Third Prize

Nios II-Based Air-Jet Loom Control System

Institution: **Donghua University**

Participants: **Yu-Bin Lue, Hong Chen, and Bin Zhou**

Instructor: **Ge-Jin Cui**

Design Introduction

Widely used in the textile industry, the air-jet loom is one of the fastest, shuttleless looms today. The loom has significant control system requirements. With multiple tasks (for example, the ZA205 loom's control board needs 61 I/O ports) and demanding response time requirements, the control system's performance usually determines the air-jet loom's capability. The loom usually needs to complete a series of movements in a few milliseconds, such as opening, weft insertion, beating up, warp let-off, fabric take-up, weft cutting, weft storage, etc.

The advanced air-jet looms overseas use 32-bit embedded systems, while China still uses 8-bit monolithic processors and digital integrated circuits. Because other control systems have a high price and maintenance costs, a reliable substitute with excellent price/performance is of significance to China in loom production and maintenance cost reduction compared to imported looms. The air-jet loom does not have a customized 32-bit microcontroller (MCU), so the MCU application inevitably wastes resources.

With the rapid development of electronic technology, system-on-a-programmable-chip (SOPC) performance has improved. As a leader in the industry, Altera has developed the Nios II processers. The latest Nios® II processor provides performance up to 200 DMIPS. The user can select the core type to balance performance and size. Additionally, designers can use multiple Nios II CPUs, custom instruction sets, and hardware acceleration, improving performance. The µC/OS-II multi-tasking, RTOS provides flexible, efficient applications.

In industrial sites, FPGAs integration capability and rich I/O resources improve the system's anti-interference capability and satisfy I/O requirements. By studying the ZA205 air-jet loom's control system, we were able to implement a Nios II-based control board.

Because of time and other conditions, the system is restricted to a simulated platform. We give priority to the warp tension control to implement constant tension control without overshoot, because this is a key factor affecting the cloth quality. In the system, the signals output by the photoelectric coder use the remaining FPGA resources to build a small Nios II system that generates Gray code, with the relay's control signal indicated by a LED.

Function Description

The air-jet loom computer control system is a typical direct digital control (DDC) system. It has four layers: collection/execution, I/O channel, main control loom, and peripheral devices. The control board belongs to the main control loom and precisely runs all organization and fittings. This section describes the ZA205 air-jet loom's control system and the control board functions.

Warp Let-off Control

To ensure continuity of the production process, the loom must be constantly supplied with tensioned warp yarns. The knitting must be drawn away from the opening while the system winds the knitting onto the roller. The system has the following controls:

- *Tension control*—The servomotor controls the warp yarn tension and slack according to the signal collected by the tension sensor. This process ensures that the warp yarn's tension is controlled properly during opening, warp loosening, and warp let-off.
- *Manual warp yarn loosening and tightening*—When the machine is not operating, the user can manually tighten or loosen the warp by controlling the servomotor's forward and reverse rotator.
- *Fabric take-up control*—This function pulls the finished cloth away from the opening in time as required by the weft density so that the position of the opening does not vary with the new weft-yarn, ensuring successful production.

Crank Angle Collection

The loom's crank angle is an indicator of the loom's operation state. It is collected by the 256-line absolute photoelectric coder in the signal format of binary Gray code. According to this signal, the control board indicates the loom's operational state and send commands accordingly.

Picking Control According to the Crank Angle

The air-jet loom is a shuttleless loom that inserts the weft with the airflow's frictional traction force on the weft yarn. Most air-jet looms today use main and auxiliary nozzles to relay air jetting and weft insertion. Therefore, weft insertion control mainly focuses on controlling the valve solenoid of the main and auxiliary nozzles.

- *Main nozzle control*—This control draws the weft yarn from the weft feeder using a high-speed airflow and completes the first phase weft insertion.
- *Auxiliary nozzle control*—The auxiliary nozzles pull out the yarns introduced by the main nozzle, relaying weft insertion.
- *Weft stop pin control*—After weft insertion, this process sends the weft stop pin signal to the FDP system (weft feeder) to make it ready for the next round of picking.
- *Nozzles cutting control*—Uses the electronic knives to cut the weft yarns.

Main Motor Loop Control

The main motor drives the loom by a pulley and uses a gear to drive the other mechanisms such as the cam harness mechanism, crank beating mechanism, and electric take-up mechanism. The air-jet loom's

operational states are divided into quick action and slow action; the slow action can be further divided into forward and backward. This section describes control circuit functions of the main motor.

- *Star-delta start control*—Starting the main motor too strongly jeopardizes the electric network and motor, so the system adopts a star-delta press reduction start mode. The system connects in a delta when starting the motor and then switches the relay when it receives the control signal from the I/O port of the control system. As the system comes up to speed, the motor connects in a star.

- *Slow forward and reverse control*—When the loom needs to run slowly, the AC switches from 50 Hz to 5 Hz, so that the loom can accurately stay at a certain angle when faults such as broken warp and weft occurs.

- *Braking control*—This function uses different braking methods when it receives different braking signals, such as:

  - *Emergency braking*—Stops the loom using the main motor while shutting off all power.

  - *Safe standstill or scram*—Loosens the clutch and then brakes suddenly to slow down the motor. After a while, it brakes slowly to stop the motor. Finally, it stops the motor at a certain angle by slow action.

Network Remote Monitoring Function

To facilitate monitoring, the loom's control system transfers its operation state and parameters into the server through the network. The following table shows the control board's I/O.

Control Board I/O

Control Board I/O	Number of Routes	Direction
Crank angle	8	Input
State collection	11	Input
Button signal	10	Input
Main nozzles control	2	Output
Auxiliary nozzles control	12	Output
Weft stop pin signal	2	Output
Yarn shield pin signal	2	Output
Cutting nozzles control	2	Output
Main motor control	8	Output
Alarm light signal	4	Output

Performance Parameters

The control parameters are described below.

- *Tension control density*—< 5%

- *Touch screen operation response time*—0.1 second

- *Speed of the simulated main motor*—3,000 RPM

- *System power supply*—±5 V, 10 V
- *IPV4 data frame send frequency*—10 Hz

Design Architecture

Figure 1 shows the control structure block diagram.

Figure 1. Overall Block Diagram

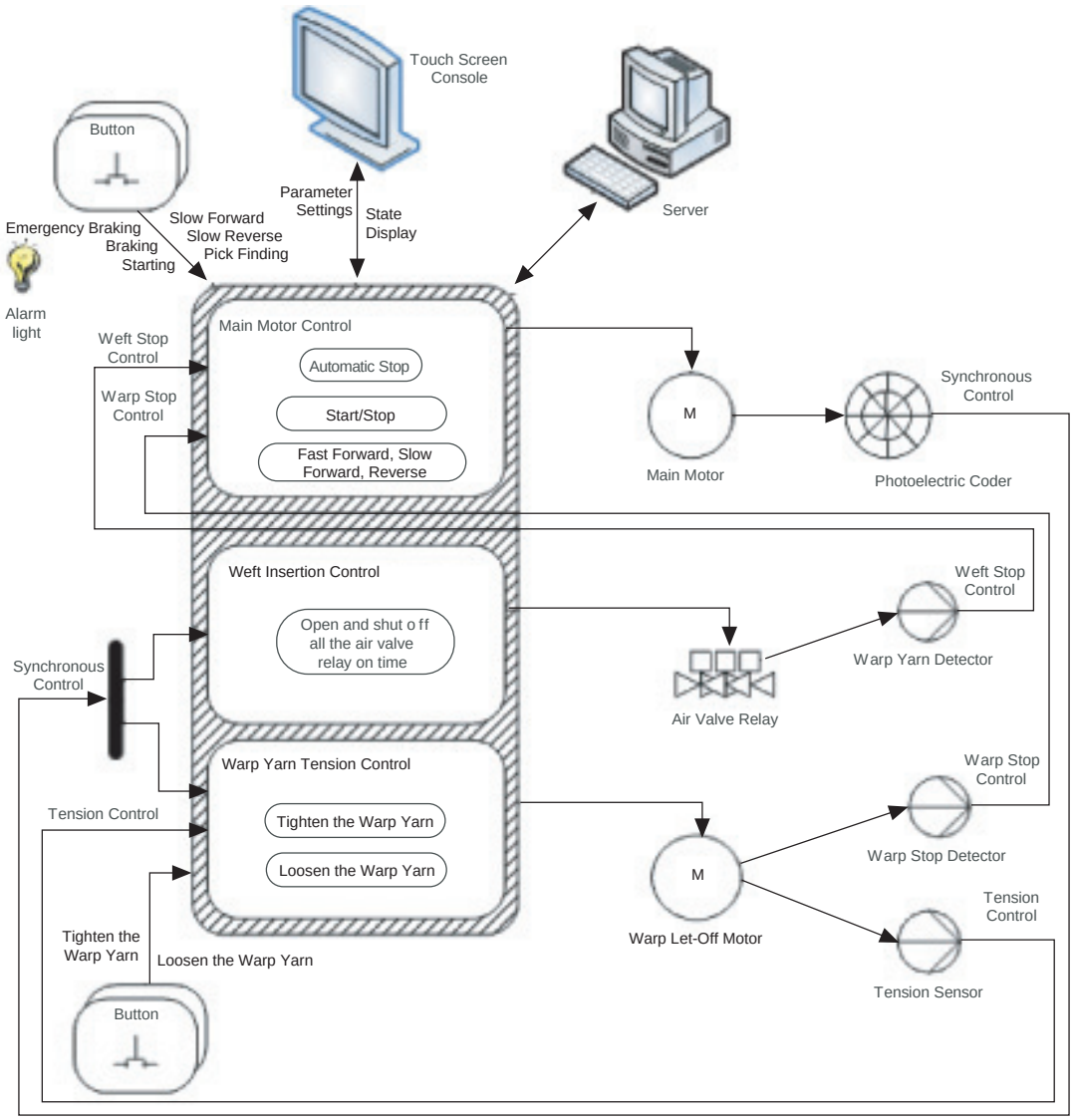


Figure 2 shows the warp let-off motor control block diagram.

Figure 2. Warp Let-off Motor Control Block Diagram

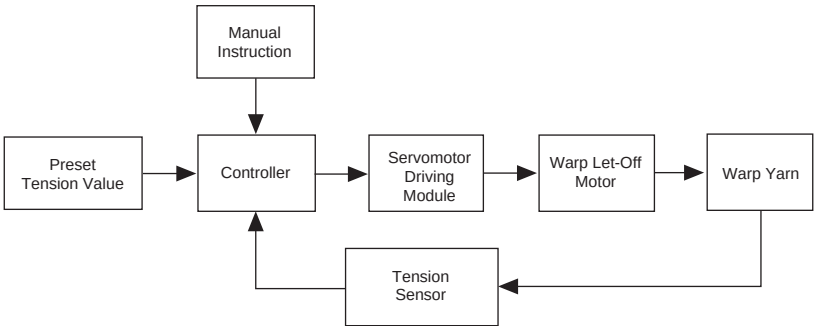


Figure 3 shows the hardware block diagram.

Figure 3. Hardware Block Diagram

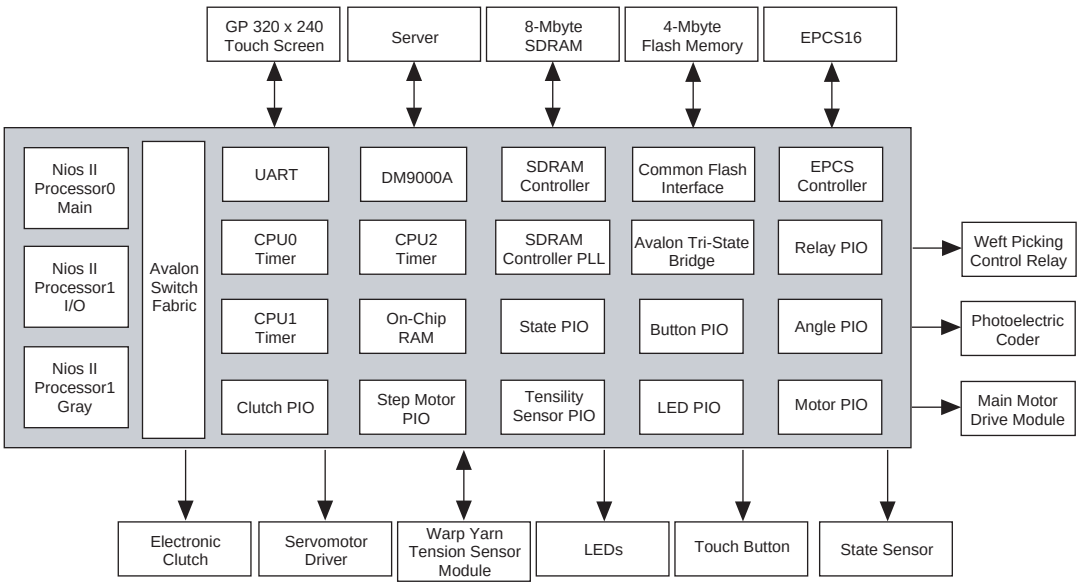
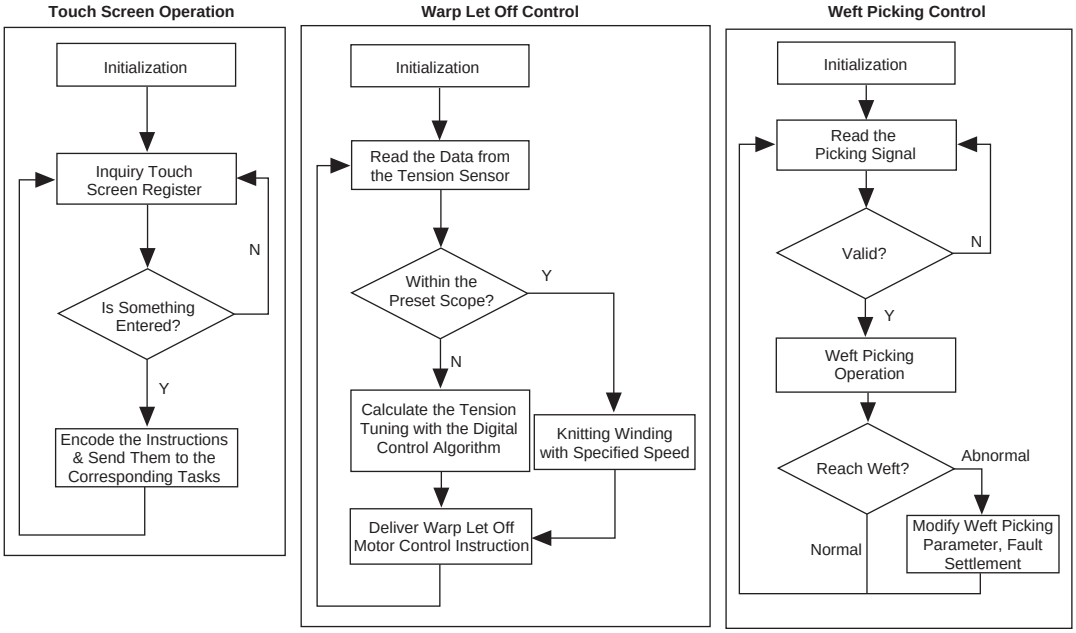


Figure 4 shows the main software flow.

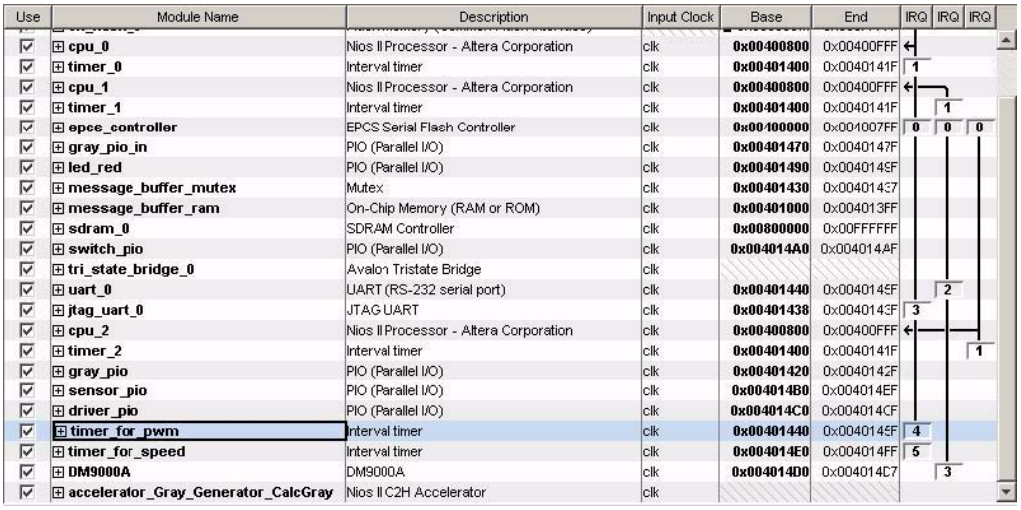
Figure 4. Main Software Flow



Design Methodology

The design provides multi-tasking with multiple Nios II processors and shares data between the processors. It adds a variety of peripherals to the bus. See Figure 5.

Figure 5. SOPC Builder



Use	Module Name	Description	Input Clock	Base	End	IRQ	IRQ	IRQ
☑	cpu_0	Nios II Processor - Altera Corporation	clk	0x00400800	0x00400FFF	←		
☑	timer_0	Interval timer	clk	0x00401400	0x0040141F	1		
☑	cpu_1	Nios II Processor - Altera Corporation	clk	0x00400800	0x00400FFF	←		
☑	timer_1	Interval timer	clk	0x00401400	0x0040141F		1	
☑	epce_controller	EPCS Serial Flash Controller	clk	0x00400000	0x004007FF	0	0	0
☑	gray_pio_in	PIO (Parallel I/O)	clk	0x00401470	0x0040147F			
☑	led_red	PIO (Parallel I/O)	clk	0x00401490	0x0040149F			
☑	message_buffer_mutex	Mutex	clk	0x00401430	0x00401437			
☑	message_buffer_ram	On-Chip Memory (RAM or ROM)	clk	0x00401000	0x004013FF			
☑	sdram_0	SDRAM Controller	clk	0x00800000	0x00FFFFFF			
☑	switch_pio	PIO (Parallel I/O)	clk	0x004014A0	0x004014AF			
☑	tri_state_bridge_0	Avalon Tristate Bridge	clk					
☑	uart_0	UART (RS-232 serial port)	clk	0x00401440	0x0040145F		2	
☑	jtag_uart_0	JTAG UART	clk	0x00401438	0x0040143F	3		
☑	cpu_2	Nios II Processor - Altera Corporation	clk	0x00400800	0x00400FFF	←		
☑	timer_2	Interval timer	clk	0x00401400	0x0040141F			1
☑	gray_pio	PIO (Parallel I/O)	clk	0x00401420	0x0040142F			
☑	sensor_pio	PIO (Parallel I/O)	clk	0x004014B0	0x004014EF			
☑	driver_pio	PIO (Parallel I/O)	clk	0x004014C0	0x004014CF			
☑	timer_for_pwm	Interval timer	clk	0x00401440	0x0040145F	4		
☑	timer_for_speed	Interval timer	clk	0x004014E0	0x004014FF	5		
☑	DM9000A	DM9000A	clk	0x004014D0	0x004014D7		3	
☑	accelerator_Gray_Generator_CalcGray	Nios II C2H Accelerator	clk					

The system's functions are warp yarn tension control, touch screen communication, network monitoring, main motor and nozzle control, etc. It uses three CPUs, which are described as follows:

- *CPU0*—Warp yarn tension reading, servo motor control, and nozzle control.
- *CPU1*—Touch screen and network communication.
- *CPU2*—Main motor movement simulation and Gray code generation.

The following sections describe these functions.

Warp Yarn Tension Control

The following sections describe the warp yarn tension control function.

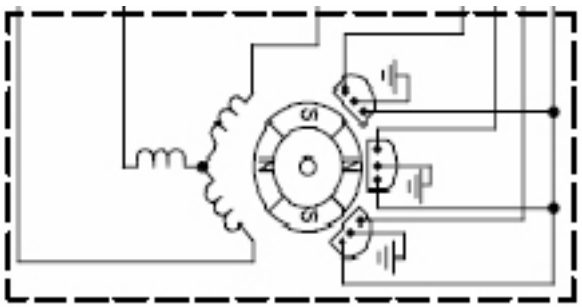
Data Collection

The warp yarn tension is indicated by the sensor of its pulse signal duty cycle. After the signal is reshaped by a Schmitt trigger, the system filters it to obtain the tiny DC level proportional to the tension. Then it is enlarged by a multi-stage amplifier to obtain the 0 to 3 V valid signal (after adding the DC offset). The system performs analog-to-digital (A/D) conversion of the valid signal. Because of limited I/O resources, the A/D converter selects the serial output TLC0832 device. Because the A/D converter can only work in slave mode, the system uses Verilog HDL to write the driving interface module. After A/D conversion, the sampled digital signals are sent into the shared memory for the CPUs to read.

Servomotor Control

The system compares the tension sensor reading with a preset value and then adjusts the warp let-off motor by the digital process ID (PID). The warp let-off motor is REDCOM's 57ZW-10W brushless DC servomotor that has three Hall output routes. Figure 6 shows the schematic diagram.

Figure 6. Warp Let-off Motor Schematic



The schematic uses Sanken's SLA6023 chip, which is used specifically for brushless DC motors. The chip is composed of three Darlington doubled pipes, as shown in Figure 7.

Figure 7. Sanken SLA6023 Chip

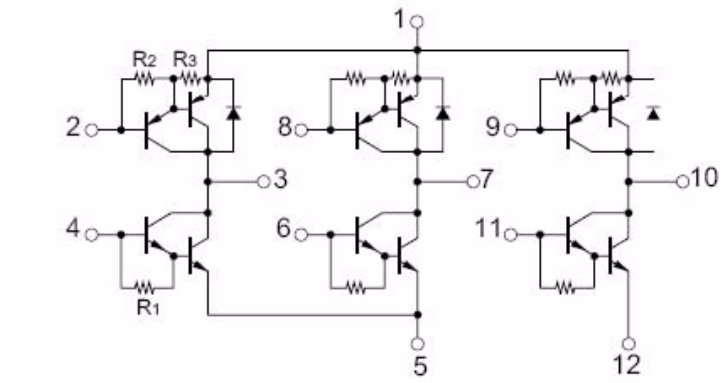
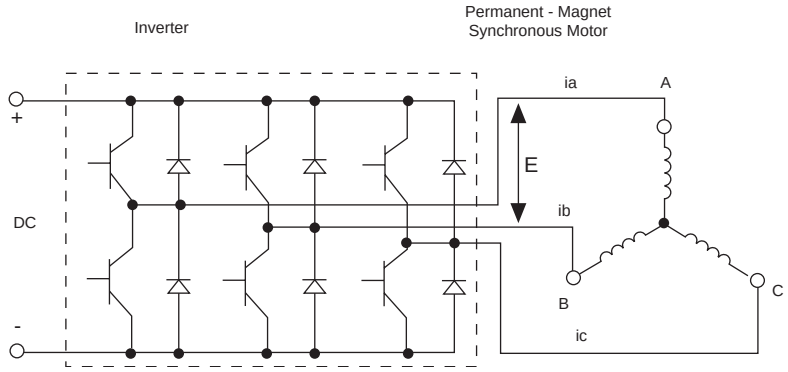


Figure 8 shows the motor-driven connection lines.

Figure 8. Motor-Driven Connection Lines



To drive the motor, the system determines the wheel rotor's position by reading the three-route Hall sensor. It controls the current direction of the A, B, and C stator winding by turning on/off the Darlington pipes according to the drive timing.

The timer interruption mode uses pulse width modulation (PWM) and initializes according to the timer core's register map. The following code describes the initialization process:

```
//Clear TO
IOWR(TIMER_1_BASE,0,0x0);
//Set START,CONT,ITO?start the timer and make cycle count, and
//generate system interruption on the crossover point
IOWR(TIMER_1_BASE,1,0x07);
//Registration interruption responsive funtion
alt_irq_register(TIMER_1_IRQ,edge_capture_ptr,handle_timer1_interrupts);
```

The following code sets the count cycle in the `handle_timer1_interrupts` interrupt response function:

```
IOWR(TIMER_1_BASE,2,periodl);
IOWR(TIMER_1_BASE,3,periodh);
```

Where `periodl` is the starting value's low count address and `periodh` is the high address.

During program operation, the system dynamically alters the values in the two registers according to the PID-calculated output, implementing the corresponding PWM duty cycle control.

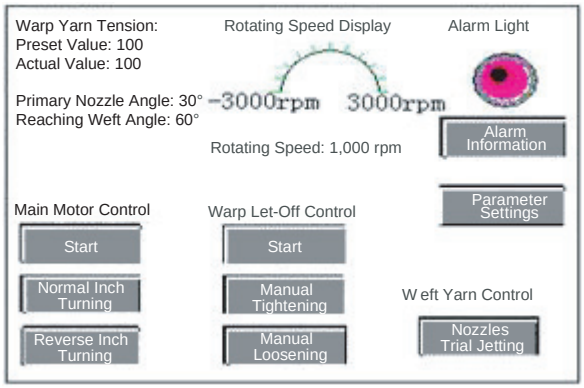
Touch Screen Communications

The touch screen is Digital's 37W2 6-inch blue LCD for industrial graphics, featuring 320 x 240 resolution and 16 x 12 touch keys on each screen. It also has a graphic interface development environment, which is very convenient for the user. The screen has an RS-232 communications interface, which operates slowly. So that it does not affect the overall speed, the system usually sets independent threading for the serial port reads/writes in the PC's WIN32 program.

In this design, a designated CPU communicates with the screen. It polls the touch screen's registers, reads the corresponding data when there are operations, and sends operation instructions (such as starting and stopping) into the data sharing area, so that the other CPUs can perform as instructed by the instructions they receive. The user can also use the screen to change the parameters and password. The system writes data to the flash chip to initiate the touch screen when the power is on, which saves the parameters in the event of a power failure. Additionally, if the system becomes abnormal, the screen sends an instruction to control the alarm interface.

Figure 9 shows the touch screen interface.

Figure 9. Touch Screen Interface

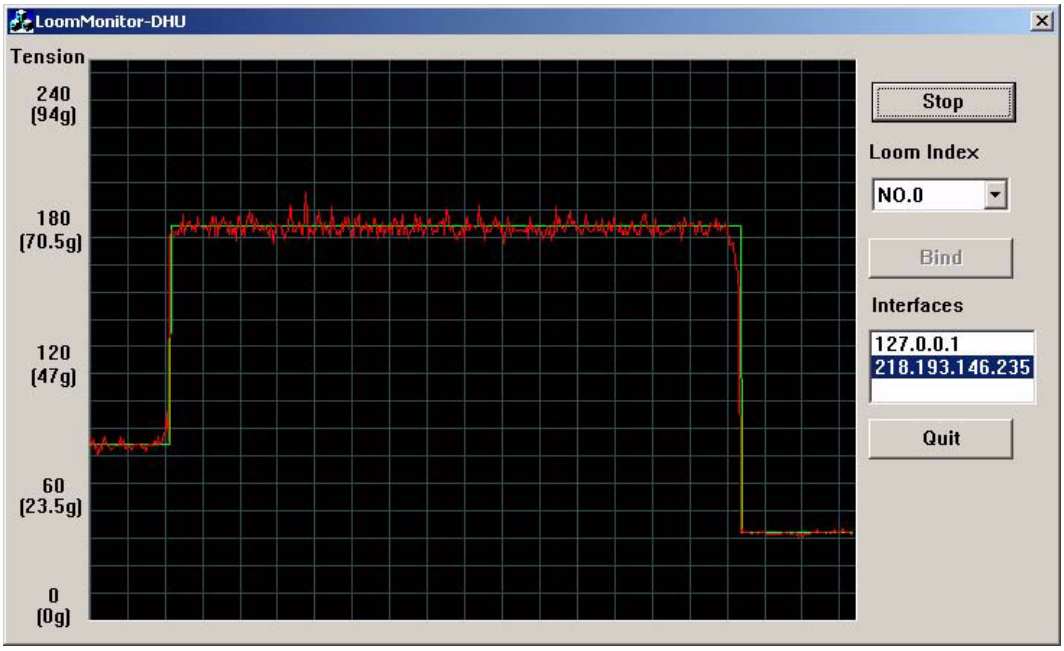


Network Monitoring

The current loom is an independent entity, requiring the workers to walk back and forth to watch the state of each loom. We tried a few methods to solve this problem, and decided to add a control system with a network monitoring function. Even without an operating system, the system can generate a network data frame, packaging the data in IPV4 frame format. Then, it delivers the data to a server via the network interface.

We used Visual C++ 6.0 to develop the server side monitoring software, which displays the warp yarn tension's preset value and actual value in dynamic curves, and monitors the cooresponding loom according to its number. Figure 10 shows a section of the software, in which the green curve is the tension's preset value and the red curve is the actual value (the curve data sampling frequency is 10 Hz).

Figure 10. Server Monitoring Software



Main Motor and Picking Nozzle Control

The loom is driven by the main motor. Its working state is shown by the photoelectric coder on the crankshaft that is connected to the motor. To reduce the peak current pulse caused by the coding, the output codes adopt the most reliable coding method, which is binary Gray code.

Our system is a simulated platform without actual photoelectric coding components. Therefore, it uses an independent CPU to simulate the main motor movements, generating 8-bit Gray codes to simulate the photoelectric coder output. Its primary task is to generate Gray codes and put them into the data sharing area according to the main motor's operational state after obtaining the touch screen operation instruction. It has four states: starting, stopping, normal inch turning, and reverse inch turning. The main CPU turns on/off the nozzles according to the Gray code value and the loom's state (simulated by the seven-segment LED).

In the main motor simulation of the above tasks, the `CalcGray()` function we used for Gray code calculation has many cycles, is independent, and is used frequently. Therefore, we used the Altera® C2H compiler and generated the SOPC Builder component **accelerator_Gray_Generator_CalcGray**. SOPC Builder automatically generates the function interface for the new component, which is very convenient for the user.

Design Features

Our design has the following features:

- Achieves data sharing communication between the three Nios II processors and uses a large CPU instruction cache.
- Generates and delivers the IPV4 frame format network information packet.
- Uses the C2H compiler to perform hardware acceleration for frequently used functions.

- Uses many other peripherals, which are connected to the Avalon® bus.

- The traditional multiple monolithic computers architecture are affected by electronic interference in industrial sites, and are unable to maintain a stable system. The Nios II processor integrates most of the system on a FPGA, increasing the system's stability and enhancing its anti-interference capability.

- This system involves the communication between modules. The configurability of the Nios II processor let us add a UART, SDRAM, the LCD interface, and a custom interface, avoiding the resources wasted by MCU applications, accelerating research and devlopment, and lowering costs.

Conclusion

The Altera Quartus® II software and Nios II Integrated Development Environment (IDE) are revolutionary in system development. They have played a leading role in software-like-hardware design and simplified the design process.

During the competition, we spent a lot of time figuring out how to initiate the multiple CPUs simultaneously. We experimented with many methods, including modifying SDRAM clock deviation, bringing a phase-locked loop (PLL) from the inside to outside of the project, modifying CPU types, and modifying the code storage position in flash. Despite the strong support we received from Altera via the MySupport site, some problems remained unresolved until the end of the competition. In the end, we modified our plan according to the advice we received, canceling one CPU that was to be used for servomotor control and only using three. Meanwhile, we integrated the tasks into the main CPU. Although this solution increases stability at system start-up, other problems occur, such as too many tasks in the main CPU and non-specific labor division. Because of these reasons, we were not able to spend enough time on software development and peripheral tuning to complete all of the functions we planned.