DRM

HW/SW Co-Design of Viterbi Decoder

Will O’Donnell and Juan Garcia

Agenda

- Overview
- System Design
- SW Design
- HW Design
- Metrics
- Summary and Conclusion

Overview

- Introduction
- Market & Product Requirements
- Team Assignment

Introduction

- DRM - Digital Radio Mondiale
  - Digital replacement for analog radio
  - Uses Viterbi to encode/decode signals
- Hardware acceleration vs. full software implementation
- Design effort worth the effort?

Market & Product Requirements

- DRM SoC to integrate into MP3 player or 4G cell phone
- Transmit/receive data in the AM bands
- Flexible and scalable platform based architecture

Team Assignment

Will

- VD sim and syn
- In/Out buffer implm.
- RAM optimizations
- Integration (top.v)
- FPGA debugging

Juan

- HAL/Device Driver
- Bus Interface
- Program and Test FPGA

Agenda

- Overview
- System Design
- SW Design
- HW Design
- Metrics
- Summary and Conclusion

System Design

- Algorithm Design and Profiling
- Scheduling and Optimizations
- SW & HW Partitioning

Algorithm Design and Profiling

Profiling DRM code - gprof

- Code Versions
  - Floating point and Fixed Point versions
- Platforms
  - Complied/executed in Linux and ARM
  - OVP/SystemC sim model of TLL5000-TLL6219
- Bottlenecks
  - Viterbi Decoder
  - Floating point Arithmetic

Scheduling and Optimizations

CatapultC Synthesized Code Optimizations

- two loops that are of primary interest: METRIC_INIT_VALUE_LOOP & BUTTERFLY_LOOP
- found the optimum unroll values for each loop individually
- constraints of Unrolling two loops at the same time
- Pipelining was also attempted but was not possible due to dependencies and resource constraints

SW/HW Partitioning

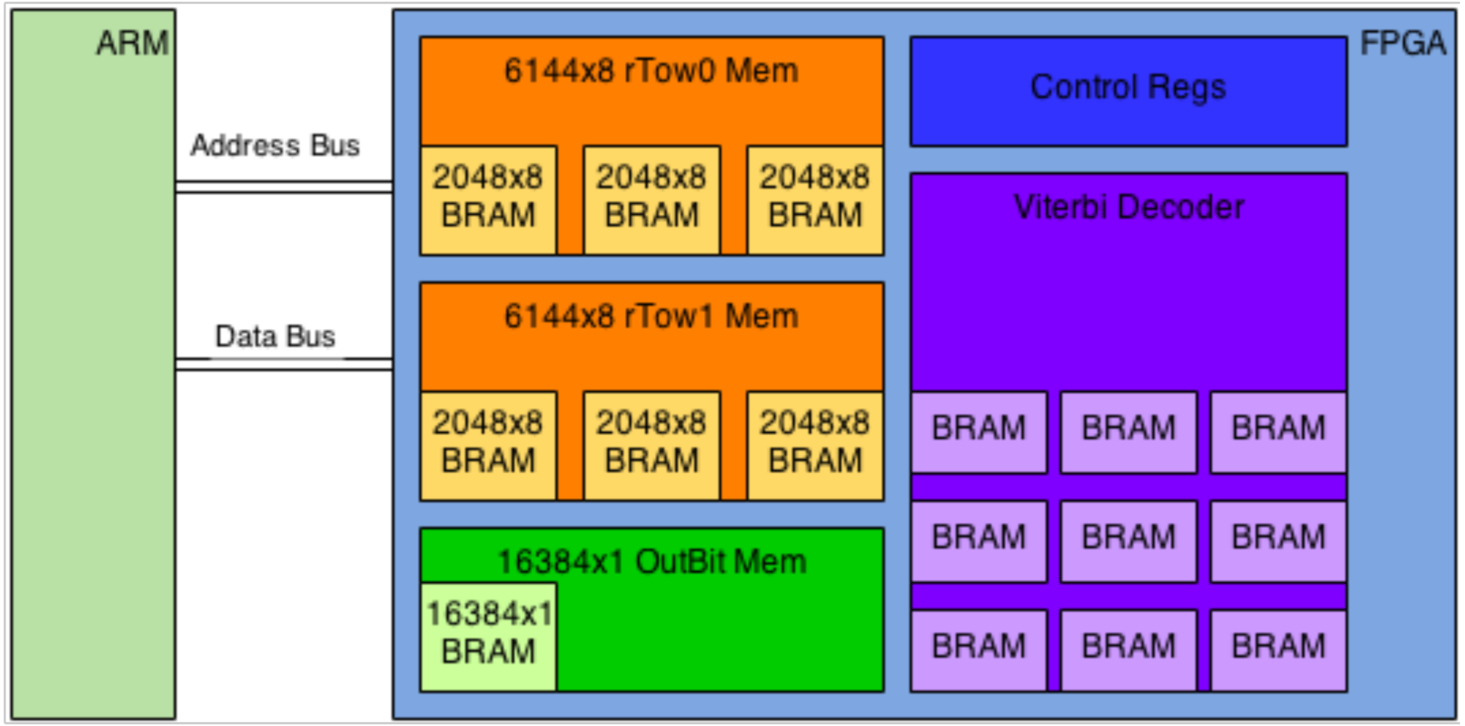
Software

- Manage initialization
- Input data packaging
- Output data unpackaging

Hardware

- Puncture pattern generation
- Butterfly algorithm
- Input/Output buffer

System Block Diagram



Agenda

- Overview
- System Design
- SW Design
- HW Design
- Metrics
- Summary and Conclusion

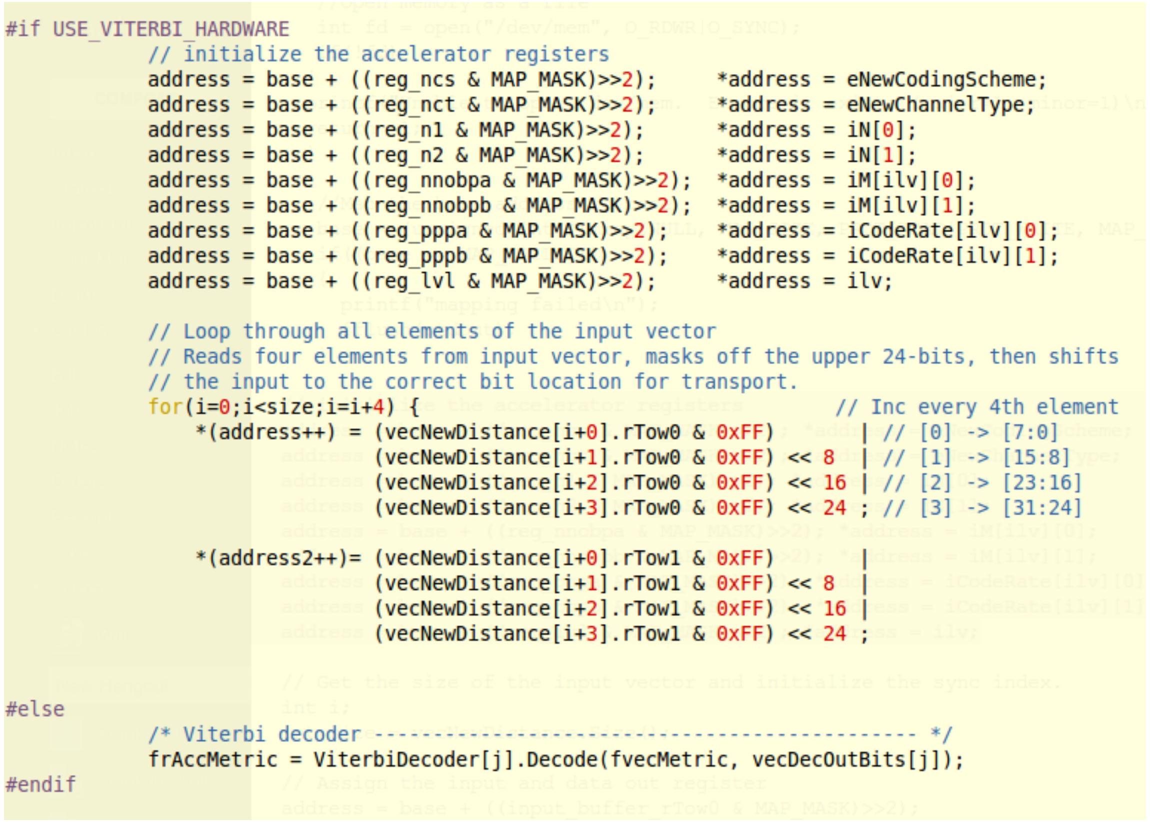
SW Design

- Software Implementation
- External SW/HW Communication Interface
- Data Packaging

Software Implementation

- Stand-alone Viterbi
  - Compiled with g++
  - ViterbiDecoder.cpp
  - Header files
- Test bench
  - Quick functional testing
  - Confirm optimizations didn’t break functionality

Software Implementation



```
#if USE_VITERBI_HARDWARE
        // initialize the accelerator registers
        address = base + ((reg_ncs & MAP_MASK)>>2);     *address = eNewCodingScheme;
        address = base + ((reg_nct & MAP_MASK)>>2);     *address = eNewChannelType;
        address = base + ((reg_n1 & MAP_MASK)>>2);      *address = iN[0];
        address = base + ((reg_n2 & MAP_MASK)>>2);      *address = iN[1];
        address = base + ((reg_nnobpa & MAP_MASK)>>2);  *address = iM[ilv][0];
        address = base + ((reg_nnobpb & MAP_MASK)>>2);  *address = iM[ilv][1];
        address = base + ((reg_pppa & MAP_MASK)>>2);    *address = iCodeRate[ilv][0];
        address = base + ((reg_pppb & MAP_MASK)>>2);    *address = iCodeRate[ilv][1];
        address = base + ((reg_lvl & MAP_MASK)>>2);     *address = ilv;

        // Loop through all elements of the input vector
        // Reads four elements from input vector, masks off the upper 24-bits, then shifts
        // the input to the correct bit location for transport.
        for(i=0;i<size;i=i+4) {                                       // Inc every 4th element
            *(address++) = (vecNewDistance[i+0].rTow0 & 0xFF)       | // [0] -> [7:0]
                           (vecNewDistance[i+1].rTow0 & 0xFF) << 8  | // [1] -> [15:8]
                           (vecNewDistance[i+2].rTow0 & 0xFF) << 16 | // [2] -> [23:16]
                           (vecNewDistance[i+3].rTow0 & 0xFF) << 24 ; // [3] -> [31:24]

            *(address2++)= (vecNewDistance[i+0].rTow1 & 0xFF)       |
                           (vecNewDistance[i+1].rTow1 & 0xFF) << 8  |
                           (vecNewDistance[i+2].rTow1 & 0xFF) << 16 |
                           (vecNewDistance[i+3].rTow1 & 0xFF) << 24 ;


#else
        /* Viterbi decoder ------------------------------------------------ */
        frAccMetric = ViterbiDecoder[j].Decode(fvecMetric, vecDecOutBits[j]);
#endif
```

External SW/HW Communication

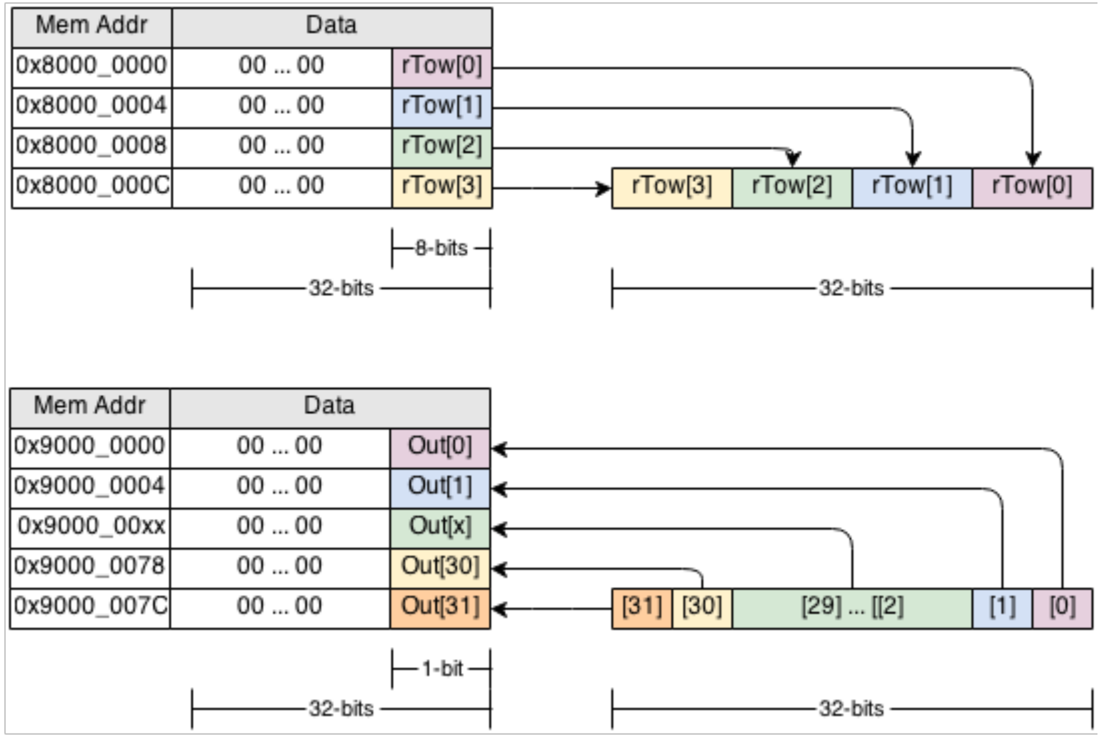
Memory-Mapped I/O

- Function arguments into registers
- Input/output buffers into block rams
- Base address 0xd3000000

Example Mem. Map

- reg_run: 0x0000
- reg_ncs: 0x0010
- reg_n1: 0x0018
- rTow0: 0x2000-37FF

SW Data Packaging



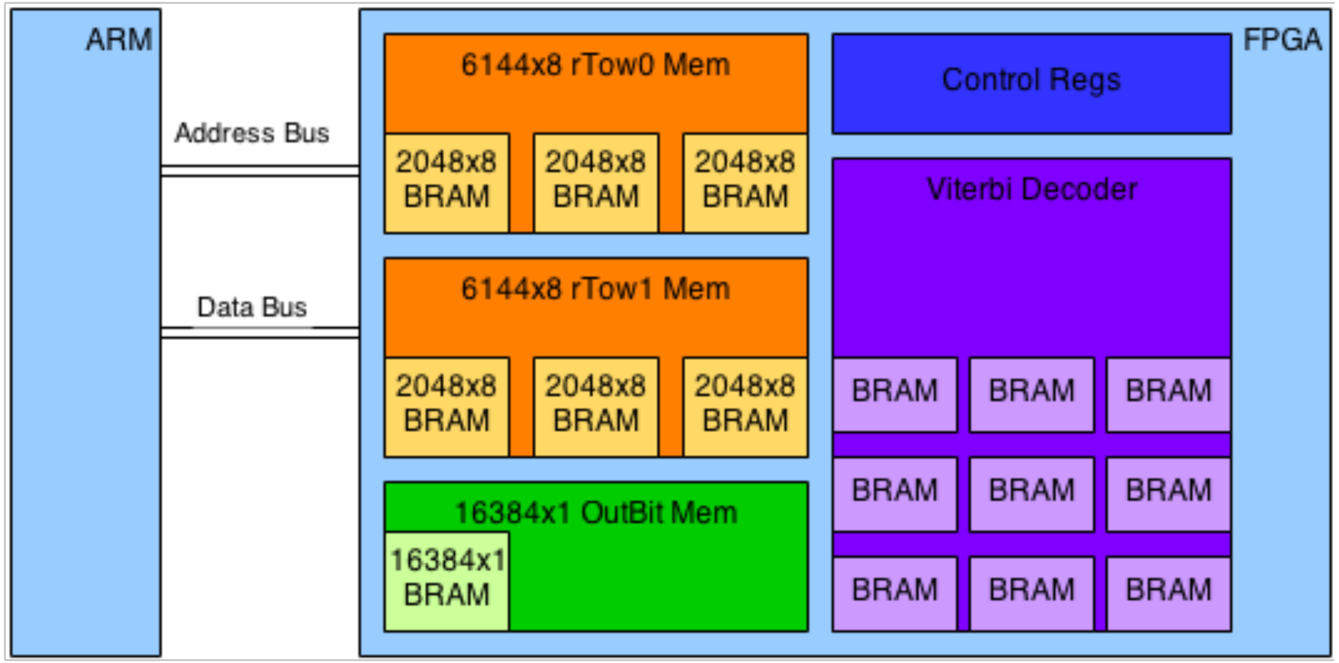
Agenda

- Overview
- System Design
- SW Design
- HW Design
- Metrics
- Summary and Conclusion

HW Design

- Accelerator Architecture Overview
- Hardware Implementation
- External HW/SW Communication Interface

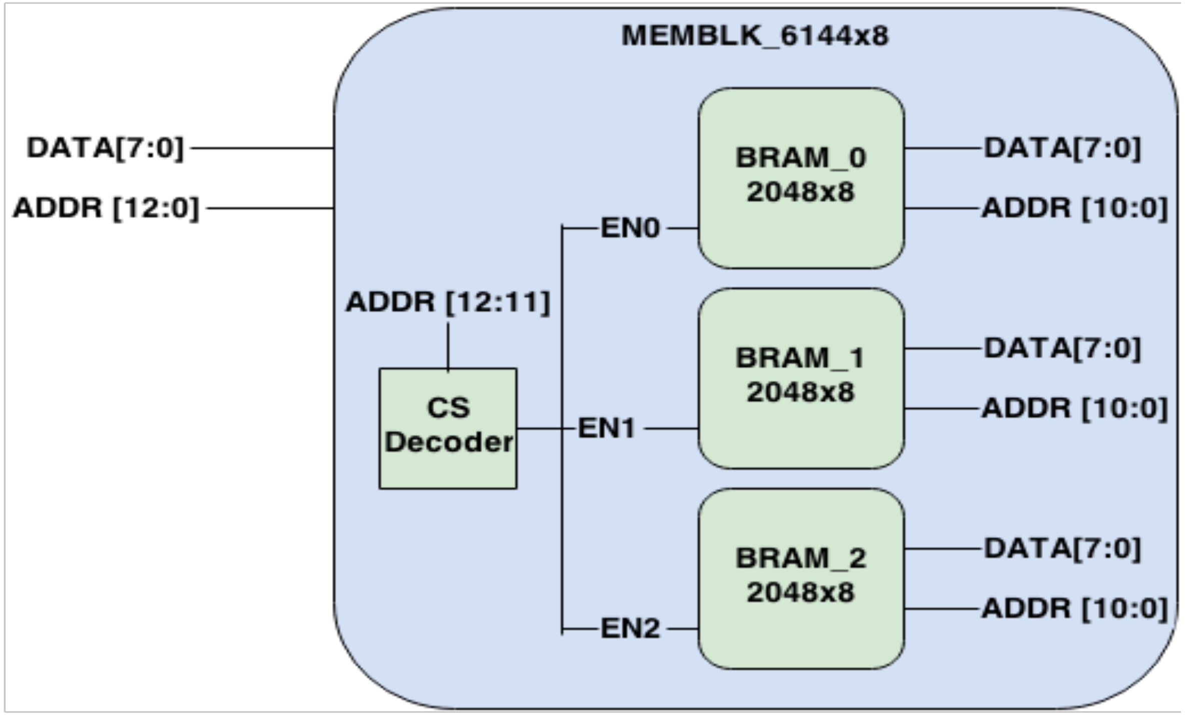
Accelerator Architecture Overview



Hardware Implementation

- Catapult generated RTL from C++
- Precision synthesized RTL
  - Target Xilinx Spartan-3 XC3S1500
  - 32 block 16Kb Block RAMS
- Optimizations
  - Rolled-up Butterfly logic into for-loop
  - Packaged input/output data

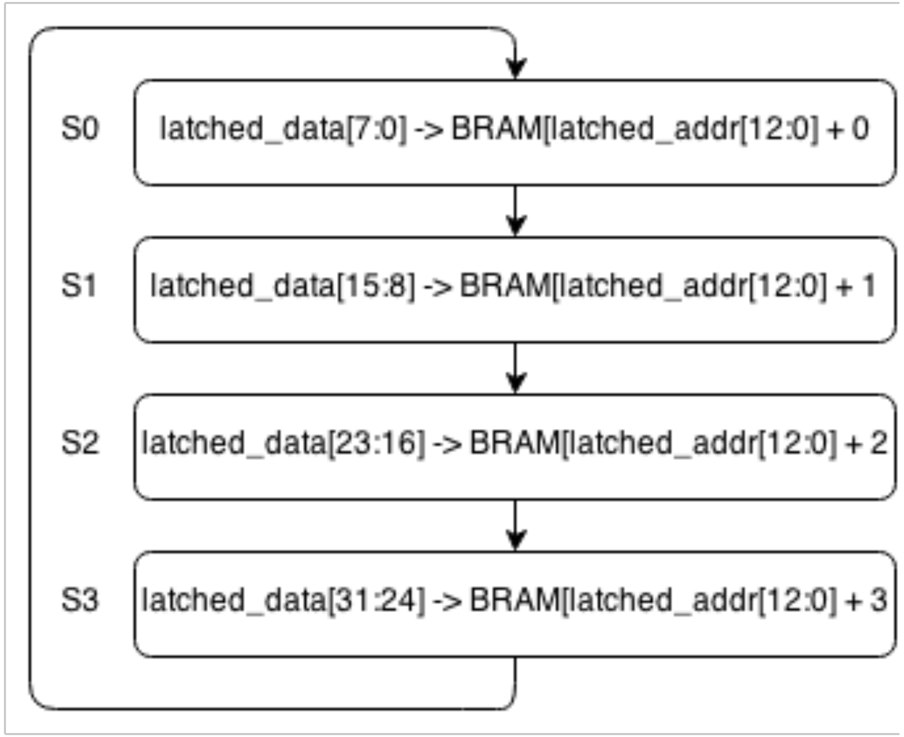
Hardware Implementation



Hardware Implementation

Resource	Used	Available	Utilization
IOs	100	487	20.53%
Global Buffers	2	8	25.00%
LUTs	1661	26624	6.24%
CLB Slices	831	13312	6.24%
Dffs or Latches	1008	28085	3.59%
Block RAMs	32 (actually 26+7)*	32	100.00%
Block Multipliers	0	32	0.00%
Block Multiplier Dffs	0	1152	0.00%

External HW/SW Communication


S0
latched_data[7:0] -> BRAM[latched_addr[12:0] + 0
S1
latched_data[15:8] -> BRAM[latched_addr[12:0] + 1
S2
latched_data[23:16] -> BRAM[latched_addr[12:0] + 2
S3
latched_data[31:24] -> BRAM[latched_addr[12:0] + 3

FPGA Testing

- Synthesized RTL ready for test
- DRM HAL and device driver not quite ready
- Not able to test or verify functional design on board

Agenda

- Overview
- System Design
- SW Design
- HW Design
- Metrics
- Summary and Conclusion

Metrics

- Functional Verification
- Area Estimation and Power Estimation
- Optimization

Functional Verification

- Viterbi SW verified
  - Golden input/output from OVP simulation
  - Verified in stand-alone executable
- Viterbi HW planned to use Verilog testbench
  - Send golden input
  - Compare Viterbi return buffer with golden buffer
  - Use pass/fail signal to indicate failure

Area and Power Estimates

Area

- Total: 29718 um^2
  - ~0.02 0.172 x 0.172 mm^2
  - 0.172 x 0.172 *mm*
  - *Memory not included*

Power

- Dynamic: 3.51 *mW*
- Leakage: 106 *nW*

Using Design Compiler 45nm Artisan library and typical-cell database

Optimization Opportunities

- Streaming solution
  - Efficient BRAM usage
- Puncture Pattern
  - Repeating pattern
- Ping-pong buffers
  - SW and HW busy simultaneously

Agenda

- Overview
- System Design
- SW Design
- HW Design
- Metrics
- Summary and Conclusion

Summary and Conclusion

- SW model verifies functionality for HLS
- Use Catapult to generate RTL
- Synthesized to FPGA
  - Used 32 Block Rams
  - Design fit into Spartan-3 XC3S-1500
- FPGA prototyping not complete

Summary and Conclusion

Lessons Learned (JG)

- Learn how to program effectively
  - Insert error checking as you go
  - Practice C++ & HDL
- Methodical system level design
  - Power in OOP: ESL, TLM
  - HLS - rapid architecture exploration

Summary and Conclusion

Lessons Learned (WO)

- Early optimizations will reduce design cycles
- Profiling is key to identifying optimization opportunities
- Optimizing black box is difficult
- Understand tools to be efficient
- Test bench at each stage of integration

Questions?