ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO TỔNG QUAN VỀ RISC-V

MÔN: ĐỒ ÁN 1

Lớp: CE201.O21.MTCL

Giảng viên hướng dẫn:

- Trương Văn Cương

Sinh viên thực hiện:

- Phan Trọng Nhân 21522408

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

Giới thiệu về RISC-V

RISC-V (Reduced Instruction Set Computing - Five): là kiến trúc tập lệnh mở và đang nhận được nhiều sự chú ý trong những năm gần đây do tính linh hoạt, tính mô-đun và khả năng mở rộng của nó trong thiết kế. Không giống như những kiến trúc độc quyền, RISC-V cho phép truy cập bản thiết kế và tùy chỉnh để phù hợp với nhu cầu người thiết kế cho mục đích hoặc ứng dụng nhất định (IOT, AI,..). Với RISC-V, lợi ích là: bộ xử lý tùy chỉnh tiết kiệm chi phí, ứng dụng cải tiến và triển khai bảo mật mạnh mẽ. Công nghệ này được coi là xử lý tương lai, có thể tùy chỉnh trong tay mỗi chúng ta.

Lịch sử và sự phát triển RISC-V

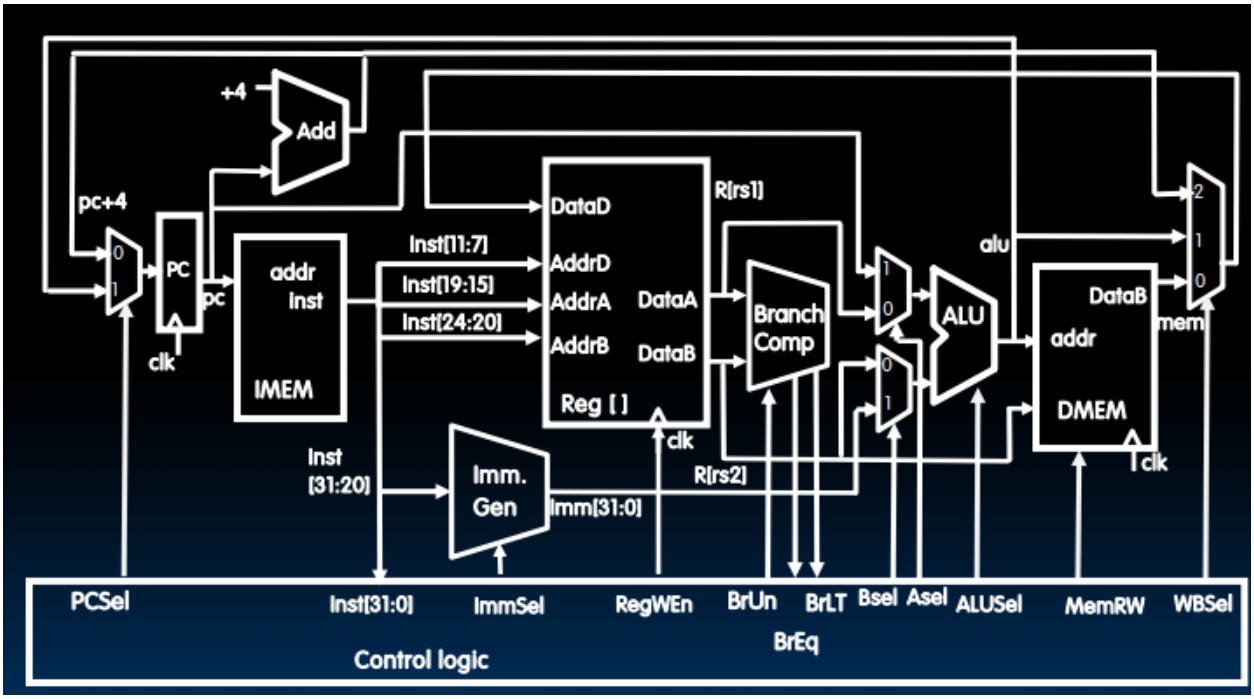
Năm 2010 các nhà phát triển của Berkeley đã phát minh ra kiến trúc tập lệnh mới đơn giản, mở, miễn phí cấp phép và miễn phí bản quyền đó là RISC-V vì lúc bấy giờ những kiến trúc như x86, ARM đã trở nên phức tạp và cồng kềnh qua nhiều năm, cùng với giới hạn truy cập và áp dụng phí cấp quyền lên những kiến trúc này. Thêm vào đó việc sử dụng mã RTL cho bộ xử lí ARM không được hổ trợ và mã nguồn cho bộ xử lí x86 không có sẵn. Các kiến trúc khác như SPAC, OpenRISC cũng ít nhiều có vấn đề. Kiến trúc máy tính và kiến trúc tập lệnh đã rất hoàn thiện trong nhiều thập kỷ, nhưng các tổ chức nghiên cứu như Berkeley không thể chọn kiến trúc tập lệnh phù hợp để sử dụng. Do đó, các phát triển của Berkeley đã quyết định phát minh ra một kiến trúc tập lệnh mới từ đây.

Sự phát triển của RISC-V được thúc đẩy bởi nhiều yếu tố có thể kể ra như sau: nhu cầu tùy chỉnh và tính linh hoạt của bộ xử lí, mong muốn giảm sự phụ thuộc thuộc vào kiến trúc tập lệnh độc quyền và nhu cầu ngày càng tăng về các giải pháp điện toán tiết kiệm năng lượng và chi phí,… Có thể nói, RISC-V đã mở ra một hướng đi mới trong thiết kế bộ xử lý và có tiềm năng định hình lại cục diện của ngành công nghiệp bán dẫn.

Kiến trúc RISC-V

Datapath và Control của RV32I

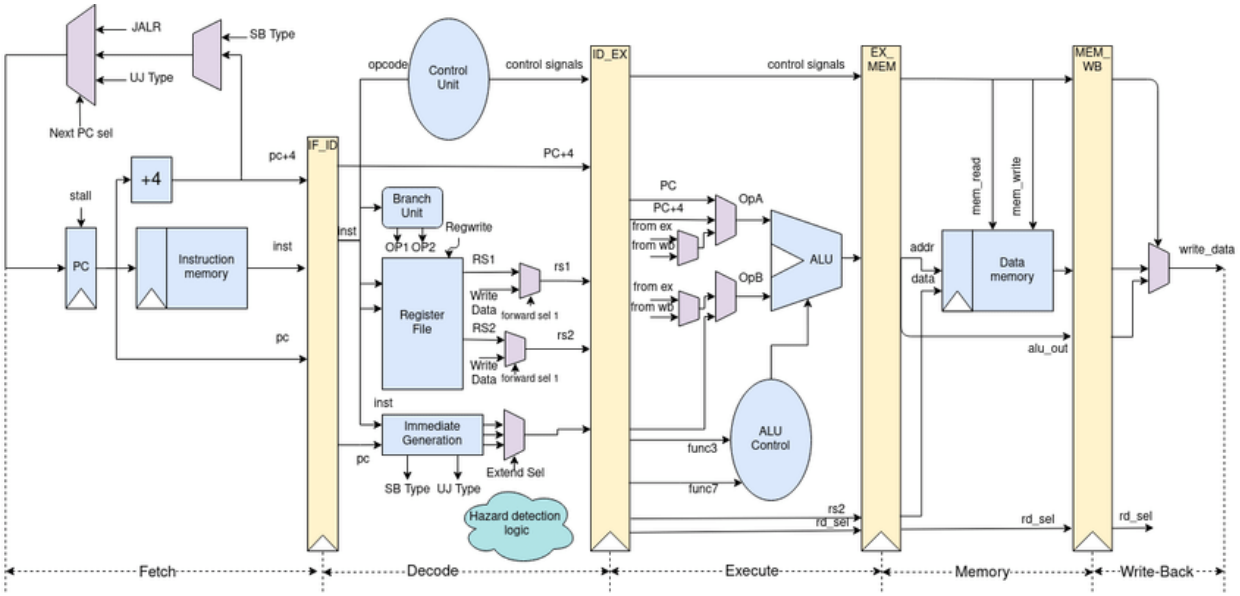
Kiến trúc đơn chu kì



Hình 1. Datapath và Control của RV32I đơn giản

Đây là kiến trúc cơ bản của RISC-V thực hiện mỗi lệnh trong một chu kì xung clock. Từ hình ảnh ta có thể thấy nó không khác nhiều so với những kiến trúc mở trước đây những thành phần chính như: Register File, PC, Instruction Memory, ALU, Data Memory vẫn được giữ nguyên và nó là cốt lõi. Tuy nhiên, do có nhiều lệnh riêng biệt về nhảy và rẻ nhánh nên giữa nó và các kiến trúc khác xuất hiện 2 khối: Immediate Generator, Branch Compare để thực hiện những tác vụ phù hợp với nhu cầu người thiết kế.

Kiến trúc Pipeline 5 tầng



Hình 2. Kiến trúc pipeline 5 tầng của RV32I

Như đã đề cập ở trên những thành phần chính của kiến trúc RISC-V vẫn được giữ nguyên và RISC-V 5 tầng được phát triển lên từ kiến trúc RISC-V cơ bản. Tuy nhiên, với kiến trúc pipeline này bao gồm 5 giai đoạn: Lấy lệnh, Giải mã, Thực thi, Truy cập bộ nhớ và Ghi lại, giúp tăng tốc độ xử lý và giảm tiêu thụ năng lượng.

Việc chia nhỏ quá trình xử lý lệnh thành 5 giai đoạn nhỏ, giúp tăng hiệu quả xử lý bằng cách thực hiện song song các giai đoạn khác nhau từ đó sử dụng bộ dự đoán nhánh để dự đoán hướng thực thi tiếp theo của chương trình, giúp giảm thời gian chờ khi thực hiện các lệnh rẽ nhánh. Ngoài ra, sử dụng bộ đệm dữ liệu để lưu trữ dữ liệu truy cập gần đây, giúp tăng tốc độ truy cập dữ liệu.

Tập lệnh con mô-đun

Tập lệnh số nguyên cơ sở

Phân loại tập lệnh	Tên tập lệnh	Số tập lệnh	Mô tả
Cơ bản	RV32I	47	- Lệnh số nguyên - Không gian địa chỉ 32 bit, 32 thanh ghi 32 bit - Bao gồm các lệnh số học, logic và truy cập bộ nhớ.
	RV32E	47	- Các lệnh tương tự như RV32I, ngoại trừ số lượng thanh ghi chỉ có 16 đối với môi trường nhúng.
	RV64I	59	- Lệnh số nguyên - Không gian địa chỉ 64 bit, 32 thanh ghi 64 bit - Bao gồm các lệnh số học, logic và truy cập bộ nhớ.
	RV128I	71	- Lệnh số nguyên - Không gian địa chỉ 128 bit, 32 thanh ghi 128 bit - Bao gồm các lệnh số học, logic và truy cập bộ nhớ.

Bảng 1. Phân loại lệnh số nguyên

Phần cơ bản nhất của tập lệnh mà RISC-V yêu cầu là tập hợp con lệnh số nguyên cơ bản được biểu thị bằng chữ cái I. Với tập hợp con lệnh số nguyên này, một trình biên dịch phần mềm hoàn chỉnh có thể được triển khai do tính đơn giản của nó và những thành phần đã được quy định từ trước. Tập lệnh số nguyên cơ sở, có thể gọi là tập lệnh "RV32I" , "RV64I" hay "RV128I", tùy thuộc vào kích thước không gian địa chỉ và mục đích tổng thể mà chức năng yêu cầu.

Đối với tập lệnh số nguyên thì định dạng lệnh được chia thành 6 loại bảng 2 sẽ cho ta thấy chức năng của từng loại lệnh này.

Loại	Mô tả
R-type	Được sử dụng cho các phép toán giữa các thanh ghi, chẳng hạn như phép toán số học và logic. (add $s1, $t0, $t2)
I-type	Được sử dụng cho các phép toán tức thời, chẳng hạn như các phép toán số học và logic có giá trị tức thời.(add $s1, $t0, 10)
S-type	Dùng để lưu trữ dữ liệu từ thanh ghi lên bộ nhớ. (sw $s0, (0)$s1)
B-type	Được sử dụng cho các phép toán rẻ nhánh có điều kiện, chuyển điều khiển sang một lệnh khác dựa trên một điều kiện. (beq $s0,$s0, EXIT)
U-type	Được dùng để nạp một giá trị nhất thời vào thanh ghi. (lui $s0, 100)
J-type	Được sử dụng cho các pháp toán nhảy vô điều kiện, chuyển điều khiển sang lệnh khác một cách vô điều kiện. (J EXIT)

Bảng 2. Các kiểu của lệnh số nguyên

Tập lệnh con mở rộng

Ngoài tập lệnh số nguyên cơ sở, RISC-V còn bao gồm một tập hợp các phần mở rộng tiêu chuẩn cung cấp chức năng chuyên biệt cho các ứng dụng cụ thể. Các tiện ích mở rộng này có thể được thêm vào bộ xử lý RISC-V khi cần, cho phép tùy

chỉnh và tối ưu hóa ở mức độ cao. Một số tiện ích mở rộng tiêu chuẩn đáng chú ý nhất là:

Phân loại tập lệnh	Tên tập lệnh	Số tập lệnh	Mô tả
Mở rộng	M	8	Thực hiện nhanh các phép tính nhân và chia số nguyên, dùng trong xử lý tín hiệu và mã hóa.
	A	11	Cho phép truy cập bộ nhớ cùng lúc mà không cần đồng bộ phức tạp, hữu ích cho hệ đa nhân và đa luồng
	F	26	Chứa các lệnh dấu phẩy động có độ chính xác đơn
	D	26	Chứa các lệnh dấu phẩy động có độ chính xác kép
	Q	26	Chứa các lệnh dấu phẩy động có độ chính xác gấp bốn lần
	C	46	Giúp giảm kích thước code bằng bộ lệnh 16-bit, cải thiện hiệu năng cho hệ thống hạn chế bộ nhớ.

Bảng 3. Các kiểu lệnh mở rộng

Ví dụ về tập lệnh mở rộng nếu RV32IM được triển khai, tập lệnh cơ bản 32 bit và tập lệnh mở rộng nhân chia sẽ được triển khai.

RISC-V Register File

Bộ nhớ tạm RISC-V (register file) là thành phần quan trọng lưu trữ dữ liệu trong khi CPU xử lý lệnh. Do đó, Register file được tổ chức thành tập thanh ghi số nguyên hay thanh ghi dấu phẩy động, tùy thuộc vào phần mở rộng được triển khai trong bộ xử lý.

Thanh ghi số nguyên

Những thanh ghi này được sử dụng để lưu trữ và thao tác các giá trị số nguyên trong quá trình thực hiện các lệnh. Chúng ta có thể thực hiện các phép tính như cộng, trừ, nhân, chia, thao tác bit và so sánh bằng cách sử dụng các thanh ghi này.

Quy ước đặt tên cho thanh ghi:

Thanh ghi	ABI Name	Mô tả
x0	zero	Thanh ghi được gán cứng vào giá trị 0 và không thể sửa đổi được.
x1	ra	Thanh ghi địa chỉ trả về, được sử dụng để lưu trữ địa chỉ trả về trong các lệnh gọi hàm.
x2	sp	Thanh ghi con trỏ stack, được sử dụng để quản lý stack.
x3	gp	Thanh ghi con trỏ toàn cục, được sử dụng để truy cập dữ liệu toàn cục.
x4	tp	Thanh ghi con trỏ luồng (Thread), được sử dụng để lưu trữ cục bộ luồng.
x5-7	t0-2	Các thanh ghi tạm thời, được sử dụng để lưu giữ các giá trị trung gian trong quá trình tính toán.
x8	s0/fp	Thanh ghi đã lưu, trỏ khung.
x9	s1	Thanh ghi đã lưu.
x10-11	a0-1	Thanh ghi đối số được sử dụng để truyền đối số của hàm và trả về giá trị hàm
x12-17	a2-7	Được sử dụng để truyền đối số của hàm.
x18-27	s2-11	Các thanh ghi đã lưu, được sử dụng để bảo toàn các giá trị trong các lệnh gọi hàm.

x28-31	t3-6	Thanh ghi bổ sung.

Bảng 4. Quy ước tên các thanh ghi số nguyên

Bằng những quy ước đối với thanh ghi ở trên giúp đơn giản hóa việc phát triển trình biên dịch và các công cụ phần mềm khác tạo mã cho bộ xử lý.

Thanh ghi dấu phẩy động

Các thanh ghi này khả dụng khi phần mở rộng F hoặc phần mở rộng D được triển khai trong bộ xử lý. Các thanh ghi dấu phẩy động được đặt tên theo quy ước "f" theo sau là một số, chẳng hạn như f0, f1, f2, v.v., cho đến f31. Thanh ghi dấu phẩy động được tổ chức thành nhiều loại dựa trên mục đích sử dụng của chúng:

Loại thanh ghi	Mô tả
ft0-ft7 (f0-f7)	Thanh ghi tạm thời, lưu trữ giá trị trung gian
fs0-fs11 (f8-f9, f18-f27)	Các thanh ghi đã lưu, được sử dụng để bảo toàn các giá trị trong các lệnh gọi hàm.
fa0-fa7 (f10-f17)	Dùng để truyền đối số cho hàm
ft8-ft11 (f28-f31)	Thanh ghi tạm thời bổ sung, lưu trữ giá trị trung gian

Bảng 5. Quy ước đặt tên thanh ghi dấu phẩy động

Giống như thanh ghi số nguyên thanh ghi dấu phẩy động cũng nhờ vào những quy ước đối với thanh ghi ở trên giúp đơn giản hóa việc phát triển trình biên dịch và các công cụ phần mềm khác tạo mã cho bộ xử lý.

RISC-V Memory Model

Mô hình bộ nhớ đóng một vai trò quan trọng trong hiệu suất và hiệu quả của bộ xử lý, vì nó xác định cách tổ chức và truy cập dữ liệu trong quá trình thực hiện các lệnh. Trong kiến trúc RISC-V, mô hình bộ nhớ hỗ trợ cả bộ nhớ ảo và vật lý, cung cấp một khuôn khổ linh hoạt và hiệu quả để quản lý tài nguyên bộ nhớ.

Bộ nhớ ảo

Với việc phân tách bộ nhớ bộ nhớ ảo làm cho mỗi tiến trình có không gian địa chỉ riêng biệt, đảm bảo bảo mật và ổn định. Thêm vào đó, bộ nhớ ảo còn quản lý bộ nhớ giúp hệ điều hành phân bổ và giải phóng bộ nhớ hiệu quả hơn, giảm phân mảnh.Ngoài ra, nó cũng mở rộng không gian địa chỉ truy cập nhiều bộ nhớ hơn dung lượng vật lý, sử dụng đĩa như lưu trữ thứ cấp.

Trong RISC-V, bộ nhớ ảo thường được triển khai bằng phần mềm, nghĩa là không có sự hỗ trợ cụ thể từ phần cứng cho cơ chế này. Các hệ điều hành và trình quản lý bộ nhớ của RISC-V sẽ quản lý bộ nhớ ảo bằng cách sử dụng các kỹ thuật như phân trang hoặc phân đoạn. Tuy nhiên đối với những thiết kế không quá lớn thì bộ nhớ ảo không quá cần thiết trong hệ thống.

Bộ nhớ vật lý

Là bộ nhớ thực tế của hệ thống (RAM, ROM). Được tổ chức thành không gian địa chỉ phẳng, mỗi địa chỉ tương ứng với một vị trí duy nhất. Bộ xử lý truy cập thông qua mô hình bộ nhớ, dịch địa chỉ ảo thành địa chỉ vật lý khi cần. Được thực hiện bởi đơn vị quản lý bộ nhớ (MMU). MMU sử dụng bảng trang đa cấp để dịch địa chỉ ảo sang địa chỉ vật lý. MMU đảm bảo bảo mật và kiểm soát truy cập bộ nhớ.

Kiến trúc RISC-V hỗ trợ nhiều kiểu địa chỉ:

Địa chỉ + offset: Tính toán dựa trên giá trị thanh ghi gốc và offset.

Địa chỉ tức thời: Địa chỉ được cung cấp trực tiếp trong lệnh.

Địa chỉ gián tiếp qua thanh ghi: Địa chỉ được lưu trữ trong một thanh ghi.

Chế độ đặc quyền của RISC-V

Các cấp độ này được thiết kế để cung cấp một môi trường an toàn và được kiểm soát để chạy phần mềm, đảm bảo rằng các ứng dụng và thành phần hệ thống chỉ có thể truy cập vào tài nguyên và thực hiện các hoạt động mà chúng được phép. Kiến trúc RISC-V xác định ba chế độ làm việc, còn được gọi là chế độ đặc quyền:

Chế độ	Mô tả
Chế độ máy (M - mode)	- Mức đặc quyền cao nhất: Truy cập toàn bộ tài nguyên hệ thống. - Sử dụng cho phần mềm hệ thống: Quản lý phần cứng, tạo môi trường an toàn. - Khả năng: Truy cập/sửa đổi thanh ghi, bộ nhớ, thực hiện bất kỳ lệnh nào.
Chế độ giám sát (S - mode)	- Mức đặc quyền trung gian: Hạn chế quyền truy cập tài nguyên. - Sử dụng cho phần mềm hệ thống: Hệ điều hành, màn hình ảo. - Kiểm soát truy cập: Đảm bảo hoạt động hợp lệ, ngăn chặn truy cập trái phép.
Chế độ người dùng (U - mode)	- Đặc quyền thấp nhất: Kiểm soát chặt chẽ tài nguyên và chức năng. - Dành cho ứng dụng: Chạy với quyền hạn hạn chế, do phần mềm giám sát kiểm soát. - Ngăn chặn: Truy cập trái phép và sửa đổi tài nguyên hệ thống, ứng dụng khác.

Bảng 6. Các chế độ cấp quyền của RISC-V

Trong kiến trúc RISC-V, Chế độ máy là bắt buộc và hai chế độ còn lại là tùy chọn. Các hệ thống khác nhau có thể được thực hiện thông qua sự kết hợp các phương thức khác nhau vì nó là chế độ cơ bản để thực thi các chương trình.

Trình biên dịch (Compiler)

RISC-V là một kiến trúc tập lệnh (ISA) phần cứng mã nguồn mở, được thiết kế để linh hoạt, hiệu quả và dễ dàng triển khai. Để chạy chương trình trên bộ xử lý RISC-V, chúng ta cần một bộ biên dịch để chuyển đổi mã nguồn (thường được viết bằng c, c++,..) thành ngôn ngữ assembly, từ ngôn ngữ này tiếp tục qua assembler để biên dịch thành mã máy (gọi là lệnh RISC-V) mà bộ xử lý có thể hiểu và thực thi. Trong quá trình bộ biên dịch RISC-V thực hiện chuyển đổi mã nguồn sang mã máy RISC-V trình biên dịch giúp phân tích cú pháp và ngữ nghĩa, tối ưu hóa mã (loại bỏ mã thừa, sắp xếp lệnh, thực hiện phép toán hằng số), và kiểm tra lỗi (phát hiện lỗi cú pháp và ngữ nghĩa) để giúp gỡ lỗi chương trình dễ dàng hơn.

Triển khai RISC-V

Triển khai trên lõi mềm

Thiết kế RISC-V tổng hợp được viết bằng HDL cho phép linh hoạt, chi phí thấp và phát triển nhanh chóng trên FPGA/CPLD, tuy nhiên hiệu suất sẽ thấp hơn so với lõi cứng. Ưu điểm của nó bao gồm khả năng dễ dàng sửa đổi, thiết kế theo yêu cầu, tiết kiệm chi phí phát triển và thời gian phát triển, kiểm tra và triển khai nhanh hơn. Nhược điểm là hiệu suất thấp hơn so với lõi cứng trên chip silicon. Các ứng dụng phù hợp cho thiết kế này bao gồm tạo mẫu và thử nghiệm, dự án nhỏ, phát triển ý tưởng và đưa sản phẩm ra thị trường nhanh chóng. Ví dụ về các thiết kế RISC-V tổng hợp bao gồm Spike, trình mô phỏng RISC-V mã nguồn mở với trình biên dịch đơn giản, và Rocket Chip, lõi RISC-V được thiết kế tại UC Berkeley.

Triển khai lõi cứng

Thiết kế RISC-V tích hợp trực tiếp vào chip silicon tùy chỉnh (ASIC, SoC) cho hiệu suất cao hơn, tiêu thụ điện năng thấp hơn và tích hợp cao hơn, tuy nhiên chi phí và thời gian phát triển sẽ cao hơn so với thiết kế tổng hợp trên FPGA/CPLD. Ưu điểm của nó bao gồm khả năng đạt hiệu suất tối ưu, giảm thiểu tiêu thụ điện năng và tích hợp nhiều chức năng vào cùng một chip. Nhược điểm là chi phí phát triển cao hơn và thời gian phát triển lâu hơn do yêu cầu thiết kế và sản xuất chip phức tạp hơn. Những ví dụ về triển khai lõi cứng có thể kể đến là: SiFive E2: SoC RISC-V hiệu suất cao được thiết kế cho các ứng dụng nhúng Kendryte K210: SoC RISC-V giá rẻ được thiết kế cho các ứng dụng IoT.