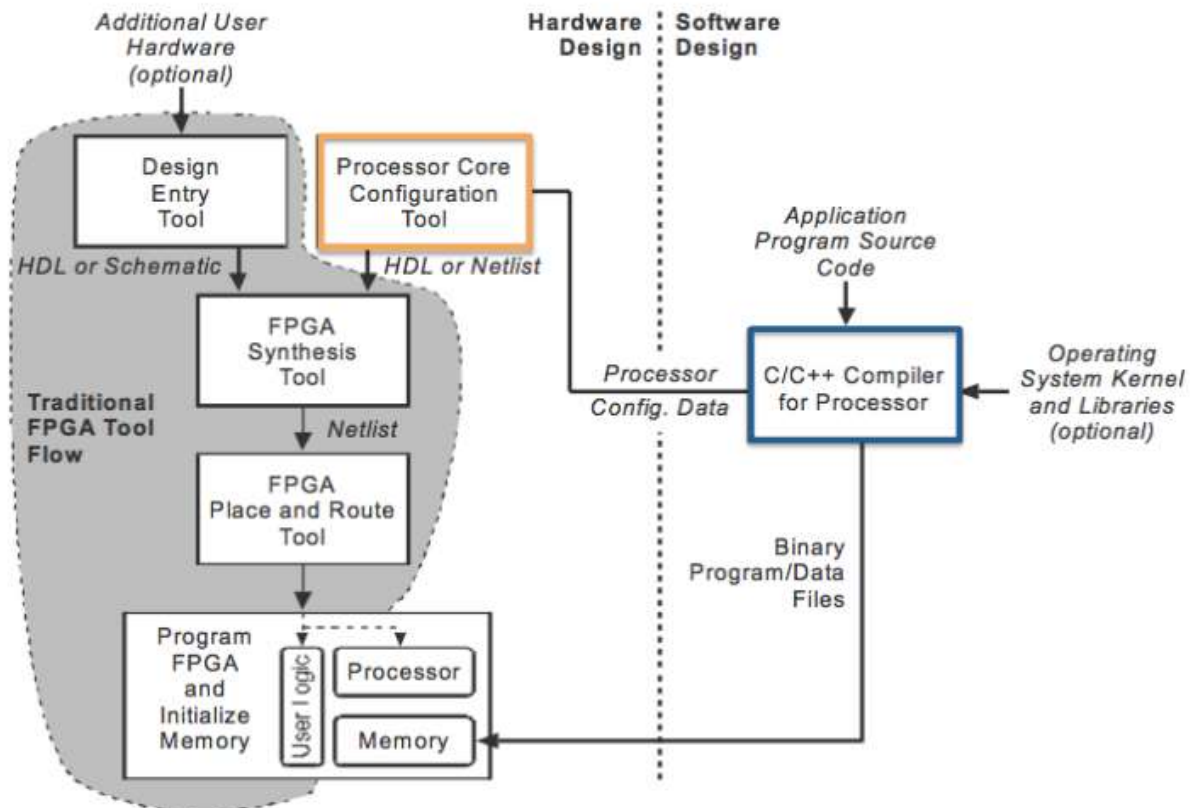
Thiết kế một hệ thống SoPC thường bao gồm hai phần: Thiết kế phần cứng và thiết kế phần mềm chạy trên phần cứng đã được thiết kế. Hình 1.9 trình bày lưu đồ thiết kế một hệ thống SoPC.

Trong quá trình thiết kế phần cứng, người thiết kế có thể tích hợp các IP (bao gồm các IP được nhà sản xuất cung cấp hoặc các IP được phát triển riêng) bằng công cụ tích hợp hệ thống ví dụ như phần mềm Qsys (SoPC Builder). Kết quả ở giai đoạn này thường là thiết kế được đặt tả ở dạng ngôn ngữ mô tả phần cứng. Sau đó, người thiết kế dùng file kết quả này để tiếp tục quy trình thiết kế trên FPGA với phần mềm Quartus II như trên.

Sản phẩm ở bước thiết kế phần cứng thường bao gồm CPU, bộ nhớ, các bộ tính toán phần cứng chuyên dụng và các giao tiếp ngoại vi.

Thiết kế phần mềm: Dùng phần mềm NIOS II Embedded Design Suite (EDS) – Eclipse. Phần mềm thường được viết bằng ngôn ngữ C/C++.

Lưu ý là so với quy trình thiết kế trên FPGA tổng quát ở phần II.3 ở trên thì thiết kế SoPC còn thêm giai đoạn phát triển phần mềm. Phần mềm này sẽ chạy trên CPU nhúng (NIOS II hoặc ARM) đã được sử dụng trong giai đoạn thiết kế Logic.



Hình 1.9: Lưu đồ thiết kế SoPC

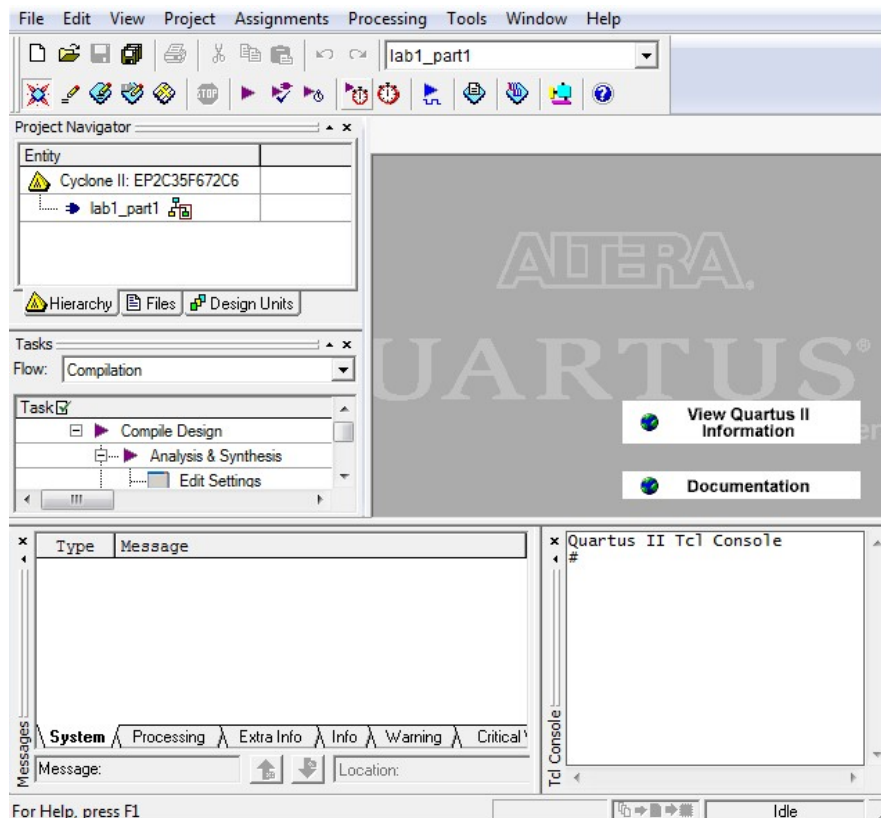
IV. Minh họa thiết kế

Các bước bên dưới trình bày quá trình thiết kế một phần cứng SoPC đơn giản gồm CPU NIOS II, bộ nhớ và giao tiếp vào ra (hỗ trợ quá trình Debug) cũng như viết một chương trình ứng dụng đơn giản cho phần cứng được thiết kế.

Cũng cần lưu ý là giao diện hay các bước thực hiện có thể khác đi đôi chút tùy theo phiên bản phần mềm sử dụng để thiết kế cũng như FPGA dùng trong thiết kế.


Bước 1: Tạo project trên Quartus II

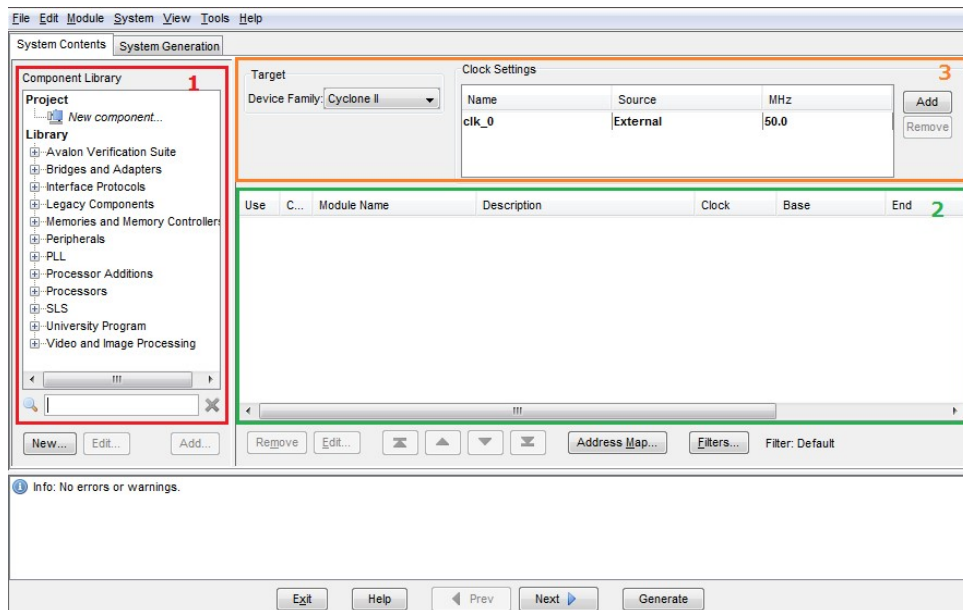
Trong công cụ Quartus II → chọn File → New Project Wizard, đặt tên project là lab1_part1 (lưu ý: đường dẫn đến thư mục lưu project không chứa khoảng trắng). Chọn FPGA sẽ dùng trong thiết kế (ví dụ Cyclone II EP2C35F672C6). Project sau khi tạo xong có giao diện như Hình 1.10.



Hình 1.10: Giao diện sử dụng của Quartus II

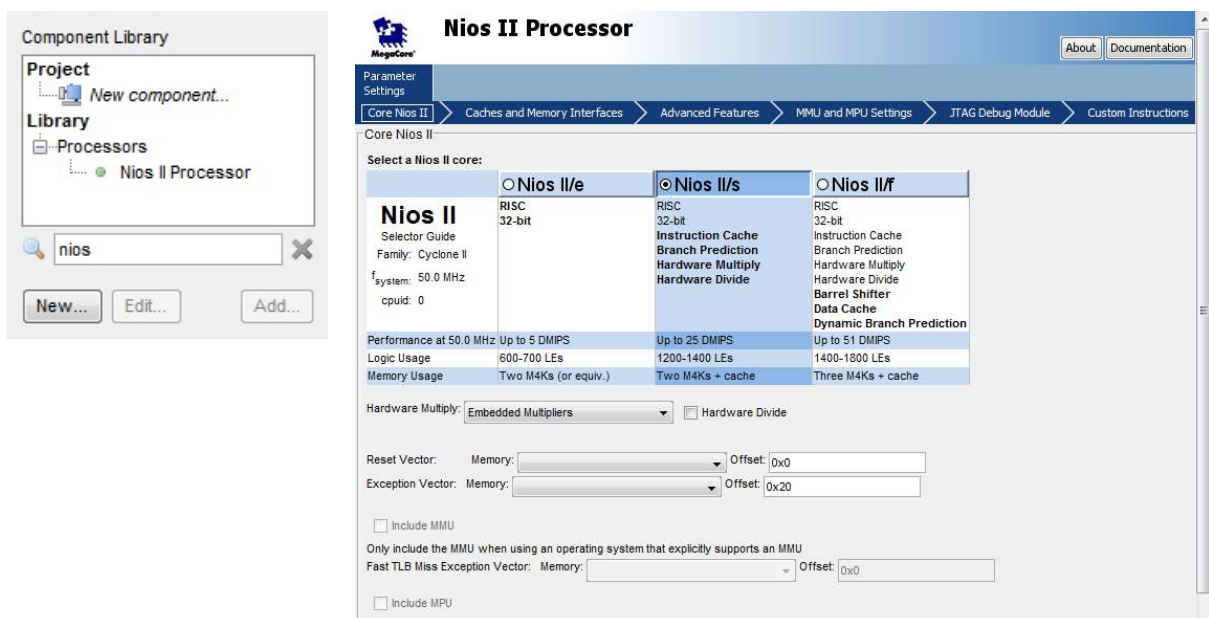
Bước 2: Tạo hệ thống SoPC

- Trong công cụ Quartus II → chọn Tools → SoPC Builder hoặc nhấn vào biểu tượng 
- Trong công cụ SoPC Builder → Đặt System Name là nios_sys, Targer HDL là Verilog. SoPC Builder gồm 2 tabs System Contents và System Generation. Tab System Contents có giao diện như Hình 1.11
 - ✓ (1) Component Library: chứa các IP Cores đã được thiết kế sẵn
 - ✓ (2): chứa các IP Cores dùng được tích hợp vào hệ thống
 - ✓ (3) Target/Clock Settings: chứa các thông số hệ thống



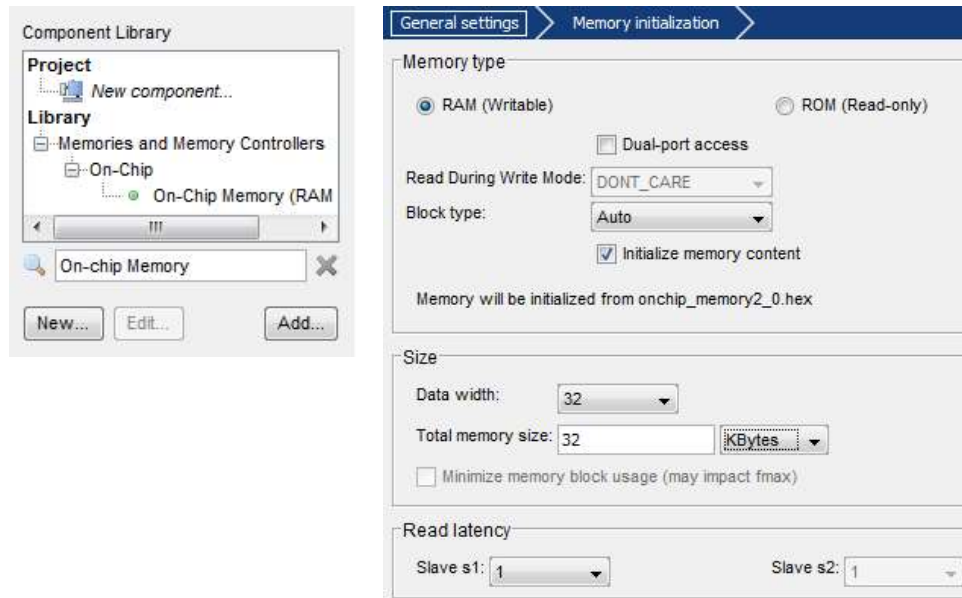
Hình 1.11: Giao diện sử dụng của SoPC Builder

- Thêm CPU NIOS II vào hệ thống bằng cách gõ “nios” vào khung tìm kiếm, chọn Nios II Processor → Add → chọn NIOS II/s → Finish. Thông số cấu hình được minh họa như Hình 1.12



Hình 1.12: Cấu hình CPU NIOS II

- Thêm bộ nhớ On-chip Memory vào hệ thống bằng cách gõ “On-chip Memory” vào khung tìm kiếm, chọn On-chip Memory → Add → chọn Total Memory size là 32 KB → Finish. Thông số cấu hình được minh họa như Hình 1.13.



Hình 1.13: Cấu hình Onchip Memory

- Thêm giao tiếp JTAG UART vào hệ thống bằng cách gõ “JTAG UART” vào khung tìm kiếm, chọn JTAG_UART → Add → Finish.
- Cài đặt lại thông số cho Nios II Processor bằng cách double-click vào cpu_0 → chọn Reset Vector và Exception Vector trở vào on-chip_memory2_0 → Finish. Thông số cấu hình được minh họa như Hình 1.14.



Hình 1.14: Cấu hình vector cho CPU NIOS II

- Chọn System → Auto-Assign Base Addresses và Auto-Assign IRQs.
- Hệ thống SoPC sau khi thiết lập xong được minh họa như Hình 1.15.
 - ✓ Cột Use báo hiệu thành phần nào được kết nối vào hệ thống SoPC.
 - ✓ Cột Connection mô tả kết nối master/slave giữa các thành phần trong hệ thống.
 - ✓ Cột Description mô tả tên gọi và kết nối của các thành phần trong hệ thống.
 - ✓ Cột Clock báo hiệu xung clock cấp cho từng thành phần trong hệ thống.

- ✓ Cột Base, End báo hiệu địa chỉ bắt đầu và kết thúc của các thành phần trong hệ thống.
- ✓ Cột IRQ mô tả ngắt (interrupt) được sử dụng trong hệ thống. Các ngắt được đánh số từ 0 trở đi. Ngắt 0 có mức ưu tiên cao nhất, rồi đến ngắt 1, 2, ...

Use	Conn...	Module Name	Description	Clock	Base	End	Tags	IRQ
<input checked="" type="checkbox"/>		cpu_0	Nios II Processor					
		instruction_master	Avalon Memory Mapped Master	clk_0				
		data_master	Avalon Memory Mapped Master					
		jtag_debug_module	Avalon Memory Mapped Slave					
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)					
		s1	Avalon Memory Mapped Slave	clk_0	0x00008000	0x0000ffff		
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART					
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00011000	0x00011007		

Hình 1.15: Hệ thống SoPC đơn giản

- Kiểm tra hệ thống SoPC còn lỗi hay không thông qua cửa sổ **Info: No errors or warnings.** Information
- Chọn Tab System Generation → nhấn vào nút Generate → Save. Nếu Generate hoàn tất sẽ hiện ra thông báo

2011.10.01 12:32:18 (*) SUCCESS: SYSTEM GENERATION COMPLETED.
 Info: System generation was successful.

Bước 3: Tổng hợp và biên dịch hệ thống SoPC để nạp xuống FPGA

- Trong công cụ Quartus II → project lab1_part1, tạo file lab1_part1.v với nội dung như sau:


```

1 module lab1_part1 (
2
3     input        CLOCK_50,
4     input [0:0]  KEY
5
6 );
7
8 // =====
9 // nios_sys.v
10 // =====

```

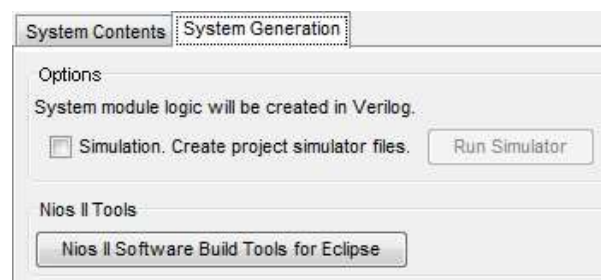
9	nios_sys NIOS_SYS
10	(
11	.clk_0 (CLOCK_50),
12	.reset_n (KEY[0])
13);
14	
15	Endmodule

Hình 1.16: File top-level của project

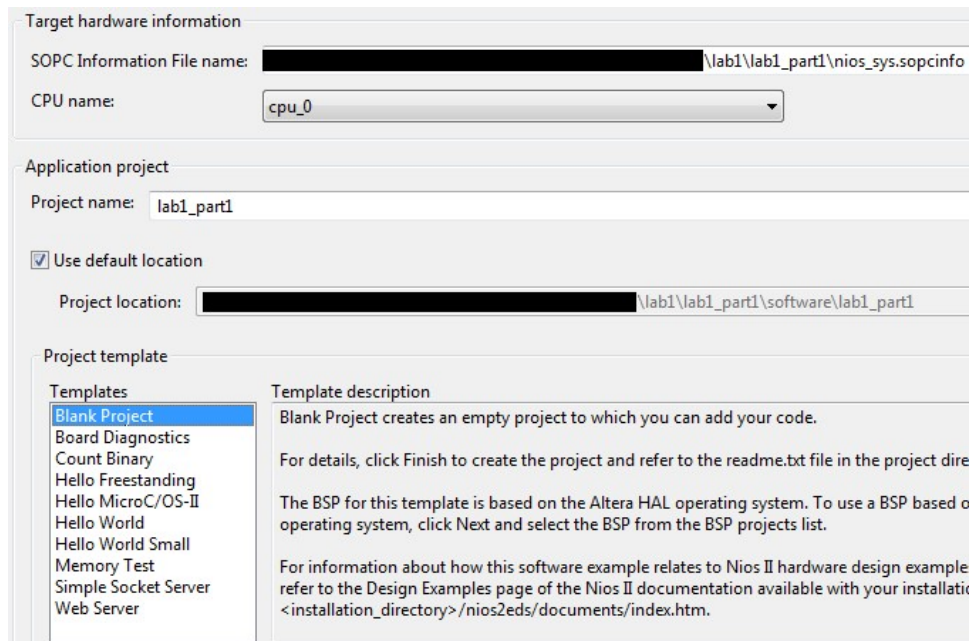
- Dòng 9 → 13: copy nội dung trong file nios_sys_inst.v vào, thay đổi “nios_sys nios_sys_inst” thành “nios_sys NIOS_SYS”, clk_0 thành CLOCK_50, reset_n thành KEY[0] như Hình 1.16.
- Gán chân (pin) cho FPGA bằng cách vào Assignments → Import Assignments ... → add file DE2_pin_assignments.csv vào.
- Tiến hành biên dịch và tổng hợp bằng cách vào Processing → Start Compilation hoặc nhấn Ctrl+L hoặc nhấn 

Bước 4: Phát triển phần mềm trên NIOS II

- Trong công cụ Altera SoPC Builder → System Generation → NIOS II Software Build Tools for Eclipse.

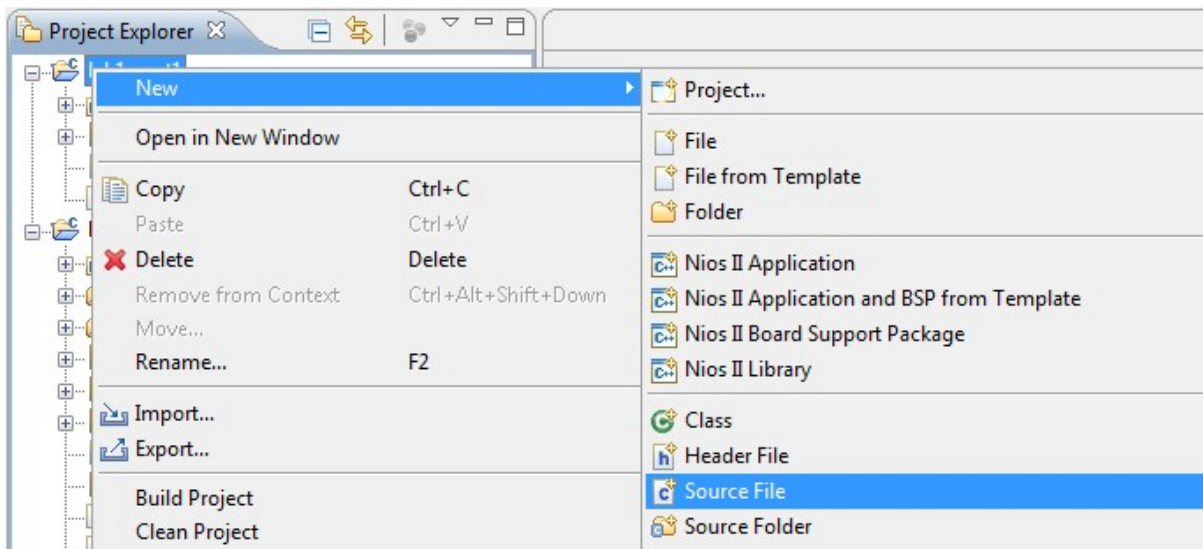


- Trong công cụ NIOS II – Eclipse Platform → chọn File → New → NIOS II Application and BSP from Template → chọn SOPC Information File name là nios_sys.sopcinfo (nằm trong thư mục ...lab1_part1) → đặt Project name là lab1_part1 → Project template là Blank Project → Finish.



Hình 1.17: Tạo project trong NIOS II Eclipse

- Trong tab Project Explorer, click phải vào lab1_part1 → New → Source File → đặt tên Source File là lab1_part1.c → Finish.



Hình 1.18: Tạo file trong NIOS II Eclipse

- File lab1_part1.c có nội dung như sau

1	void main()
---	--------------------

2	{
3	volatile int a = 0;
4	volatile int b = 2;
5	volatile int c = 5;
6	
7	a = b + c;
8	b = a - c;
9	}

Hình 1.19: Chương trình điều khiển trong NIOS II Eclipse

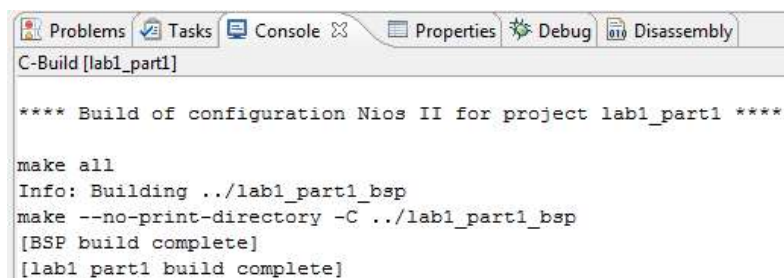
- Chọn Project → Build Project hoặc click chuột phải vào lab1_part1 trong tab Project Explorer → Build Project hoặc nhấn Ctrl+B để biên dịch chương trình.

Lưu ý: do NIOS II 9.1 không hỗ trợ hoàn toàn Windows 7 nên để biên dịch chương trình ta thực hiện các bước sau:

- Click chuột phải vào lab1_part1 → Nios II → Nios II Command Shell ... → gõ lệnh make all. Nếu biên dịch thành công màn hình sẽ báo hiệu

```
Info: <lab1_part1.elf> 18 KBytes program size (code + initialized data).
Info:      13 KBytes free for stack + heap.
Info: Creating lab1_part1.objdump
nios2-elf-objdump --disassemble --syms --all-header --source lab1_part1.e
1_part1.objdump
[lab1_part1 build complete]
```

- Sau đó, click chuột phải vào lab1_part1 → Build Project.
- Nếu biên dịch thành công, sẽ xuất hiện thông báo như Hình 1.20.



```
C-Build [lab1_part1]

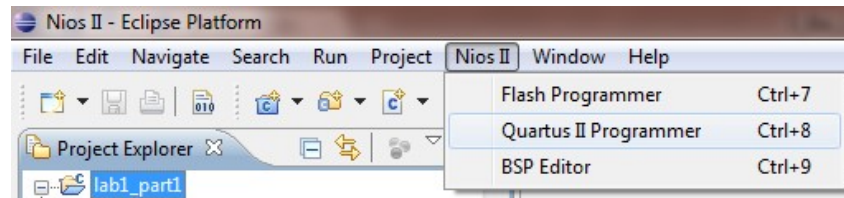
**** Build of configuration Nios II for project lab1_part1 ****

make all
Info: Building ../lab1_part1_bsp
make --no-print-directory -C ../lab1_part1_bsp
[BSP build complete]
[lab1_part1 build complete]
```

Hình 1.20: Dòng lệnh thông báo sau khi biên dịch thành công

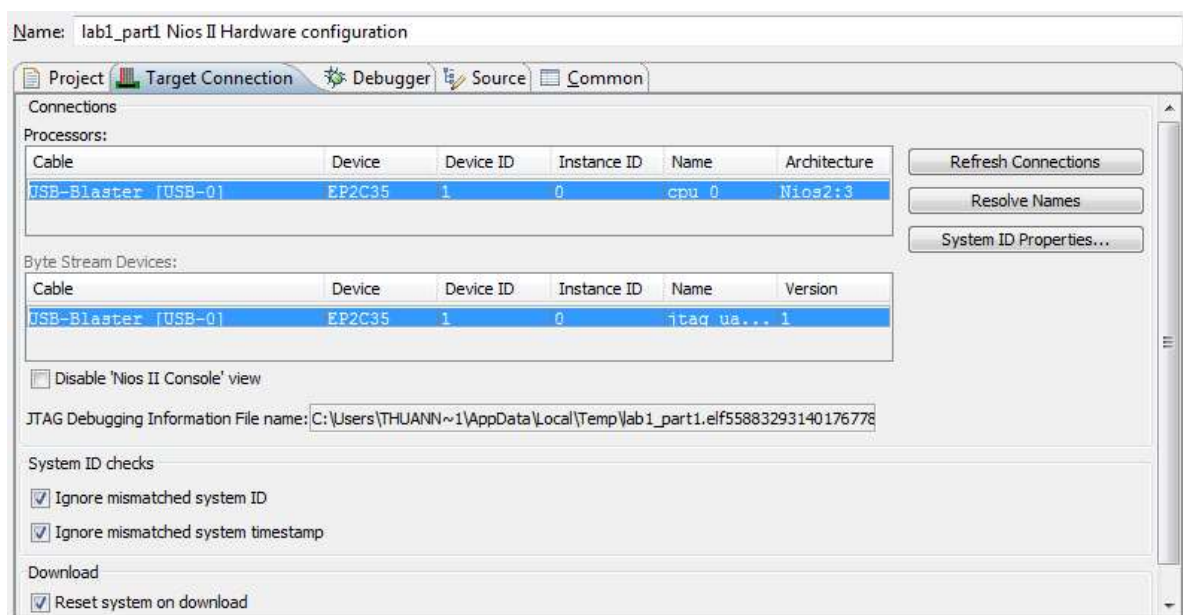
Bước 5: Kiểm tra hoạt động của hệ thống SoPC trên FPGA

- Trong công cụ NIOS II – Eclipse Platform → chọn Nios II → Quartus II Programmer hoặc nhấn Ctrl+8.



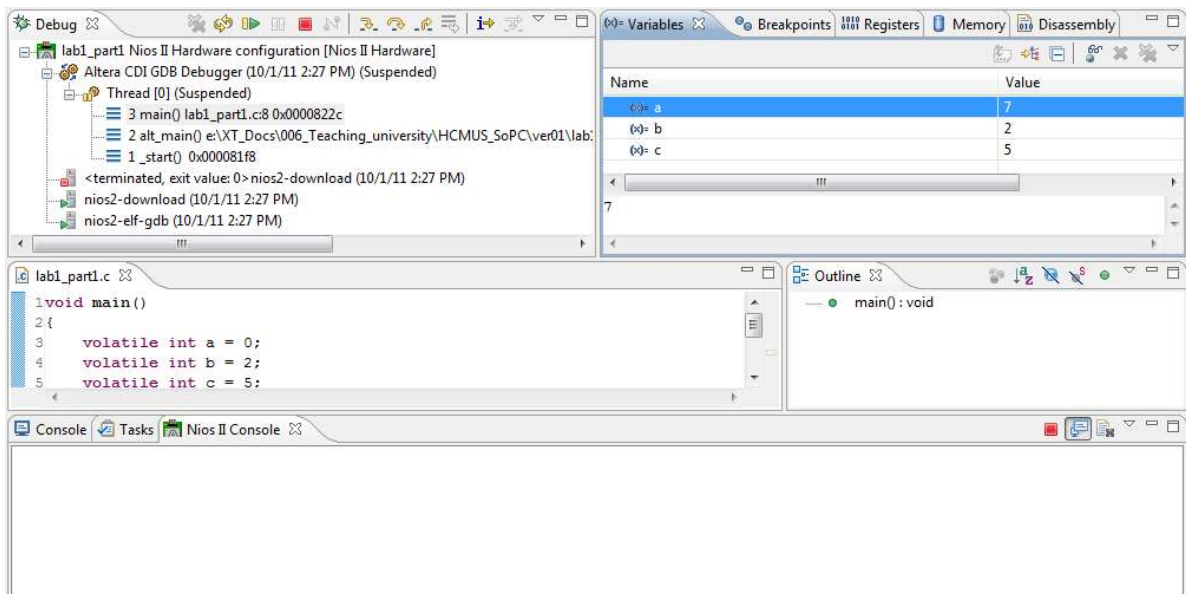
Hình 1.21: Công cụ Quartus II Programmer

- Trong công cụ Quartus II Programmer → Add File ... → .../lab1/lab1_part1/lab1_part1.sof → Start.
- Trong công cụ Nios II – Eclipse Platform → click phải vào lab1_part1 trong tab Project Explorer → chọn Run As → 2 Nios II Hardware nếu muốn thực thi project hoặc chọn Debug As → 2 Nios II Hardware nếu muốn debug project.
- Trong cửa sổ Debug Configurations → click Refresh Connections đến khi nào tab Connections xuất hiện USB-Blaster → thiết lập các thông số như Hình 1.22 → Debug.



Hình 1.22: Cấu hình nạp xuống FPGA trên NIOS II Eclipse

- Giao diện debug được minh họa như Hình 1.23



Hình 1.23: Giao diện debug trên NIOS II Eclipse

V. Bài tập

- Câu 1. Trình bày các thành phần của một hệ thống nhúng?
- Câu 2. Trình bày điểm khác nhau giữa các hệ thống nhúng dựa trên MCU, DSP và FPGA?
- Câu 3. Phân biệt khái niệm SoPC và SoC FPGA?
- Câu 4. Trình bày quy trình thiết kế FPGA?
- Câu 5. Quy trình thiết kế SoPC và các phần mềm thiết kế SoPC của hãng Intel?

Tài liệu tham khảo chương 1

1. <https://www.safaribooksonline.com/library/view/programming-embedded-systems/0596009836/ch01.html>
2. Introduction_to_the_Altera_SOPC_Builder.pdf, hãng intel
3. Tài liệu về FPGA họ Cyclone IV:
<https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-53001.pdf>
4. Tài liệu về Board thí nghiệm DE2 115:
<https://www.intel.com/content/www/us/en/programmable/solutions/partners/partner-profile/terasic-inc-/board/altera-de2-115-development-and-education-board.html>

5. Tài liệu “Thực hành môn SoPC”, khoa Điện tử - Viễn thông, trường Đại học Khoa học Tự nhiên Tp. HCM