

High-Level Synthesis of a Queue Manager Block - a Case Study

Maneesh Soni

Akila Subramaniam

Texas Instruments, Inc.

Per Edstrom

Steve Brown

Cadence Design Systems, Inc.

Motivation for High Level Synthesis

Design productivity

- Code at algorithmic level
- Easy to modify design and optimize micro-architecture
- Faster architecture definition, iteration, implementation
- Rapid Prototyping for SW and performance analysis

Verification productivity

- Fewer RTL bugs due to higher level of abstraction
- Faster simulation
- Improved complexity management

Requirements For HLS Adoption

Return on Investment

- Equal or better than hand-coded
- Order of magnitude improvement in design productivity
- Equal or better workload for verification

Maturity and ease-of-use

- Help designer understand when a design is optimal
- Unified/integrated solution for control and data-path
- Extensive debug support and visibility into high level implementation
- Tight coupling with industry accepted synthesis tools
- Equivalence Check
- Incremental synthesis for ECO
- Clean RTL output
- Integration of structural components (memories, etc.)

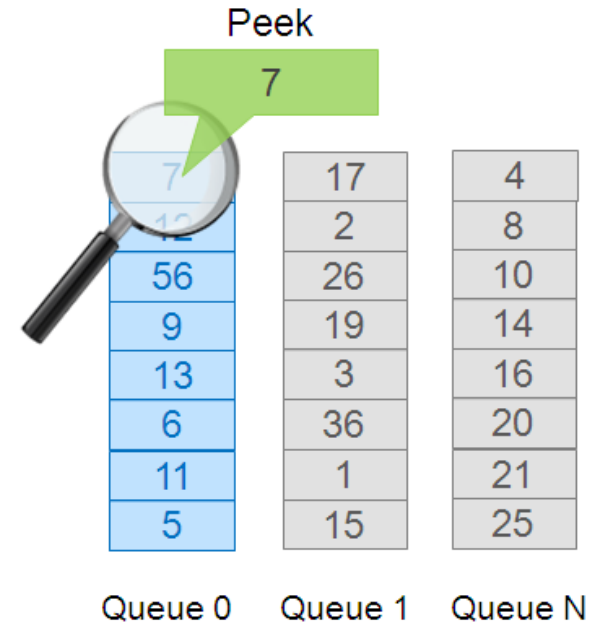
Queue Manager Functional Overview

Manages descriptors and pointers across multiple queues

- Push
 - Queue descriptor to a queue
 - Write sideband Information
- Pop
 - De-queue a descriptor from a queue
 - Read sideband information
- Peek
 - Just look at what is on a queue
- Status
 - Per queue status based on preset thresholds

Why use Queue Manager for case-study?

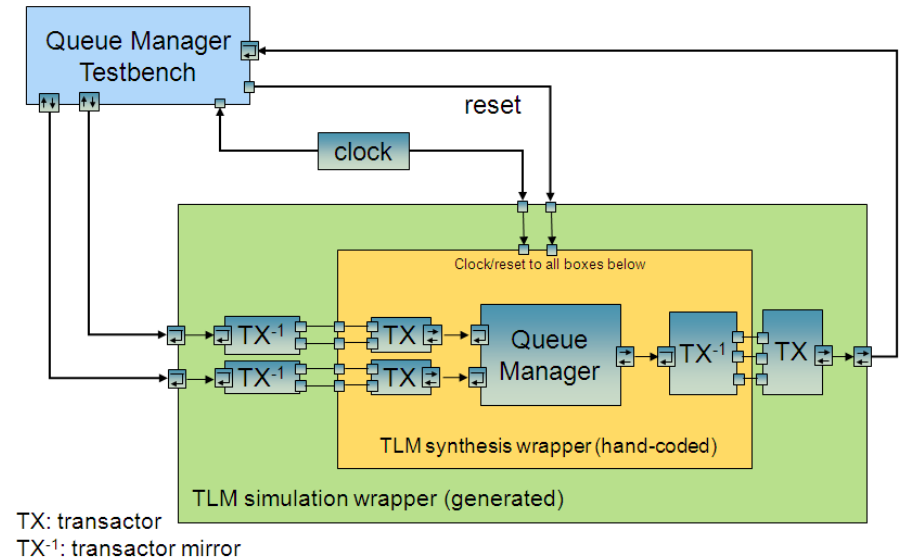
- Reasonably complex
- Mix of control logic, state machines, bus interfaces, RAMs, Data-path and Arithmetic
- Three months of manual design and verification effort
- Easy to articulate performance goals



TLM Based Simulation Setup

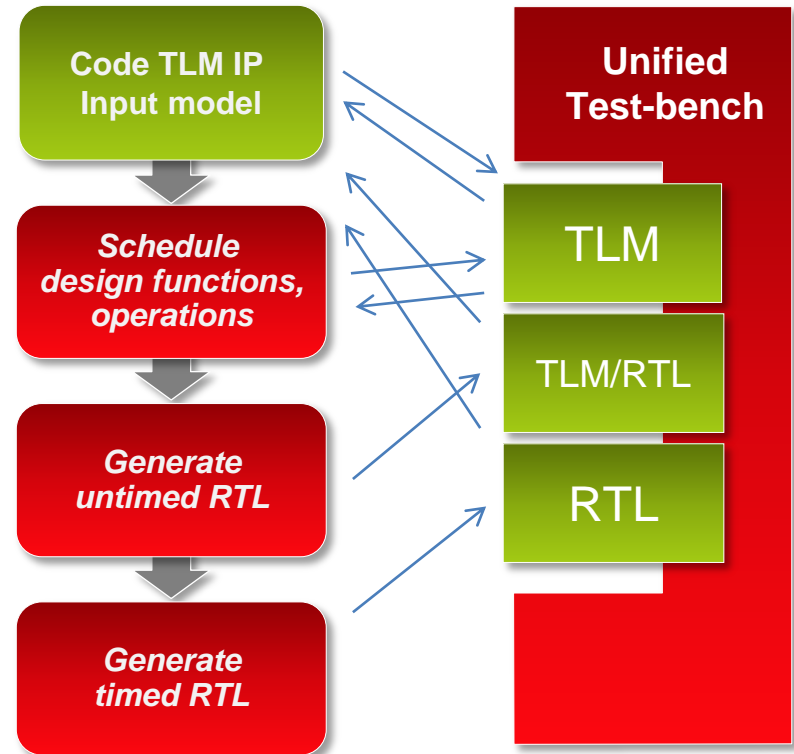
Rationale for Transaction-level Modeling (TLM)

- Enables structured design approach
- Design and Virtual Platform teams can use same TLM model
- C-to-S works with TLM model to implement the hardware
 - Fast enough for virtual platform simulations
- Quick and Easy implementation
- TLM enabled test-bench reuse at multiple levels (C, RTL, Gate)



Design & Verification flow with C-to-Silicon Compiler

- SystemC 2.1 for both “golden” design and test-bench
 - TLM1.x model for TI bus protocol
- C-to-S GUI for analyzing, debugging scheduling issues
 - Technology libraries provided to CtoS and RC for timing analysis
- Single test bench with TLM adaptors to support all levels of design



Case Study Results

Metric	Goal / RTL Comparison	With C-to-Silicon
Cycle Budget	10 cycle push/pop	Met. Iterative process.
Operating Frequency	400 MHz at 40nm node	410 MHz Used RC as synthesis engine
Area	0.69 mm ²	0.6 mm ²
Lines of Code	4500 lines of Verilog	1900 lines of System C
Simulation/Debug	All RTL variables are visible	System C debug support is a work in progress
RAM Integration	Automated / Scripted	Simple flow. Automatic RAM inference and scripted instantiation

C-to-Silicon Compiler “Report Card”

Criteria	Grade	Remarks
Productivity Gain	A	Shorter time from concept to design, faster visibility of results
User Learning Curve	A-	Prior familiarity with C helps but conceptual challenges hard
Quality of Results	A-	Quite impressive but not very intuitive to achieve
Verification Flow	B+	Tremendous resistance to adoption
Tool Maturity	B	Promising prospects but needs work
Ease of Adoption	B-	Engineering discontinuity, management buy-in, risk assessment and mitigation

General Observations & Conclusions

Productivity Gains

- About 2-3x more productive than RTL implementation
- Additional benefit when design is retargeted to different area/frequency/performance targets
- No bugs related to mundane RTL issues
- Parameterization much easier than in Verilog

Design Flow Challenges

- About 2-3x more productive than RTL implementation
- Steep learning curve for System C, TLM, CtoS and associated flows
- Conceptual challenges understanding “Control Flow Graph” and fixing “scheduling conflicts”
- Limited debug and visibility into C models during simulation (improving in newer versions)
- Initial model development is very fast (enabling quick delivery for software development)...but hitting timing and performance targets can be challenging

Verification Challenges

- RTL is not readable. Can't easily correlate RTL issues with C code.
- Must simulate at C level (analogous to gate-level simulation)