Software Defined Radio (SDR) for Parallel Satellite Reception in Mobile/Deployable Ground Segments

Small Satellite Conference, Logan, Utah
11/08/2015

1Mamatha R. Maheshwarappa
2Dr Mark Bowyer & 1Dr Christopher P. Bridges

1Surrey Space Centre, University of Surrey, United Kingdom
2Airbus Space and Defence Ltd., United Kingdom
Background & Motivation
Problem Definition & Proposed Solution
Implementation & Analysis
Future Work & Conclusions
Problem Statement:
There is no generic platform/embedded solution to provide a configurable communication module to support multiple signals from multiple-satellites.
**Software Defined Radio (SDR)** – “Radio in which some or all of the physical layer functions are software defined”


Mamatha R. Maheshwarappa
**SDR Architecture**

![SDR Architecture Diagram]

- **S-Band VHF/UHF**
- **RF Programmable Transceiver SOC**
  - e.g., Linxtera LM570 quad M/A Analog Devices AD9361
- **ZYNQ**
- **Dual ARM Cortex - A9**

*Receive*
- Active IQ + DC offset correction
- Down-conversion
- Decimation

*Transmit*
- Active IQ + DC offset correction
- Up-conversion
- Interpolation
- Frequency/Time/Phase correction
- Demodulation
- Packet decoding
- User Interface
- Modulation
- Packet encoding
- User Interface
- SPI Control
- GPIO Control

Digital Interface

32 KB SRAM

I Q

32 KB SRAM

I Q

ZYNQ Diagram Description:

- **ZYNQ**: Central processing unit for SDR implementation.
- **Dual ARM Cortex - A9**: Provides processing capabilities for both receive and transmit paths.
- **Receive** Path:
  - Active IQ + DC offset correction
  - Down-conversion
  - Decimation
- **Transmit** Path:
  - Active IQ + DC offset correction
  - Up-conversion
  - Interpolation
  - Frequency/Time/Phase correction
  - Demodulation
  - Packet decoding
  - User Interface
  - Modulation
  - Packet encoding
  - User Interface
- **Digital Interface** connects the SDR components to the RF section.
Test-bed
Test-bed
Background & Motivation
Problem Definition & Proposed Solution
Implementation & Analysis
Future Work & Conclusions

Test-bed

© Analog Devices

© Phil Karn
Signal Received at Different Data Rates
1. File generated: System.bit; Tool Used: Vivado 2014.2

2. File generated: First Stage Boot Loader (FSBL.elf); Tool Used: SDK 2014.2

3. The u-boot.elf(bootloader)

4. File generated: BOOT.BIN; Tool Used: Bootgen in SDK 2014.2

5. File generated: uImage and Devicetree Tool Used: Linux Terminal

6. uEnv.txt contains the base address.

SD Card for Zedboard Boot up

(H/w & s/w lines are blurring)

Mamatha R. Maheshwarappa
### Background & Motivation

**Problem Definition & Proposed Solution**

**Implementation & Analysis**

**Future Work & Conclusions**

---

<table>
<thead>
<tr>
<th>Original Design (Software DDC)</th>
<th>With DDC Block on FPGA</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Power:</strong></td>
<td><strong>Power:</strong></td>
<td><strong>Power:</strong></td>
</tr>
<tr>
<td>• Total On-Chip Power: 2.2 W</td>
<td>• Total On-Chip Power: 2.231 W</td>
<td>• Total On-Chip Power: 1.4%</td>
</tr>
<tr>
<td>• Dynamic Power : 2.03 W</td>
<td>• Dynamic Power : 2.06 W</td>
<td>• Dynamic Power : 1.47%</td>
</tr>
<tr>
<td>• Device Static : 0.17 W</td>
<td>• Device Static : 0.171 W</td>
<td>• Device Static : 0.58%</td>
</tr>
</tbody>
</table>

| Post Implementation:          | Post Implementation:      | Post Implementation:  |
| • Flip Flop : 19%             | • Flip Flop : 19%          | • Flip Flop : 0       |
| • LUT : 24%                   | • LUT : 24%                | • LUT : 0            |
| • Memory LUT : 4%             | • Memory LUT : 5%          | • Memory LUT : 1%    |
| • I/O : 61%                   | • I/O : 61%                | • I/O : 0           |
| • BRAM : 6%                   | • BRAM : 6%                | • BRAM : 0          |
| • DSP48 : 31%                 | • DSP48 : 33%              | • DSP48 : 2         |
| • BUFG : 28%                  | • BUFG : 28%               | • BUFG : 0          |
| • MMCM : 50%                  | • MMCM : 50%               | • MMCM : 0          |

---

![Implementation on FPGA Fabric](image1.png)

![Implementation on FPGA Fabric](image2.png)

---

**Implementation on FPGA Fabric**

---

Mamatha R. Maheshwarappa

14

SSC15-VI-2
Spectrum
Background & Motivation

Problem Definition & Proposed Solution

Implementation & Analysis

Novelty & Conclusions

\[ \text{Power Received} \propto \frac{E_b}{N_0} \]

Variables:
- \( E_b \) = Energy/bit
- \( N_0 \) = Noise
- Other changes: Gains
BER v/s Eb/No

Background & Motivation
Problem Definition & Proposed Solution
Implementation & Analysis
Novelty & Conclusions

\[ T = T_R + \left( \frac{m \cdot CR_v \cdot CR_{rs}}{\text{Data Rate}} \right) + \text{Decode} + T_w \]

- \( m \) = Modulation Scheme
- \( CR_v \cdot CR_{rs} \) = Forward Error Correction Codes
- \( T_R, T_w \) = Read & Write Time
Background & Motivation
Problem Definition & Proposed Solution
Implementation & Analysis
Novelty & Conclusions

Hardware is Digital Down Converter (DDC) from profiling results
Problem statement: Need for an embedded solution to provide a configurable communication module to support multiple signals from multiple-satellites.

Proposed solution: SDR with open source hardware & software implemented on low resource embedded systems with a new pipeline architecture.

Objectives realised:
- Implementation of adaptive SDR architecture for different data rates from 1k2 to 19k2.
- Obtained performance results demonstrate the need to move blocks demanding higher computation capacity.

Potential applications:
- Ground station for multi-satellite communications.
- Deployable mobile ground station network
- Distributed satellite systems.
Mamatha R. Maheshwarappa
SDR for Parallel satellite Reception in Mobile/Deployable Ground Segments
Surrey Space Centre
E-mail: m.maheshwarappa@surrey.ac.uk
Website: http://www.surrey.ac.uk/ssc/research/onboarddata/index.htm

Questions?
Software Defined Radio (SDR) for Parallel Satellite Reception in Mobile/Deployable Ground Segments

Small Satellite Conference, Logan, Utah
11/08/2015

1Mamatha R. Maheshwarappa
2Dr Mark Bowyer & 1Dr Christopher P. Bridges

1Surrey Space Centre, University of Surrey, United Kingdom
2Airbus Space and Defence Ltd., United Kingdom
# Profiling - Transmitter

<table>
<thead>
<tr>
<th></th>
<th>Dell Optiplex 745</th>
<th>ODROID XU LITE</th>
<th>Zedboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel x86</td>
<td>ARM Cortex A15 &amp; A7</td>
<td>ARM Cortex A9</td>
</tr>
<tr>
<td>Number of Cores</td>
<td>Dual</td>
<td>Octa – Quad A15 &amp; Quad A7</td>
<td>Dual</td>
</tr>
<tr>
<td>CPU Frequency</td>
<td>2.13 GHz</td>
<td>A15 – 1.4 GHz &amp; A7 – 1.2 GHz</td>
<td>700 MHz</td>
</tr>
<tr>
<td>Linux Version</td>
<td>3.13.0</td>
<td>3.4.98</td>
<td>3.15.0</td>
</tr>
<tr>
<td>System type</td>
<td>64-bit</td>
<td>32-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td>Application</td>
<td>Dell Optiplex 745</td>
<td>ODROID XU LITE</td>
<td>Zedboard</td>
</tr>
<tr>
<td>Processor</td>
<td>Identical Application from Source</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image)

**Absolute CPU Consumption - Transmitter**

- **ARM Cortex - A9**
- **ARM Cortex - A15**
- **Intel x86**
Profiling - Receiver

Absolute CPU Consumption - Receiver

Success Rate Comparison on Different Architectures
Profiling – Receiver (Contd..)

Profiling Results on Dual Core Intel x86

Profiling Results on Octa Core ARM Cortex A15 & A7

<table>
<thead>
<tr>
<th></th>
<th>Dell Optiplex 745</th>
<th>ODROID XU LITE</th>
<th>Zedboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Compiler</td>
<td>GCC 4.8.2</td>
<td>GCC 4.8.2</td>
<td>GCC 4.6.3</td>
</tr>
<tr>
<td>No. of different Instructions</td>
<td>88</td>
<td>150</td>
<td>144</td>
</tr>
<tr>
<td>No. of Similar Instructions across the platforms</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Dominant Instructions (top 5)</td>
<td>mov(693)</td>
<td>add(238)</td>
<td>ldr(256)</td>
</tr>
<tr>
<td></td>
<td>callq(186)</td>
<td>ldr(235)</td>
<td>add.w(246)</td>
</tr>
<tr>
<td></td>
<td>add(130)</td>
<td>mov(156)</td>
<td>movw(176)</td>
</tr>
<tr>
<td></td>
<td>movss(123)</td>
<td>movw(146)</td>
<td>add(169)</td>
</tr>
<tr>
<td></td>
<td>cmp(100)</td>
<td>movt(141)</td>
<td>mov(155)</td>
</tr>
</tbody>
</table>
Implementation