Mobile Data Communication Device
Project

- Project was carried out under the Open Source Policy of the ICT R&D Fund

- It was a nine month project
  - January – October 2009

- Project was done in collaboration with Institute of Space Technology (IST)
  - Joint Project Director: Mr. Iftekhar Mehmood
Project Components

- Three Major parts of the project
  - System on Chip (SoC) Framework
    - RTL Development
    - Simulation & Verification
  - Firmware Development
  - Board Development
    - To prove the SoC Framework is working as required
Framework

- A basic framework and architecture for SoC development
  - Processor Independent
  - Palmbus is being used to interconnect the various modules
RTL Development

- Intellectual Property (IP) developed
  - Processor Interface
  - Palmbus Interface
  - Universal Asynchronous Receiver and Transmitter (UART)
  - SPI Master/Slave
  - I2C Master/Slave
  - Interrupt Controller
  - Timer
  - Watchdog Timer
  - Real Time Clock (RTC)
  - OLED Controller
  - Power Management Unit (PMU)
  - GPIO
  - Clock Controller
  - Memory Controller - IST
Features

- **Processor Interface**
  - Interfaces to the
    - On-Board Processor
    - Palmbus Controller

- **Palmbus Controller**
  - Performs the address decode
  - Generates the read/write cycle to individual block
  - Support for 16 blocks

- **Power Management Unit**
  - If a module is idle for a certain period of time than clock is stopped to that module to conserve power
  - Programmable 16-bit counters implemented for each module
  - When a new request is received clocks are enabled again
Features

UART
- 16550 register-set compatible
- 5-8 data bits, Parity even/odd/none, 1, 1.5 or 2 stop bits
- Support for modem communication

OLED
- Programmable 8-bit or 16-bit LCD/OLED Screen Support
- Programmable timing values
- Programmable polarity of control signals
- Support for 8080 and 6800 Mode

Interrupt Controller
- Programmable interrupt priority
- Programmable polarity – Rising or Falling edge
- Programmable Edge or Level triggered interrupts
- Independent enable and disable for each interrupt
Features

- **I2C**
  - Master and Slave Mode
  - Multi-Master support
  - Programmable Clock Rate
  - 8 or 10-bit Addressing
  - Programmable Data Sizes

- **SPI**
  - Master and Slave mode
  - Programmable Clock Rate, and Polarity
  - Programmable bit ordering (MSB First, LSB First)
  - Programmable payload length
Features

- **Real Time Clock (RTC)**
  - Maintains system date and time
  - Programmable interval timer
  - Programmable time-and-date interrupt

- **Timer**
  - Programmable 32-bit general purpose timer
  - Code Profiling
  - PWM Mode

- **Watchdog Timer**
  - Programmable 32-bit timer
  - Keeps track of overall system health
  - Reset's the system if it expires
Features

- **General Purpose IO**
  - 4 8-bit Ports for General Purpose IO
  - Programmable Input, Output or Special signal Mode
  - Interrupt can be generated for each input

- **Memory Controller**
  - Designed by IST students
  - SRAM
  - SPI Flash Interface
  - Palmbus Interface for Register Programming
  - Memory Bus Interface for Read/Write Commands
Features

- **SRAM Controller**
  - Programmable timing parameters
  - Support for single and burst transactions

- **SPI Flash Interface**
  - Read/Write Commands require no CPU intervention
    - Reduces CPU overhead
  - Commands performed under firmware control
    - Write Disable
    - Read Status Register
    - Write Status Register
    - Sector Erase
    - Read Chip Manufacture Identification
  - Programmable Clock Rate, and Polarity
  - Programmable bit ordering (MSB First, LSB First)
  - Programmable payload length
Verification

- System level verification methodology was setup.

- Simulation Models were developed for the following external interfaces
  - Processor Interface
  - OLED
  - I2C
  - Interrupt Generation Logic
  - UART
  - SPI

- Verification Plan was developed
  - To figure what needs to be tested
  - Test Suites Coded based on the verification plan
Verification – Setup

- Oscillator / Reset Controller
- External Models (CPU, I2C model, SPI model, etc.)
- Chip I/Os
- Clock Reset
- WatchDog Timer
- Real Time Clock
- Power Management Unit
- CPU Interface
- PalmBus Controller
- Interrupt Controller
- UART
- I2C
- SPI
- OLED
- Timer
- General Purpose I/O

PalmChip
www.palmchip.com
## Coverage Results

<table>
<thead>
<tr>
<th>Block Name</th>
<th>Statements Coverage</th>
<th>Branches Coverage</th>
<th>Conditions Coverage</th>
<th>Expressions Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Watchdog Timer</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Interrupt Controller</td>
<td>99.8%</td>
<td>94.4%</td>
<td>92.4%</td>
<td>96.6%</td>
</tr>
<tr>
<td>General Purpose I/O</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>OLED Controller</td>
<td>99.7%</td>
<td>99.3%</td>
<td>94.2%</td>
<td>96.5%</td>
</tr>
<tr>
<td>Real Time Clock</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>I2C</td>
<td>98.2%</td>
<td>97.6%</td>
<td>90.2%</td>
<td>98.6%</td>
</tr>
<tr>
<td>SPI</td>
<td>97.9%</td>
<td>97.3%</td>
<td>93.5%</td>
<td>97%</td>
</tr>
<tr>
<td>UART</td>
<td>99%</td>
<td>100%</td>
<td>92%</td>
<td>90%</td>
</tr>
<tr>
<td>Power Management Unit</td>
<td>96.4%</td>
<td>95.5%</td>
<td>93.8%</td>
<td>100%</td>
</tr>
</tbody>
</table>

*The percentage number in each column is calculated by toggled (expressions, branches, statements, conditions) divided by the total number of (expressions, branches, statements, conditions) inside a block.*
Board
Board Interfaces

- Following physical interfaces have been designed and verified
  - JTAG Interface
  - OLED
  - GSM Modem
  - UART
  - SPI Master/Slave
  - I2C Master/Slave
  - Keypad
  - SPI Flash
  - SDRAM
Future

- Following interfaces have been designed for future use
  - Audio Interface
  - MMC/SD/SDIO
  - NAND Flash
  - SDRAM (FPGA/CPU)
  - Ethernet (CPU)
Firmware
Firmware Development

- Low level functions to transfer data and handle interrupts for all the blocks implemented in the FPGA
  - Universal Asynchronous Receiver and Transmitter (UART)
  - SPI Master/Slave
  - I2C Master/Slave
  - Interrupt Controller
  - Timer/Watchdog Timer
  - Real Time Clock (RTC)
  - OLED Controller
  - GPIO (Keypad)

- Interrupt handler has been developed

- CPU and Peripheral Initialization
void I2C_Configure(i2cRegs *i2cPtr, 
    guint32 i2c_clk_spd, 
    guint32 i2c_dasz, 
    guint32 i2c_dsz, 
    bool i2c_repeat_start, 
    bool i2c_addr_mode, 
    bool i2c_master)

void I2C_StartRead(i2cRegs *i2cPtr)
{
    guint32 i2c_reg;

    // I2C Read
    i2c_reg = i2cPtr->Ctrl;
    i2cPtr->Ctrl = (guint32)(i2c_reg | (1 << 12));

    // I2C Start
    i2c_reg = i2cPtr->Mode;
    i2c_reg &= 0xFFFFFFFF7;
    i2cPtr->Mode = (guint32)(i2c_reg | (1 << 3));
}

guint32 Uart_Write(uartRegs *uart, gint8 data)
{
    volatile guint32 linestatus;

    // Wait until transmitter is busy
    do {
        linestatus = uart->LSR;
    } while (!((linestatus & URT_TX_BUF_EMPTY)));

    // Transmit a given character
    uart->TBR = (guint32)data;

    return linestatus;
}
Power on Self Test

- Power on Self Test (POST) tests have been developed for all modules
  - Universal Asynchronous Receiver and Transmitter (UART)
  - SPI
  - I2C
  - Interrupt Controller
  - Timers
  - Watchdog Timer
  - Real Time Clock (RTC)
  - OLED Controller
  - Power Management Unit (PMU)
  - GPIO
  - Clock Controller
Power on Self Test

```c
trace_LOG(trace_ERROR, " Initialization Failed --\n\r");
return 1;
else {
    trace_LOG(trace_WARNING, " CPUIF(CS2) Initialized --\n\r");
}

trace_LOG(trace_WARNING, " Registers Tests --\n\r");
errCount += TimRegsTest(timPtr);
errCount += WdtimRegsTest(wdtimPtr);
errCount += IntctrlRegsTest(intctlPtr);
errCount += UartRegsTest(uart0Ptr);
errCount += GpioRegsTest(gpio0Ptr);
errCount += SpiRegsTest(spiPtr);
errCount += I2cRegsTest(i2cPtr);
errCount += UartRegsTest(uart1Ptr);
errCount += WdtimRegsTest(wdtimPtr);
errCount += MemCtrlRegsTest(memctlPtr);
errCount += RtcRegsTest(rtcPtr);
if (errCount > 0) {
    // Display on screen registers tests failed
    OledDisplayRegRdWrFail(oledPtr);
    trace_LOG(trace_WARNING, " Failed --\n\r");
} else {
    // Display on screen registers tests passed
    OledDisplayRegRdWrPass(oledPtr);
    trace_LOG(trace_WARNING, " Passed --\n\r");
}

trace_LOG(trace_WARNING, " FPGA FIQ Interrupt Enabled --\n\r");
ATF_ConfigureFTT;
```
Application Development

- The main application has been developed.
  - Main purpose is to demonstrate working of RTL
  - Application developed using the low level functions developed for each block.
  - Application is loaded into the Flash of the Processor.

- Application consists of
  - CPU initialization
  - POST tests of all blocks implemented in FPGA
  - SMS send/receive from a PC through the device to the Modem
Application Data Flow
Desktop Application

MobileCD Demo Application

Number Tab: 03459469059

Message Tab:
well come to mobileCD project.

Count Tab: 30

Unread Message IDs:
10
11
12

Count/id Tab
Received Message id Tab
Port Tab
GSM Connection Indicator
Progress Bar

SEND  CLEAR  RCV  DELETE  CONFIG PORT  6

www.palmchip.com
MobileCDFirmware Documentation

1. INTRODUCTION
   This document describes the high level design for the Mobile Communication Device project. The focus of this document will remain on the microprocessor. The document will include the communication protocol of the microprocessor with other units of the project. These include the GSM modem, FPGA and the HyperTerminals running on a PC.
   - Purpose and Scope
     The purpose of this project is to build a proof of concept board that will demonstrate its capability. For demonstration purpose and application will run on the board that will communicate with other peripheral such as GSM modem and FPGA to send and receive SMS messages.
   - Intended Audience
     This document is written of the developers to understand the high level design and how the microcontroller is interacting with other devices and the state machine running on the microcontroller.
     This document can also be useful source for project managers to understand the high level design.
   - Definitions, Acronyms and Abbreviations
     - GSM Global System for Mobile communications.
     - RISC Reduced instruction set computer.
     - ARM Advance RISC machines.
     - SAM Smart ARM-based Microcontrollers.

2. APPROACH AND STRATEGY FOR DESIGN IMPLEMENTATION
   In this project the microcontroller will control the data flow between the connected devices. It will take input from the PC and communicate with the FPGA and GSM modem and display the processed data back to the PC.
   The code will be written such that the PC can later be replaced with on board LCD and keypad without changing the design flow of the code.
   Microcontroller will communicate with the GSM modem using serial port. This serial port has been synthesized on the FPGA. The communication between microcontroller and the FPGA is done via the external memory interface of the controller.
   The controller will appear to the PC as a serial port using the USB CDC serial interface.

3. DATA DESIGN
   The controller will only save temporary the SMS data that includes the number that SMS is going to and the message. This data will be saved until the delivery status of the message is received.
   All other data coming from the serial port (GSM modem) or the USB port (PC) will be processed on the fly. The controller will parse the incoming data and perform the necessary action.

4. ARCHITECTURE AND COMPONENT LEVEL DESIGN
   The code has following main parts:
   - USB communication using CDC serial interface.
   - UART communication with GSM modem
   - Access to FPGA using external memory interface.

---

PALMCHIP
www.palmchip.com
Tools Used

- **Open Source**
  - Xilinx ISE Webpack
  - Eclipse IDE
  - GCC Compiler

- **Commercial**
  - Modelsim
  - Altium
  - Visual C++ (For PC Application)
Open Source Package

- **Package**
  - SoC Framework
    - RTL Source Code
    - Verification Environment, and Test Suites
    - Functional Specifications
    - Verification Plan
  - Board
    - Schematics
    - Gerber Files
    - Bill of Material
    - Design Specifications
    - Board Bring Up
  - Firmware
    - Low Level Function Source Code
    - Application Source Code
    - Design Specifications
Future Work

- Project has been developed keeping in mind future development
  - 2 Million Gate FPGA on board
    - Currently only 20% resources being used
  - Expansion Connector has been provided on board
    - Daughter Card can be designed and connected to the Main Board
  - Lot of interfaces designed into the board for Future IP development
  - Operating System can be Ported to the board
  - Drivers can be developed for all peripherals
Acknowledgements

- ICT R&D Fund
Thank You
The following physical interfaces have been verified on the board:

- CPU Interface
- JTAG Interface
- OLED
- GSM Modem
- UART connected to HyperTerminal
- SPI Master/Slave
- I2C Master/Slave
- GPIO (Keypad)
- SPI Flash
- SDRAM
Coverage Results

- **Statement coverage** - This indicates how many lines of the source code have been executed.

- **Branch coverage** - How many control structures (such as if-else, case statements) have been evaluated both to true and false.

- **Condition coverage** - Condition coverage analyzes the decision made in "if" and ternary statements and is an extension to branch coverage.

- **Expression coverage** - Expression coverage analyzes the expressions on the right hand side of assignment statements and counts when these expressions are executed.