EventQueue: Giao tiếp liên quá trình dựa trên sự kiện và có nhận thức ưu tiên cho các hệ thống nhúng

Fabian Mauroner và Marcel Baunach
Viện Tin học Kỹ thuật Đại học Công
nghệ Graz, Graz, Áo
Email: {mauroner, baunach}@tugraz.at

Trừu tượng-Các hệ thống nhúng hiện đại đang nhắm mục tiêu cho các nhiệm vụ riêng biệt và cho một Giao tiếp liên quy trình (IPC) hiệu quả. Tuy nhiên, thống nhất cả hai yêu cầu không phải là một thách thức nhỏ. Trong bài báo này, chúng tôi đề xuất EventQueue thống nhất cả hai yêu cầu với sự hỗ trợ của phần mở rộng phần cứng. Với hàng đợi tin nhắn, dữ liệu được chuyển từ tác vụ này sang tác vụ khác. Do đó, việc gửi dữ liệu được thực hiện trong phần cứng, giúp loại bỏ vấn đề Đảo ngược mức độ ưu tiên của hệ điều hành (OS-PI) và đồng thời cho phép tách biệt các tác vụ với nhau. Chúng tôi đã triển khai EventQueue vào nhà thờ Hồi giáoMCU, chạy trong Mảng cổng lập trình trường (FPGA). Điều này được minh họa trong các đánh giá hiệu suất. Đánh giá cho thấy sự cải thiện thông lượng đáng kể, điều làm cho EventQueue rất phù hợp với các hệ thống nhúng thời gian thực trong tương lai.

Điều khoản lập chỉ mục—Những hệ thống nhúng; Truyền thông liên quy trình; Hệ điều hành-Nhận thức; Triển khai FPGA

Tôi tôiGIỚI THIỆU

Trong các hệ thống nhúng thời gian thực ngày nay, Hệ điều hành (OS) thời gian thực vẫn là một lớp cơ bản của toàn bộ hệ thống; và do đó, vẫn là một phần của nghiên cứu ngày nay. Trong hệ điều hành, một tập hợp hữu hạn các bộ phận phần mềm, được gọi lànhiệm vụs, được chọn để lên lịch trên Bộ xử lý trung tâm (CPU). Qua đó, HĐH chọn một tác vụ theo chính sách lập lịch của HĐH. Chính sách lập lịch của HĐH thời gian thực nhằm đáp ứng tất cả các ràng buộc thời gian thực của nhiệm vụ; do đó, một nhiệm vụ phải được bắt đầu và / hoặc hoàn thành trong một giới hạn thời gian thấp hơn và / hoặc trên tương ứng. Việc vi phạm các giới hạn thời gian này có thể dẫn đến những tình huống kịch tính, nơi máy móc bị đập phá hoặc thậm chí tính mạng con người bị ảnh hưởng. Do đó, phải đảm bảo rằng tất cả các ràng buộc thời gian thực trong hệ thống nhúng thời gian thực đều được thỏa mãn.

Phân tích khả năng lập lịch (ví dụ, [1]) chứng minh nếu một tập hợp các nhiệm vụ không vi phạm các ràng buộc thời gian thực. Phân tích tính toán việc sử dụng toàn bộ hệ thống yêu cầu, trong số các tham số khác, Thời gian thực hiện trường hợp tồi tệ nhất (WCET) của mỗi tác vụ. WCET dài hơn làm tăng việc sử dụng toàn bộ hệ thống; và do đó, việc sử dụng có thể vượt quá ngưỡng để đảm bảo lập kế hoạch khả thi. Ví dụ, để giảm mức sử dụng, phải sử dụng CPU nhanh hơn hoặc tần số CPU cao hơn, tuy nhiên, điều gì sẽ làm tăng mức tiêu thụ điện năng. Tuy nhiên, các hệ thống nhúng, như trong Internet of Things (IoT) hầu hết đều bị hạn chế về sức mạnh. Do đó, một giải pháp hiệu quả hơn để cải thiện việc sử dụng là giảm WCET của các tác vụ.

978-1-5386-4155-2 / 18 / $ 31,00©c 2018 IEEE

Hệ điều hành thời gian thực hiện đại hỗ trợ vô số tính năng để hỗ trợ các nhà phát triển ứng dụng trong việc triển khai ứng dụng của họ và đạt được các chức năng không thể thực hiện được ở lớp ứng dụng. Do đó, hệ điều hành thời gian thực hỗ trợ, ví dụ, quản lý tác vụ, quản lý bộ nhớ, quản lý tệp và Giao tiếp liên quá trình (IPC). IPC đại diện cho việc trao đổi dữ liệu giữa các tác vụ cục bộ (tức là đơn lõi) hoặc toàn cầu (tức là, phân phối hệ thống dưới dạng đa lõi). Một hệ điều hành có thể nhận ra một IPC trong các biến thể khác nhau. Linux [2] hỗ trợ IPC thông qua tệp, ổ cắm, đường ống, bộ nhớ dùng chung và hàng đợi tin nhắn. Thông thường, các phương pháp này yêu cầu các nguyên tắc đồng bộ hóa để tránh các điều kiện chủng tộc (ví dụ: đối với bộ nhớ dùng chung). Do đó, các phương pháp IPC này ảnh hưởng đến hiệu suất của toàn hệ thống [3]. Đặc biệt đối với các kênh nhỏ (ví dụ: [4], [5]),

Bộ Bảo vệ Bộ nhớ (MPU) có thể cho phép khả năng cô lập dữ liệu nhiệm vụ bí mật với nhau. Do đó, một tác vụ đưa dữ liệu quan trọng vào vùng bộ nhớ được bảo vệ và dữ liệu không trọng yếu vào vùng bộ nhớ không được bảo vệ. Tuy nhiên, không phải tất cả các phương pháp IPC đều phù hợp để chuyển dữ liệu quan trọng từ tác vụ này sang tác vụ khác (ví dụ: bộ nhớ dùng chung). Để truyền IPC được bảo vệ, người gửi phải có khả năng gửi dữ liệu bí mật đến vùng bộ nhớ được bảo vệ của tác vụ người nhận. Điều này có thể thực hiện được với cơ chế IPC một hướng và khả năng bảo vệ bộ nhớ phù hợp. Do đó, các IPC trong hàng đợi tin nhắn sẽ là một cơ chế phù hợp để thực hiện chuyển IPC an toàn.

Dữ liệu mới nhận được nhận dạng bằng cách bỏ phiếu hoặc bằng cách nhận một ngắt. Việc thăm dò có thể tiêu tốn rất nhiều chu kỳ CPU chỉ để kiểm tra, điều gì ảnh hưởng xấu đến WCET của một tác vụ. Việc nhận một ngắt có thể giới thiệu nghịch đảo ưu tiên đơn điệu tỷ lệ [6], trong đó nhiệm vụ có mức độ ưu tiên cao được ưu tiên bởi Yêu cầu ngắt (IRQ) được giải quyết cho nhiệm vụ có mức độ ưu tiên thấp hơn. Trong [7], chúng tôi đã khái quát định nghĩa nghịch đảo ưu tiên đơn điệu tỷ lệ thông qua đảo ngược ưu tiên do HĐH gây ra, như một cuộc gọi tổng hợp thực hiện các hoạt động cho nhiệm vụ có mức độ ưu tiên thấp hơn. Chúng tôi đặt tên cho hiện tượng này là Đảo ngược ưu tiên hệ điều hành (OS-PI).

Bài báo này trình bày EventQueue, một IPC dựa trên hàng đợi tin nhắn cho các hệ thống nhúng thời gian thực. EventQueue dựa trên cách tiếp cận sự kiện để tránh vấn đề OS-PI, điều dẫn đến thực tế là các tác vụ có mức độ ưu tiên cao hơn không được ưu tiên trước trong khi nhận dữ liệu mới từ mức độ ưu tiên thấp hơn

các nhiệm vụ. Hơn nữa, EventQueue giảm số lượng các cuộc gọi tổng hợp gửi thực thi trong chế độ hạt nhân, điều này giúp cải thiện WCET của các tác vụ. EventQueue không giới hạn số lượng kênh IPC, thường bị giới hạn trong các giải pháp phần cứng khác. Do đó, nó đã sẵn sàng cho các hệ thống thời gian thực nhúng thích ứng cao trong tương lai. Như một tác dụng phụ, cách tiếp cận được đề xuất cho phép khả năng truyền dữ liệu một cách an toàn cho các tác vụ, được cách ly với nhau, bằng cách sử dụng một MPU.

Phần còn lại của bài báo được sắp xếp như sau: Đầu tiên, Phần II trình bày các cách tiếp cận IPC dựa trên hàng đợi tin nhắn tương tự trong phần cứng. Thứ hai, Phần III hiển thị kiến trúc cơ bản, nơi EventQueue được áp dụng và Phần IV hiển thị chi tiết EventQueue. Trong khi Phần V hiển thị các đánh giá hiệu suất và kết quả tổng hợp trong Mảng cổng lập trình trường (FPGA), Phần VI thảo luận về hiệu suất, bảo mật và các khía cạnh chung của EventQueue. Cuối cùng, Phần VII kết thúc bài báo này.

II. Rcông việc đã đạt được

Hơn 25 năm trước, tin nhắn đầu tiên truyền qua hệ điều hành đã được phát triển, với vô số cuộc gọi IPC. Lúc đầu, các IPC phần mềm thuần túy rất chậm; do đó, sau đó đã bắt đầu một số dự án nghiên cứu với mục đích hỗ trợ các IPC bằng bộ đồng xử lý tin nhắn [3], [8]. Giải pháp phần cứng cho thấy sự gia tăng hiệu suất rất lớn, nhưng nó được thiết kế cho các ứng dụng máy chủ trong đó không gian và tiêu thụ điện năng không liên quan như trong các hệ thống nhúng. Hơn nữa, các ràng buộc thời gian thực vẫn chưa được xem xét, điều gì khiến các IPC này không thể áp dụng cho các hệ thống thời gian thực.

Công việc trong [9] trình bày một IPC hàng đợi thông báo với sự hỗ trợ phần cứng cũng xem xét các ràng buộc thời gian thực bằng cách lưu ý các ưu tiên. Phần cứng hỗ trợ một số kênh hàng đợi tin nhắn giới hạn, có thể được sở hữu bởi một tác vụ. Do đó, số lượng kênh IPC bị giới hạn bởi phần cứng. Điều này không làm cho phần cứng này phù hợp với các hệ thống nhúng phức tạp ngày nay, hầu hết yêu cầu vô số kênh IPC.

Trong [10], các tác giả đã đề xuất một cách tiếp cận của cơ chế IPC có thể dự đoán được cho các hệ thống nhúng. Giải pháp của họ dựa trên cấu trúc bộ nhớ chia sẻ hai cấp. Một mức bộ nhớ cục bộ cho mỗi tác vụ và một mức bộ nhớ chung để kết hợp các mức cục bộ. Dữ liệu được di chuyển giữa hai mức bộ nhớ, theo đó mức bộ nhớ cục bộ, ngược lại với mức bộ nhớ chung, không yêu cầu mức nguyên thủy đồng bộ hóa. Do đó, khi đồng bộ hóa lớp bộ nhớ chung, thông báo đi qua chi phí sẽ tăng lên cùng với số lượng tác vụ. Để khắc phục sự cố này, họ đã điều chỉnh bộ điều khiển Truy cập Bộ nhớ Trực tiếp (DMA) để thực hiện việc truyền dữ liệu giữa hai lớp và giảm độ trễ để có được nguyên thủy đồng bộ hóa của lớp bộ nhớ chung. Tuy nhiên, cách tiếp cận này yêu cầu sao chép dữ liệu từ lớp bộ nhớ cục bộ sang lớp bộ nhớ chung và trở lại lớp cục bộ của tác vụ máy thu. Cách tiếp cận này dẫn đến một IPC có thể dự đoán được, nhưng mỗi lần truyền yêu cầu hai lần chuyển bộ nhớ dẫn đến chi phí đáng kể trên nhiều IPC chuyển nhượng. Hơn nữa, dữ liệu có khả năng bí mật được đưa vào bộ nhớ chung và có thể được đọc bởi các tác vụ không đáng tin cậy.

Hỗ trợ hàng đợi tin nhắn trong phần cứng hiếm khi được tìm thấy trong các kiến trúc máy tính thương mại. ARM [11] chỉ định mô-đun IPC của nó và NXP [12] cung cấp Trình quản lý hàng đợi, cả hai đều được triển khai trong phần cứng. Mô-đun IPC của ARM dựa trên một số hộp thư cụ thể. Trong mỗi hộp thư, một phần tử có thể được gửi đến người nhận và người nhận sẽ được IRQ thông báo. Do đó, IRQ làm gián đoạn dòng chương trình hiện tại, điều có thể dẫn đến OS-PI cần phải tránh đối với các hệ thống thời gian thực. Trình quản lý hàng đợi trên chip NXP là một phần mở rộng rất lớn với mục đích chuyển bất kỳ gói tin nào đến bất kỳ bộ tăng tốc hoặc lõi nào trong Hệ thống trên chip (SoC). Phần mở rộng được xây dựng dựa trên các khung xác định vị trí và kích thước dữ liệu trong bộ nhớ dữ liệu và các bộ mô tả, xử lý việc lưu vào bộ đệm hoặc xếp hàng của các khung. Đây là một tiện ích mở rộng phần cứng mạnh mẽ,

Do sự hỗ trợ phần cứng hiếm hoi trong các kiến trúc máy tính thương mại, cách phổ biến để triển khai IPC trong các hệ thống thời gian thực nhúng là sử dụng các giải pháp phần mềm thuần túy. Đối với bối cảnh ô tô, trong AUTOSAR [13], IPC được thực hiện với các sự kiện. Sự kiện là một nguyên thủy đồng bộ hóa tác vụ mà qua đó các tác vụ có thể trao đổi thông tin giữa nhau. Do đó, chi phí hệ điều hành được tăng lên do nhiều chuyển mạch ngữ cảnh; và hơn nữa, tất cả các tác vụ đều có toàn quyền truy cập vào bộ nhớ dùng chung, nơi thông tin được trao đổi. Miền avionic [14] cung cấp cùng một cách tiếp cận IPC cộng với một cách tiếp cận hàng đợi. Tuy nhiên, để gửi dữ liệu vẫn phải thực hiện cuộc gọi tổng hợp, điều này dẫn đến chuyển đổi ngữ cảnh và có thể dẫn đến OS-PI.

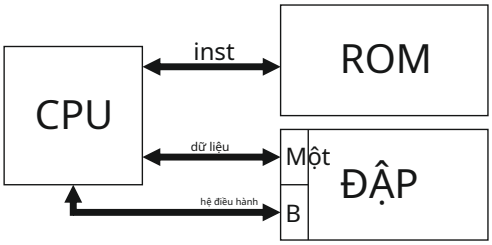
Cách tiếp cận EventQueue được đề xuất không hạn chế số lượng hàng đợi cũng như không dẫn đến OS-PI, điều mà không công trình nào trước đây có thể xử lý trong một giải pháp.

III. MộtKIẾN TRÚC

A. Thuật ngữ và giả định

Đối với EventQueue, chúng tôi giả định một CPU lõi đơn với hệ điều hành đa tác vụ đang chạy một tác vụ $\tau_{chạy} \in T$trong số các nhiệm vụ$TMỗi nhiệm vụ \tau \in T$sở hữu một mức độ ưu tiên tĩnhhP_τ được xác định tại thời điểm biên dịch và tuân theo các quy tắc lập lịch biểu Rate Monotonic (RM) [1] (nghĩa là thời hạn ngắn hơn có mức độ ưu tiên cao hơn). Để đồng bộ hóa các tác vụ với nhau, HĐH hỗ trợ các sự kiện. Một sự kiện$e \in E$là một nguyên thủy đồng bộ hóa một hướng để thông báo một tác vụ τ, đang chờ sự kiện cụ thể đó. Nếu sự kiệneđược đặt, nhiệm vụ được ưu tiên cao nhất trong hàng đợi sự kiện$q_e \subset T$được tiếp tục.

Đơn vị tính toán sở hữu nhận thức về hệ điều hành ở cấp độ phần cứng. Do đó, đơn vị tính toán luôn nhận thức được mức độ ưu tiên của tác vụ hiện đang chạy$P_{chạy}$và thông tin khác của nhiệm vụ, chẳng hạn như kích thước ngăn xếp tối đa. Với kiến thức này, đơn vị tính toán hỗ trợ HĐH đạt được các thuộc tính mà ở cấp độ HĐH không thể đạt được hoặc chỉ với một nỗ lực tính toán lớn. Để kích hoạt nhận thức về hệ điều hành, chúng tôi giả định một kiến trúc hệ thống như được mô tả trong Hình 1. Ở đó, đơn vị tính toán sở hữu một bus hướng dẫn đến Bộ nhớ Chỉ đọc (ROM)



Hình 1. Kiến trúc hệ thống để hỗ trợ cách tiếp cận EventQueue.

để đọc lệnh và một bus dữ liệu tới Bộ nhớ Truy cập Ngẫu nhiên (RAM). Bên cạnh kết nối dữ liệu, một kết nối xa hơn với RAM được thực hiện để hỗ trợ tất cả các chức năng nhận biết hệ điều hành. Để hỗ trợ kiến trúc như vậy, bộ nhớ dữ liệu phải là một cổng kép. Bộ nhớ cổng kép cho phép truy cập bộ nhớ với hai tín hiệu địa chỉ và dữ liệu độc lập và sở hữu logic phân xử để xử lý các ghi đồng thời có thể có vào cùng một địa chỉ bộ nhớ. Do kiến trúc này, nhận thức về hệ điều hành không can thiệp vào hoạt động bình thường của đơn vị tính toán bằng cách hoạt động đồng thời với nó.

B. mosartMCU

Để thực hiện phương pháp EventQueue, chúng tôi sử dụng nhà thờ Hồi giáoMCU[1]dự án triển khai nhận thức về hệ điều hành vào các hệ thống đa lõi nhúng (ví dụ: [7], [15]). Cơ sở cho nhà thờ Hồi giáoMCU là vScale mã nguồn mở[2], là một kiến trúc RISC-V [16], được chỉ định bởi Đại học California, Berkeley. vScale triển khai tất cả32 bitsố nguyên và các lệnh nhân / chia từ đặc tả Kiến trúc Tập lệnh (ISA) [17] và thực thi các lệnh trong một đường ống ba giai đoạn. Đặc tả định nghĩa 32 thanh ghi, trong khi trình biên dịch không sử dụng thanh ghitp. Thanh ghi này cho biết Khối điều khiển tác vụ (TCB) của tác vụ hiện đang chạy $\tau_{chạy}$, chứa thông tin về nhiệm vụ bao gồm mức độ ưu tiên của nó$P_{chạy}$. Chúng tôi đã mở rộng triển khai cơ bản với thao tác đọc tự động, được kích hoạt bởi phần cứng nếu đăng kýtpbị thay đổi. Do đó, phần cứng luôn nhận thức được mức độ ưu tiên của tác vụ hiện đang chạy. Đồng thời với quá trình thực thi bình thường, thao tác đọc được thực hiện tự động bằng cách sử dụng kết nối bổ sung với bộ nhớ dữ liệu thông qua bộ nhớ cổng kép. Bộ nhớ cổng kép này cũng được sử dụng bởi một số phần mở rộng nhận thức về hệ điều hành khác. Hơn nữa, thông số kỹ thuật RISC-V xác định ba chế độ hoạt động khác nhau, trong khinhà thờ Hồi giáoMCU chỉ hỗ trợ những người không có đặc quyềnngười sử dụng-chế độ và đặc quyềnhạt nhân- cách thức. Các chế độ hoạt động này xác định quyền đối với một số hướng dẫn và để truy cập Thanh ghi trạng thái điều khiển (CSR), là các thanh ghi phần cứng được sử dụng để cấu hình và lấy thông tin từ CPU.

bên trongnhà thờ Hồi giáoMCU chạy phần cứng / phần mềm đa tác vụ phủ đầu đầy đủ được thiết kế theo mãnhà thờ Hồi giáoMCU-OS. Các nhà thờ Hồi giáoMCU-OS là một hệ điều hành thời gian thực được nhúng hỗ trợ tất cả các phần mở rộng nhận thức về hệ điều hành củanhà thờ Hồi giáoMCU. Kernel được thiết kế như một microkernel; do đó, IPC là một tính năng cần thiết cần thiết để giao tiếp giữa các tác vụ.

C. EventIRQ

EventIRQ [7] là một phần mở rộng phần cứng của cơ bản nhà thờ Hồi giáoMCU, dựa trên EventQueue. EventIRQ là một cách tiếp cận xử lý IRQ để tránh sự gián đoạn không thể đoán trước của các IRQ. Để đạt được điều đó, tất cả các ngắt được ánh xạ tới các sự kiện hệ điều hành, mà một tác vụ τ đang chờ đợi. Do đó, việc xử lý IRQ được chuyển từ Quy trình Dịch vụ Ngắt (ISR) sang một nhiệm vụ. Trên IRQ, phần mở rộng phần cứng truy cập TCB của tác vụ được kích hoạt bằng cách biết cấu trúc dữ liệu hệ điều hành bên trong. Tất cả các hoạt động này được thực hiện đồng thời với quy trình thực thi bình thường, bằng cách sử dụng kết nối bổ sung với bộ nhớ dữ liệu. Khi kết thúc truy cập TCB, EventIRQ nhận thức được mức độ ưu tiên của nhiệm vụ hiện đang chạy$P_{chạy}$và mức độ ưu tiên của nhiệm vụ được kích hoạt. Do đó, tác vụ hiện đang chạy chỉ bị gián đoạn vì mức độ ưu tiên của tác vụ được kích hoạt cao hơn tác vụ hiện đang chạy. Nếu không, tác vụ sẽ được thêm vào một danh sách, sẽ được hệ điều hành bắt kịp sau này. Với việc hoãn xử lý IRQ này, thời gian phản hồi của IRQ có thể tăng lên; tuy nhiên, tránh được sự gián đoạn không thể đoán trước của một nhiệm vụ có mức độ ưu tiên cao và không có OS-PI nào xảy ra hoặc ít nhất là bị giới hạn kịp thời.

Đối với tất cả các IRQ, ngoại trừ bộ đếm thời gian hệ thống, các tác vụ chờ sự kiện được sắp xếp theo mức độ ưu tiên. Do đó, trên mộtthiết lập sự kiện, nhiệm vụ được ưu tiên cao nhất tiêu thụ sự kiện. Tuy nhiên, đối với bộ đếm thời gian hệ thống, tất cả các tác vụ được sắp xếp theo thời gian chờ của nó. Để xử lý hàng đợi thời gian chờ đúng cách và tránh sự cố OS-PI, EventIRQ bổ sung đặt bộ hẹn giờ hệ thống với thời gian chờ mới của tác vụ chờ tiếp theo trong hàng đợi.

Để tránh OS-PI do thiết lập sự kiện phần mềm gây ra, sự kiện này được chuyển hướng đến nhiệm vụ có mức độ ưu tiên thấp hơn, EventIRQ mở rộng ISA cơ sở với sự kiện đã đặtsevhướng dẫn. Hướng dẫn này thực hiện các hoạt động tương tự như quy trình đã đề cập cho IRQ, nhưng thay vì hoạt động trên một sự kiện IRQ, nó hoạt động trên sự kiện phần mềm.

IV. EventQueue

EventQueue cho phép giao tiếp giữa các tác vụ, dựa trên cơ chế hàng đợi tin nhắn và cách tiếp cận EventIRQ đã đề cập. Để gửi dữ liệu cho một tác vụ, một hàng đợi $\rho \in Q$, của tập hợp hàng đợiQ,bắt buộc. Hàng đợi ρ được biểu diễn dưới dạng một bộ

$$\rho := (e_\rho, \text{ệ}_\rho, s_r, \text{đọc}_\rho, \text{viết}_\rho, r_\rho, b_\rho). \quad (1)$$

Dữ liệu trong hàng đợi ρ được lưu trữ trong bộ đệmb_ρ. Kích thước của bộ đệmb_ρđược xác định với kích thước bộ đệmS_ρ. Các chỉ số đọc$_\rho$và viết$_\rho$được sử dụng để đọc và ghi nội dung trong bộ đệmb_ρ, tương ứng. Chiều dài bộ đệml_ρđại diện cho số phần tử hợp lệ được lưu trữ trong bộ đệmb_ρ. Bộ đệm

[1]MCU thời gian thực nhận biết hệ điều hành đa lõi
[2]https://github.com/ucb-bar/vscale

chiều dàil$_\rho$được tính toán với các chỉ sốviết$_\rho$vàđọc$_\rho$như sau:

$$l_\rho := (\text{viết}_\rho - \text{đọc}_\rho) \bmod S_\rho \quad (2)$$

Chiều dài bộ đệml_ρsẽ là0cho một bộ đệm trống hoặcSnếu bộ đệmb_ρsẽ hỗ trợ để lưu trữ$S_\rho$các yếu tố. Để phân biệt giữa bộ đệm trống và bộ đệm đầy, bộ đệm$b_\rho$có thể lưu trữ lên đếnS_ρ-1các yếu tố. Do đó, đối với một bộ đệm trống điều kiện

$$\text{trống rỗng}_\rho := \begin{cases} \text{thật} & \text{nếu đọc}_\rho = \text{viết}_\rho \\ \text{sai} & \text{khác} \end{cases} \quad (3)$$

và để có bộ đệm đầy đủ thì điều kiện

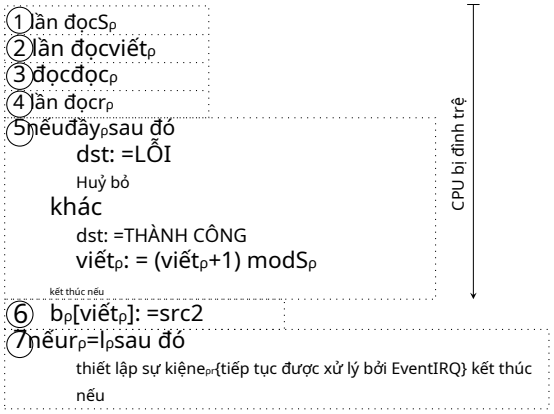
$$\text{đầy}_\rho := \begin{cases} \text{thật} & \text{nếu đọc}_\rho = (\text{viết}_\rho + 1) \bmod S_\rho \\ \text{sai} & \text{khác} \end{cases} \quad (4)$$

phản ánh trạng thái của chúng. Để nhận dữ liệu qua hàng đợi ρ, kích thước được yêu cầur_ρ: = [1,S_ρ-1]được định nghĩa; do đó, người nhận không được thông báo bởi sự kiện$e_{\rho r}$miễn là chiều dài bộ đệml_ρ không đạt đến số lượng của kích thước yêu cầur_ρ. Sự kiện $e_\rho$thông báo cho người gửi về tình trạng bộ đệm không còn đầy nữa. Cả người gửi và người nhận đều bị tạm ngưng miễn là bộ đệm đầy hoặc kích thước được yêu cầu$r_\rho$không đạt được. Do đó, việc kích hoạt các sự kiện sẽ tiếp tục các nhiệm vụ bị treo nếu các điều kiện trở thành sự thật. Các phần tiếp theo sẽ trình bày việc nhận ra EventQueue trongnhà thờ Hồi giáoMCU và trongnhà thờ Hồi giáo MCU-OS.

A. Hỗ trợ phần cứng

EventQueue mở rộngnhà thờ Hồi giáoMCU với một hướng dẫn bổ sungqwr dst, src1, src2,mà cũng phải được hỗ trợ bởi trình biên dịch. Lệnh kích hoạt hoạt động, chuyển dữ liệu đến bộ đệm của hàng đợi, trong phần cứng. Vì vậy, lệnh này sử dụng hai thanh ghi nguồn để tham chiếu đến hàng đợi ρ và để chuyển một nội dung dữ liệu. Hàng đợi được tham chiếu được mô tả trong Khối điều khiển hàng đợi (QCB), là một cấu trúc dữ liệu được lưu trữ trong RAM, với tất cả thông tin của hàng đợi ρ. Thanh ghi đích lưu trữ giá trị trả về của lệnh cung cấp thông tin về sự thành công của quá trình chuyển giao. Sau khi gọiqwr hướng dẫn các bước sau được thực hiện (cũng được mô tả trong Hình 2) với sự hỗ trợ của nhận thức về hệ điều hành trongnhà thờ Hồi giáoMCU:

① Kích cỡS_ρcủa hàng đợi ρ được đọc từ QCB. Đọc ghi chỉ
② mụcviết$_\rho$, trong hàng đợi ρ, trong QCB. Đọc chỉ mục đã
③ đọcđọc$_\rho$, trong hàng đợi ρ, trong QCB. Đọc kích thước
④ được yêu cầur_ρ, trong hàng đợi ρ, trong QCB. Chỉ sốviết$_\rho$
⑤ được tăng dần và được lưu trữ trong QCB. Thanh ghi đích được điền với một giá trị cho biết thành công và EventQueue tiếp tục với bước tiếp theo nếu điều kiện đệm đầy$_\rho$là không đúng sự thật. Nếu không, chỉ số ghiviết$_\rho$ không được cập nhật, thanh ghi đích chứa mã lỗi và lệnh đã kết thúc. Để tránh các điều kiện chạy đua, đơn vị tính toán không được phép tiếp tục, trong vòng năm hoạt động của bộ nhớ. Do đó, đường ống ngăn chặnnhà thờ Hồi giáoMCU cho4các chu kỳ.



Hình 2. Mã giả được kích hoạt bởi mộtqwr dst, src1, src2hướng dẫn.

⑥ Nếu hàng đợi ρ không đầy, nội dung trong nguồn thứ hai đăng ký sẽ được lưu trữ vào bộ đệmb_ρtại chỉ mụcviết$_\rho$.
⑦ Nếu chiều dài bộ đệml_ρđạt đến kích thước yêu cầur_ρ, EventQueue kích hoạt sự kiện người nhận$e_\rho$nó nằm trong QCB của hàng đợi ρ. Sau đó,thiết lập sự kiện cách tiếp cận của EventIRQ được thực thi.

Tất cả các hoạt động này, bao gồm EventIRQ, được thực hiện trên kết nối hệ điều hành với RAM. Do đó, sau khi dừng đường ống trong bốn lệnh (nghĩa là, lệnh này tiêu thụ đúng 4 chu kỳ), các bước xử lý hàng đợi và xử lý sự kiện còn lại được thực hiện đồng thời với quy trình thực thi thông thường. Lệnh này thay thế lệnh gọi hệ điều hành (tức là lệnh gọi của một chức năng hệ điều hành, chạy ở chế độ hạt nhân), do đó nó tránh được các chuyển mạch ngữ cảnh và cải thiện thời gian thực thi. Hơn nữa, EventQueue đảm bảo, do EventIRQ, tác vụ hiện đang chạy chỉ bị gián đoạn vì mức độ ưu tiên của tác vụ đang chờ vượt quá mức độ ưu tiên của nó. Do đó, cách tiếp cận EventQueue cũng tránh được các vấn đề OS-PI khi chuyển dữ liệu từ tác vụ này sang tác vụ khác.

B. Trách nhiệm phần mềm

Hệ điều hành chịu trách nhiệm triển khai hàng đợi thông báo được xử lý trong phần mềm và phải xem xét việc sử dụng hướng dẫn đã giới thiệuqwr,đã đề cập trước đó. Hướng dẫn qwrcho phép gửi một từ cho mỗi lệnh gọi mà không có thời gian chờ. Để thuận tiện hơn cho việc lập trình, cần có một hàm gọi lệnh và cho phép gửi một số từ cụ thể với thời gian chờ cao hơn. Do đó, chức năng đó không chạy ở chế độ hạt nhân, mà ở chế độ người dùng. Do đó, nó tránh được cuộc gọi tổng hợp, do đó làm giảm số lượng chuyển mạch ngữ cảnh. Nếu hàng đợi ρ đã đầy (tức là,đầy$_\rho$=thật),cuộc gọi tổng hợpwait_event_until ($e_{\rho s}$, t)được gọi, dẫn đến nhiệm vụ gửi để chờ sự kiện$e_{\rho s}$trong thời gian tối đa làtDo đó, nhiệm vụ bị tạm dừng cho đến khi người nhận ném sự kiện$e_\rho$do người nhận đọc một số dữ liệu trong bộ đệm$b_\rho$hoặc hết thời gian chờ đợi.

Bộ thu được thực hiện hoàn toàn trong phần mềm như một cuộc gọi tổng hợp và được thực thi ở chế độ hạt nhân. Cuộc gọi tổng hợp kiểm tra xem số lượng dữ liệu hợp lệ có ít nhất là kích thước được yêu cầu hay khôngr $_\rho$, nếu không thì nó gọi là cuộc gọi tổng hợpwait_event_until ($e_{\rho r}$, t)

và đặt kích thước được yêu cầu$r_\rho$trong QCB. Ở đó, tác vụ bị tạm dừng cho đến khi EventQueue đặt sự kiện người nhận$e_{\rho r}$ nếu độ dài hàng đợil_ρđạt đến kích thước yêu cầur_ρ, hoặc cho đến khi hết thời gian chờ.

Điều đáng chú ý là mã người gửi không truy cập QCB của hàng đợi ở chế độ người dùng. Tất cả các truy cập bộ nhớ cho QCB được thực hiện trong phần cứng hoặc trong syscall, được thực thi ở chế độ hạt nhân. Do đó, để bảo vệ QCB trong số các tác vụ, tác vụ nhận phải đặt QCB của hàng đợi trong vùng được bảo vệ MPU. Với cách tiếp cận đó, các nhiệm vụ được cách ly với nhau và có thể giao tiếp an toàn qua IPC với hiệu suất cao và tránh được OS-PI.

ĐÃĐỊNH GIÁ

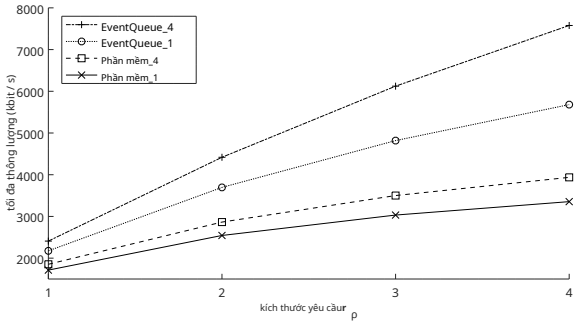
Chúng tôi đã triển khai EventQueue vào nền tảng nghiên cứu của mìnhnhà thờ Hồi giáoMCU trong đónhà thờ Hồi giáoMCU-OS chạy. Các nhà thờ Hồi giáoMCU được triển khai thành Xilinx Artix-7 FPGA, được lắp ráp trên bo mạch Nexys 4 DDR của Digilent. Chúng tôi đã so sánh việc triển khai hàng đợi phần mềm thuần túy với cách tiếp cận EventQueue về thông lượng tối đa và thời gian thực thi để gửi dữ liệu. Hơn nữa, chúng tôi đã điều tra việc sử dụng tài nguyên trong FPGA và mức tiêu thụ bộ nhớ cho hệ điều hành.

A. Đánh giá hiệu suất

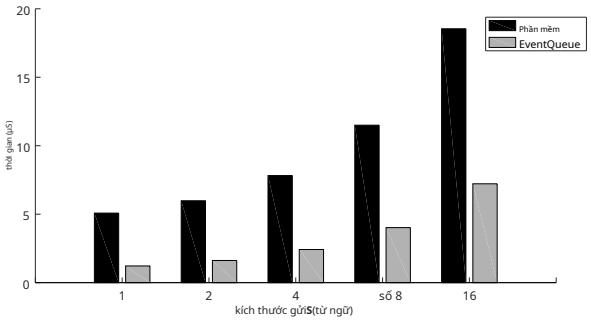
Đánh giá hiệu suất đầu tiên điều tra thông lượng tối đa trongnhà thờ Hồi giáoMCU. Ba nhiệm vụT: = {τ_S, τ_r, τ_m} được khởi tạo; theo đó, nhiệm vụ τ_Sgửi dữ liệu cho nhiệm vụ τ_r qua hàng đợi ρ. Kích thước của hàng đợi ρ làS_ρ=5,có nghĩa là bộ đệmb_ρcửa hàng lên đến4các yếu tố. Nhiệm vụ τ_mđặt đếm ngược và chờ đợi nó. Nếu hết hạn, nhiệm vụ $\tau_m$in ra số bit được truyền.

Chúng tôi đã điều tra đánh giá với các cấu hình khác nhau: Đối với người nhận, chúng tôi đặt kích thước yêu cầur_ρđến1, 2, 3,và 4.Người gửi gọi hàmsend_queue_until (ρ, d, s, t),gửi các từ truyền đidkích thước gửiSvào hàng đợi ρ. Nếu bộ đệm đầy, hàm sẽ đợi cho đến khi bộ đệm không đầy nữa hoặc hết thời gian chờthết hạn. Chúng tôi đã đánh giá hiệu suất với kích thước gửiScủa1 và4.Tất cả các tổ hợp đều được thử nghiệm với phương pháp IPC và EventQueue phần mềm thuần túy và kết quả được mô tả trong Hình 3. Với kích thước yêu cầu ngày càng tăng$r_\rho$và kích thước gửiS thông lượng tăng cho cả hai cách tiếp cận. EventQueue sở hữu thông lượng tối đa có thể đạt được gần như gấp đôi so với giải pháp phần mềm thuần túy. Điều này là do các cuộc gọi tổng hợp giảm và thời gian thực thi ngắn hơn để gửi một từ qua hàng đợi ρ.

Đánh giá hiệu suất thứ hai điều tra thời gian thực hiện củasend_queue_untilhoạt động trong phần mềm thuần túy và trong giải pháp EventQueue. Hình 4 mô tả thời gian thực hiện củasend_queue_untilchức năng gọi với các kích thước gửi khác nhauStruyền từ. Đối với EventQueue, hàm yêu cầu 0,4µScho mỗi từ bổ sung; cho giải pháp phần mềm thuần túy0,9µS.EventQueue không gọi một cuộc gọi tổng hợp; do đó nó tránh được việc di chuyển vào kernelmode. Do đó, EventQueue cải thiện việc chuyển IPC tổng thể



Hình 3. So sánh thông lượng tối đa của việc triển khai phần mềm thuần túy và EventQueue với các cấu hình khác nhau.



Hình 4. Thời gian thực hiện củasend_queue_until (ρ,d, s, t)với số lượng kích thước gửi khác nhauS.

màn biểu diễn; hơn nữa, nó tránh OS-PI trong chuyển IPC. Đặc biệt là các kênh vi mô, vốn đang sử dụng nhiều IPC làm khái niệm cơ sở của chúng, sẽ được hưởng lợi từ EventQueue.

B. Tổng hợp và kết quả hệ điều hành

Phần thứ hai của đánh giá nghiên cứu kết quả tổng hợp của FPGA và mức tiêu thụ bộ nhớ hệ điều hành. Qua đó, chúng tôi so sánh bản gốcnhà thờ Hồi giáoMCU, EventIRQ mở rộng và phiên bản mở rộng EventQueue. Bảng I liệt kê các kết quả tổng hợp, được báo cáo bởi Xilinx Vivado 2017.3.

Các lát cắt của Bảng Tra cứu (LUT) và Flip-Flop (FF) tăng lên với phần mở rộng của EventIRQ và một lần nữa với phần mở rộng của EventQueue. Việc triển khai ban đầu không chứa nhận thức về hệ điều hành; và do đó, nó không yêu cầu kết nối bổ sung với RAM. Do đó, cách tiếp cận EventIRQ và EventQueue yêu cầu các phần bổ sung. EventQueue yêu cầu các lát bổ sung để xử lý lệnhqwrvà

BẢNG I
SKẾT QUẢ YÊU CẦU CỦA BẢN GỐC VÀ PHẦN MỞ RỘNGnhà thờ Hồi giáoMCU PHIÊN BẢN.

	nhà thờ Hồi giáoMCU		
	Nguyên bản	EventIRQ	EventQueue
LUT lát	2799	3543	3982
FF lát	2078	2413	2632
tối đa tần số	73,992MHz	72.124MHz	73,373MHz
Động lực	16mW	18mW	19mW

BẢNG II
MSO SÁNH EMORY CỦA BẢN GỐC VÀ PHẦN MỞ RỘNG
nhà thờ Hồi giáoMCU-OSPHIÊN BẢN.

	nhà thờ Hồi giáoMCU-OS		
	Nguyên bản	EventIRQ	EventQueue
ROM	3952B	4316B	4216B
ĐẬP	12B	136B	136B

các bước để truy cập dữ liệu của QCB qua kết nối RAM được sử dụng bởinhà thờ Hồi giáoNhận thức về hệ điều hành của MCU. Tần số tối đa có thể đạt được hầu như không đổi cho cả ba cách triển khai. Công suất tiêu thụ phụ thuộc vào các lát yêu cầu; do đó, mức tiêu thụ điện năng của EventQueue là cao nhất.

Chúng tôi cũng đã điều tra mức tiêu thụ bộ nhớ tĩnh của hệ điều hành, được báo cáo bởigcctrình biên dịch với mức tối ưu hóa đầu tiên (tức là -O1)đã được kích hoạt. Bảng II liệt kê việc sử dụng bộ nhớ của phiên bản gốc và các phiên bản mở rộng. Phiên bản gốc tiêu thụ ít ROM và RAM hơn, vì EventIRQ yêu cầu mã bổ sung để bắt kịp các tác vụ được kích hoạt và vì bảng vectơ ngắt được chuyển từ ROM vào RAM. Phương pháp EventQueue thay thế việc triển khai phần mềm để gửi một phần tử bằng cách sử dụngqwr hướng dẫn; do đó, yêu cầu ROM được giảm xuống, so với EventIRQ.

VI. DBÀN LUẬN

Sự phức tạp ngày càng tăng của các hệ thống nhúng thời gian thực ngày nay đòi hỏi nhiều tính năng bảo mật hơn nữa để ngăn chặn việc truy cập vào thông tin bí mật được lưu trữ trong một tác vụ. Do đó, việc bảo vệ bộ nhớ sẽ tìm thấy đường vào các hệ thống nhúng thời gian thực ngày nay. Tuy nhiên, việc thực hiện IPC sau đó bị hạn chế. Bộ nhớ dùng chung là một giải pháp khả thi, nhưng nó đòi hỏi các nguyên tắc đồng bộ hóa bổ sung và dẫn đến các vấn đề khác, chẳng hạn như vấn đề đảo ngược ưu tiên để quản lý tài nguyên [18]. EventQueue cho phép gửi dữ liệu giữa các tác vụ được cách ly với nhau, vì chỉ người nhận, phần cứng và HĐH mới có quyền truy cập vào QCB của hàng đợi. Người gửi chỉ sử dụng các địa chỉ của QCB để kích hoạt phần mở rộng phần cứng để chuyển tiếp dữ liệu của nó. Do đó, người gửi không thể đọc dữ liệu trong bộ đệmb_ρ; và do đó, EventQueue thích hợp để chuyển tiếp thông tin bí mật.

Bên cạnh khả năng cô lập các tác vụ với nhau, EventQueue cải thiện thông lượng để gửi thông tin từ tác vụ này sang tác vụ khác. Hiệu suất của IPC là một đặc tính quan trọng, đặc biệt là đối với các kênh nhỏ, vốn đang sử dụng nhiều IPC. Các thiết kế nhân của hệ thống nhúng chủ yếu dựa trên các kênh vi mô; do đó, cách tiếp cận EventQueue rất phù hợp cho các hệ thống nhúng thời gian thực ngày nay và tương lai, như trong lĩnh vực ô tô hoặc IoT.

EventQueue cũng có thể áp dụng trong kiến trúc máy tính khác, để bảo vệ dữ liệu của QCB và để cải thiện hiệu suất. Tuy nhiên, chỉ bằng cách sử dụng EventQueue cùng với EventIRQ, bạn sẽ đạt được toàn bộ khả năng tránh được các vấn đề OS-PI, điều chỉ có thể thực hiện được với phương pháp OSawareness trongnhà thờ Hồi giáoMCU. Hơn nữa, như đã đề cập trong Phần II, các giải pháp IPC nghiên cứu và thương mại được thực hiện trong phần cứng, giới hạn số lượng kênh IPC. Theo đó, EventQueue không giới hạn số lượng kênh IPC, số lượng nhiệm vụ cũng như số lượng sự kiện. Điều này có thể thực hiện được thông qua việc truy cập bộ nhớ trực tiếp vào RAM bằng kết nối hệ điều hành bổ sung. bên trongnhà thờ Hồi giáoMCU, các chức năng của OSawareness đang truy cập trực tiếp vào cấu trúc dữ liệu hệ điều hành trong RAM và không có thanh ghi phần cứng nào được sử dụng cho các kênh IPC, sự kiện hoặc tác vụ như trong các hoạt động trước đây đã đề cập.

VII. CKẾT LUẬN VÀOUTLOOK

Trong bài báo này, chúng tôi đã trình bày EventQueue cho IPC giữa các tác vụ trong một lõi. EventQueue dựa trên EventIRQ, tránh sự cố OS-PI thông qua nhận thức về OS trong nhà thờ Hồi giáoMCU. Chúng tôi đã mở rộng EventIRQ với một lệnh bổ sung thực hiện việc chèn một phần tử vào hàng đợi thay vì một giải pháp phần mềm thuần túy. Bên cạnh khả năng cô lập dữ liệu của các tác vụ với nhau, EventQueue thừa nhận một giao tiếp giữa các tác vụ. Chúng tôi đã triển khai cách tiếp cận vàonhà thờ Hồi giáoMCU và điều tra hiệu suất, kết quả tổng hợp và các yêu cầu của hệ điều hành. Thông lượng của EventQueue để chuyển dữ liệu từ tác vụ này sang tác vụ khác gần như tăng gấp đôi so với giải pháp phần mềm thuần túy. Kết quả tổng hợp của EventQueue cho FPGA và mức tiêu thụ bộ nhớ trong MCU hầu như không thay đổi, so với EventIRQ, ngoại trừ các lát LUT và FF được yêu cầu bởi logic phần cứng bổ sung.

Ở cấp độ phần cứng, bước tiếp theo là mở rộng khái niệm Nhận thức về Hệ điều hành của chúng tôi cho các hệ thống đa lõi. Do đó, chúng tôi cần hỗ trợ EventIRQ và EventQueue cho các hệ thống nhúng đa lõi, để cho phép IPC xếp hàng không chỉ giữa các tác vụ trong cùng một lõi mà còn giữa các tác vụ trên các lõi khác nhau, trong khi vẫn tránh hoặc ít nhất là hạn chế vấn đề OS-PI .

MộtTHÔNG BÁO

Dự án nghiên cứu này được tài trợ một phần bởi AVL List GmbH và Bộ Khoa học, Nghiên cứu và Kinh tế Liên bang Áo (bmwfw).

RTIỀN TỆ

[1] CL Liu và JW Layland, “Các thuật toán lập lịch trình cho đa chương trình trong môi trường thời gian thực cứng”Tạp chí ACM (JACM), vol. 20, không. 1, trang 46–61, tháng 1 năm 1973.

[2] “Quỹ Linux,” https://www.linuxfoundation.org/. Lần truy cập cuối: 01.12.2017.

[3] U. Ramachandran, M. Solomon và M. Vernon, “Hỗ trợ phần cứng cho giao tiếp giữa các quy trình”, trongProc. của kỳ Int. hàng năm lần thứ 14 Hội nghị chuyên đề về Kiến trúc Máy tính (ISCA),ser. ISCA '87. ACM, 1987, trang 178–188.

[4] “Hệ điều hành QNX,” http://blackberry.qnx.com/en/products/ neutrino-rtos / index. Lần truy cập cuối: 01.12.2017.

[5] “ThreadX”, https://rtos.com/solutions/threadx/.Lastaccess:01.12.2017.

[6] LE Leyva-del Foyo, P. Mejia-Alvarez và D. de Niz, “Quản lý ngắt và tác vụ tích hợp cho các hệ thống thời gian thực”ACM Trans. trên Hệ thống máy tính nhúng,vol. 11, không. 2, trang 32: 1–32: 31, tháng 7 năm 2012.

[7] F. Mauroner và M. Baunach, “EventIRQ: Xử lý IRQ dựa trên sự kiện và có mức độ ưu tiên cho Môi trường đa tác vụ,” trongProc. của Hội nghị Euromicro lần thứ 20 về Thiết kế Hệ thống Kỹ thuật số (DSD),Tháng 8 năm 2017, trang 102–110.

[8] J.-M. Hsu và P. Banerjee, “Bộ đồng xử lý truyền thông báo cho máy tính đa máy tính bộ nhớ phân tán,” trongProc. của Hội nghị ACM / IEEE về Siêu máy tính (SC) năm 1990,ser. Siêu máy tính '90. IEEE Computer Society Press, 1990, trang 720–729.

[9] J. Furunäs, J. Adomat, L. Lindh, J. Starner và P. Voros, “Một nguyên mẫu cho hỗ trợ giao tiếp giữa các quy trình, trong phần cứng,” trongProc. của Hội thảo Euromicro lần thứ 9 về Hệ thống thời gian thực.IEEE, tháng 6 năm 1997, trang 18–24.

[10] S. Srinivasan và DB Stewart, “Giao tiếp liên xử lý thời gian thực có hỗ trợ phần cứng tốc độ cao cho vi điều khiển nhúng,” trong Proc. của Hội nghị IEEE lần thứ 21 về Hội nghị chuyên đề về hệ thống thời gian thực (RTSS),ser. RTSS'10. IEEE Computer Society, 2000, trang 269–279.

[11]Mô-đun truyền thông liên bộ xử lý PrimeCell (PL320),ARM Limited, 2004.

[12] K. Johnson, “Nền tảng QorIQ: Ưu điểm của Kiến trúc,” Bài trình bày, tháng 10 năm 2013.

[13] “AUTOSAR,” https://www.autosar.org. Lần truy cập cuối: 20.11.2017.

[14] “ARINC-653,” https://www.arinc.com. Lần truy cập cuối: 20.06.2017.

[15] F. Mauroner và M. Baunach, “StackMMU: Chia sẻ ngăn xếp động cho các hệ thống nhúng,” trongProc. của IEEE Int. lần thứ 22 Hội nghị về các công nghệ mới nổi và tự động hóa nhà máy (ETFA).IEEE, tháng 9 năm 2017, trang 1-9.

[16] RISC-V Foundation, “RISC-V,” https://riscv.org/. Lần truy cập cuối: 01.12.2017.

[17] A. Waterman, Y. Lee, R. Avizienis, D. Patterson và K. Asanovic,Hướng dẫn sử dụng Bộ hướng dẫn RISC-V,2016.

[18] L. Sha, R. Rajkumar và JP Lehoczky, “Giao thức kế thừa ưu tiên: Phương pháp tiếp cận đồng bộ hóa thời gian thực”IEEE Trans. trên Máy tính,vol. 39, không. 9, trang 1175–1185, tháng 9 năm 1990.