CHAPTER 4

ERROR CONTROL USING RECONFIGURABLE CRC

4.1 IMPLEMENTATION OF RECONFIGURABLE HARDWARE

The implementation of target reconfigurable hardware (RHW) on output processing block (OPB) slave bus is presented. Xilinx IP line hub is used to interface OPB bus in simpler way for the user model. This interface device is connected in both OPB and PLB buses. The adaptation of target RHW to the PLB bus is an easier task due to IP line hub. Control and configuration of the RHW is done through register Write operations in C++. Genome values are written to registers which are again connected to the configuration inputs of each functional module. Registers are also provided for RHW for getting inputs and for storage of outputs. This is shown in figure 4.1.

Figure 4.1 RHW design
The functional unit in Figure 4.1 is the core of RHW, which performs programmable functionalities and carry out mathematical operation on inputs, and perform storage of output. Each and every functional unit gets three inputs and the input unit can perform any of the four functions: BUF, MUX, AND, XOR. The functional units and its three inputs are configurable and determined by genetic programming. The functional units of the genome string are encoded and shown in table 4.1.

**Table 4.1 Functional Units of Single Input Data**

<table>
<thead>
<tr>
<th>Input data 1 (4 bits)</th>
<th>Input data 2 (4 bits)</th>
<th>Input data 3 (4 bits)</th>
<th>Function Unit (2 bits)</th>
</tr>
</thead>
</table>

Out of the 4 bits for each input, one bit can be inverted in toggling, remaining 3 bits used for the output from the previous layer. The array uses 6 layers with 8 units; the genome string is 672 bit long. The array is built in parallel approach, and registers are interfaced to the output of each layer. The training vector calculated at any instant of time is one.

**4.2. ARM BASED RECONFIGURABLE ARCHITECTURE (RA)**

The RA takes the input from the sender and transfers it to physical layer that takes the input and converts to binary form and perform the different operations needed for error recovery. The Verilog implementation code is listed in appendix-I.

**4.2.1 Interface unit with SWD/JTAG**

JTAG mechanism is traditionally used for the debug connections for ARM7/9 ports, but with the Cortex-M family, ARM introduced the serial wire debug (SWD) Interface and this is followed in this work. SWD reduce the pin count by two required for debug. In addition, one of the pins freed up by this can be used for single wire
viewing (SWV), which is a low cost tracing technology. The connection details are shown in table 4.2 and table 4.3.

Table 4.2 The SWD/SWV Pins Overlaid on Top of JTAG pins

<table>
<thead>
<tr>
<th>JTAG Mode</th>
<th>SWD Mode</th>
<th>Signal</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCK</td>
<td>SWCLK</td>
<td>Clock into the core</td>
<td>10K-100K Ohm pull down resistor to GND</td>
</tr>
<tr>
<td>TDI</td>
<td>--</td>
<td>JTAG Test Data Input</td>
<td>10K-100K Ohm pull-up resistor to V_CC</td>
</tr>
<tr>
<td>TDO</td>
<td>SWV</td>
<td>JTAG Test Data Output / SWV trace data output</td>
<td>10K-100K Ohm pull-up resistor to V_CC</td>
</tr>
<tr>
<td>TMS</td>
<td>SWDIO</td>
<td>JTAG Test Mode Select / SWD data in/out</td>
<td>10K-100K Ohm pull-up resistor to V_CC</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

Table 4.3 20-Pin JTAG/SW Interface

<table>
<thead>
<tr>
<th>Pin no</th>
<th>Pin details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V_CC</td>
</tr>
<tr>
<td>2</td>
<td>V_CC(optional)</td>
</tr>
<tr>
<td>3</td>
<td>TRST</td>
</tr>
<tr>
<td>5</td>
<td>TDI</td>
</tr>
<tr>
<td>7</td>
<td>SWDIO/TMS</td>
</tr>
<tr>
<td>9</td>
<td>SWCLK/TCLK</td>
</tr>
<tr>
<td>11</td>
<td>RTCK</td>
</tr>
<tr>
<td>13</td>
<td>SWO/IDO</td>
</tr>
<tr>
<td>15</td>
<td>RESET</td>
</tr>
<tr>
<td>17,19</td>
<td>N/C</td>
</tr>
<tr>
<td>4,6,8,10,12,14,16,18,20</td>
<td>GND</td>
</tr>
</tbody>
</table>
The complete reconfigurable architecture is shown in figure 4.2.

**Figure 4.2 Reconfigurable architecture**
4.3. RA BASED IMPLEMENTATION OF CRC BLOCK

CRC is the most preferred method of error detection scheme since it provides very efficient protection against commonly occurring burst errors, and is easily implemented. CRC can detect all one bits and two bits errors as well as all odd number of bits in error. Since CRC is a technique for detecting errors, but not for making corrections when errors are detected, the whole packet data will be retransmitted if error occurs.

![Figure 4.3 Structure of CRC block](image)

4.3.1 Specification of CRC block

The specification of the CRC block, the input and output ports details is listed in table 4.4.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Port Details</th>
<th>Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Input ports</td>
<td>Clock, reset, write, Shift_enable, Data_in</td>
</tr>
<tr>
<td>2.</td>
<td>Output port</td>
<td>Data_out</td>
</tr>
<tr>
<td>3.</td>
<td>Frequency</td>
<td>&lt;=250 KHz</td>
</tr>
</tbody>
</table>
4.3.2 Implementation of CRC control block

The CRC control block performs the one’s and two’s complement on the input data. The control module of the CRC block includes the

- Data Control and
- Register Control

4.3.2.1 Data control in CRC block

The data control access initially runs under the write mode. Once the mode is started data transfer takes place. The timing sequence of this block is shown in figure 4.4.

![Figure 4.4 Timing sequence of data control in CRC block](image1)

4.3.2.2 Register control in CRC block

The timing status of the register control is shown in figure 4.5.
4.3.2.3 Shift enable logic in CRC block

The shift enable logic used in the CRC block is implemented using barrel shifters. The ‘n’ flip-flops are connected in series with the last output being feedback to the first input. The following code implements the n-bit barrel shifter:

```verilog
module barrel n bit(Data_out, Data_in, clock, reset, write);
    output [n-1:0] Data_out;
    input [n-1:0] Data_in;
    input clock;
    input reset;
    input write;
    reg [n-1:0] Data_out;
    always @(posedge reset or posedge clock)
    begin
        if (reset == 1'b1) Data_out <= n'0;
        else if (write == 1'b1) Data_out <= Data_in;
        else Data_out <= {Data_out[n-2:0], Data_out[n-1]};
    end
end module
```

4.4 SIGNATURE ANALYSER BLOCK FOR SECURE NODE ENTRY

In this work, integrating the front end signature check for the individual node with the CRC block is focused via, software hardware co design. Secured transmission insists the various technical aspect of self-healing transmission process. Apart from the concept too is also made use of the existing secured transmission nodes travel either toward the advancement, logics, techniques and algorithms of the software issues or towards the assuring aspects of the hardware. Hence, data could be protected, via the software
hardware co-design, from the intruders. The focus is to identify the untrusted nodes in the initial stage itself into the transmission from the physical and data link layer as shown in figure 4.6.

![Diagram](image_url)

**Figure 4.6 Proposed integrated architecture for secure data transmission**

The hardware chip present along with the master node in the transmission side possesses the hostname and the unique address of the trustworthy nodes. When a slave node demands for a data, the master node sends a request via the hardware e-mote, which checks for the validation of the unique address. If the corresponding unique address is present in the list of available unique address in the e-mote then the corresponding hostname is said to be a trustworthy node and is guaranteed to initiate transmission. If the corresponding unique address is not available with the list of unique address present in the master node e-mote, then that particular node is suspended to be an untrustworthy
node. Then a key is made for its certificate. If the slave node sends a valid certificate, then the corresponding unique address is a validated and is decided to be a trustworthy user. Hence, the unique address and hostname of that particular node is also added as a trustworthy user. If the certificate is not received/ received certificate is identified to be fake, then the corresponding node is identified to be an untrustworthy user and so initialization of communication is made as shown in figure 4.7.

When the corresponding user is identified to be trustworthy, the requested data need to be sent. Before transmission, the data are provided added security using an encryption algorithm (like DES) and the encrypted text is transmitted.

Figure 4.7 E-mote Interface for initial signature check (transmitter side)
4.4.1 Receiver node/ E-mote interface

When a slave node is in need of a data, it checks for the unique address of the master node from the list of available unique address contained in the e-mote and makes a request to the corresponding master node. If the master node doesn’t contain the unique address of the slave node unique address then, when the master makes demand for the certificate, the slave node has to read its certificate for verification purpose. If the certificate has not been sent then the corresponding node is identified to be an untrustworthy node as shown in figure 4.8.

Figure 4.8 Receiver interface
4.4.2 Self-healing architecture based error detection in networks

The architecture for error detection includes master node, slave node, embedded mote, transmission medium and error recovery algorithm (Figure 4.9). A particular system is the network, say, slave node, request for a confidential data. The master node which already possesses a list of registered hostname validates the hostname and the unique address of the slave node. It the master node contains the details of the particular slave node, and then the corresponding node is said to be trustworthy user. Once the slave node is identified to be a valid user, then the communication could be made and the data could be transferred. The data is not sent in the raw form, but encrypted using the data encryption standard [DES], is transmitted through the channel using the regulations of the protocol suite. The key used to encrypt the plain text too is transmitted access, via, Wi-Fi. The hostname and the unique address of the genuine users are stored in a chip, called embedded mote. The embedded mote, heritages, referred as E-Mote is a hardware chip connected to each of the systems in the network.

Figure 4.9 Proposed architecture of error detection in networks
4.4.3 E-mote to master node interface

The E-Mote present in the master node contains the hostname and the unique address of the different slave node identified as trusted in the network. When a slave node request occurs on a master node, it gets access to its E-Mote. The E-Mote validates and contains mechanism to prevent (E-Mote stores all the hostname of trustworthy marked nodes) a particular slave node if it is untrustworthy as illustrated in the flow chart of figure 4.10.

Figure 4.10 Flow chart for receiver node interface with master node
4.5 HARDWARE IMPLEMENTATION

An embedded system sets software and hardware in one system, which has characteristics such as small size of software code, high degree of automation, and high response speed. These are particularly suited to multi-task and real-time control in industrial areas. The ARM Cortex™-M3 processor is the latest generation of ARM processors for embedded systems. It has been developed to provide a low-cost platform that meets the needs of MCU implementation, with a reduced pin count and low-power consumption, while delivering outstanding computational performance and an advanced system response to interrupts.

The ARM Cortex™-M3 32-bit RISC processor features exceptional code-efficiency, delivering the high-performance expected from an ARM core in the memory size usually associated with 8- and 16-bit devices. To meet the requirements of rapid and complex algorithms in the device STM32F103 ARM processor with low power consumption based on Cortex™-M3 from ST Microelectronic is chosen as control core of the device. The processor provides a wealth of hardware and software resources for users. The developed ARM board for implementing the master-slave concept described in section 4.5.2 along with the functional unit is shown in figure 4.11.
4.5.1 Hardware resources

1) 128 Kbytes of embedded Flash is available for storing programs and data.

2) 20 Kbytes of embedded SRAM accessed (read/write) at CPU clock speed with zero wait states.

3) The STM32F103RB performance line embeds a nested vectored interrupt controller able to handle up to 43 maskable interrupt channels (not including the 16 interrupt lines of Cortex™-M3) and 16 priority levels.

4) System clock selection is performed on startup; however the internal RC 8 MHz oscillator is selected as default CPU clock on reset. An external 4-16 MHz clock can be selected, in which case it is monitored for failure.

5) At startup, boot pins are used to select one of three boot options:
a. Boot from User Flash
b. Boot from System Memory
c. Boot from embedded SRAM

6) In build 18Mbps SPI interfaces, 2.25/4.5 Mbps USART, I2C/SM Bus interface, CAN Bus, USB Bus, GPIO and ADC.

The ARM Board 10-pin connector detail is shown in table 4.5.

<table>
<thead>
<tr>
<th>Pin no</th>
<th>Pin details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VCC</td>
</tr>
<tr>
<td>2</td>
<td>SWDIO</td>
</tr>
<tr>
<td>3,5,9</td>
<td>GND</td>
</tr>
<tr>
<td>6</td>
<td>SWO</td>
</tr>
<tr>
<td>7,8</td>
<td>N/U</td>
</tr>
<tr>
<td>10</td>
<td>RESET</td>
</tr>
</tbody>
</table>

### 4.5.2 Slave Functionality in Hardware

The arm board consisting of the transmitter, receiver and controller peripheral devices are usually attached to the processor by mapping their physical registers into arm memory space or into the coprocessor space or connecting to another device (a bus) which in turn attaches to the processor. Coprocessor accesses have lower latency and some peripherals are designed to be accessible in both ways (through memory and through coprocessors). Sometime the arm board can act as an interrupt. The arm board includes the hardware debugging facilities within them. So it can be utilized to find the
trust worthy nodes among the sensor network in the transmission. The arm board consists of two units. They are

- ARM based master node (Server)
- ARM based slave node (Client)

The master node and slave node implementation are discussed in the next section. The integrated hardware unit is shown in figure 4.14.

![ARM board slave node (Client Hardware)](image)

Figure 4.12 ARM board slave node (Client Hardware)
The code performing the data synchronizing function of slave module is shown in appendix-IV.

4.5.3 Master Functionality in Hardware

![ARM board based on master node (Server hardware)](image)

**Figure 4.13** ARM board based on master node (Server hardware)
The code performing the data synchronizing function of master module is described in appendix-III.

4.5.4 Master slave transmission unit

![Figure 4.14 Transmission process between master and slave node in sensor network](image)

**Figure 4.14 Transmission process between master and slave node in sensor network**
4.5.5 Internal architecture of master node

The ARM board internal architecture of master node is shown in figure 4.15.
4.5.6 Internal architecture of slave node

The ARM board internal architecture of slave node is shown in figure 4.16.

Figure 4.16 Internal architecture of ARM board (Slave node)
4.6 CHAPTER CONCLUSION

In this chapter a RHW based signature check integrated with the CRC to remove the untrusted nodes and the error present in the data is designed. The RHW was implemented on a Xilinx FPGA based data processing board. The reconfigurable architecture enables the addition of new features, allows rapid implementation of new standards and protocols on an as-needed basis and protects the investment in computing hardware. It functions as a programmable hardware with higher performance and allows the flexibility of a software based solution while retaining the execution speed of a more traditional hardware based approach.