Synthetic Aperture Radar on Low Power Multi-Core DSP

Dan Wang, Murtaza Ali

Texas Instruments
Outlines

• Basics of Synthetic Aperture Radar (SAR)
• *TMS320C6678* Architecture
• SAR System Implementation on DSP
  − Modulization
  − Data flow
• Implementation Profiling
  − Module profiling
  − Single core vs. multi-core
  − Comparison with alternative platforms in literature: GPU, FPGA
• Conclusion
SAR Geometry

• Use one antenna in time-multiplex

• Use Doppler shift to obtain fine azimuth resolution

• Two dimensions
  – Range (cross-track, fast time)
    ▪ Line-of-sight distance from radar to target
  – Azimuth (along-track, slow time)
    ▪ Parallel to radar motion track

SAR Algorithm

• SAR types
  – Airborne or spaceborne
  – Strip SAR, spot SAR, etc.
  – Platforms: CPU, GPU, FPGA, etc

• Diverse algorithms
  – Range-azimuth algorithm
  – Chirp-scaling algorithm
  – Ω-k algorithm

• Range-azimuth algorithm
  – Achieve block processing efficiency
  – Separability of processing in two directions
  – Limited for low squint case

Strip mapping
Spot mapping
Multicore DSP (TMS320C6678): Functional Diagram

- Multicore KeyStone architecture SoC
- Fixed/Floating corePac
  - 8 CorePac @ 1.25 GHz
  - 4.0 MB Shared L2
  - Performance: 320GMAC, 160GFLOPs, 60GDFLOPs
  - Power ~10W@1GHz
- Navigator
  - Queue Manager, Packet DMA
- Multicore shared memory controller
  - Low latency, high bandwidth memory access
- 3-port GigE switch (Layer 2)
- PCIe gen-2, 2-lanes
- SRIO gen-2, 4-lanes
- HyperLink
  - Support connection to other keystone devices providing resource scalability
  - Provide a 50Gbps chip-level interconnect
C66x – Core Architecture

• VLIW architecture
  – Can issue 8 instructions per cycle

• 2 data paths
  – 4 units per data path
  – L, S, D, M
  – Access cross data path

• 64 registers (32 bit)
  – 32 per data path
  – Can be arranged in dual (64 bit) or quad (128 bit) registers

• Single Instruction Multiple Data (SIMD)
  – Dual or quad multiplies (64 or 128 bits)
Multicore Performance Single Core Simplicity

Develop

- Code Composer Studio
- Code Generation Tools
  - Compiler, Linker
- Programming Model
  - MCSDK, OpenMP, OpenCL

MCSDK
OpenMP
OpenCL

Customer Applications
- Master Thread
  - Slave Thread
- Master Thread
  - Slave Thread
- Master Thread
  - Slave Thread
- Master Thread
  - Slave Thread

OpenMP Programming Layer
- Compiler Directives
- OpenMP Library
- Environment Variables

Application Binary Interface
- OpenMP ABI
- OpenMP Programming Interface

Parallel Thread Interface

Operation System
- SYS/BIOS

Memory Subsystem
- CorePac
- C66X DSP
- L1
- L2
- C66X DSP
- L1
- L2
- C66X DSP
- L1
- L2
- C66X DSP
- L1
- L2
- C66X DSP
- L1
- L2

Network CoProcessors
- Crypto
- Packet Accelerator

IP Interfaces
- GbE Switch
- SGMII

Peripherals & IO
- SRIO
- PCIe
- EMIF
- TSP
- SPI
- UART

System Elements
- Power Management
- Slotbus
- SGLA

TeraNet

Code Composer Studio

HyperLink

8 x CorePac

Texas Instruments
• 1 c6678
• 160 Gflops
• 1GByte DDR3
• 10W

• 4 c6678
• 512 Gflops
• 4GByte DDR3
• 54W

• 8 c6678
• >1 Teraflop
• 16 GByte DDR3
• 110 W
Range-azimuth Algorithm

- Main steps
RCM Correction

• Cause
  - Instantaneous range change leads to variation in the range delay that could be larger than a range sample space

• Range-Doppler correction
  - Interpolation in range-Doppler domain
Range-azimuth Algorithm Implementation
Data Flow

RCMC and Azimuth Compression

[Diagram showing the data flow process with nodes labeled as: RCMC, Azimuth, Compression, FFT, MUL, IFFT, DDR3, Sinc coeff. table, Twiddle, ref, L2, in_B, out_B.]
Multicore Mapping

- Allocate memory for each core
- OpenMP for multiple threads running simultaneously
- DMA read from DDR3
- Local processing
- DMA write to DDR3
Implementation Profiling

• Setups
  – TMDXEVM6678L
  – Compiler: v7.4.0.A12012
  – OpenMP: v1.0.0.34_eng
  – DSPLIB:v 3.1.0.0, (FFT, complex transpose)
  – L1D cache 32K, L1P cache 32K
  – L2 cache: 128K
  – Data size: 4096*4096, complex single precision
Module Profiling

• Five modules
Module Profiling

- DDR3 transfer time: 19 ms
- Transpose saturate around: 34 ms

Graphs showing:
- Range comp. 7.9x
- Azimuth comp. 7.9x
- Azimuth FFT 4.5x
- RCMC 4.0x
- Transpose 2.9x

Time in ms vs number of cores.
Total Time

- 1.4 s
- 1 core
- 8 cores = 5.6
- 0.25 s
- DDR3 time = 94 ms
- 94 ms = 37.6%
- 250 ms
Time Breakdown

- Range compression: 29%
- Transpose: 14%
- Azimuth FFT: 18%
- RCMC: 27%
- Azimuth compression: 13%
Comparison

• GPU benchmark: Nvidia Tesla C1060 (Bisceglie’10)
  – Core clock= 1.296GHz; Processor core #=240; memory= 4GB @ 800MHz
  – Testing algorithm: Range-azimuth algorithm, FFT size 4096

• FPGA: Xilinx VIRTEX-5 (Pfitzner’11)

• Comparison
Image Example

RADARSAT: 38km*23km
Conclusions

• Contributions
  – Implementation of range-azimuth algorithm for SAR on TMS320C6678
  – Parallel processing on multiple DSP cores
  – Provide benchmark results for a typical 4k by 4k SAR image
  – Achieve real-time performance with 4 frames/sec

• Next steps
  – Extra modules
    • Doppler parameter estimation, geometry information
    • Auto focusing, computational intensive
  – Larger size FFT (>4K)
  – Evaluation with 4 DSPs connected to server with PCIe
Thanks & Questions