A 34.7-mW Quad-Core MIQP Solver Processor for Robot Control

Hiroki Noguchi, Junichi Tani, Yusuke Shimai, Masanori Nishino,
Shintaro Izumi, Hiroshi Kawaguchi, and Masahiko Yoshimoto
Kobe University, Kobe, 657-8501 Japan
h-nog@cs28.cs.kobe-u.ac.jp

Abstract— **We propose a quad-core mixed integer quadric programming (MIQP) solver processor. The MIQP solver is applicable to hybrid control systems including real-time control robotics. Using multi-core architecture, fixed-point calculations, and branch-and-bound method with high-dispersion performance while processing a 50-variable problem, our design achieves 34.7-mW operation at a frequency of 52 MHz in measurement results, although a core 2 duo PC requires 3.16 GHz to solve it as rapidly.**

Keywords—MIQP, multi core, real-time hybrid control

I. Introduction

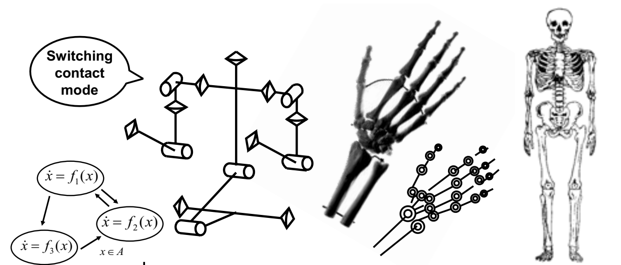
Recently, control for robots has attracted great interest. Robots with improved control are expected to enrich human life. Many studies have examined them; hybrid system control by solving mixed integer quadratic programming (MIQP) is one such system. Hybrid system control is applicable to various systems. It is therefore possible to control a fuel cell reactor and gas turbine [1], multi-vehicle pass planning [2], and robot manipulation [3]. An earlier report [4] presented derivation of the hybrid system for solving an MIQP problem. However, in general, it takes a long time to solve an MIQP problem. Consequently, a high-speed MIQP solver is sought for real-time robotic control.

We specifically examine future dexterous robotics control using a hybrid system. Fig. 1 portrays examples of the multiple degree-of-freedom (DoF) robotics. These are categorized as complex hybrid systems with continuous time and discrete events, which realizes dexterous operation. Using the mixed logical dynamical (MLD) system model, we can formulate this hybrid system control problem as an MIQP problem (Eq. 1) [5]. The mathematical definition of the MIQP problem is as shown below.

$$\begin{aligned}
&\underset{\mathbf{x}}{\text{minimize}}\ \ f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x} + \mathbf{g}^{\mathrm{T}}\mathbf{x}\\
&\text{subject to}\, \mathbf{A}_{\mathbf{E}}\mathbf{x} = \mathbf{b}_{\mathbf{E}}\\
&\qquad\qquad \mathbf{A}_{\mathbf{I}}\mathbf{x} \le \mathbf{b}_{\mathbf{I}}\\
&\qquad \text{where}\, \mathbf{x} \in \mathbf{R}^{n_c} \times \mathbf{Z}^{n_b}, \mathbf{H} \in \mathbf{R}^{n \times n}\\
&\qquad\qquad \mathbf{A}_{\mathbf{E}} \in \mathbf{R}^{m_E \times n}, \mathbf{A}_{\mathbf{I}} \in \mathbf{R}^{m_I \times n}\\
&\qquad\qquad \mathbf{b}_{\mathbf{E}} \in \mathbf{R}^{m_E}, \mathbf{b}_{\mathbf{I}} \in \mathbf{R}^{m_I}.
\end{aligned} \tag{1}$$

Variables	Description
n	$n_c + n_b$
n_b	# of integer variables
n_c	# of continuous variables
$\mathbf{H}$	A symmetric positive definite matrix
$\mathbf{g},\mathbf{b}_E,\mathbf{b}_I,\mathbf{A}_E,\mathbf{A}_I$	Constant vectors and matrices
$\mathbf{R}$	The set of real numbers
$\mathbf{Z}$	The set of integers
$f(\mathbf{x})$	The objective function

Therein, $f(x)$ is called an objective function; the MIQP has a linear equality constraint and an inequality constraint. If x satisfies all constraints, then it is called a feasible point. The integer entries in x are called integer variables. The programming problem is classified as a binary programming (BP) problem if every integer variable has either 0 or 1. The MIQP problem is, however, an NP-hard problem. In practice, the computational complexity increases drastically with the number of the variables (presented in Fig. 2). In addition, the number of DoFs is correlated to the number of variables in the MIQP problem. For the most extreme example, shown in Fig. 1, the number of human DoFs is about 400, which engenders more than 2,400 variables in MIQP (each joint is assumed to have six DoFs). The computational cost of the MIQP problem is quite high, but the hybrid system control with the MIQP solver gives mobile robotics intelligence. Complex planning and movements must be supported online for real time. Consequently, low-power calculations are necessary: a high-speed but low-power MIQP solver is important for mobile robot applications. Such a supreme MIQP solver provides a new paradigm for modeling, planning, and controlling robots.



	Multi contact robot	Forceps manipulator	Biped walker
Degree of freedom (DoF)	15 ~	25 ~	400 ~
# of variables in MIQP problem	90 ~	150 ~ 300	2,400 ~

Fig. 1. Examples of multi-DoF robotics and normalized computational amounts.

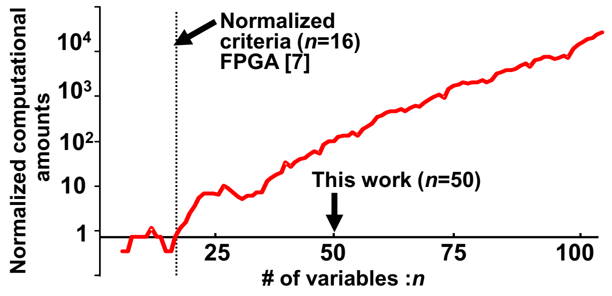


Fig. 2. Examples of multi-DoF robotics and normalized computational amounts.

II. Distributed Algorithm for MIQP Solver

We divide a problem into child sub-problems by branching. Then we solve the child sub-problems. To distribute computations to several cores equally and to solve an MIQP problem efficiently, we must first schedule the assignment of quadratic-programming (QP) sub-problems. We apply the branch-and-bound method to the MIQP solver to produce a distribution. The branch-and-bound method has been used to solve mixed integer programming problems in a grid-computing system [6]. Consequently, the MIQP problem is divisible into QP sub-problems. We can achieve decentralization of the computation. The number of generated QP sub-problems depends on the MIQP problem. Fig. 3 presents an example of a QP sub-problem grouping. Each core is assigned adaptively to one group.

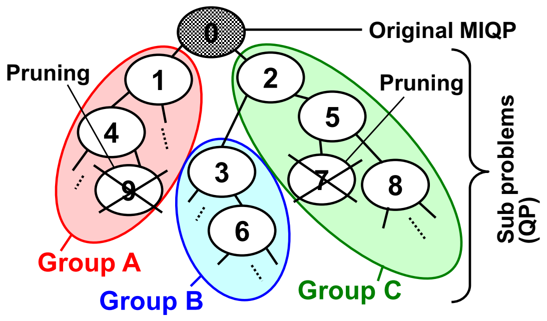


Fig. 3. Example of the branch-and-bound search tree.

III. Core Architecture

We have already implemented the MIQP solver that can solve problems with 16 variables onto the FPGA [7]. In contrast, the designed quad-core MIQP solver processor can solve problems with 50 variables. Furthermore, compared with the implementation onto the FPGA, the bit-width of fraction of fixed-point can be increased from 22 to 26. In doing so, the error margin can be decreased for judgment of whether the target variable is an integer or not. Each core solves different QP sub-problems based on the dual active set method [8]. The input to each core is the QP sub-problem (= 327.2 kb); the output is its solution (= 2 kb). Each core has a dedicated SRAM and multiply and accumulation (MAC) unit, and various other arithmetic units including a distance calculator, a square root calculator, and fixed-point divider for adding and deleting the active constraints. All calculating units are implemented with pipelining and 26-b fixed-point computing to reduce the total computational cycles and amounts. Among them, the fixed-point divider has a critical path. To guarantee the computation at 120-MHz operation frequency, the multi-cycled CLKs are set to the fixed-point divider (Fig. 5). The multi-cycled fixed-point divider needs a 5-cycle delay at every calculation, but it maintains a sufficient-speed operation capability. Actually, 100-MHz operation enables the MIQP processor to solve the 50-variable MIQP problem, which is targeted according to omnidirectional mobile robotics, and which can be controlled at every 100 ms.

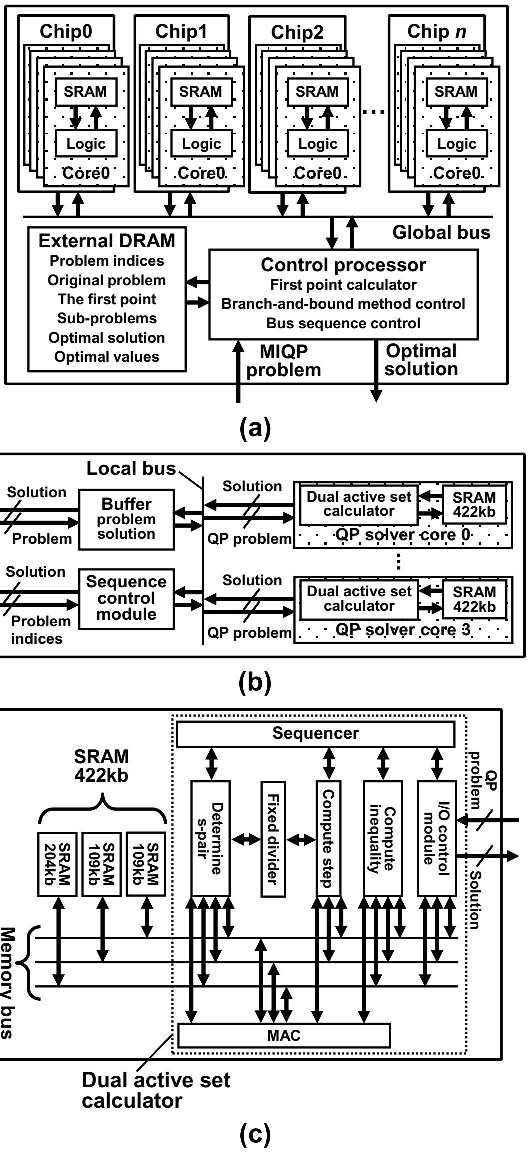


Fig. 4. Block diagrams of (a) multi-chip MIQP solver, (b) MIQP solver chip, and (c) QP solver core.

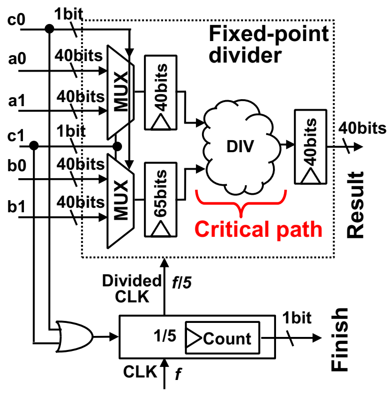


Fig. 5. Multi-cycled fixed-point divider.

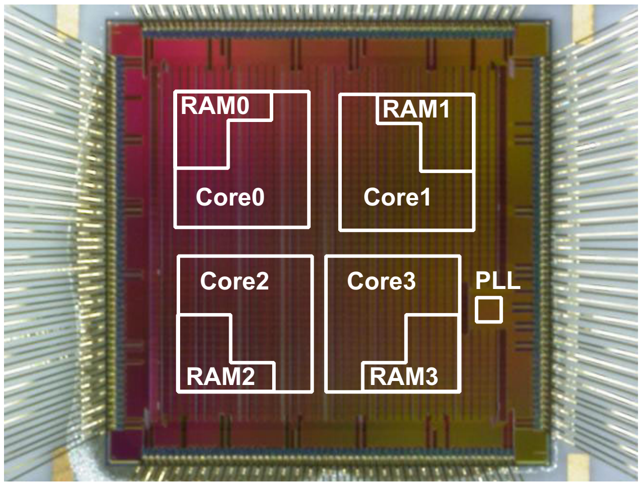


Fig. 6. Chip micrograph of a Quad-core MIQP solver processor designed with 65-nm technology.

IV. Multi-Cored Flexibility

To parallelize the QP solver cores efficiently, their idle cycles must be reduced. Each QP solver core is given a different QP problem through the external control processor (portrayed in Fig. 4). Each core solves it asynchronously. The solutions are sent to external processor; then it updates the optimal solution and the optimal values in external DRAM. Simultaneously, child sub-problems that must be solved later are generated and saved in DRAM. The total communications traffic is a sum of the input QP problems and the results of their problems. In our 65-nm design, we implemented quad cores on a chip. In reality, the core number depends on an area constraint and the communication traffic limits. As portrayed in Fig. 4, all chips that have quad cores connect to an external branch-and-bound module through a global bus; QP cores can solve an MIQP problem all together.

V. Hardware Results

We designed the 65-nm quad-core MIQP solver processor test chip depicted in Fig. 6. This chip is optimized for a QP problem with 50 entries of variables and runs at a maximum operating frequency of 120 MHz. The core size is 1.1 × 1.2 mm^2. Each core has 422 kbits SRAM as a local working memory. The total size, including the quad cores and PLL circuit, is 3.2 × 3.2 mm^2; the number of logic gates is 1.36 M. Fig. 7 presents the evaluation environment—the FPGA board, LSI chip, external DRAM, and power supplies—used for the experiments described herein. Fig. 8 shows the measurement power consumptions at several operation frequencies. Fig. 9 shows the required frequencies, which are times that the MIQP solver would finish calculations before PCs need to process them: (core 2 duo 3.16 GHz (Intel Corp.) and 3 GB memory). Table I presents specifications of each implementation. When considering the 16-variable MIQP problem, the single-chip implementation indeed yields over a tenth higher performance than the FPGA implementation. Even with a much more complicated problem, a 50-variable MIQP problem, the 3-chip implementation achieves 33 MHz operation, which is a 50.7% reduction of the required frequency. To advance that reduction, more MIQP solver cores are effective.

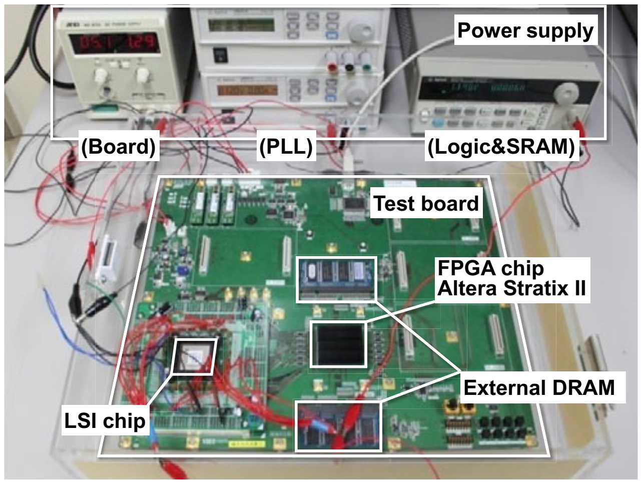


Fig. 7.Evaluation environments for a test chip using an FPGA, dual external-DRAMs, and power supplies.

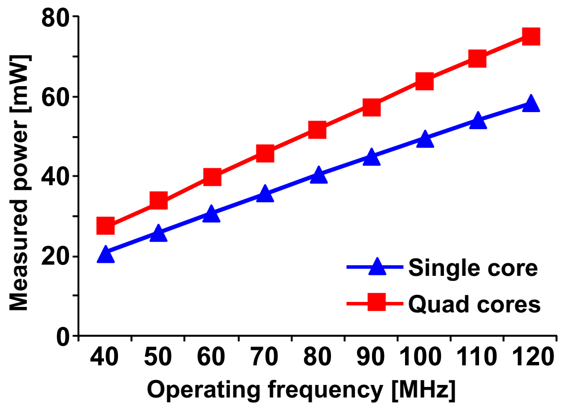


Fig. 8. Measurement results of frequency vs. power consumption.

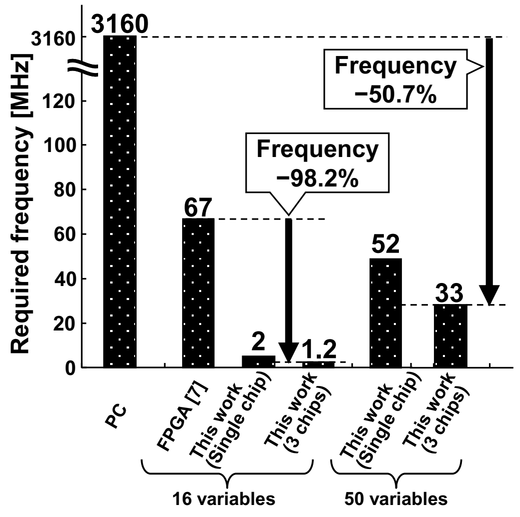


Fig. 9. Frequency and power comparisons.

Fig. 10 presents a comparison of the power consumption values shown for a PC, FPGA, single chip, and three-chip implementations. Operating frequencies are set, according to Fig. 9. When we consider a 50-variable problem, the single-chip implementation achieves 35-mW operation in actual measurements, which represents a 99% power reduction compared with the FPGA implementation.

TABLE I. MIQP solver specifications

	PC	FPGA [7]	This work
# of variables	50	16	50
# of equality constraints	10	16	10
# of inequality constraints	100	32	100
# of active set nodes	50	32	50
The bit width of the fraction	Float	22	26
The bit width of the integer	Float	14	14
Maximum operating frequency	3.16GHz	67MHz	120MHz

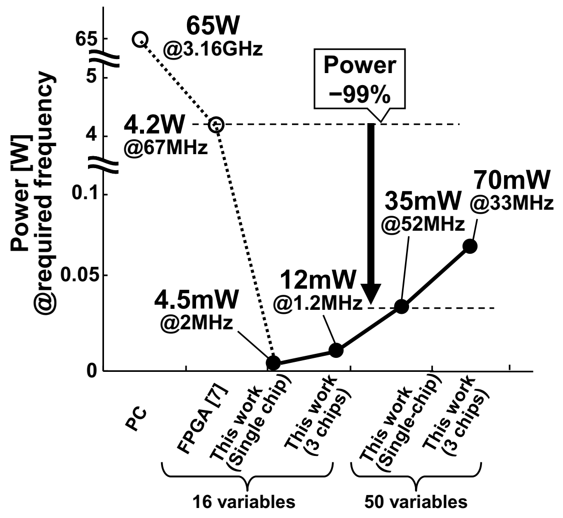


Fig. 10. Frequency and power comparisons.

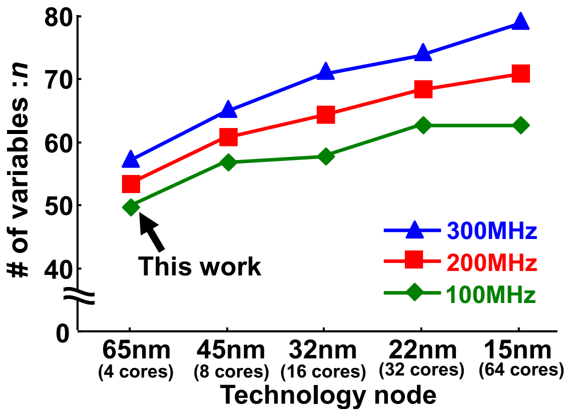


Fig. 11. Future performance prediction with a single chip.

VI. Prediction

Fig. 11 shows the scale prediction of MIQP problems that is solvable within 100 ms using the future process technology on a single chip. The gate transistor size was shrunk, and the number of cores and the SRAM memory size in the 3.2 × 3.2 mm^2 area are considered. At 22-nm and later nodes, the over-70-variable problem will be handled, which means that advanced robotics with higher DoFs can be controlled with a single chip. When involved with our proposed multi-chip solution, more dexterous robotic control can be achieved in the same technology node.

VII. Summary

The branch-and-bound method is applied to the multi-core MIQP solver. A 26-b fixed-point computing is implemented to reduce the total computational amounts. The multi-cycled fixed-point divider is adopted to eliminate the critical-path timing problem. Using these conditions and arrangements, our designed processor achieves 34.7 mW at a 52-MHz operation. At a frequency of 100 MHz, our quad-core processor can achieve almost twice faster calculation than a PC with a 50-variable MIQP problem. We also propose a multi-chip solution, which increases the number of MIQP problems that are solvable simultaneously.

Acknowledgments

The VLSI chip in this study was fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with STARC, e-Shuttle, Inc., and Fujitsu Ltd.

The authors thank Prof. Zhi-Wei Luo for valuable discussion related to hybrid system control and predictive control using the MIQP problem.

References

[1] P.Costamagna, L. Magistri, and A. F. Massardo, “Design and part-load performance of a hybrid system based on a solid oxide fuel cell reactor and a micro gas turbine,” Journal of Power Sources 96 (2001) 352-368.

[2] M. Mukai, T. Azuma, and M. Fujita, “A Collision avoidance Control for Multi-Vehicle Using PWA/MLD Hybrid System Representation,” Proc. of the IEEE Conference on Control Applications (ICCA), pp. 872-877, Sep. 2004.

[3] D. Dimitrov, P. Wieber, O. Stasse, H. J. Ferreau, and H. Diedam, “An Optimized Linear Model Predictive Control Solver for Online Walking Motion Generation,” Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1266-1271, May 2009.

[4] D. Axehill, “Applications of Integer Quadratic Programming in Control and Communication,” Linköping Studies in Science and Technology, thesis no. 1218, pp. 9-45, 2005.

[5] Y. Yin, S. Hosoe, and Z. Luo, “A mixed logic dynamical modeling formulation and optimal control of intelligent robots,” Optimization and Engineering (Springer), Vol. 8, No. 3, pp. 321-340, Sep. 2007.

[6] K. Aida, and T. Osumi, “A case study in running a parallel branch and bound application on the grid,” Proc. of the IEEE Symposium on Applications and the Internet, pp. 164-173, Jan. 2005.

[7] Y. Shimai, J. Tani, H. Noguchi, H. Kawaguchi, and M. Yoshimoto, “FPGA implementation of mixed integer quadratic programming solver for mobile robot control,” Proc. of IEEE International Conference on Field-Programmable Technology (FPT), pp. 447-450, Dec. 2009.

[8] D. Goldfarb, and A. Idnani, “A numerically stable dual method by solving strictly convex quadratic programs,” Mathematical Programming (Springer), Vol. 27, pp. 1-33, 1983.