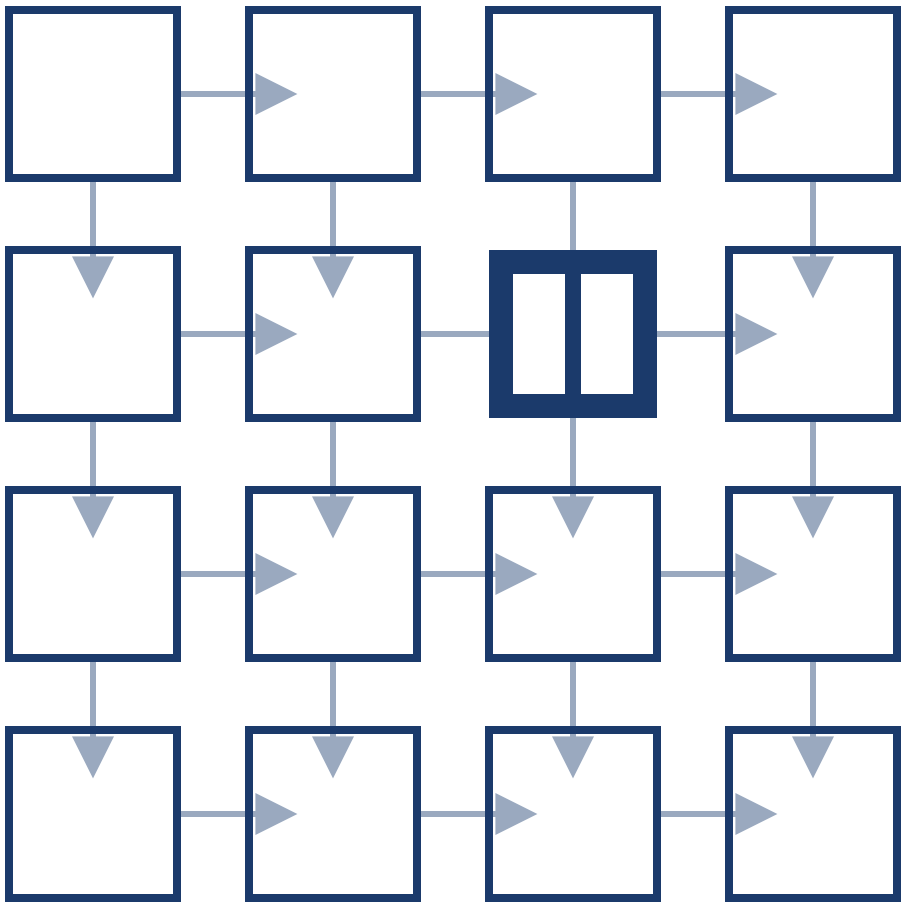


把 supercomputer 做进一颗芯片

iWarp: An Integrated Solution to High-Speed Parallel Computing

Shekhar Borkar et al. · 16 位作者 · Carnegie Mellon University &
Intel Corporation

Supercomputing '88 · IEEE · pp. 330–339



iWarp cell 阵列 · 点对点单向 pathway
互连

焦点 cell: computation ||
communication 双 agent

约 **600,000**

transistors / 单颗芯片

20 MHz

clock · one clock = 50 ns

10 MFLOPS

double-precision 峰值 / 单个 cell

从挑战到实证：六个层次读懂 iWarp

01	02	03
挑战与动机	解决方案	具体架构
分布式内存并行计算在 1980 年代末的四个系统性矛盾	把 computation 与 communication 两个引擎集成进单颗 VLSI 芯片	component → cell → system 三层；双 agent 与四大功能单元
04	05	06
通信模型与机制	实验验证与结果	总结与展望
message passing systolic；pathway / message 统一抽象与硬件实现	retargeted compiler + architecture simulator；Livermore Loops 与 NETtalk	一类新并行架构的开场；affordable supercomputing 的 building block

1980 年代末的分布式内存并行计算：卡在计算与通信难以兼得的系统性矛盾

C1 计算—通信失衡

处理器数值能力持续提升，通信却跟不上：阵列整体效率被 interprocessor communication 拖住。论文的原话是——iWarp must be **equally efficient in communication**。

C2 两种粒度，一套硬件

message passing 以完整消息为单位、通信伙伴运行时才确定；**systolic** 细到单个 word、要求专用低抖动路径。需求迥异，却要在同一颗芯片上同时高效支持。

C3 前任 Warp 的工程天花板

Warp 是 board-level 离散实现：cost、reliability、power、physical size 都难再压，阵列规模也难扩大。iWarp 的目标是至少 **一个数量级** 的改善，且 clock 翻倍。

C4 专用效率 vs 可编程性

special-purpose systolic array 高效但难编程；general-purpose array 需要 compiler 与 programming model 支撑。拓扑上 1D 易建但 hop 数 $O(n)$ ，大规模系统必须走向 2D / torus。

iWarp 的回答：把 computation 与 communication 两个引擎，集成进单颗芯片

iWarp component · 单芯片 $\approx 600,000$ transistors

computation agent || communication agent：同芯片、独立控制、紧密耦合

+ **local memory** (≤ 64 MBytes 直接寻址)

iWarp cell

加内存芯片即成完整 building block——一台处理器节点

+ **cell $\times$ N** (点对点 physical bus 互连)

iWarp system

1D / ring / 2D / torus / hexagonal：从几个到上千个 cell

设计目标 · 论文 §1

G1 **special-purpose 与 general-purpose 阵列通吃**：高 bandwidth 与 programmability 都要

G2 **host centric 与 array centric 两用**：可 attached，可自治完成全部计算与 I/O

G3 **tightly 与 loosely coupled 并行都支持**：systolic 与 message passing 同机共存

G4 **拓扑与规模不设限**：1D array 到 torus，数个到数千个 processor

20 MFLOPS

单 cell 计算峰值 (single precision)

320 MBytes/sec

单 cell 聚合通信带宽

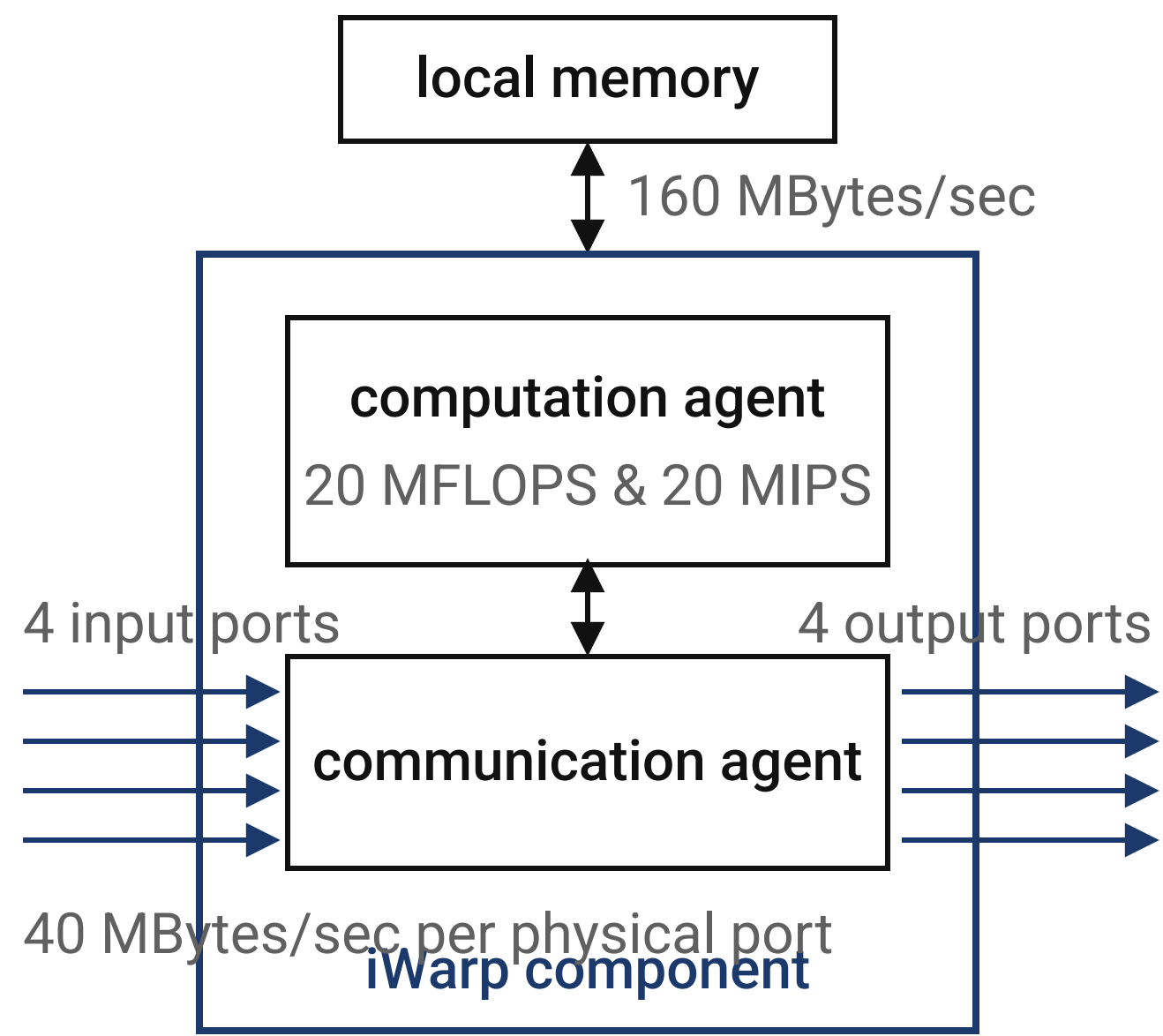
100–150 ns

through-a-cell 通信延迟

$\geq 10\times$ improvement

相对 Warp：cost / reliability / power / size

iWarp cell: component 加 local memory, 双 agent 同芯片并行



重绘自论文 Figure 1 · iWarp component overview

集成度 single chip	约 600,000 晶体管；clock 20 MHz (one clock = 50 ns)
computation agent LIW 控制	FP adder + FP multiplier + integer unit, 一条 long instruction word 并行调度；non-pipelined, vectorizable 与否都持续高速
communication agent 独立控制	4 入 + 4 出 ports；与计算并行——through-cell 通信 不占用 cell 程序
local memory off-chip	数据与指令共用；≤ 64 MBytes 直接寻址；24-bit address + 64-bit data bus
on-chip program store	256-word cache RAM + 2K-word ROM；32- 与 96-bit 指令

来源：论文 §2.1、Figure 1–2

双 agent 独立 = 计算与通信可重叠执行

communication agent: 把通信做成硬件保障的一等公民

physical ports

点对点 physical bus
4 入 + 4 出; 每 port **40 MBytes/sec**; word-by-word 硬件 flow control, 无同步延迟

pathway unit

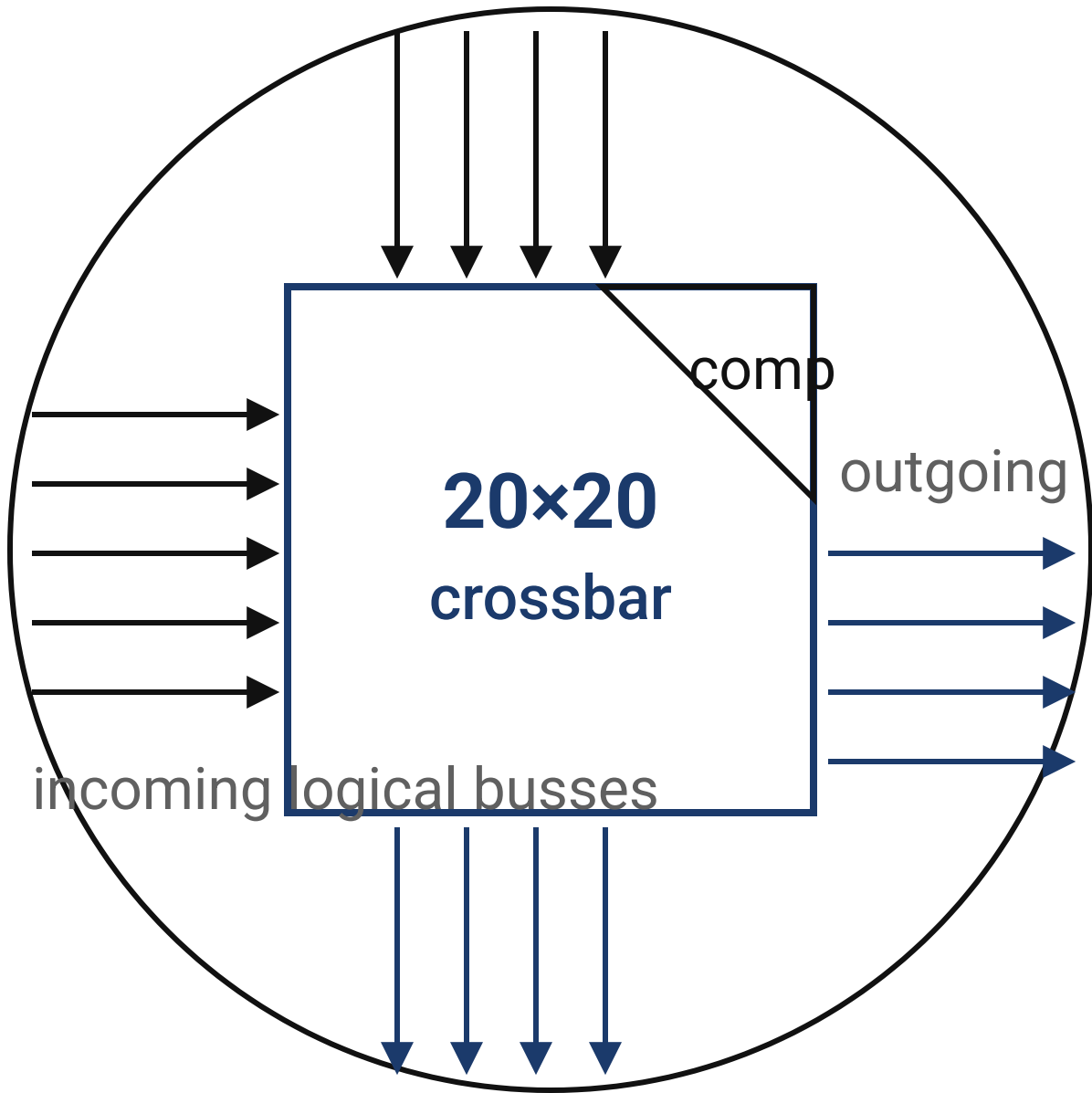
硬件路由
1D / 2D 配置路由; 支持 **wormhole** 与 **streetsign** routing; 建路全程硬件完成

logical busses

word-level 多路复用
每条 physical bus 复用多条 logical bus; 单 component 同时维持 ≤ 20 条 **incoming pathways**; 空闲不占带宽

streaming & spooling

计算 $\leftrightarrow$ 通信接口
computation agent 经 streaming gates 逐 word 直读通信数据; 或经 spooling gates 硬件搬运到 local memory



重绘自论文 Figure 8(b) · crossbar

320 MBytes/sec

aggregate data transfer rate

20 busses

incoming logical busses 上限 / cell

20x20

crossbar: incoming $\rightarrow$ outgoing 直连

0 consult

发送方用预分配 free bus 发起, 无需询问接收方

computation agent: 一条 LIW 指令并行调度三个单元，不需要 vectorization

floating-point multiplier non-pipelined · IEEE 754 · div / rem / sqrt	10 / 5 MFLOPS · 32/64-bit
floating-point adder non-pipelined · IEEE 754	10 / 5 MFLOPS · 32/64-bit
integer / logic unit 8/16/32-bit · arithmetic / logical / bit	20 MIPS peak
shared multiported register file · 128 words operand bypassing · hardware zero-overhead looping · 高带宽低延迟 memory (no striding penalty)	

20 + 20 MFLOPS + MIPS
三单元并行的 peak computing rate

8 → 4 operands → results
单条 C&A 指令的数据通路宽度

2 clocks
single-precision C&A (double = 4 clocks)

COMPUTEANDACCESS (C&A) · LIW 主力指令
最多 **8 个 operands** → **4 个 results**; 含 ≤2 次 memory access (带非单位 stride 地址计算) 与 **4 次 gate access** (直读通信数据) ; read-ahead / write-behind pipeline 隐藏内存延迟。

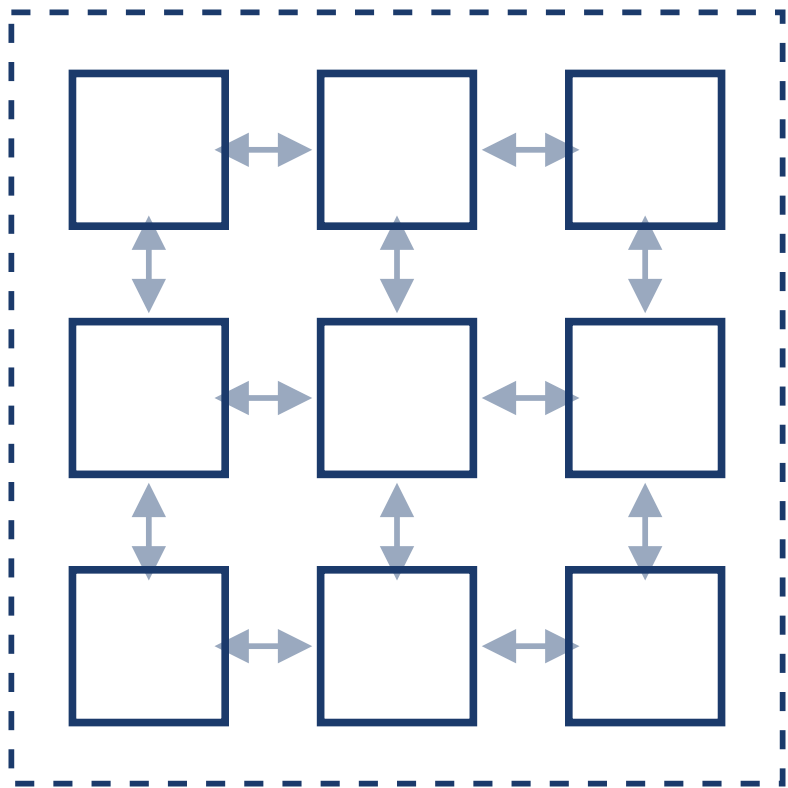
```
FOR i := 0 TO n-1 DO BEGIN
    f := (A[i]*B[3*i]) + f;
END;
```

循环体编译为**单条 C&A**: single precision 2 clocks、double 4 clocks 一次迭代——循环以 peak rate 运行，无论是否 vectorizable。
另有全套 2/3-address 「short」 RISC-like 指令保证 general programmability。**三单元一条指令内并行** → peak = 20 MFLOPS + 20 MIPS。

同一种 cell，靠互连拓扑跨越从加速器到超算的全谱系

2D torus

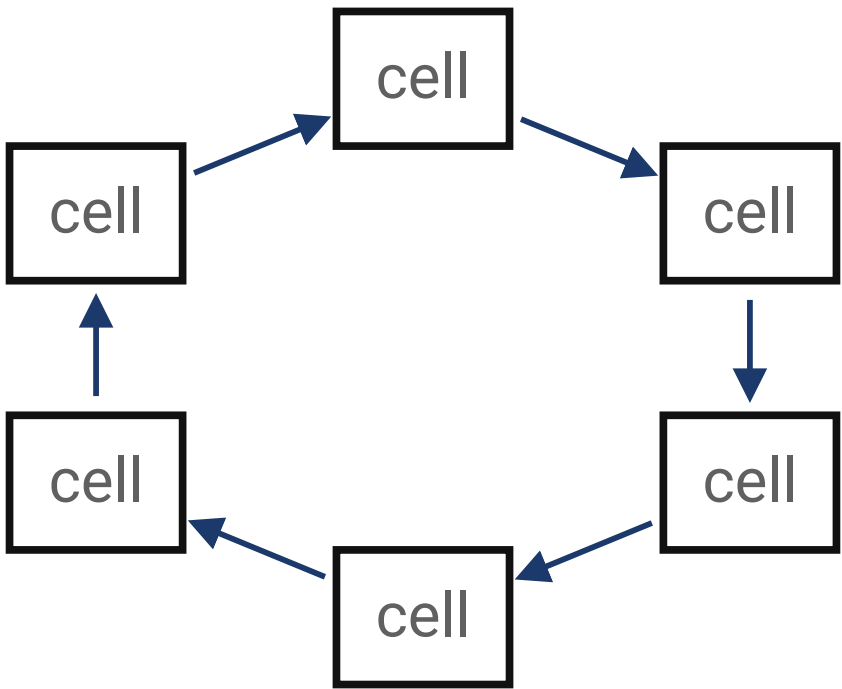
通用并行 · 演示系统形态



重绘自 Figure 5 · 虚线 = wraparound

ring / 1D array

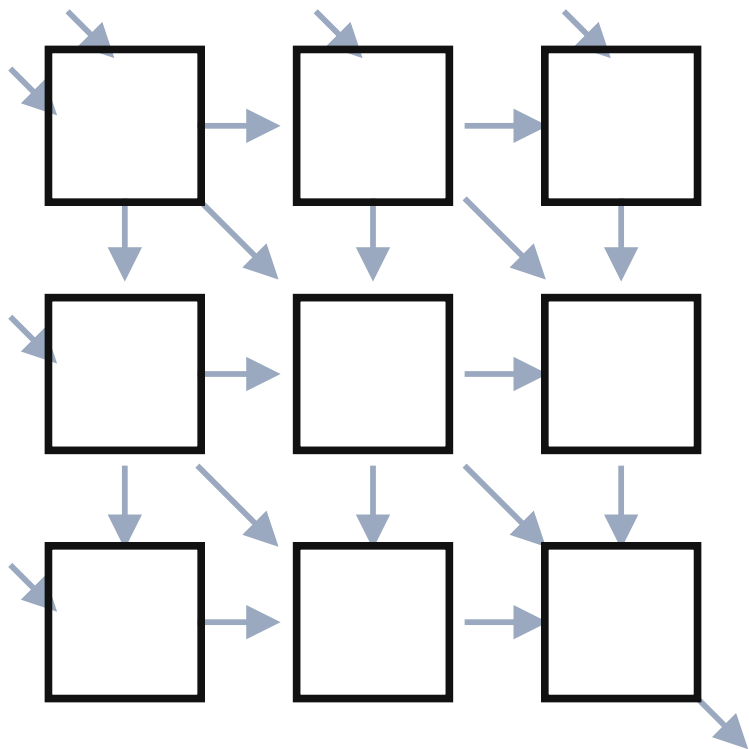
workstation attached processor



重绘自 Figure 6 · 单向 pathway 环

hexagonal array

special-purpose systolic



重绘自 Figure 7 · 单向 physical busses

8×8 torus · 1990

演示系统：32 MBytes
SRAM · 1,280 MFLOPS

20,480 MFLOPS

同设计扩展至 32×32 torus
的 aggregate peak

1,024 cells

系统规模上限

64 MBytes

每 cell 可直接寻址内存上限

message passing 与 systolic：两种粒度，被同一套硬件需求收敛

message passing

coarse-grain

处理单位 = **完整消息**：源 cell 攒齐、整包传输、到目的地内存后才可操作；伙伴与时机常由 run time 决定。

高效支持所需能力

spooling	收发由独立控制流搬运，不阻塞计算进程
wormhole routing	逐 word 转发、不在中间 cell 整包缓冲
multiplexing	多路径复用一条 physical bus（virtual channel 防 deadlock）
door-to-door	直写接收方目标地址，免除二次拷贝

统一抽象 →

wormhole routing 同时占用多条硬件链路，恰似 systolic 中连接非邻居进程的路径——两种模型在 iWarp 上被同一对编程抽象高效支持：**pathway**（源到目的的直接连接）与 **message**（header + data words + end marker）。硬件实现见下页。

systolic

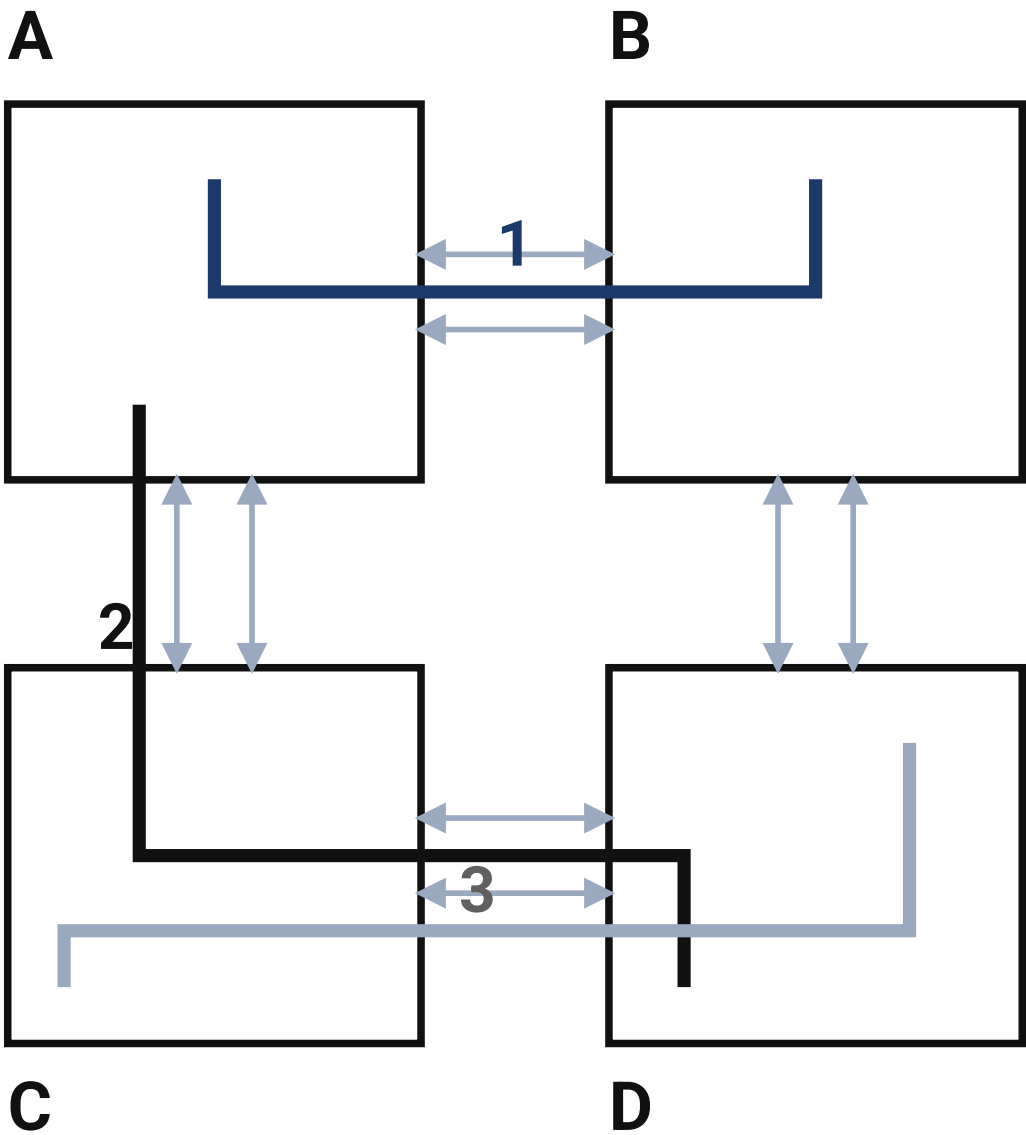
fine-grain

处理单位可细到**单个 word**：源边产生边发送，目的第一个 word 到达即开始计算；如一个 cell 加法器的输出直接作另一个的操作数。

高效支持所需能力

dedicated paths	邻居或非邻居间专用路径，成本恒定、抖动小
FIFO queuing	沿路径队列松弛收发耦合；空读/满写才 stall
直接耦合	计算单元直读通信数据，不经 local memory（≤4 words/clock）
spooling overflow	队列不够深时（如 2D 卷积存整行）自动溢出到 local memory

pathway 与 message: open、传字、close, 全程硬件完成



重绘自论文 Figure 9 · 三条 pathway (2 与 3 在 C→D 复用总线)

open pathway marker · streetsign routing —— 携带地址序列 ("go to Jones, turn right, ..., stop" = streetname + action) ; 每跳 communication agent 用 **Address Match CAM** 硬件识别, 只处理 "next" streetname

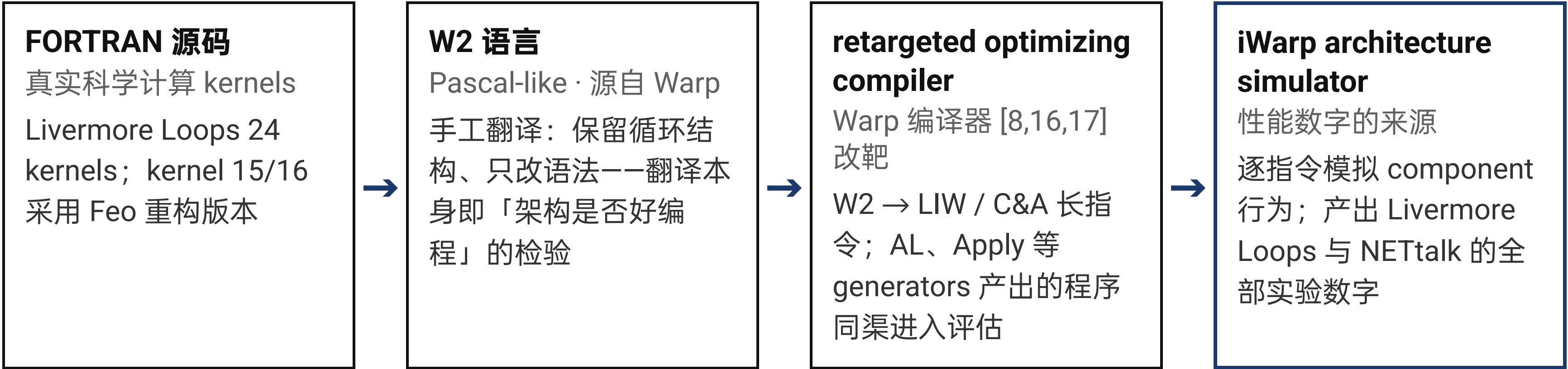
零附加延迟建路 · hardware only —— through-a-cell **100 ns**、corner turning **150 ns**; 新建 pathway segment 不引入额外延迟; close marker 逐跳拆除

word-level multiplexing · logical busses —— 多条 pathway segment 按 word 粒度共享一条 physical bus; 空闲不占带宽; source 用自己的 free logical bus 发起通信, 无需协商

streaming gates · 2 in + 2 out —— 读 gate = 消耗 input message 的下一个 word, 写 gate = 产生 output message 的下一个 word; 空读自动 spin 直到数据到达——同步即副作用

spooling gates · queue ↔ memory —— 硬件在 pathway unit 与 local memory 间搬数据; 低开销、对并行计算透明 (仅 cycle stealing 与内存访问干扰)

芯片流片之前，retargeted compiler + architecture simulator 已把真实程序跑了一遍



对照实验：NETtalk 神经网络 benchmark 在 **10-cell Warp 真机** 与 **同 cell 数 iWarp 模拟** 上对跑——16.5M → 36M connections/sec、70 → 153 MFLOPS，验证「clock 翻倍 + I/O 带宽同步翻倍 ⇒ 效率不降」的论证链。

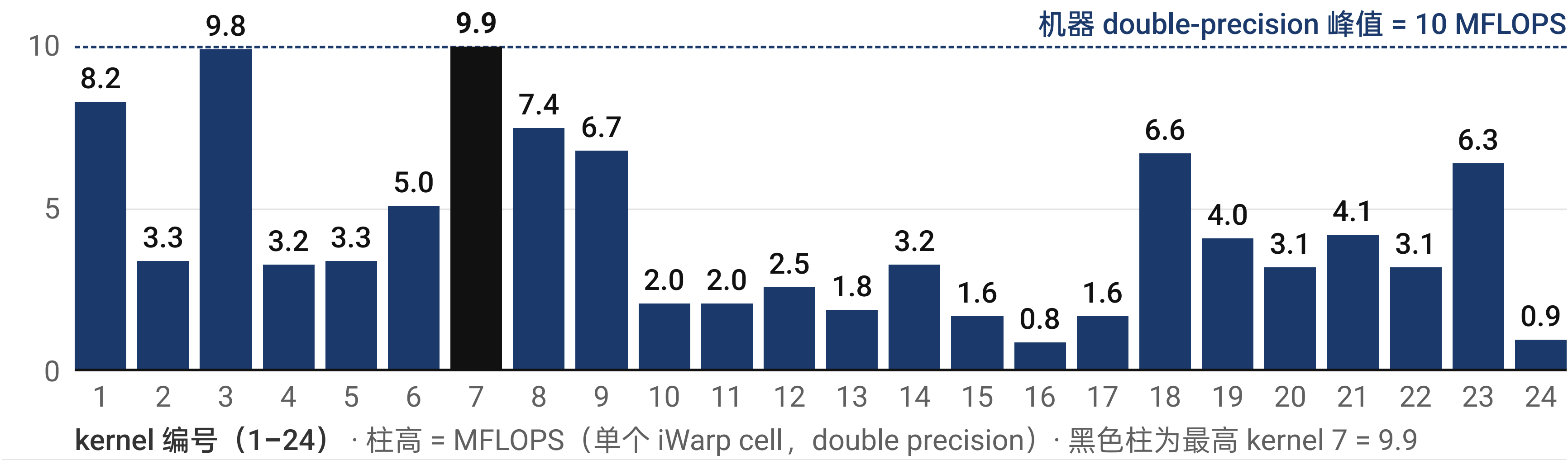
1988 · 07
architecture 与 logic design 完成; 编译器 retarget 完成

1989 · 年底
prototype iWarp system 预期 operational

1990 · 年中
三套 8×8 torus 演示系统 (>1.2 GFLOPS) 预期 operational

EXPERIMENTS · LIVERMORE LOOPS (TABLE 6-1) · DOUBLE PRECISION · 单个 iWarp cell

24 个 kernels 实测：unweighted mean 4.2 MFLOPS, scalar processor 不依赖 vectorization



解读：iWarp 是 scalar processor，性能不依赖 vectorization。限制因素：**data dependency**（如 kernel 5 的递归依赖把峰值压到 5 MFLOPS）与 **critical resource bottleneck**（最忙的资源决定速度；无乘法的程序 ≤ 5 MFLOPS）。

NETTALK · 10-CELL
WARP

Warp **70** MFLOPS
iWarp **153** MFLOPS
conn/sec: 16.5M $\rightarrow$ 36M

4.2 **2.7** **10**
unweighted mean harmonic mean 峰值 MFLOPS
单位 MFLOPS · $\sigma = 2.6$ · Table 6-1

计算与通信不是对立需求——在 iWarp 上，它们互相加强

贡献 · 论文 §7

- K1 一类新并行架构的开山：**单颗 VLSI component 集成 computation 与 communication
- K2 统一通信抽象：**pathway / message 覆盖 message passing 到 systolic 的谱系
- K3 LIW scalar 设计：**不依赖 vectorization 也持续高速（Livermore Loops 佐证）
- K4 商业化 building block：**强计算 + 强通信 + 可商用 = affordable supercomputing

展望 · 论文 §7

- F1 首批系统形态：**1D array、ring、2D array、torus；component 足够灵活，可用于更多组织形式
- F2 数千 cell 的阵列：**feasible、programmable，且比同算力超算便宜得多
- F3 编程生态：**cell 级 FORTRAN / C + parallel program generators（AL、Apply）降低门槛

「These requirements are not incompatible and, in fact, do reinforce each other.」
——高带宽/低延迟通信与强计算在同一芯片上的 synergy，是全文论点句

>1.2 GFLOPS
8×8 torus 演示系统（1990）

20,480 MFLOPS
32×32 torus 的 aggregate peak

1,024 cells
单一系统规模上限

谢谢。 问题与讨论

下一课我们从这颗 1988 年的芯片出发，看专用加速器如何一步步走向 TPU：当 communication 与 computation 的 synergy 被推到极致，阵列就变成了矩阵引擎。

本 deck 来源：Borkar, S. et al. "iWarp: An Integrated Solution to High-Speed Parallel Computing". Supercomputing '88, IEEE, pp. 330–339.

示意图为重绘版本（Figure 1 / 5 / 6 / 7 / 8 / 9），非论文扫描件

SELECTED REFERENCES	
[1]	Annaratone, M. et al. "The Warp Computer". <i>IEEE Trans. on Computers</i> C-36, 12 (Dec. 1987), 1523–1538.
[6]	Feo, J. T. "An Analysis of the Computational and Parallel Complexity of the Livermore Loops". <i>Parallel Computing</i> 7, 2 (June 1988), 163–186.
[12]	Kung, H. T. "Systolic Communication". Proc. Int'l Conference on Systolic Arrays, May 1988, 695–703.
[16]	Lam, M. S. <i>A Systolic Array Optimizing Compiler</i> . Ph.D. Thesis, Carnegie Mellon University, May 1987.
[20]	Sejnowski, T. J. & Rosenberg, C. R. "Parallel Networks that Learn to Pronounce English Text". <i>Complex Systems</i> 1, 1 (1987), 145–168.
[21]	Tseng, P. S., Lam, M. & Kung, H. T. "The Domain Parallel Computation Model on Warp". SPIE Vol. 977, 1988.