A Mobile Station Modem Chip for WCDMA

David Hansquine
Director, Engineering
2001 Hotchips Symposium
August 19-21, 2001
Agenda

- Overview & Chip Features
- Chip Architecture
- Problems in Designing Wireless Modems
- Power Savings Games
- Clocks
- Making the Design Flexible
- Bit Exact Simulations
- Testing & Verification
- Chip Development
Overview

• First generation WCDMA Mobile Station Modem (MSM) testchip
• What does an MSM do?
  • Contains all the digital logic necessary for a mobile phone and tightly coupled analog interfaces
  • CDMA modulation/demodulation
  • Contains Microprocessor & DSP for running applications
• What’s WCDMA?
  • Original phones were voice-only, no data
  • Second generation had limited data support (9600 bps upwards to 112 kbps)
  • WCDMA, CDMA2000, 1xEV are new 3G standards
  • Support high data rates (153.6 kbps up to 2.4 Mbps)
  • Enable new services:
    » Data (web-browsing, downloads)
    » Video on demand, videoconferencing
    » Multi-media services
Phone Architecture

RFR = RF receive chip
RFT = RF transmit chip
PA = power amplifier
PM = power management chip
MSM Features

- 3GPP Release 99 compliant
- Supports 384 kbps uplink & downlink (send & receive)
- Design is based on existing MSM products
- Much more DSP-controlled with all firmware in RAM
- Not intended to be the final product
Block Overviews (1)

- **Modem DSP (MDSP):**
  - Controls WCDMA engine
- **Vocoder DSP (VDSP):**
  - AMR vocoder
  - Runs applications (voice recognition, echo cancellation, music, etc.)
- **ARM7TDMI:**
  - Runs protocol stack for WCDMA
  - Runs applications (phone book, web browsing, image & sound downloads, games, etc.)
- **WCDMA Engine:**
  - Hardware for Searcher, Demodulator, Deinterleaver, Viterbi decoder, Turbo decoder, Modulator
- **System Timer:**
  - Keeps track of system time and aligns events in chip
  - Denotes periodic intervals by generating strobes or interrupts
Block Overviews (2)

- **Sleep Controller**
  - Controls disabling/enabling of various blocks in the chip when the phone enters/exits idle mode

- **Clock Control**
  - Selects source for each clock regime and if it is on/off

- **Peripherals**
  - USB, UARTs, SBI

- **Integrated CODEC**
  - Earphone and microphone analog interface

- **Integrated HKADC**
  - General purpose ADC for battery and temperature sensing

- **Analog RX & TX interfaces**
  - Integrated ADC and DAC for antenna interface
Problems in Designing Wireless Modems

• What’s different about designing a wireless modem?
  • Power
  • Clocks
  • Keeping the design flexible and upgradable
  • Bit Exact Simulations
  • Testing & Verification
Power Savings Games

- Focus is on low power operation (active & idle modes)
  - Chip is powered by a battery

- Run clocks at lowest speed necessary to reduce power
- Independently enable clocks per core or function
- Software has ultimate control over each regime
  - Hardware can turn off a regime locally if circuitry is not needed
    » Turbo decoder vs. Viterbi decoder, only one needed at a time
    » Not all fingers in Demodulator are constantly active
- Treat rd_n and wr_n as clocks
  - There are many thousands of software register bits
  - These are clocked with rd_n/wr_n which pulse during read or write
- Divide microprocessor bus into two
  - WCDMA blocks on one bus, peripherals on separate bus
Chip Buses

- ARM7TDMI
- DMA
- MDSP BRIDGE
- MDSP
- WCDMA ENGINE
- SYSTEM TIMER
- SLEEP CTRL
- CLOCK CTRL
- USB
- UARTs
- SBI
- MISC
- AUXMSM BRIDGE
- VDSP BRIDGE
- ARM PERIPHS

Buses:
- DMA Bus
- MSM Bus
- ASB
- AUXMSM Bus
- MDMA Bus
Problems with Clocks

- Many peripheral interfaces each clocked independently
- Clocks should be controllable (enable/disable each cycle)
- Many clocks can be sourced from multiple sources
- Meeting timing constraints
- Many, many clocks to deal with
- Design Targets:
  - TCXO (13, 19.2, 19.8 MHz) sources PLL to generate 122.88 MHz (wchipx32)
  - WCDMA cores: chipx8=30.72 MHz, chipx16=61.44 MHz
  - ARM: max 40.96 MHz
  - MDSP: max 61.44 MHz
  - VDSP: max 40.96 MHz
  - USB: 48 MHz
  - Misc & peripherals: TCXO, max 19.8 MHz
  - Sleep controller: 32 KHz
Clock Generation (1)

- Buffered clock tree
- Each tree is sourced by a Clock Source Cell (CSC)
  - Metal programmable delay to enable skew control
  - Muxes normal clock with test clock to enable ATPG
  - Supports gating the clock tree on cycle by cycle basis
Clock Generation (2)

- HSpice is used to analyze clock nets to program CSC delay
- Should we balance all clock trees?
  - Increases insertion delay
    » Smaller regimes need to be delayed as much as larger regimes
  - Increases skew
    » Harder to balance insertion delays
    » Register to register paths are likely longer
- Analysis takes longer
- Only balance clock trees which have synchronous signals going between them
  - Resynchronize or half-latch signals going between non-balanced clock regimes
Clock Generation (3)

- In ATPG mode, the clock source is the same for all clock trees
  - Do we need to then balance all clock trees anyways?
- Solution:
  - Disallow ATPG over circuit boundaries from regime to regime
    - Reduces coverage
  - Insert resynchronization logic at boundaries
    - Increases area
    - Need to figure out where to insert them and not miss even one since simulations may not catch skew problems
    - This is what we did
- In the WCDMA test chip, we have:
  - 17 separately configurable clock sources
  - 71 CSC’s
Clock Skew

- DSP target speed is 61.44 MHz = 16.28 ns period
- Initial clock analysis results showed a skew of 1.4 ns
- Skew affects amount of setup time and hold time to budget
- Options to reduce this:
  - Split clock tree into two
    » Reduces load of each tree but have more main branches
    » Reduced skew to 720 ps
  - Wide metal routing for clock nets
    » Silicon Ensemble bug prevented this from being used in time for tapeout
    » After tapeout with new Silicon Ensemble version this path showed skew reduces to 470 ps
Placement & Timing (1)

- Timing driven placement used
- Several passes of Netlist processing and Place and Route are done excluding passes for bug fixes:
  - Netlist ECO1: add test support
  - P&R Pass1: run first placement, registers are now fixed
  - Netlist ECO2: using placement info build clock tree
  - P&R Pass2: redo placement of combinatorial cells, route, then extract parasitics
  - Netlist ECO3: using parasitics, do clock analysis and balance skew by programming CSC’s, fix hold time problems
  - P&R Pass3: Update CSC’s, redo placement of combinatorial cells with fixes for hold time issues, redo routing
Placement & Timing (2)

- Combinatorial cell placement is not fixed until Pass3
- Timing varies from run to run as much as 100% !!!
- Create rudimentary physical compiler:
  - Parses a config file and primetime report files
  - Outputs a netlist ECO file to increase drive strengths of cells with big delays and/or insert buffers
  - Each P&R pass can show considerable timing differences
    » Run this tool on all primetime output files and use the output file for the next netlist ECO
    » Fixed about 400 cells during each early pass and 150 cells in the final pass
Making the Design Flexible

- Problems:
  - Standards evolve, new features come up
    - WCDMA standard has had many revisions since project started
    - Hundreds of change requests (additions/modifications)
    - Standards subject to interpretation
    - No fully functional system to compare with
    - Once systems are up, likely need to reoptimize algorithms
  - This goes into a consumer device, very expensive to recall
    - Ideally want to fix problems with SW upgrade

- Solutions:
  - Complex algorithms and portions of standard likely to change are done in firmware which is stored in RAM
  - Overdesign: design for worst-case standards interpretation
  - Many of the constants are programmable (eg. filter coefficients)
  - Lots of flexibility for programming clock sources
    - Provide limitless options to maximize power savings
Bit Exact Simulations (1)

- Problem:
  - Since DSPs and ARM run asynchronous to WCDMA engine, it’s hard to do bit exact testing
- Solutions:
  - Essentially all a modem chip does is take in a receive (RX) stream, and output a transmit (TX) stream
  - These are synchronous interfaces running independently from the ARM and DSP cores
  - If the ARM or DSP reads/writes the WCDMA engine, the precise cycle relative to the RX or TX samples is not precise
  - On the RX side, we need to initialize timers based on when external samples start, and trigger tasks at fixed times relative to that
  - On TX side, just need an indication of when the first sample is coming out (a framing boundary)
Bit Exact Simulations (2)

• To align external world with internal timing of chip, the following are provided:
  • All ARM+DSP programmable registers take effect at periodic intervals (eg. every 2048 clock cycles)
    » If ARM and DSP can guarantee task is initiated during that interval, it takes effect at precisely the next boundary
  • Bypass ADC and accept digital RX inputs
    » resync input is used with samples to align internal timers
  • Bypass DAC and transmit digital TX outputs
    » tx_stb output aligns outside world with TX samples
Testing & Verification

• Problems:
  • Standards documents are huge, sometimes subject to interpretation-- this is a first generation WCDMA product
  • Software plays a big role in how the chip is used
    » Many things happening at once at different times
    » Realtime system requirements need to be met
    » ARM & DSPs control the WCDMA engine
  • What works at physical layer may not work in full system
  • System performance can only be determined once the entire system (digital & RF H/W and S/W) is in place

• Solutions:
  • Higher pincount package to pin out a testbus
  • Supports modes for debugging MDSP, VDSP and ARM
    » Multiple modes depending on need/intrusiveness
  • Consistent method for testing on different platforms (simulations, chip tester, phone platform)
ARM & DSP Debug

• Problems:
  • Since ARM+DSPs are imbedded in chip and control the WCDMA engine, challenges are:
    » Visibility for debug, monitoring progress, checking results
    » Control for testbench interaction

• Solutions:
  • Message passing system implemented over the testbus
    » ARM and DSP can communicate with testbench
      – Can trigger events in testbench
      – Provide internal results to testbench to print or save to file
  • Testbench also monitors (prints to screen/file) either of:
    » All ARM accesses to internal buses
    » All DSP memory accesses
Overall Test Flow

• How do we test the chip consistently across platforms?

• Systems team provides two configuration files for a test containing register reads & writes, wait for events:
  • One for MDSP
  • One for ARM

• For blocklevel sims, these are converted into separate processes for the testbench which drive the ARM and MDSP interface buses

• For toplevel sims, these are converted into assembly for DSP, and C code for ARM

• Result is that the same system config files are used for both blocklevel and toplevel

• The same files are used for generating test files for the 3rd party testbox driving the phone platform
Design Flow

- Primarily VHDL
- Custom core for ARM
- Custom datapath logic for DSP
- Modeltech for simulations
- Synopsys for synthesis
- IKOS for gatelevel simulations
- Silicon Ensemble for place & route

- Many internal tools are used between each stage including test insertion, scan chain stitching, buffer resizing, floorplanning
Digital Testing

• ATPG
  • In current package 91 parallel chains are used
• Functional vectors for at-speed testing
• Memory tests
  • Common memory test SW interface used
• Tricks for test time reduction
  • ATPG MISR
    » Enables doubling of parallel chains
    » Reduces number of outputs to be checked
  • RX pattern generator
    » Fewer inputs to drive
  • TX MISR
    » Fewer high-speed outputs to sample
Analog Core Testing

- Analog test mode:
  - Pins out digital inputs & outputs of each analog core
  - There are no internal analog interface signals that need to be pinned out
- Cores can be tested independent of the rest of the chip
Floorplan
Test Chip Info

• Statistics:
  • Technology: 0.18 μm to two foundries
  • Die size: 10.5 mm x 10.5 mm
  • 45% RAM/ROM and 55% Std Logic
  • 320 pin package

• Current Status:
  • Clocking options & testbus turned out to be very much needed!
  • Makes a voice call with 3rd party test equipment