DATA CODING FUNCTIONS FOR
SOFTWARE DEFINED RADIOS
IMPLEMENTED ON R3TOS

Raúl Torrego
IK4-IKERLAN
rtorrego@ikerlan.es
Introduction
  – R3TOS
  – Software Defined Radios

Implementation
  – Data Coding Functions
  – I2CI: ICAP-based Inter-task Communication Infrastructure
  – Function context saving and restoration procedure
  – Function Parameterization strategy

Results

Conclusions and future work

Questions
R3TOS (overview)

- Reliable Reconfigurable Real-Time Operating System
  - Enables HW tasks to behave like SW tasks
  - Tasks are swapped in and out of the FPGA’s reconfigurable area (real-time) via partial reconfiguration
  - Damaged resources are avoided (scrubbing techniques)

Data coding functions for SDR implemented on R3TOS

29/08/2012
R3TOS (components)

- **ICAP controller**
  - Efficient control for the Internal Configuration Access Port
  - Composed of a finite state machine and a PicoBlaze microcontroller
  - High level functions: task loading and blanking, data feeding and collection to/from tasks, scrubbing...
  - 32 bit words at 100 MHz

- **Scheduler**
  - Decides the execution order of the tasks
  - Earliest Deadline First algorithm implemented on a PicoBlaze
  - Preemptive operating system
  - True parallel task execution
**R3TOS (components)**

- **Allocator**
  - Looks for an appropriate site within the reconfigurable area of the FPGA for the tasks
  - Input: damaged resources, needed resources, occupied resources
  - Empty Area Compaction algorithm implemented on a PicoBlaze

- **Miscellaneous**
  - MicroBlaze processor (external communications, bitstream management, task generation)
  - Recovery unit
  - Configuration guardian

Data coding functions for SDR implemented on R3TOS
Software Defined Radios

### INTRODUCTION

- **Definition:**
  Communication system where a single piece of hardware has different functionalities in different times.

- Close relation with FPGA dynamic partial reconfiguration
- High flexibility (change of parameters, change of communication standard...)
- Reliable communications

Fits with R3TOS
Data Coding Functions

– First tasks in a SDR modulator
  • Present in UMTS, WiFi and WiMAX
  • Proof-of-concept of a future full SDR

– Implemented with System Generator
  • Xilinx’s rapid prototyping tool
  • Graphical programming and automatic HDL code generation

– Two different implementation
  • Individual functions (5)
  • Full-standard implementation (single task with all the functions inside)
Data Coding Functions

- Data randomizer
  - Provides a simple encryption
  - Pseudo Random Binary Sequence generator (PRBS), plus a XOR gate
  - $x^{15} + x^{14} + 1$ polynomial
  - Coefficients stored in Flip-Flops
Data Coding Functions

- Reed-Solomon encoder
  - Part of Forward Error Correcting (FEC) coding
  - Adds redundant bits
  - Detect and correct errors
  - IP + glue logic
Data Coding Functions

- Convolutional encoder
  - Part of Forward Error Correcting (FEC) coding
  - IP + glue logic
Data Coding Functions

– Puncturer

• Some bits selectively deleted
• Usually after a convolutional encoder
• Higher code rates. More flexibility
• Dual port RAM + glue logic
Data Coding Functions

– Interleaver

• Temporal diversity against burst errors
• Ease FEC decoding
• Dual port RAM + glue logic
• Interleaving pattern stored in BRAM
Data Coding Functions

- Full-standard implementation
  - Coarse grained execution
  - Simulates a complete communication standard change

Data coding functions for SDR implemented on R3TOS

29/08/2012
I2CI: ICAP-based Inter-task Communication Infrastructure

- No “wired” communication system
  - Higher area performance
  - More freedom for the allocator
- Direct data access via ICAP
  - Hardware Semaphore (LUT)
  - BRAM communication
  - System Generator link
    - Clock in, clock enable, clock enable clear
- Snake strategy
  - Reuse out data BRAM for next task
Function context saving and restoration procedure

- Preemptive operating system
- Task context needs to be saved/restored
- Issues related with Flip-Flops discovered
  - FF are not initialized after a partial reconfiguration
  - FF readback gets the initial values by default

- Context saving
  - GCAPTURE command (ICAP)
  - Updates the INIT/VALUE bits to the current state of the FF
  - Applied to the whole FPGA (caution!)
Function context saving and restoration procedure

- Context restoration
  - GRESTORE command (ICAP)
    - Applied to the whole FPGA -> Not valid
    - Working on a protection technique
  - Local reset
    - SR pin
    - Initializes FF with the values in SRMODE bit
    - Manual VHDL coding to grant access to SR
    - Connected to the HW semaphore to automate the procedure
Function Parameterization strategy

- Aims to reduce reconfiguration time (preserve tasks)
- Similar but not equal functions used
  - Different generation polynomial (conv. encoder)
  - Different interleaving pattern
  - Different data sizes
- With a correct task design
  - There is only ONE compatible function
  - Minor resource reconfiguration is carried-out
  - Small overhead may appear
- Examples
  - LUT update (1 frame) in convolutional encoder
  - BRAM update (64 frame) in interleaver
Results

- Resource utilization

<table>
<thead>
<tr>
<th>Function</th>
<th>Resource utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SLICEs</td>
</tr>
<tr>
<td>Randomization</td>
<td>58</td>
</tr>
<tr>
<td>RS encoder</td>
<td>176</td>
</tr>
<tr>
<td>Conv. Encoder</td>
<td>80</td>
</tr>
<tr>
<td>Puncturing</td>
<td>39</td>
</tr>
<tr>
<td>Interleaver</td>
<td>483</td>
</tr>
<tr>
<td>Full-standard</td>
<td>798</td>
</tr>
<tr>
<td>R3TOS (full)</td>
<td>5571</td>
</tr>
<tr>
<td>R3TOS (stand-alone)</td>
<td>1793</td>
</tr>
</tbody>
</table>

- BRAM use
- Glue logic effect in the individual functions
- R3TOS overhead
Results

– Task configuration time comparison

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameterized version</th>
<th>Normal version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv. Encoder</td>
<td>0.855 us</td>
<td>75 us</td>
</tr>
<tr>
<td>Interleaver</td>
<td>116 us</td>
<td>413 us</td>
</tr>
</tbody>
</table>

• Convolutional encoder: 1 LUT update
  - 98% reduction with parameterization
• Interleaver: 1 BRAM update
  - 70% reduction with parameterization
Results

- Task execution times

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Configuration</th>
<th>Data feeding</th>
<th>Processing</th>
<th>Data recovery</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomization</td>
<td>75 us</td>
<td>58 us</td>
<td>164 us</td>
<td>58 us</td>
<td>355 us</td>
</tr>
<tr>
<td>RS encoder</td>
<td>163 us</td>
<td>58 us</td>
<td>164 us</td>
<td>58 us</td>
<td>443 us</td>
</tr>
<tr>
<td>Conv. Encoder</td>
<td>75 us</td>
<td>58 us</td>
<td>164 us</td>
<td>58 us</td>
<td>355 us</td>
</tr>
<tr>
<td>Puncturing</td>
<td>42 us</td>
<td>58 us</td>
<td>164 us</td>
<td>58 us</td>
<td>322 us</td>
</tr>
<tr>
<td>Interleaver</td>
<td>413 us</td>
<td>58 us</td>
<td>168 us</td>
<td>58 us</td>
<td>697 us</td>
</tr>
<tr>
<td>Total funct. by funct.</td>
<td>768 us</td>
<td>348 us</td>
<td>824 us</td>
<td>348 us</td>
<td>2,32 ms</td>
</tr>
<tr>
<td>Total funct. by funct. (Snake)</td>
<td>768 us</td>
<td>116 us</td>
<td>824 us</td>
<td>116 us</td>
<td>1,82 ms</td>
</tr>
<tr>
<td>Full-standard design</td>
<td>673 us</td>
<td>116 us</td>
<td>227 us</td>
<td>116 us</td>
<td>1,13 ms</td>
</tr>
</tbody>
</table>

- Full-standard execution obtains the best time (fully parallel)
- The snake strategy reduces the execution time a 20%
- Execution of one task at a time (time for 2 tasks at a time = 1.63 ms)
- Smaller task ease allocator’s work
Conclusions

- R3TOS is suitable for SDR implementations
- The SDR features take care of a secure communication while R3TOS guarantees efficient and reliable hardware utilization
- Importance of parameterization

Future work

- Improve the state saving and restoration procedure
- Implement the remaining functions that make up a whole SDR system
- Develop a design methodology that chooses the optimum task size