DOI:10.19651/j.cnki.emt.1802032

基于 MIC 的 Turbo 码译码并行加速

赵梦伟[1] 陈永锋[2] 刘 凯[1] 孙超群[1]
(1.上海大学特种光纤与光接入网重点实验室 上海 200444; 2.西南电子电信技术研究所上海分所 上海 200434)

摘 要: Turbo 码因具有卓越的纠错能力和接近香农理论极限的性能而受到广泛关注。为了满足译码的实时性需求,利用 Intel 公司的众核处理器的多核并行处理和迭代运算能力,并结合 OpenMP 在编译层面可以自动将程序并行化的能力,对 Turbo 译码的 CPU 程序进行众核移植和并行化。在保证译码性能的同时,使用折线逼近的 Log-MAP 算法,并在代码层面进行调整和优化。针对实际卫星信号,采用基于数据的 MIC 多线程并行处理模式,相比利用 CPU 进行处理,使用 Intel Xeon Phi Coprocessor 7120 众核处理器在计算速度上有将近 60 倍的提升,并且可以实现 8 路突发信号的实时处理。

关键词: Turbo 码;译码算法;众核处理器;并行加速
中图分类号: TN927 **文献标识码**: A **国家标准学科分类代码**: 520.3020

Parallel acceleration of turbo-code decoding based on MIC

Zhao Mengwei[1] Chen Yongfeng[2] Liu Kai[1] Sun Chaoqun[1]
(1.Key Laboratory of Specialty Fiber Optics and Optical Access Networks, Shanghai University, Shanghai 200444, China;
2. Shanghai Branch of South-West Electron and Telecom Technology Institute, Shanghai 200434, China)

Abstract: Turbo-code has received wide attention due to excellent error correction ability and performance near to the Shannon limit. In order to satisfy the real-time requirement of decoding, the multi-core parallel processing and iterative computing capabilities of Intel′s many integrated core (MIC) processor and OpenMP′s ability to automatically parallelize programs are used to decode Turbo-code. Then MIC migration and parallelization are used to the CPU program. While ensuring the decoding performance, the Log-MAP algorithm with the polyline approximation is used and adjusted and simplified at the code level. For the actual satellite signal, using the data-based MIC multi-thread parallel processing mode. Compared to the CPU processing, the calculation speed of the Intel Xeon Phi Coprocessor 7120 Many Integrated Core processor has nearly 60 times improvement. What is more, it can achieve 8 Real-time processing of the burst signal.

Keywords: Turbo-code; decoding algorithm; MIC processor; parallel acceleration

0 引 言

Turbo 码是一种级联码,有着几乎接近香农理论极限的译码性能[1]。迭代译码是 Turbo 码译码的主要特征,它采用软输出迭代译码来逼近最大似然译码[2],提高了译码性能。Turbo 码译码算法中的修正的最大后验概率译码算法(maximum a posteriori algorithm, MAP)[3],虽然性能最优,但计算复杂且运算量大。Log-MAP 算法为对数形式的 MAP 算法[4],采用了最大后验概率设计 Turbo 译码器。折线逼近的 Log-MAP 算法[5]将 Log-MAP 算法的校正函数进行折线拟合逼近,在降低计算复杂度的同时可以保证译码性能。

众核(many integrated core,MIC)架构是 Intel 公司专门为高性能计算设计的。MIC 芯片是将数十个精简的 x86 核心集成在一起的处理器,通常作为协处理器,提供高度并行的计算能力[6-7]。利用 CPU 实现 Turbo 串行译码,远远无法达到对多路卫星信号的实时译码的需求,这就需要进行硬件移植和并行加速。目前 DSP、FPGA 以及 GPU 芯片技术的发展,为 Turbo 译码的实现提供了有力条件,但需要针对硬件设计新的程序结构,工程上实现较为耗时耗力。而 MIC 芯片采用对称多处理结构[6],这种设计保证了 MIC 和 CPU 程序的共享和兼容,并且有高效的并行计算能力[8],只需简单的编译指导语句便可以实现 CPU 程序的众核移植和并行化,减少了维护成本。

收稿日期:2018-08-20

本文采用折线逼近的 Log-MAP 算法实现了实际卫星信号的 Turbo 译码。并根据实际工程应用的需要,利用 MIC 的多核并行处理和迭代运算能力,对 CPU 程序进行众核移植和 OpenMP 并行化。并使用 VTune 分析工具有针对性地在代码层面进一步做调整和优化处理,从而大幅提高运行速度,实现了多路卫星信号的实时译码。

1 Turbo 码译码算法

Turbo 译码器最主要的特点是迭代译码[9],其译码原理如图 1 所示,其中 $\boldsymbol{y}_i=[y_i^{p1},y_i^s,y_i^{p2}]$ 为接收到的数据,y_i^s 为对应的系统位数据,y_i^{p1} 和 y_i^{p2} 为对应的校验位数据。$Le1$ 和 $Le2$ 为经过分量译码器计算得到的外信息,$La1$ 和 $La2$ 为对应比特的先验信息,LLR 为对数似然比。

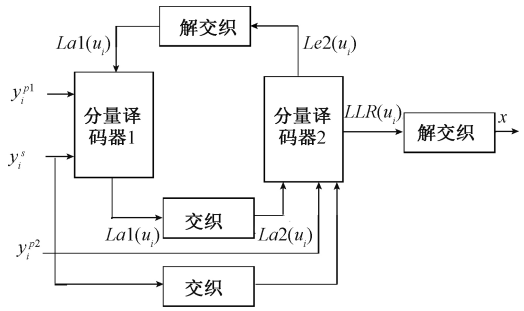

图 1 Turbo 译码原理

Turbo 译码中比特信息 $La1$、$Le1$、$La1$ 和 $La2$ 通过分量译码器 1 和 2 交织和解交织的迭代过程如图 1 所示。在迭代满足一定次数或者某个准则之后则停止,然后将得到的 LLR 经过解交织之后输出,通过判决得到译码信息。

Turbo 分量译码算法中的修正的 MAP 算法是在 BCJR 算法的基础上改进得到的[10]。设译码器输入为 y,定义关于 u_k 的对数似然比为:

$$L(u_k)=\ln\frac{P(u_k=1\mid y_1^N)}{P(u_k=0\mid y_1^N)} \tag{1}$$

利用全概率和贝叶斯公式,可以得到软信息量 $L(u_k)$ 的计算公式。

$$L(u_k)=\ln\frac{\sum\limits_{(s',s),u_k=1}\tilde{\alpha}_{k-1}(s')\gamma_k(s',s)\tilde{\beta}_k(s')}{\sum\limits_{(s',s),u_k=0}\tilde{\alpha}_{k-1}(s')\gamma_k(s',s)\tilde{\beta}_k(s')} \tag{2}$$

其中 $\tilde{\alpha}_k(s)$、$\tilde{\beta}_{k-1}(s')$ 和 $\gamma_k(s',s)$ 为 k 时刻不同条件下由不同状态变化到 s 状态的信息比特的概率。得到 $L(u_k)$ 后可以按照如下规则进行判断:

$$\begin{cases}u_k=1 & L(u_k)\geqslant 0\\ u_k=0 & L(u_k)<0\end{cases} \tag{3}$$

由于计算 $L(u_k)$ 过程乘法、对数和指数运算量大,限制了译码的规模和速度。将 MAP 算法变换到对数域,并使用如式(4)所示的近似。

$$\log\left(\sum_i e^{x_i}\right)\approx \max_i(x_i) \tag{4}$$

这种算法虽然相比 MAP 算法复杂性降低,但是因为使用了近似表示,所以也导致了误差的产生。Log-MAP 算法引入了校正函数,如式(5)所示,对 Max-log-MAP 算法做出了修正。

$$f_c(x)=\log(1+e^{-|x|}) \tag{5}$$

文献[5]将 Log-MAP 算法的校正函数进行折线拟合逼近,如表 1 所示。

表 1 拟合迭代函数

拟合	区 间		
幂次	[0,4]	[0,1.38]	[1.39,4]
一阶	$-0.191\ 2x+0.585\ 3$	$-0.337\ 5x+0.657\ 2$	$-0.073\ 2x+0.281\ 5$
二阶	$-0.063\ 7x^2-0.407x+0.673\ 6$	—	—
三阶	$-0.012\ 6x^3-0.133x^2-0.498\ 6x+0.692\ 5$	—	—

实验结果表明,折线逼近的 Log-MAP 算法简化了算法并降低了计算复杂度,而译码性能却没有随之降低,是一种实用有效的简化方案。本文利用这种译码算法,设计相应的 CPU 串行程序,对实际的卫星信号进行 Turbo 译码。但由于译码的速度依然无法达到 8 路实时的要求,本文利用 MIC 架构的高度并行计算能力对 CPU 译码程序进行并行加速。

2 MIC 处理器

2.1 MIC 硬件架构

MIC 芯片采用对称多处理结构结构,是一种多处理机硬件架构,即多个相同的处理器共享同一主存,并由同一个操作系统控制,可以带来很高的并行度。使用 MIC 架构在并行编程时,通常使用线程粒度的任务划分,兼容性、高度并行运算和编程可控方面更具优势。

如图 2 所示,MIC 协处理器上包含多个片上高速互联的 MIC 架构计算核心(MIC core),每个 Intel 架构微处理器核都能够执行 IA 指令(即 IA x86 指令),并且功能齐全、彼此独立。MIC 芯片拥有数十个核心,每个核可以支持 4 个硬件线程,每个线程可以作为真正的核心来运行。这些核心都能够执行统一的英特尔架构上的标准化 C、C++和 FROTRAN 源代码,从而通过共享的开发库和代码重用降低操作和编程开支。最新的 MIC 处理器搭载 72 个内核,提供了超过 3 tera FLOPS(每秒浮点运算)的双精度峰值。另外,MIC 芯片还具有 8 个内存控制器,共支持 16 个 GDDR5 传输通道。

2.2 MIC 与 GPU 对比

近年来,由于强大的浮点运算能力,GPU 也被广泛地

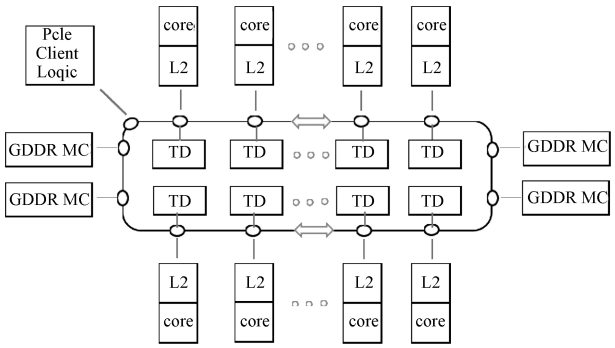

图 2　MIC 处理器硬件架构

应用在并行计算领域，利用 CUDA 编程语言使多个线程执行相同的指令流，从而获得高度并行能力。GPU 并行计算适合相互独立且没有依赖关系的大规模计算，Turbo 译码需要迭代运算，虽然已有人利用高度并行的 GPU 芯片实现 Turbo 译码[11-12]，但需要针对硬件设计新的程序结构，就工程应用而言，代码变动大，维护成本高。

MIC 与 GPU 都是相对于 CPU 具有较高性价比的高性能解决方案。表 2 所示为两者主要参数对比，其中最大的区别在于“核”的概念。GPU 中的核，以 CUDA 为例，是指一个流处理器(stream processor，SP)单元，只有密集计算的功能，而不具备逻辑计算功能[13]。而 MIC 的每个“核”，都可以简单看作是一个 x86 核心，其功能较 GPU 的核更强大。因此，MIC 的定位虽然也是协处理器，但可以作为独立的节点。MIC 处理器可以作为主机端也可以作为设备端，一般情况下使用 CPU＋MIC 主从模式[14]。在整个接收机系统中，MIC 作为协处理器专注译码模块，进一步释放 CPU 的计算资源。

表 2　MIC 与 GPU 主要参数对比

属性	NVIDA GPU	Intel MIC
单核	流处理器/CUDA core 每个核运行一个线程	X86 core 每个核上最多支持 4 个硬件线程
主频	接近 1 GHz	1.0～1.5 GHz
核心数	数十个到数千个	61～72 个
并行度	Grid、block、thread 多级并行 细粒度并行(线程数》核数)	线程＋向量化 线程数＜＝(核数－1)×4
编程语言	CUDA、OpenCL、OpenACC	OpenMP、OpenCL、Cilk、OpenACC
编程模式	Offload	Offload、Native、Symmetric

2.3　分析工具

Intel 为开发者配备了至强融核™协处理器控制面板，可以监视 MIC 卡的运行状态。除了温度、内存用量等常规信息，可提供平均内核利用率以及每个 MIC 核的运行状态。

图 3 所示包括系统即数据传输所消耗资源和用户即计算所消耗资源，所使用众核的卡的 61 个核中有 1 个被系统单独占用。在控制面板中可以开启 Turbo 设置选项，使 MIC 卡以最高频率持续运行，在一定程度上可以提高计算效率。

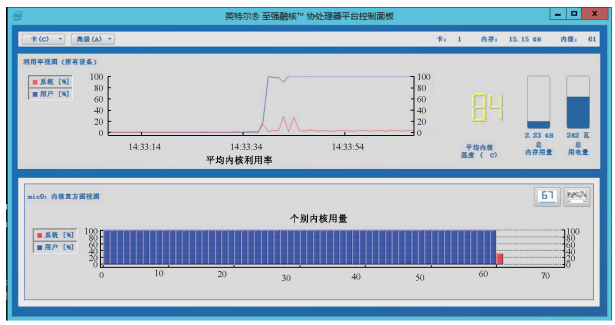

图 3　至强融核™协处理器控制面板

VTune Amplifier XE 是一款实用的多线程程序开发及性能优化指导工具[6]。如使用 VTune 可以查看串行或并行执行情况；多核的使用情况和利用率；定位程序中或者整个系统中时间消耗最多的函数或未有效利用资源的函数等。帮助开发者快速定位程序的应用瓶颈并做出调优决策，使计算资源利用率最大化，提升软件性能和扩展性。

另外，Intel C＋＋编译器提供了向量化报告和优化报告，在编译完成后可以显示出具体的代码块的优化情况。这些报告可以查明编译器没有添加向量化或能够进行优化的区域。它可能会修改代码，通过自动生成 SIMD 指令或向编译器传达更多信息，使应用生成的代码实现向量化或优化。

3　基于众核的并行加速

3.1　众核移植

MIC 程序不需要独立的编程语言，它是对传统 CPU 程序的扩展和移植。MIC 程序可以和 CPU 共用相同的执行代码，只需要添加相应的编译指导语句。众核移植中最重要的关键字是 offload，表示在其作用范围内的代码，将会在众核上执行。offload 语句一般作用在 CPU＋MIC 主从模式，即 CPU 作为主机 MIC 作为协处理器的模式。

在声明的时候将声明部分包括在以下两句代码中即可：

```
#pramga offload_attribute (push, target(mic))
#pramga offload_attribute (pop)
```

需要注意的是 const 类型和宏定义不需要此类声明；函数定义部分不需要再次声明；包含头文件部分不需要此类声明；所有调用的函数只要会在 MIC 上运行就必须进行声明。在众核移植过程中，由于 MIC 编译指导 offload 语

句和 windows.h 头文件不兼容,如果在 CPU 程序中调用了该头文件中的函数,则应尽量避免调用或将这些函数单独提取出来。

3.2 OpenMP 并行化

众核拥有数十个功能齐全的精简的 x86 核心,每个核心有 4 个"硬件线程",只有将程序并行化,才能体现出众核的性能优势[15]。众核程序的并行化一般有数据并行和任务并行两种并行方式。数据并行是将数据划分到不同的线程,共用同一执行代码的并行,线程的选择较为灵活,有很好的扩展性。MIC 是共享内存并行系统众核处理器[7],因此,数据并行是 MIC 上并行优化的很好的选择。

将需要高度并行的任务,在串行 CPU/MIC 程序上做并行化处理时,一般可选用 OpenMP。OpenMP 是一种 API,采用引语的方式对程序进行并行化,最常用的方式是对循环(例如 C/C++的 for 循环)进行并行化:

```
#pragma omp parallel for
```

其中 parallel 内的 for 循环区域就是并行区域,但需要注意的是循环变量不同取值时应相互独立,不应存在数据依赖和内存共用情况,否则无法实现并行化,程序还是按照串行执行。另外,可以在 parallel 后面添加属性指令来控制数据和并行区域,同时也可以选择线程的分配方式。

3.3 异步传输

众核移植时先数据 offload 到 MIC 中,然后将数据切分并采用基于数据的 MIC 多线程并行处理模式进行计算。变量和数组在 CPU 和 MIC 之前数据传输有 in、out、inout 和 nocopy 4 种方式。由于 offload 速度较慢,甚至有时数据传输时间和计算时间在同一个数量级,如果采用数据传输和计算顺序执行的方式,这样并不能体现出 MIC 并行计算的性能优势。

众核编程提供了异步传输机制,利用 signal 和 wait 异步传输语法,使 MIC 端无须等待 offload 语句返回,即可异步运行后续的代码。一般的异步传输机制为在数据传输的同时进行另一个数据块的 MIC 并行计算,即将数据传输时间"隐藏"在计算时间内,进一步提升整体运行效率。异步传输时把需要在 MIC 运行的数据(变量或指针)用以下代码包括在内:

```
#pramga offload target(mic:0) signal()
#pramga offload_wait target(mic:0) wait()
```

可以使其中的代码与后续的代码并行执行。例如将本次 MIC 计算、上一次结果输出(MIC ⟶CPU 内存⟶硬盘)、下一次数据传入(硬盘⟶CPU 内存⟶MIC)并行执行,如图 4 所示。在异步传输时需要注意:数据传输与计算需避免冲突,即如果对数据传输和计算并行,要确保计算部分不包含数据的传输。比如开辟内存时使用 nocopy,这样不会有数据的传输。

3.4 接收机设计

在如图 5 所示的通信接收机系统中,有 DDC、检测、同步、解调和译码等过程。为了实现接收机的整体设计,将每一个流程模块化,即将不同部分的可执行代码进行动态链接库(DLL)封装。动态链接库的导出有添加函数声明和模块定义两种方式。由于在导出过程中,使用 Intel C++编译器会出现 declspec(dllexport)语句与 MIC 指导语句 offload 冲突无法通过编译的情况,所以 MIC 程序一般都使用模块定义文件(.def)的动态链接库导出方式。这样不同的进程都可以调用同一个 DLL,而不需要改变其中的代码,便于大型工程的实现,有助于数据和资源的共享。

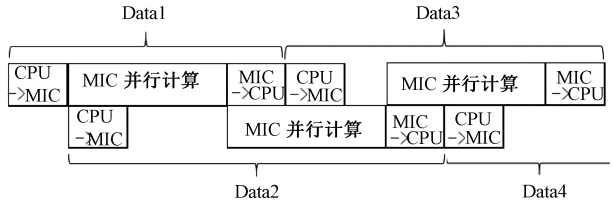

图 4 MIC 异步传输示意图

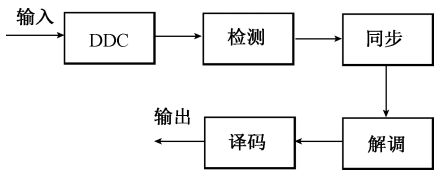

图 5 接收机设计流程

接收机设计中,使用进程间通信的方式实现不同流程之间的信号对接,如图 6 所示。例如,将 GPU 软解调之后的数据进行 MIC 译码,使用共享内存的方式进行数据传递,CPU 作为主机。将不同的流程视为不同的进程,调用同一 DLL 进行数据的传递。每个进程各自有不同的用户地址空间,数据和内存不能直接共享,所以进程之间要交换数据通过 DLL 中开辟的内存缓冲区,进程 A 把数据从用户空间拷到内存缓冲区,进程 B 再从内存缓冲区把数据读出。

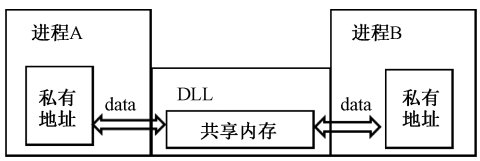

图 6 进程间通信示意图

4 实验结果与分析

实验平台采用的 CPU 是 Genuine Intel(R) CPU,MIC 卡型号为 Intel Xeon Phi Coprocessor 7120(61 核),编程模式采用 CPU+MIC 主从模式,实时要求为 8 路数据 307.76 s,实验平台参数配置如表 3 所示。

将 CPU 程序移植到众核,并利用 OpenMP 并行化,采用基于数据的 MIC 多线程(240 线程)并行处理模式。程序

表 3　实验平台参数配置

配置	参数
MIC	Intel Xeon Phi Coprocessor 7120，1.3～1.5 GHz，61 核
CPU	Genuine Intel(R) CPU，1.8 GHz
RAM	16 GB
操作系统	Windows Server2012R2
开发环境	Visual Studio 2012
编译器	Intel C++15.0

运行时，可以在控制面板看到 61 个核全部为运行的状态，而且由于开启最大 Turbo 频率选项，所有核均持续满负荷运行。将 CPU 程序众核移植和并行化后，译码时间由 30 min提升到 87 s 左右。虽然还未达到实时要求，但已经有将近 20 倍的提升，体现出众核并行加速的便捷性和高效性。

然后使用 VTune 分析工具观察资源消耗情况，在 Top Hotpots 页中直接看出耗时最多的函数为 4 个 Update 函数，大约占据了 95%的计算时间。这些更新函数中都调用了 LogMap 函数，占据了总时间的 48%。针对这些函数在代码层面进一步做优化处理并调整相应配置。

表 4 所示为 CPU 和 MIC 时间对比，包括开辟内存、数据传输和译码时间，10 次测试结果取平均值。待译码数据为文件大小为 252，116，992 字节，共 3 847 帧，每帧长 32 768，读入数据类型为 short 类型，每帧的实际采样时间为 80 ms。从表 4 中可以看出，8 路数据 MIC 并行总时长为 254.08 s，满足实时要求的 307.76 s。

表 4　CPU 和 MIC 时间对比

数据	帧数	帧长	时间/s
1 路 CPU 串行	3 847	32 768	1 800(约)
1 路 CPU 并行	3 847	32 768	347.89
1 路 MIC 并行	3 847	32 768	36.62
8 路 MIC 并行	30 776	32 768	254.08

由于 8 路数据量大，相比只运行 1 路数据，减少了等待迭代次数较多的线程的平均时间，平均每路为 31.76 s。但不是数据量越大越好，应该综合考虑内存占用和计算资源。相比原始版本 1 路数据 CPU 串行译码时长约 30 min，众核并行提升近 60 倍；相比优化后最终 CPU(16 核)1 路数据并行时间 347.89 s，众核并行也有十多倍的提升。

5　结　　论

本文针对实际卫星信号，使用了折线逼近的 Log-MAP 算法对 Turbo 码进行译码，这种算法在简化算法和降低复杂度的同时几乎没有降低译码性能，是一种实用有效的简化方案。为了在工程实现中对多路卫星信号进行实时译码，本文利用 Intel 公司的 MIC 协处理器，采用基于数据的多线程并行处理模式对的 CPU 串行程序进行众核移植和 OpenMP 并行化，编程模式为 CPU+MIC 主从模式。然后使用 VTune 分析工具观察资源消耗情况，针对性地在代码层面做了进一步优化处理。相比 CPU 串行译码，MIC 并行译码在计算速度上有将近 60 倍的提升，并且可以实现 8 路突发信号的实时处理。

参考文献

[1] BERROU C. Near Shannon limit error-correction coding and decoding[J]. Proc Icc Geneve Switzerland May, 1993.

[2] BENEDETTO S, MONTORSI G, DIVSALAR D, et al. Soft-output decoding algorithms in iterative decoding of turbo codes[J]. The Telecommunications and Data Acquisition Progress Report, 1995: 63-87.

[3] BAUCH G, KHORRAM H, HAGENAUER J. Iterative equalization and decoding in mobile communications systems [C]. European Personal Mobile Communications Conference, 1997: 307-312.

[4] BENEDETTO S, DIVSALAR D, MONTORSI G, et al. A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes[J]. Telecommunications & Data Acquisition Progress Report, 1996, 127: 1-20.

[5] 赵旦峰，张英，陶磊岩. Turbo 译码中的 Log-MAP 折线逼近法[J]. 吉林大学学报(工学版)，2009(5)：1364-1368.

[6] 王恩东. MIC 高性能计算编程指南[M]. 北京：中国水利水电出版社，2012.

[7] JEFFERS J, REINDERS J. Intel xeon phi coprocessor high performance programming [M]. Morgan Kaufmann Publishers Inc. 2013.

[8] STEPHEN BLAIR-CHAPPELL, ANDREW STOKES. Intel Parallel Studio 环境下的并行程序设计[M]. 北京：清华大学出版社，2013.

[9] CUI Z P, GAO J, DOU G Q. Research and analysis of Turbo-code decoding[J]. Communications Technology, 2016.

[10] 陶李. Turbo 编译码方案研究与实现[D]. 南京：东南大学，2017.

[11] LI A, MAUNDER R G, ALHASHIMI B, et al. Implementation of a fully-parallel Turbo decoder on a general-purpose graphics processing unit [J]. IEEE Access, 2016, 4: 5624-5639.

[12] 蒋俊，杨全，刘凯，等. 基于 CUDA 的多路突发信号检测[J]. 工业控制计算机，2018，31(5)：13-15，17.

[13] 巨涛，朱正东，董小社. 异构众核系统及其编程模型与性能优化技术研究综述[J]. 电子学报，2015，43(1)：111-119.

[14] 陈钢.众核 GPU 体系结构相关技术研究[D].上海:复旦大学,2011.
[15] 吕慧伟,程元,白露,等.众核处理器和众核集群的并行模拟[J].计算机研究与发展,2013,50(5):1110-1117.

作者简介

赵梦伟,硕士研究生,主要研究方向为 Turbo 码译码、Multi-h CPM 调制识别。
E-mail:zhaomengwei@shu.edu.cn

陈永锋,工程师,主要研究方向为卫星移动通信信号处理,软件无线电。

刘凯,副教授,主要研究方向为雷达信号处理、通信信号处理、免携带室内定位技术等。

孙超群,硕士研究生,主要研究方向为译码过程中的参数估计和码型识别。

Mobile Industrial Robots 宣布与佛吉亚展开战略合作,助其优化全球内部物流

自主移动机器人行业的先行者 Mobile Industrial Robots (MiR)宣布与全球领先的汽车技术公司佛吉亚 (Faurecia) 展开全球战略合作,为佛吉亚全球制造工厂部署自主移动机器人,助其将整个企业的自动化水平提升到一个全新高度。

多年来,佛吉亚一直在研究和测试不同的自动化内部运输解决方案,以优化生产力和内部工作流程,进而提升竞争优势。通过与 MiR 建立合作伙伴关系,佛吉亚将得以重新规划全球生产基地的内部物流,可借助灵活的自主移动机器人提高物流流程效率。

佛吉亚绿动智行系统事业部的供应链与数字化企业副总裁 Eric Moreau 表示:“我们之所以与 MiR 建立战略合作伙伴关系,是因为他们拥有足够的技术能力和深入的专业知识,可以协助我们精简和优化物流运营。MiR 的技术得到了广泛认可,许多大型跨国企业都在使用他们的机器人来提高物流效率。虽然从整体上来说,我们企业的自动化水平较高,但物料搬运始终是一大挑战。我们认为在物料搬运方面实施自动化存在巨大潜力。我们经常需要调整工厂的设施布局以满足众多的小批量生产需求。这种调整需要高度灵活的内部物流系统提供相应支持。而 MiR 正好可以满足我们的需求,MiR 自主移动机器人可轻松更新内部地图和导航系统,不需要任何指导。”

MiR 提供一系列能与人类协同工作的自主移动机器人,将人类从体力劳动中解放出来,去执行更富有价值的任务。MiR 移动机器人非常人性化,操作无需编程经验。操作员可轻松通过机器人界面与机器人交互,这个界面可与智能手机、平板电脑或个人电脑相连,操作员只需按一下按钮便可使机器人开始执行任务。此外 MiR 移动机器人操作简单,不仅可简化部署,还能降低总体成本。

客户也可以根据自身需求为 MiR 机器人定制不同的顶部模块。MiR 移动机器人的多功能性是佛吉亚选择与 MiR 合作的另一个原因。Eric Moreau 补充道:“MiR 移动机器人不仅能够运输货物,而且能够结合装卸等其他运输流程,提高整条生产线的总体效率。这些移动协作机器人非常灵活,目前我们已经确定了几种不同的应用方式,以使用 MiR 移动机器人来执行生产线内以及生产车间和仓库之间单调的货物运输任务,从而提高生产效率。”

物流机器人正在崛起

本次新合作将助力佛吉亚优化多项运营流程,此外,根据国际机器人联盟(IFR)的最新数据显示,佛吉亚并非唯一一家看到内部物流自动化潜力的公司。2017 年所售出的专业用途服务机器人中就有 63%是物流机器人,且据 IFR 预测,在 2018 年至 2021 年间,物流机器人的销量将增加至 60 万台左右。

MiR 首席执行官 Thomas Visti 表示:“我们很高兴能够与佛吉亚达成此次合作,通过部署灵活、安全的协作式移动机器人帮助佛吉亚实现生产目标。总体而言,相较其他行业,汽车行业使用自动化技术的速度更超前,我们看到了汽车行业的巨大潜力。同时当今的工厂设置往往灵活多变,人员、设备、托盘和其他障碍物都随时可能出现在曾经的开放通道区域。我们的移动机器人特别适合这种不断变化的环境。借助我们机器人的协作性与自主导航能力,自动化物料运输不仅变得灵活和易于调整,且不会产生额外费用或出现工作流程中断的情况。”

MiR 拥有全球分销网络和众多当地办事处,能够在全球范围内为佛吉亚提供支持。欲了解更多相关信息,请访问 www.mobile-industrial-robots.com。