arXiv:2303.00532v1 [cs.RO] 1 Mar 2023

fpgaDDS：面向 ROS 2 机器人应用的 FPGA 内数据分发服务

Christian Lienen[1], Sorel Horst Middeke[1],和 Marco Platzner[1]

摘要——**现代机器人应用的计算平台由一组异构组件构成，例如多核 CPU、嵌入式 GPU 和现场可编程门阵列。现场可编程门阵列是可重新编程的硬件设备，能够对机器人领域中的许多相关任务执行快速且高能效的计算。ROS 是机器人领域事实上的编程标准，可将应用分解为一组相互通信的节点。ReconROS 是一种先前提出的方法，可以将完整的 ROS 节点映射到硬件中以实现加速。由于 ReconROS 依赖标准 ROS 通信层，在硬件映射的节点之间交换数据可能会成为性能瓶颈。**

本文介绍了 fpgaDDS，这是一种面向硬件映射的 ROS 2 节点的精简型数据分发服务。fpgaDDS 采用定制且静态生成的、基于流的通信架构。我们详细介绍了该通信架构及其组件，并概述其优势。我们在一个测试示例和一个更大规模的自动驾驶车辆案例研究中评估了 fpgaDDS。与软件中的 ROS 2 应用相比，我们实现了最高 13.34 倍的加速比，并将抖动降低了两个数量级。

I. 引言

近年来，机器人应用的计算需求不断增加，这是因为现代机器人会收集越来越多的环境感知信息，并采用复杂的算法进行决策。先进机器学习技术在机器人领域的快速应用进一步加速了这一趋势。

现代机器人硬件架构由异构的计算单元组成，以能效优先的方式提供所需的性能。这些计算单元包括多核 CPU、嵌入式通用 GPU 和现场可编程门阵列。现场可编程门阵列是可重新编程的硬件设备，具有很强的灵活性，因为它们能够为各种工作负载提供高效加速。

机器人操作系统（ROS）是机器人应用领域事实上的标准。遵循 ROS 的设计理念，机器人应用被分解为多个节点，每个节点实现整个应用的子功能。这种功能分解有助于简化开发和维护，并提高代码的可复用性。ROS 节点可以利用不同类型的通信来进行数据交换，例如多对多发布-订阅通信：一个或多个节点在特定主题上发布数据，订阅这些主题的其他节点则接收数据。

[1]Christian Lienen、Sorel Horst Middeke 和 Marco Platzner 就职于德国帕德博恩 33098 的帕德博恩大学计算机科学系christian.lienen@upb.de, sorel.middeke@gmail.com, platzner@upb.de

近期，有人提出将基于现场可编程门阵列的硬件加速器集成到基于 ROS 的应用中。提出的方法要么将硬件加速器内核集成到 ROS 节点中，并以类似远程过程调用的方式触发内核执行 [1], [2] ，要么将完整节点移入硬件域 [3]。

后者具有相当大的灵活性，并使用标准 ROS 通信层进行数据传输。然而，这可能会产生显著的开销，因为所有数据都要经过共享主存。即使是硬件中直接通信的 ROS 节点，数据也必须在 ROS 通信层软件的控制下经由主存传输。

本文的创新贡献在于提出了 fpgaDDS，这是一种面向映射到硬件上的 ROS 节点的精简数据分发服务。fpgaDDS 基于定制的流式通信架构，并集成到 ReconROS 架构和编程模型中 [3]。所提出的方法在执行 ROS 应用时显著提升了性能并降低了抖动，同时保持了 ROS 的编程理念和功能分解模型。

本文其余部分的结构如下：第二节介绍背景知识，尤其是 ROS 以及面向 ROS 的硬件加速方法。第三节详细阐述新型 FPGA 内数据分发服务 fpgaDDS。第四节基于测量结果报告 fpgaDDS 的实验评估。第五节总结全文，并展望未来工作。

II. 背景

本节首先介绍机器人操作系统，然后更详细地概述基于 ROS 2 的应用中的硬件加速以及 ReconROS。

A. 机器人操作系统

机器人操作系统（ROS）由一组软件库、工具以及用于设计机器人应用的方法论构成。遵循 ROS 的设计方法论，整个应用被分解为所谓的 ROS节点。ROS 节点使用不同的通信类型进行交互和数据交换。基于主题的发布-订阅机制支持 ROS 中的多对多通信。一对一通信则采用由 ROS 服务和动作所利用的远程过程调用方案。ROS 应用通常通过其计算图进行描述，

该计算图由 ROS 节点及其作为边的通信构成。

ROS 2 使用消息基础设施来实现节点之间一致的信息交换。消息通过一种简单的定义语言以层次结构进行指定，其范围从由整数或浮点数等内置数据类型组成的简单消息，到复杂的多级消息结构。默认情况下，ROS 2 的标准安装提供一组预定义消息，并支持定义自定义用户指定消息类型。

ROS 2 利用遵循数据分发服务（DDS）标准的标准通信中间件实现。由于不同 DDS 实现之间存在细微差异，ROS 中间件层（rmw）为每种 DDS 实现提供单独的适配器。此外，中间件层还为 ROS 客户端库提供通信接口，后者汇总了通用的客户端功能 [4]。图 1 展示了 ROS 2 的整体通信协议栈。

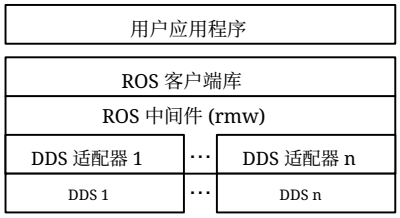


图 1. ROS 2 通信协议栈

用于网络内通信和平台内通信的中间件均已存在。网络内通信中间件的典型示例是由 eprosima 开发的 FastDDS [5]。Iceoryx [6]是平台内通信中间件的一个示例，它使用共享内存进行数据交换。此外，还有根据目标位置优化通信的混合形式。例如，CycloneDDS [7] 结合了网络内通信和平台内通信，并在内部采用 Iceoryx。

B. ROS 应用的硬件加速

近年来，已有针对基于 ROS 的应用的硬件加速器。大多数研究聚焦于 ROS 节点，识别运行时开销较大的部分，并将其移至硬件加速器。在运行时，剩余的 ROS 节点会像进行远程过程调用一样调用硬件加速器，并可能在加速器执行期间阻塞。 [1], [2], 中介绍了这种方法的示例，或该方法已应用于工业产品，例如 Xilinx KRIA [8]。

也有研究致力于加速通信。在 [9],中，作者通过在硬件中分离处理的控制与数据通信流量，减少了通信开销。后续研究 [10] 提出使用高级硬件综合工具流程解析 ROS 消息，从而缩短开发时间并生成更紧凑的代码。

在 [11], 中，作者提出了两种减少节点间的通信开销：内核融合和专用流队列。内核融合将不同 ROS 节点的加速内核合并为一个组合式加速内核。专用流队列通过数据传输接口连接加速内核。这两种方法都打破了 ROS 的节点分解方法，即原始 ROS 计算图会被转换。

FPGA-ROS [12]支持在硬件中实现 ROS 节点及其通信接口。流式网络将应用专用硬件核心连接到共享的中央以太网网关，从而在硬件中实现 TCP/IP 通信。中央管理器实例负责监督整个应用程序。

C. ReconROS

ReconROS [3] 将可重构硬件操作系统 ReconOS [13] 与 ROS 2 相结合，为 ROS2 应用提供硬件加速框架。

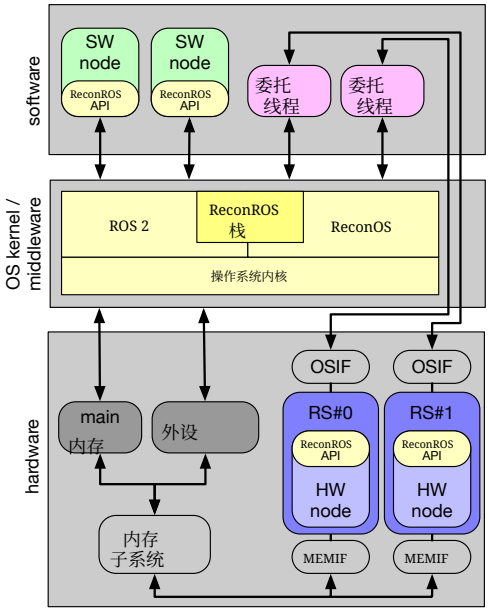


图 2. ReconROS 架构（引自 [14]）

图 2 展示了部署在平台 FPGA 上的 ReconROS 架构，该平台由多核 CPU 和可重构逻辑结构组成。以硬件实现的 ROS 2 节点（硬件节点）被映射到可重构槽位（RS），这些槽位是可重构逻辑结构上的矩形区域。硬件节点通过内存接口（MEMIF）访问共享虚拟内存空间，包括内存映射外设。此外，每个硬件节点都依赖操作系统接口（OSIF）连接到其代理线程；代理线程是一种轻量级软件线程，代表硬件节点执行操作系统调用。

ReconROS 通过 ReconOS 和 ROS 2 扩展客户操作系统（例如 Linux），并借助 ReconROS 栈和 ReconROS API，支持在硬件和软件中执行 ROS 2 节点。ReconROS 栈通过添加与 ROS 相关的对象（例如 ROS 发布者或订阅者）扩展 ReconOS。ReconROS API 适用于软件和硬件节点实现。它确保采用一致的编程模型，甚至允许动态更改

ROS 节点到软件或硬件的映射，或在运行时将硬件节点分配到特定的可重构槽位。不过，本文仅讨论静态映射。

例如，需要接收来自 ROS 主题的新消息的硬件节点，会通过其 OSIF 将相应命令发送给代理线程。代理接收该命令，将其转发给 ReconROS 栈，并阻塞等待，直到较低层的 ROS 层提供新消息。这会唤醒代理，随后代理通过 OSIF 将新消息的地址发送给硬件节点。最后，硬件节点可以使用其 MEMIF 读取消息。ReconROS 基于共享内存架构构建，所有硬件节点共享一个访问主存的端口。因此，硬件节点共享该内存端口的带宽；同时，它们还与多核 CPU 共同共享通往主存的总带宽。

ReconROS 构建流程从配置文件开始。该配置文件记录目标平台、综合工具链、ROS 版本、可重构槽位的数量、硬件节点及其 ROS 特定资源（例如订阅者和发布者），以及硬件节点到可重构槽位的映射，并生成所有必需的软件二进制文件和 FPGA 配置比特流。ReconROS 支持使用 VHDL 和 Verilog 等硬件描述语言设计硬件节点，也支持基于 C/C++ 的高级综合。

III. FPGA 内数据分发服务

ReconROS 框架使用 ROS 2 标准通信中间件在节点（包括硬件节点）之间传输数据。虽然这种方式灵活性很高，硬件节点能够与任意软件节点和硬件节点通信，但由于硬件节点通过代理线程调用 ROS 函数的机制，以及所有硬件节点之间存在竞争性内存访问，整体应用性能可能并不理想。

本文介绍了 fpgaDDS，这是一种面向 ReconROS 应用的新型轻量级 FPGA 内数据分发服务。fpgaDDS 将硬件节点之间的 ROS 通信映射到可重构逻辑结构中，从而避免大量内存访问，同时保留标准 ROS 2 发布-订阅通信的 ROS 编程模型，即无需更改 ROS 节点。图 1 概述了 fpgaDDS 及相应的 ReconROS DDS 适配器如何集成到 ROS2 通信栈中。

这种新型 FPGA 内 DDS 有两项优势：首先，将通信硬件节点之间的数据传输映射到
可重构逻辑结构可减少大量内存访问，从而提升应用性能。其次，将节点和通信从软件迁移到并行执行的硬件上，可进一步缩短应用执行时间并降低抖动，有助于在 ROS 2 下实现可预测的实时行为。

A. FPGA 内部通信架构

ROS 2 的发布-订阅通信原理启发了 fpgaDDS 通信架构的结构设计。扩展后的 ReconROS 构建

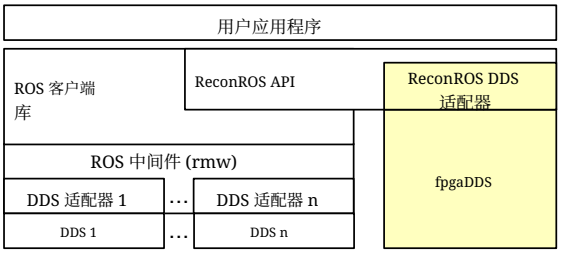


图 3. ReconROS 通信栈的扩展包括 fpgaDDS 及相应的 ReconROS DDS 适配器

流程会为每个 ROS 主题分别生成特定于应用的静态 AXI 流 (AXIS) 网络。这种基于 AXIS 的硬件映射主题 (HMT) 精简高效，因此运行时执行的通信协议相对简单，并且得益于每个 HMT 可用的总带宽，实现具有高性能。

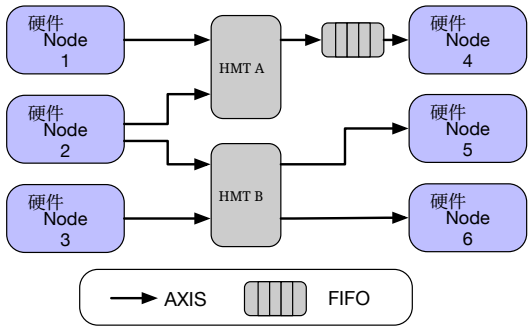


图 4. 包含两个硬件映射主题 A 和 B 的通信架构实例示例

图 4 展示了一个 fpgaDDS 示例，其中六个 Re-conROS 硬件节点通过 AXIS 连接到两个 HMT：A 和 B。图中，从硬件节点指向 HMT 的有向连接表示发布者，而从 HMT 指向硬件节点的连接表示订阅。

HMT 的内部设计取决于所需的发布者和订阅者数量。在最简单的情况下，HMT 从一个发布者接收数据，并将这些数据提供给一个订阅者，此时 HMT 只需包含一条从输入到输出的简单 AXIS 连接。如果某个 HMT 有多个发布者，则由 AXI 互连 IP 模块汇集各路输入，并对输入消息进行仲裁。AXI 互连经过配置，可基于完整消息进行仲裁，从而确保转发的始终是完整消息，保证数据一致性。如果 HMT 有多个订阅者，则由 AXI 广播 IP 模块将消息广播给所有订阅者。自然，可以将 AXI 互连和广播 IP 模块串联起来，以实现具有任意数量发布者和订阅者的 HMT。

图4还展示了硬件节点4对主题A的订阅所使用的FIFO缓冲区。这类FIFO缓冲区是HMT订阅者的可选组件。通过添加FIFO缓冲区，可以将订阅者节点的数据处理与主题解耦。这种机制可能很有帮助

例如对于异步通信，订阅者可以在不同的时间读取主题中的数据。

在 ROS 2 中，DDS 层允许开发者指定各种服务质量（QoS）参数。fpgaDDS 凭借其 AXIS 流式架构，默认实现了发布-订阅通信，并采用全部保留和可靠QoS 参数，不会丢弃任何消息。存储的消息数量取决于 FIFO 的大小。如果 FIFO 已满，传输就会阻塞。此外，由于采用静态 DDS 结构，fpgaDDS 提供无限的生命周期时长和无限的租约时长。

fpgaDDS 的通信架构在设计时根据 ReconROS 项目配置文件进行综合。首先，识别具有硬件属性的所有订阅者和发布者，并按主题分组。然后，对于每个 HMT，ReconROS 构建流程通过添加输入（订阅）端口和输出（发布）端口来扩展硬件节点，插入 HMT AXIS 基础设施，并相应地连接各端口与基础设施。整个流程对应于标准 DDS 实现中的发现过程。

B. ReconROS DDS 适配器

ReconROS DDS 适配器旨在弥合 fpgaDDS 层流式网络编程模型与 ReconROS 中发布/订阅通信机制之间的差距。因此，程序员可以使用相似的阻塞函数和非阻塞函数，与软件映射主题和硬件映射主题进行交互。

为了保持 ROS 2 提供的标准消息结构与我们的流式 fpgaDDS 通信架构所使用的消息结构兼容（该架构由高级综合工具流程生成），我们需要在发送端序列化消息对象，并在接收端对其进行反序列化。

由于 ROS 2 采用多层架构并支持多种不同的通信中间件，ROS 2 框架已在软件中提供 ROS 消息序列化功能，甚至支持复杂的嵌套消息类型。然而，由于 Vivado HLS 对指针运算、类型转换和递归的支持有限，这些函数无法直接用于硬件生成。

因此，我们从消息定义文件中提取消息的结构，即其组件和数据类型，并将所有非基本类型替换为其子组件。

这也使得解析多层嵌套消息成为可能。随后，将生成的原语和数组列表转换为用于消息发布与订阅的 C 宏。在硬件综合过程中，这些宏会被内联，并生成硬件，以阻塞和非阻塞两种方式，按顺序向 AXI 流接口写入数据和从中读取数据。

C. 执行模式

通信硬件节点可通过 fpgaDDS 以两种执行模式运行。第一种模式与使用基于软件的 DDS 时相同，依次执行接收、计算和发送阶段。图 5-a 展示了一个包含三个节点、采用发布-订阅通信的链式结构示例。

借助 fpgaDDS，我们还可以利用高级综合工具的数据流选项，创建一种实现，使各阶段尽可能重叠执行，且仅受数据流约束。图 5-b 展示了这种操作模式，它能够显著缩短整体执行时间。此模式适用于以流式方式处理数据的节点集合，例如许多底层图像处理任务便是如此。

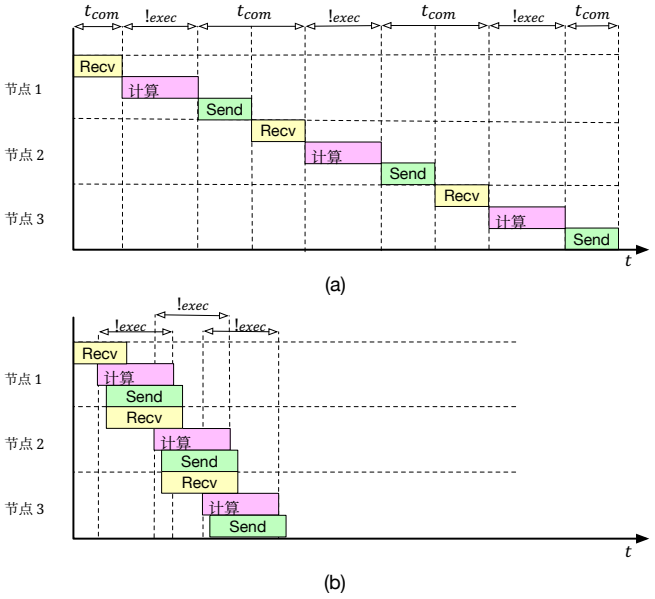


图 5. 带有通信（t_{com}）和执行（t_{exec}）阶段的硬件节点的执行模式：(a) 顺序执行；(b) 数据流执行

IV. 评估

本节报告使用 fpgaDDS 进行的实验。首先介绍实验设置，随后通过硬件节点实验展示使用 fpgaDDS 的优势。最后，给出自动驾驶实际应用的测量结果。

A. 实验设置

所有硬件节点均使用 C++ 实现，并通过高级综合工具 Xilinx Vitis 2021.2 综合为硬件。计算平台为一块 ZCU104 评估板，搭载运行 Ubuntu Linux 18.04、ReconROS 和 ROS 2 dashing 的 UltraScale+ MPSoC FPGA。ReconROS 基础设施和硬件节点的运行频率为 100 MHz。对于自动驾驶示例，我们还使用一台配备 Intel Core i5-8000 CPU、运行 Ubuntu Linux 18.04 和 ROS 2 dashing 的台式电脑，并通过千兆以太网将其连接到 FPGA评估板。

B. 硬件映射主题评估

在第一个实验中，我们实现了两个通过主题连接的硬件节点，并测量了从开始发送到接收完成所需的时间

数据传输重复进行了 1000 次，并报告平均值。表 I 展示了使用软件映射的 CycloneDDS 和硬件映射的 fpgaDDS 的硬件节点在不同消息大小下得到的传输时间 t_{avg} 及其标准差 σ 。表中还列出了 fpgaDDS 实现的加速比。加速比表明，对于较小的数据量，使用 CycloneDDS 的开销占主导地位。然而，对于较大的数据量，加速比趋近于 2，因为与通过 HMT 只需传输一次相比，数据必须通过 MEMIF 传输两次。不过，CycloneDDS 的测量结果代表了最乐观的情况。随着 MEMIF 利用率提高，由于各节点共享 MEMIF 的带宽，传输时间会增加。

增加发布者和订阅者的数量会延长 CycloneDDS 的传输时间。对于 fpgaDDS，只有增加发布者才会延长传输时间，因为 HMT 会仲裁传入的消息。然而，在 fpgaDDS 中增加订阅者不会影响传输时间。

消息大小	CycloneDDS t_{avg} (σ) [毫秒]	fpgaDDS t_{avg} (σ) [毫秒]	加速比
3 kB	0.06 (0.11)	<0.01（<0.01）	17.21
12 kB	0.07 (0.06)	0.02 (<0.01)	4.71
50 kB	0.19 (0.08)	0.06 (<0.01)	3.11
196 kB	0.63 (0.04)	0.24 (<0.01)	2.54
786 kB	2.34 (0.03)	0.98 (<0.01)	2.38
3146 kB	9.24 (0.04)	3.93 (<0.01)	2.35

表 I
CycloneDDS 和 fpgaDDS 在不同消息大小下的通信时间与加速比

C. 自动驾驶车辆示例

在本实验中，我们详细介绍并扩展了 [14] 中的自动驾驶应用。该应用受到 Turtlebot 自动赛车挑战的启发，能够沿车道行驶，并在行驶过程中处理交通信号灯。目标机器人是 Turtlebot 3 平台，配备一台 0.3 百万像素的摄像头，用于车道和交通信号灯检测。在我们的配置中，生成的 ReconROS 应用运行于 FPGA评估板上，并连接到运行 Gazebo 物理仿真环境的台式电脑。

图 6-a 展示了该应用的计算图。计算图唯一的输入是机器人的相机图像。图像补偿 节点计算输入图像的直方图，并利用它去除离群像素。随后，处理后的图像被发布到主题*A*。高斯模糊 节点订阅图像数据，应用低通滤波器，并通过主题*B*将结果转发给图像投影节点。图像投影 节点通过扭曲变换生成车道的正交视图，然后将图像调整为 1000x600 像素。最后，车道跟随 节点从图像数据中提取车道左右两侧的线条，并计算它们之间的中心点。该中心点被发送到车道控制节点，该节点根据中心点计算驾驶指令。

除高斯模糊 节点外，红色交通信号灯检测 节点和绿色交通信号灯 节点也订阅主题*A*。这两个节点都会将图像转换到 HSV颜色空间，然后检测红色或绿色交通信号灯。如果检测到红色交通信号灯，该节点便会向主题*E*发布停止指令，从而由车道控制节点停止机器人。此外，该指令还会启动绿灯检测。检测到绿灯后，该节点会向主题*F*发布启动指令，从而再次激活车道控制节点。

我们以三种不同版本实现了图 6-a 所示的 ROS2 应用：(i) 软件版本，节点在 CPU 上运行，并基于 CycloneDDS 进行通信；(ii) 硬件版本，所有节点均采用高级综合实现，但仍使用 CycloneDDS 进行通信；以及 (iii) 硬件版本，节点使用我们新颖的 fpgaDDS。第三个版本的架构图如图 6-b 所示。此外，HMT主题*E*和*F*的订阅者配备了缓冲区，从而能够通过这些 HMT 实现异步通信。

软件实现不使用任何可重构逻辑资源。表 II 总结了硬件实现对可重构逻辑结构的资源利用率。该表列出了查找表（CLB查找表）、专用存储块（BRAM、URAM）和算术功能块（DSP）等资源类型。表 II 的主要结论是，与采用 CycloneDDS 的版本相比，fpgaDDS 不需要更多的逻辑资源。这是因为基于AXIS 的通信架构所需的逻辑资源相对较少。另一方面，CycloneDDS 在实现节点内缓冲区时需要大量额外的 CLB 查找表。

实现	CLB查找表	BRAM	URAM	DSPs
硬件（CycloneDDS）	134252（58%）	250.5（80%）	95 (99%)	230 (13%)
硬件 (fpgaDDS)	44939 (20%)	255.0 (82%)	91 (95%)	230 (13%)

表 II 硬件实现的资源利用率（占 XCZU7EV-2FFVC1156 的使用量百分比）

为了评估不同通信中间件的性能，我们测量了节点链 α 的执行时间，该节点链在图 6 中以红色虚线标出。节点链从图像补偿节点开始，以车道控制节点结束。我们选择这条节点链，是因为它包含了车辆在车道跟随模式下通常执行的大多数节点，并且会定期向主题馈送消息数据。相比之下，只有检测到红色或绿色交通信号灯时，主题*E* 和*F*才会接收数据。

我们通过计算车道控制节点的发布时间与

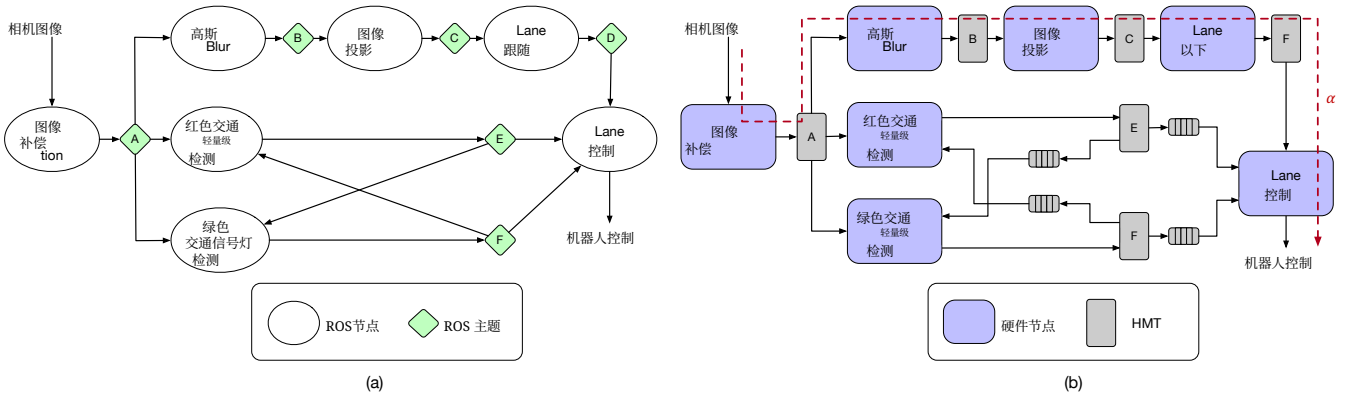


图6. 自动驾驶车辆示例的计算图 (a) 及其生成的架构图 (b)

图像补偿节点的订阅时间之差，确定了节点链 {v1}的执行时间。同样，我们重复实验 1000 次，并在表III中报告平均值和标准差。此外，该表还展示了相较于纯软件实现所获得的加速比。例如，将 ROS 节点部署在硬件上，同时保留基于软件的 CycloneDDS，可获得 2.48 的加速比。若将通信也映射到硬件上，并使用 fpgaDDS，则可获得 13.34 的加速比。此外，使用 fpgaDDS 后，与抖动相关的标准差降低了两个数量级。

实现	t_{avg} (σ) [毫秒]	加速比
软件 (CycloneDDS)	274.44 (14.96)	1.00
硬件 (CycloneDDS)	110.72 (07.73)	2.48
硬件 (fpgaDDS)	20.58 (00.13)	13.34

表III 图 6 的节点链 α 及不同实现版本的执行时间和加速比

V. 结论与未来工作

本文介绍了 fpgaDDS，这是一种面向映射到硬件上的 ROS 节点的精简型数据分发服务。fpgaDDS 是一种静态生成、定制化的流式通信架构，集成于 ReconROS 中，可确保硬件节点之间实现低延迟、高带宽通信。实验表明，该架构显著提升了性能并降低了抖动。

未来工作将沿两个方向展开：首先，我们计划扩展 fpgaDDS，以支持 ROS2 服务和动作，并支持更丰富的服务质量参数。其次，我们希望让 fpgaDDS 支持动态调度的 ROS 2 节点。ROS 2 设想了一种称为执行器的机制，可针对特定事件分派回调函数，以实现动态调度。在现场可编程门阵列平台上，部分动态重配置支持动态调度。将此类动态技术与 fpgaDDS 相结合颇具挑战，因为 fpgaDDS 目前是静态生成的。

参考文献

[1] K. Yamashina、H. Kimura、T. Ohkawa、K. Ootsu 和 T. Yokota，“CReComp：用于符合ROS标准的FPGA组件的自动化设计工具”，载于*IEEE*第十届嵌入式多核/众核片上系统国际研讨会论文集，*MCSoC 2016*。IEEE，2016年，第138–145页。[2] M. Eisoldt、S. Hinderink、M. Tassemeier、M. Flottmann、J. Vana、T. Wiemann、J. Gaal、M. Rothmann 和 M. Porrmann，“ReconfROS：在可重构片上系统上运行ROS”，载于*2021*年无人机系统工程、快速仿真与性能评估：方法与工具论文集。ACM，2021年，第16–21页。[3] C. Lienen 和 M. Platzner，“使用ReconROS设计分布式可重构机器人系统”，*ACM*可重构技术系统汇刊，第15卷，第3期，2022年12月。[4] S. Macenski、T. Foote、B. Gerkey、C. Lalancette 和 W. Woodall，“机器人操作系统2：设计、架构及实际应用”，《科学机器人》，第7卷，第66期，2022年。[5] “eprosima fast dds”，https://github.com/eProsima/Fast-DDS，访问日期：2023-02-28。[6] “iceoryx - 真正的零拷贝进程间通信”，https://github.com/eclipse-iceoryx/iceoryx，访问日期：2023-02-28。[7] “Eclipse cyclone dds”，https://github.com/eclipse-cyclonedds/cyclonedds，访问日期：2023-02-28。[8] V. Mayoral-Vilches，“Kria机器人技术栈”，
h
t
t
p
s://
w
w
w.
xilinx.com/applications/industrial/robotics/wp540-kria-robotics-stack.html，2021年，访问日期：2022-01-13。[9] Y. Sugata、T. Ohkawa、K. Ootsu 和 T. Yokota，“在符合ROS标准的FPGA组件中加速发布/订阅消息传递”，载于第八届高效加速器与可重构技术国际研讨会（*HEART2017*）论文集。ACM，2017年。[10] T. Ohkawa、Y. Sugata、H. Watanabe、N. Ogura、K. Ootsu 和 T. Yokota，“用于现场可编程门阵列的ROS协议解析与通信电路的高级综合”，载于*2019*年*IEEE/ACM*机器人软件工程第二届国际研讨会论文集（*RoSE*），2019年，第33–36页。[11] V. Mayoral-Vilches、S. M. Neuman、B. Plancher 和 V. J. Reddi，“Robotcore：面向ROS 2硬件加速的开放架构”，载于*2022*年*IEEE/RSJ*智能机器人与系统国际会议（*IROS*），2022年，第9692–9699页。[12] A. Podlubne 和 D. G¨ohringer，“FPGA-ROS：通过FPGA设计增强机器人操作系统的方法”，载于可重配置计算与现场可编程门阵列国际会议论文集（*ReConFig*），2019年。[13] E. L¨ubbers 和 M. Platzner，“ReconOS：面向可重构计算机的多线程编程”，*ACM*嵌入式计算系统汇刊，第9卷，第1期，第8:1–8:33页，2009年。[14] C. Lienen 和 M. Platzner，“使用ReconROS对硬件加速机器人应用进行任务映射”，载于*2022*年第六届*IEEE*机器人计算国际会议（*IRC*），2022年，第148–155页。