Second Prize

Black Box for Robot Manipulation

Institution: **Hanyang University, Seoul National University, Yonsei University**

Participants: **Kim Hyong Jun, Ahn Ho Seok, Baek Young Min, Sa In Kyu**

Design Introduction

Today's robot manipulators need ever more precise control capability. There are two important prerequisites for precise control: the number of fingers in the manipulator and accessable hardware/software control. In the first case, the number of system I/O pins is determined by the number of fingers that must be controlled. In the second case, the developer should be able to easily design the system, taking into account the system requirements.

Conventional systems use several CPUs because it is difficult to control all motors in the system using a single CPU. Each CPU has control over a servo motor, and the CPUs communicate with each other to simultaneously control the manipulator. This methodology can have problems with unsynchronized movements or time delays. In contrast, FPGAs have many I/O pins, and the developer can design the system to fit in one FPGA. This FPGA can then control all of the motors by distributing the servo-motor dedicated processors. We chose to use an Altera Cyclone® II FPGA, which gave us easy, natural control over the manipulator. Our design implemented the Nios® II processor with custom instructions, which helped achieve faster processing capability and precise system control, providing a robust FPGA solution. Our ultimate goal was to develop the control module using this hardware/software strategy.

The robot manipulator has two arms and uses the Development and Education (DE2) board, which contains the Altera Cyclone II FPGA. The application used the µC/OS-II RTOS and custom instructions. Communication to and from this control module is performed via an external Ethernet link.

Function Description

In a hardware/software co-designed system, the interaction of the software and hardware is critical. The hardware logic is implemented in an FPGA and tends to be fast and robust. The software logic is comparatively slow due to its sequential flow, instructions, and operand fetch stages that involve read/write memory access. Software, however, is significantly easier to develop and modify because it resides in memory and not in the dedicated gates of the switch fabric.

The black box for robot manipulation (BRM) operates via a standalone controller, which controls the DC and servo motors, and has communications ability via Ethernet and serial peripheral interface (SPI) connections. Our goal was to make a high-performance, usable system. The project involved two phases.

- In phase one, we developed and operated the manipulator like a microcontroller unit (MCU) robot system. Hardware implemention included a DC motor controller and encoder with programmable I/O (PIO), a servo motor controller with a pulse width modulator (PWM), and a hand module controller using the SPI master peripheral, and an Ethernet peripheral with the dm9000 Ethernet device.

  In the software phase, we used the μC/OS-II RTOS running on the Nios II processor. We created the design using SOPC Builder, and implemented a lightweight TCP/IP (LWIP) stack, and defined the Ethernet and SPI communication packet, and added controls for the DC motor, DC motor encoder, and servo motor at the RTOS task level.

- In phase two, we planned to use PWM generator and pid calculation using the Nios II C-to-Hardware Acceleration Compiler (C2H Compiler) and custom instructions. An important issue is whether the project can be used for home and space automation.

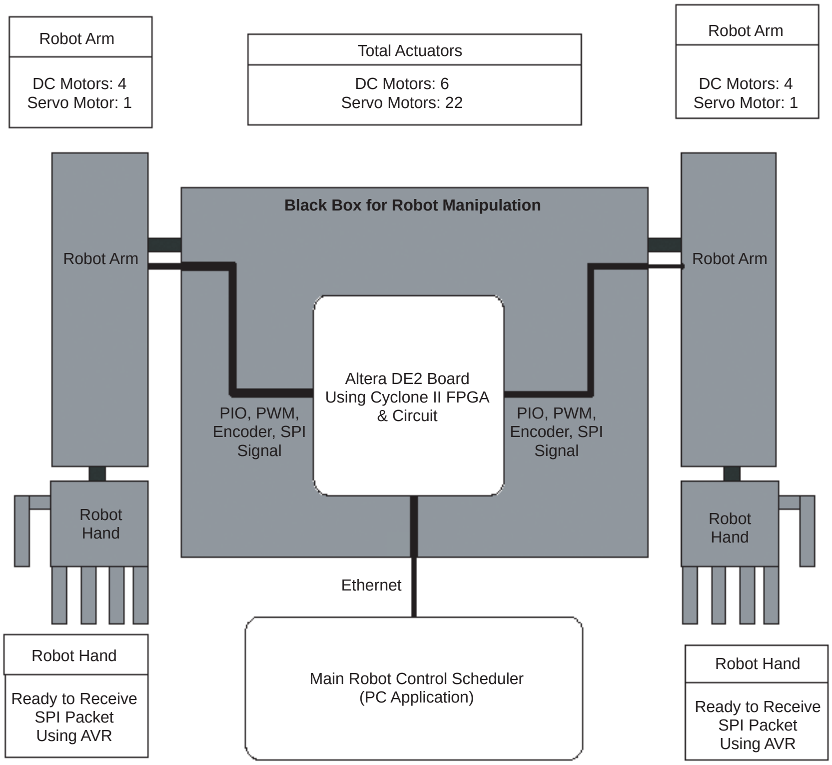
Performance Parameters

We used the Cyclone II FPGA on the Altera DE2 board, and ran the μC/OS-II RTOS on the Nios II processor. With this setup, it took one second to send a command four times to all eight DC motors, and one servo motor. It took an additional one second for SPI communication with the hand module and external Ethernet device. If a command is sent more than four times, the BRM system issues a fault indication.

Design Architecture

Figure 1 shows the BRM’s hardware block diagram.

Figure 1. Robot Manipulator Block Diagram



The external device, which is controlled by the BRM, has five components:

- DC motor
- DC motor encoder
- Servo motor
- Hand module with SPI communication
- An Ethernet device (dm9000)

Three joints in the arm use a DC motor. One wrist joint uses a servo motor. The whole system has eight joints that use DC motors, and two that use servo motors. At the end of the arm is the hand module. The AVR MCU controls it with 4 DC motors and four servo motors via SPI slave communication. Figure 2 shows a computer-aided design (CAD) drawing of the robot arm.

Figure 2. Robot Arm CAD Drawing

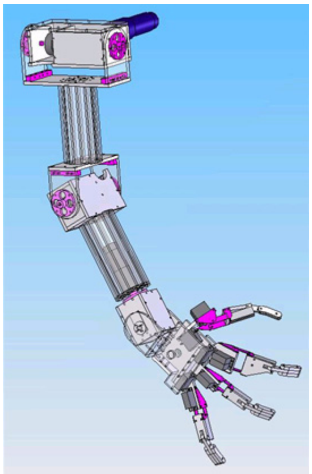
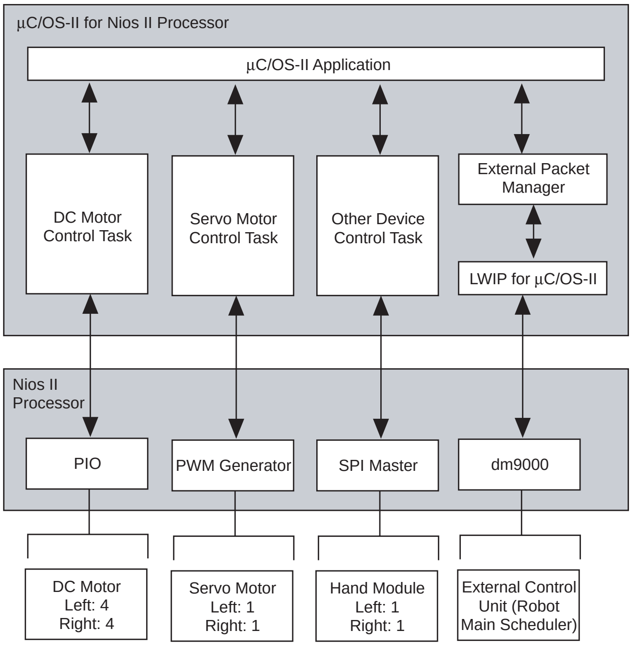


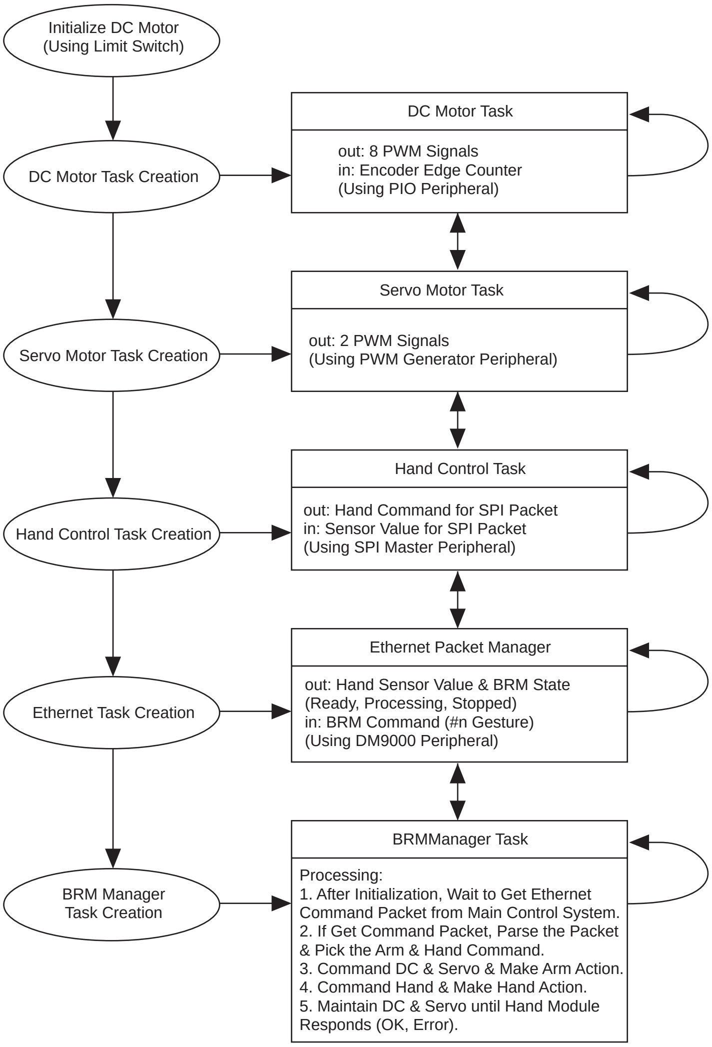
Figure 3 shows the software system block diagram.

Figure 3. System Block Diagram



The software does not have a general device driver. To maintain consistency with the RTOS, we assigned the task to the I/O subsystem. Figure 4 shows the software flow chart.

Figure 4. Software Flow Chart



Design Description

We used the Altera DE2 board, which has a Cyclone II device, to implement the design. The main clock frequency is 100 MHz, and we used a total of 39 external general-purpose I/O (GPIO) pins for the DC motor, DC motor encoder, servo motor, and the hand control of the SPI master. We modified the DE2_WEB source code, and did not change the name of the top-level design. The design has the following modules:

- *DC_MOTOR_CONTROL*—We implemented this module using the PIO output-only peripheral. It has a 32-bit data register, which we needed because one DC motor requires two signals (positive and negative).

- *DC_MOTOR_ENCODER*—We implemented this module using the PIO input-only peripheral. It catches the rising edge of the encoder pulse, and it has a 32-bit data register. The DC motor encoder requires two signals (for the forward and reverse motor states).

- *SERVO_MOTOR_CONTROL*—This module uses the PIO output-only peripheral. It has a two-bit data register because one servo motor requires one PWM signal. The operating system (OS) task generates the PWM pulse.

- *HAND_CONTROL*—The module uses the SPI master peripheral. The hand module acts in SPI slave mode. It has a clock of 11/128 MHz.

We built the Nios II processor using SOPC Builder, as shown in Figure 5.

Figure 5. Nios II implementation in SOPC Builder

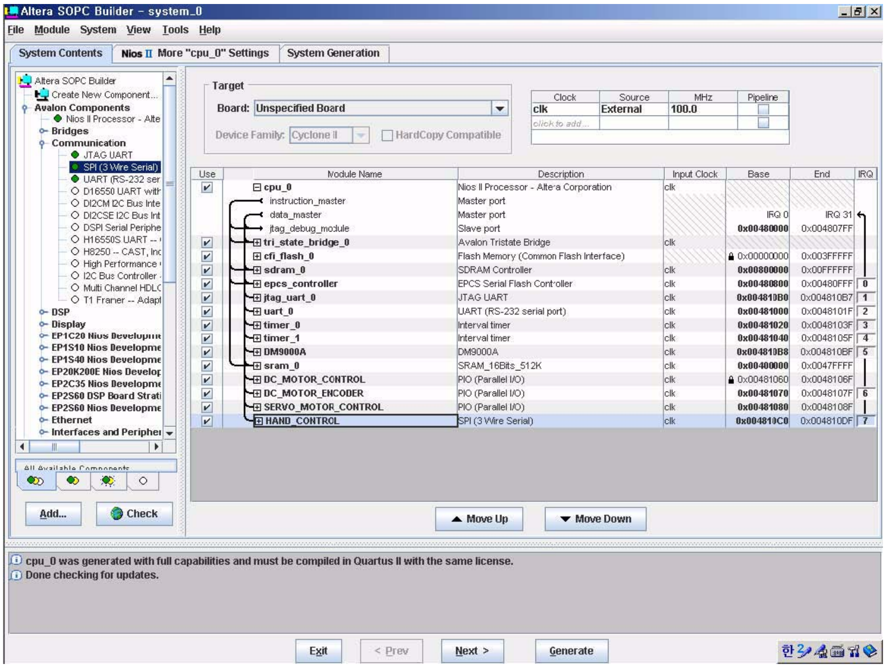
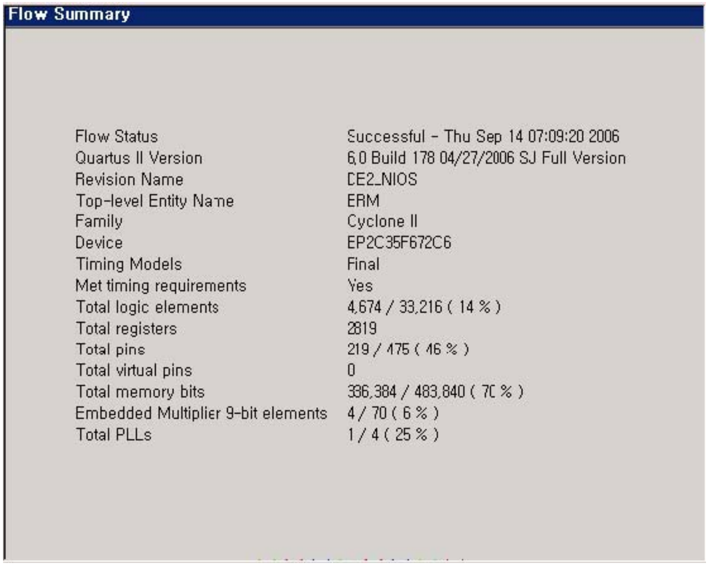


Figure 6 shows the Quartus® II-generated project summary.

Figure 6. Nios II Implementation in the Quartus II Software



Flow Summary

Flow Status	Successful - Thu Sep 14 07:09:20 2006
Quartus II Version	6.0 Build 178 04/27/2006 SJ Full Version
Revision Name	DE2_NIOS
Top-level Entity Name	ERM
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	4,674 / 33,216 (14 %)
Total registers	2819
Total pins	219 / 475 (46 %)
Total virtual pins	0
Total memory bits	336,384 / 483,840 (70 %)
Embedded Multiplier 9-bit elements	4 / 70 (6 %)
Total PLLs	1 / 4 (25 %)

Figures 7 and 8 show the circuit and hardware that are required to connect the external devices, such as the DC motor, encoder, and SPI slave module.

Figure 7. Completed Circuit

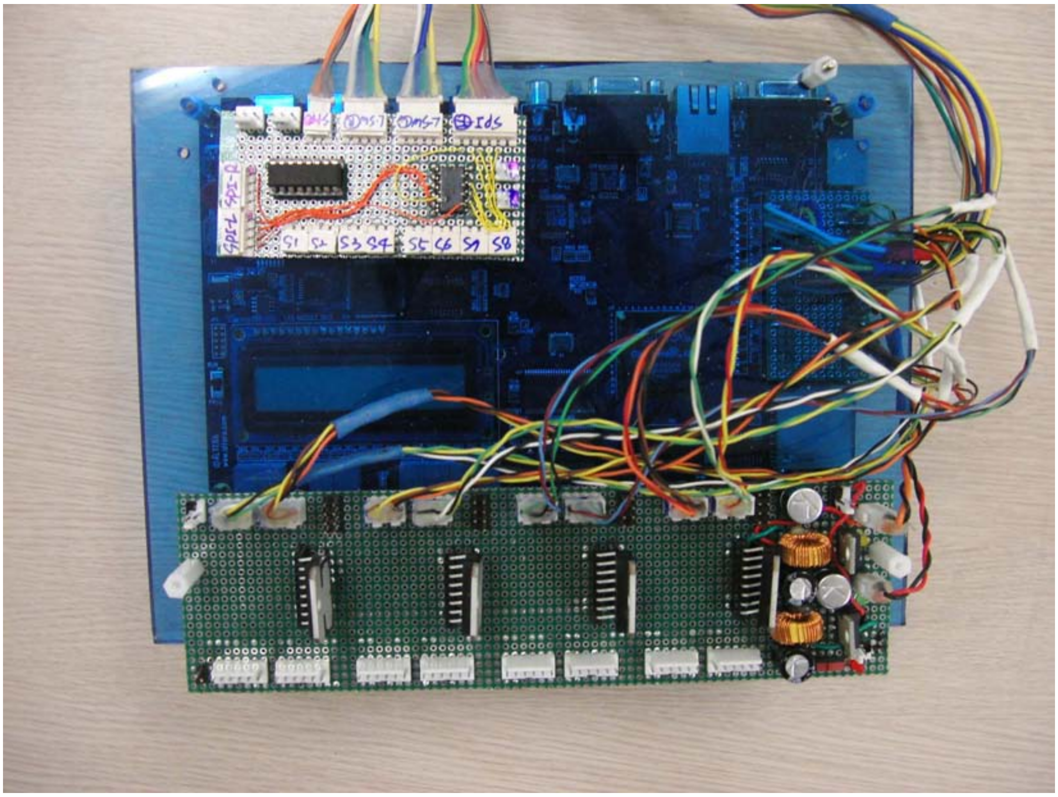
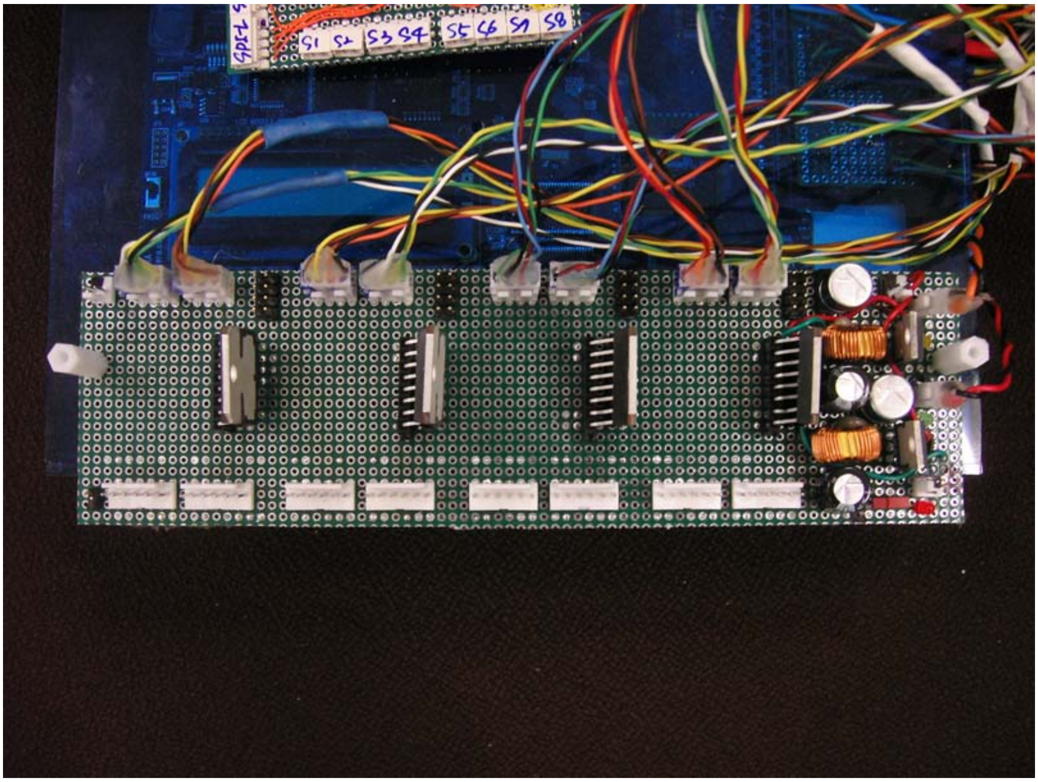


Figure 8. DC Motor Driver and Power Circuit



The DC motor driver circuit has four L298 driver devices, and the power circuit has uses the LM2576-3.3 device, which operates at 5 volts. Figure 9 shows the signal divider circuit and external DE2 header.

Figure 9. Signal Divider Circuit and DE2 External GPIO Header

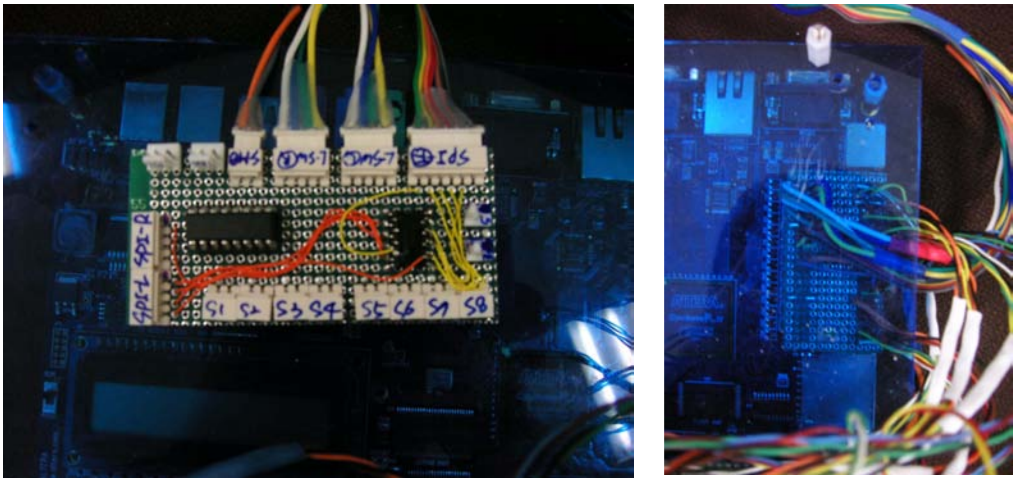
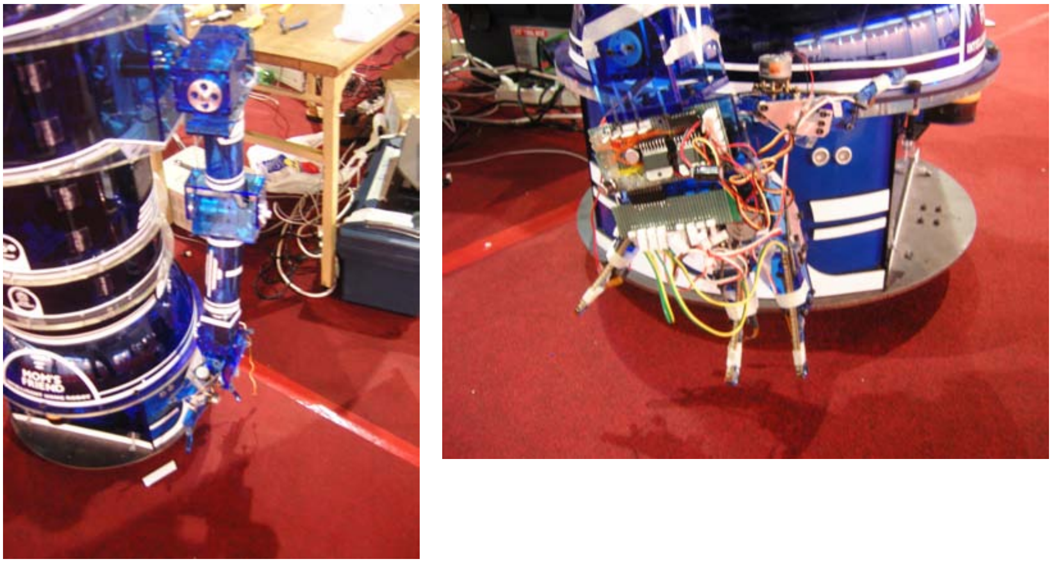


Figure 10 shows our testbench system.

Figure 10. Combination with Our Testbench Robot System



Design Features

Our design has the following features:

- *Processor*—Nios II processor
- *Operating System*—µC/OS-II RTOS using LWIP
- *Interfaces*—SPI and Ethernet communication
- *I/O Signal*—Motor control and sensing (34 I/O pins)

We developed our robot manipulator for robust control. The software generates the signals required for elaborate control while the Altera Cyclone II FPGA solves time delay problems. With this combination, the system performs well, with all of the parts working together. Furthermore, this implementation solves the synchronization problems experienced by the conventional manipulation method, which uses many CPUs to control many motors. Our control module is easy to control externally, because we use I^2C to control orders from external devices. Additionally, we implemented the robot's operating system easily by using custom instructions with the Nios II processor.

Conclusion

It was very difficult to adapt the FPGA system to the robot embedded system. To design the robot system with an FPGA, we divided the project into two parts: developing the MCU-based model and developing the FPGA-based model. Changing from MCU-based development to FPGA-based development will have a huge impact on the industry design flow. Originally, we considered using the ARM microprocessor to control the robot. However, after using the FPGA-based system, we found that we prefered using the FPGA to using ARM or an MCU. So, the design worked better than we expected. The Nios II processor, Quartus II software, SOPC Builder, and Nios II Integrated Development Environment (IDE) made it easy for us to develop our design, showing the usefulness of these tools in the design process.