

C based Design Experience using AutoPilot Synthesizing MPEG4-Decoder onto Xilinx FPGA

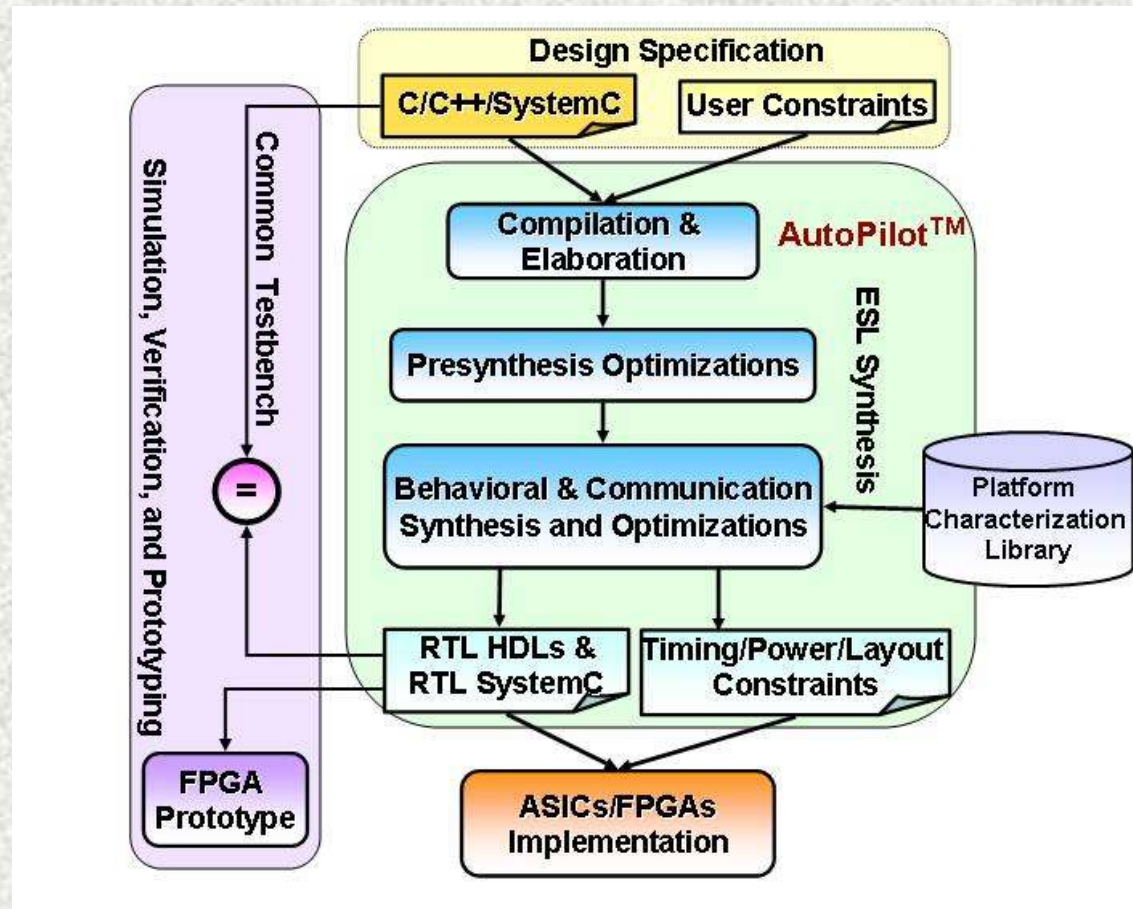
Jason Cong¹, Zhiru Zhang² and Yi Zou¹

¹UCLA Computer Science Department

²AutoESL Design Technologies

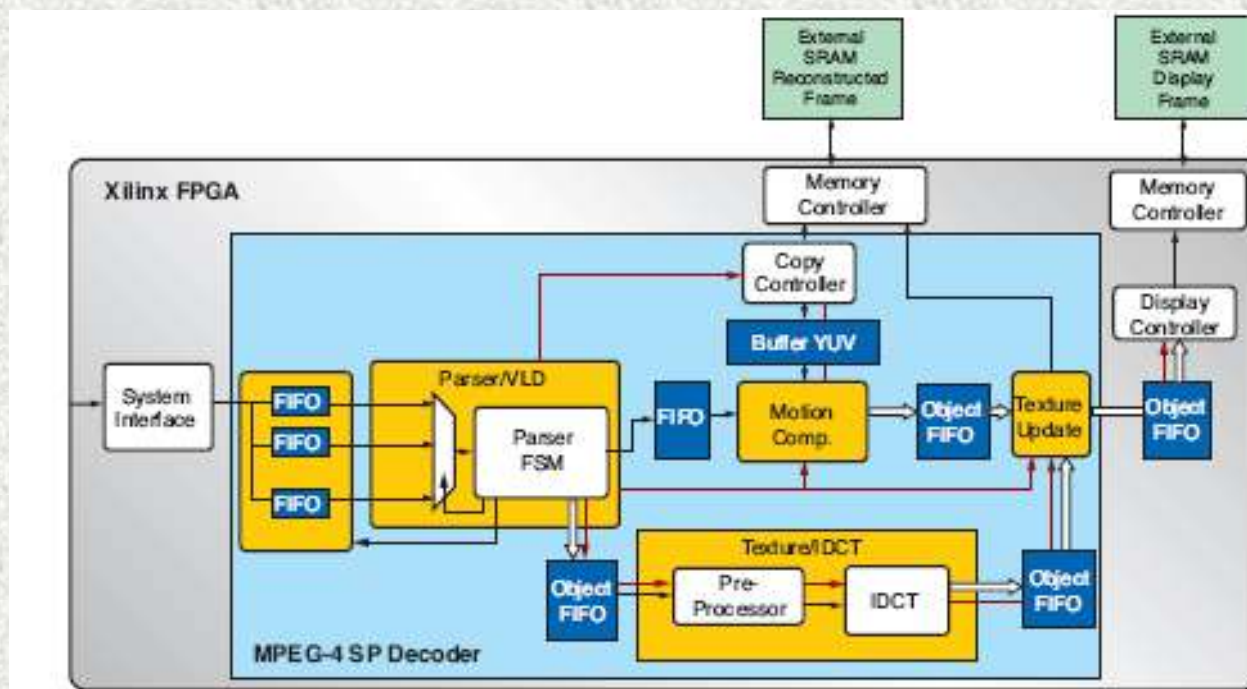
AutoPilot

- One state-of-the-art ESL tool developed by AutoESL Design Technologies
- Based on xPilot developed by UCLA



Xilinx MPEG-4 Decoder

- Reference C code that is used for their IP core development [Schumacher ICIP'05]



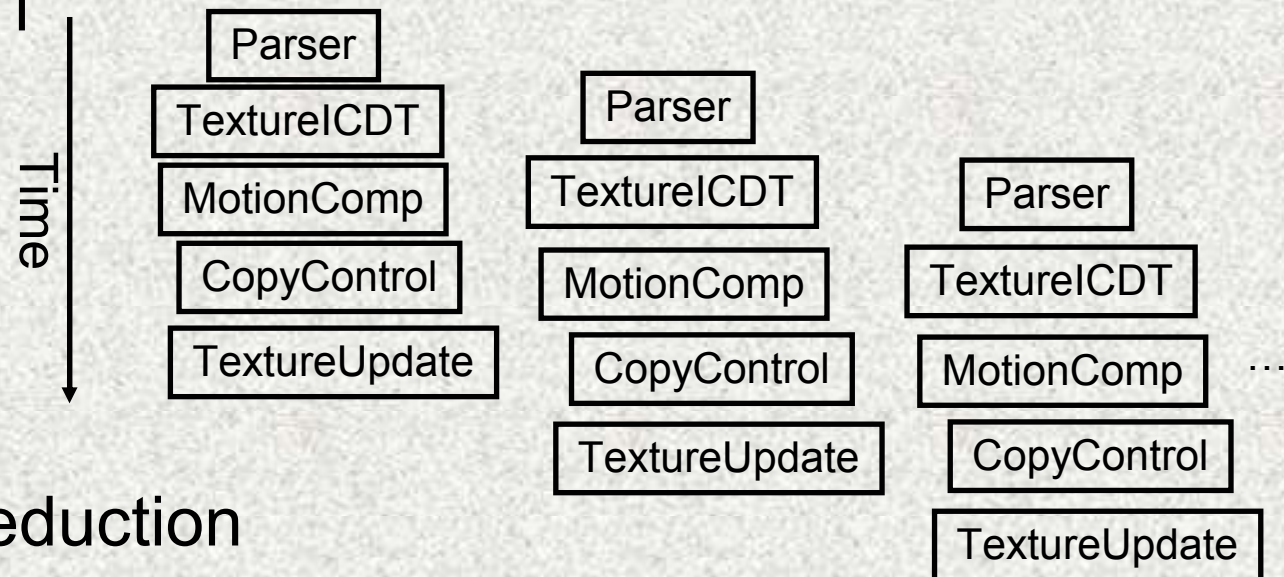
Module	Line of C codes
Parser	2530
Copy Control	287
Motion Comp	312
Texture IDCT	1819
Texture Update	220

Overview

- The paradox of ESL tools: automation versus the ability to control the hardware generation
 - AutoPilot accepts C/C++/System C with synthesis hints (pragmas)
 - These hints allows user to perform better optimizations and evaluate different trade-offs
- Performance optimizations
 - More parallelism
 - Hierarchical function-block pipelining; loop unrolling and pipelining; memory partitioning etc.
- Area optimizations
 - Better sharing and reduced bitwidth
 - Tuning of hierarchy; memory merging; control-structure specialization; bitwidth optimizations

Hierarchical Function-block Pipelining

- Multiple modules running in parallel through streaming pipeline fashion
 - Not only in top module but also in sub-module e.g., TextureIDCT



- FIFO traffic reduction
 - Redundant data/ trivial data need not be sent
 - Need to manually describe the protocols in C codes
- Latency reduction around 4~5X due to pipelining

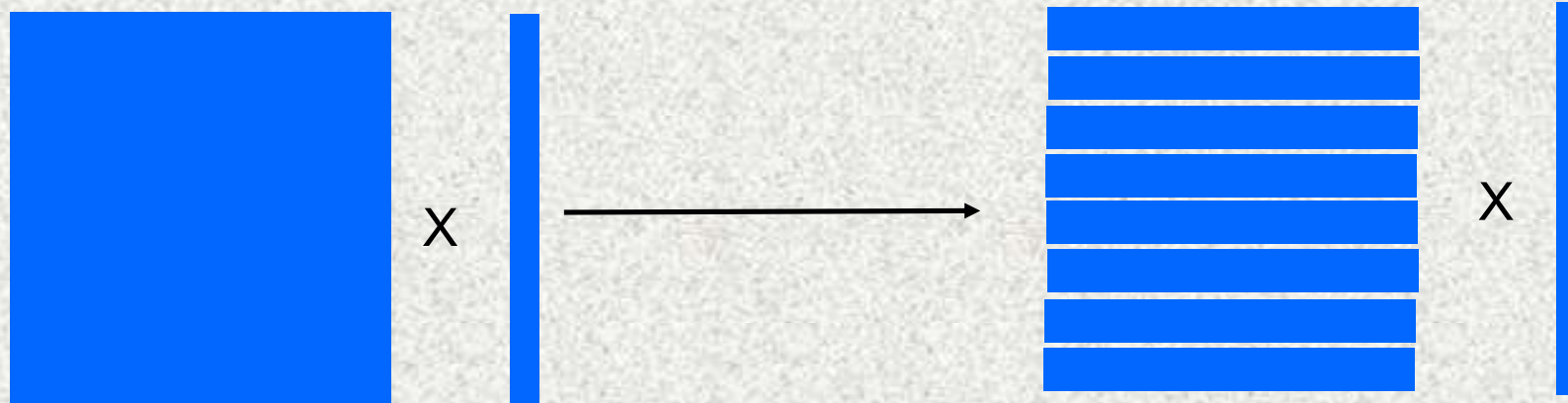
Loop-level Parallelism

- Loop unrolling
 - Partial unrolling or complete unrolling
- Loop pipelining
 - Different initiation intervals
- Associated latency-area trade-off

- Use loop pipelining for every loop
- Combined use of loop unrolling and pipelining in computational critical portion (IDCT)

Instruction- and Data-level Parallelism

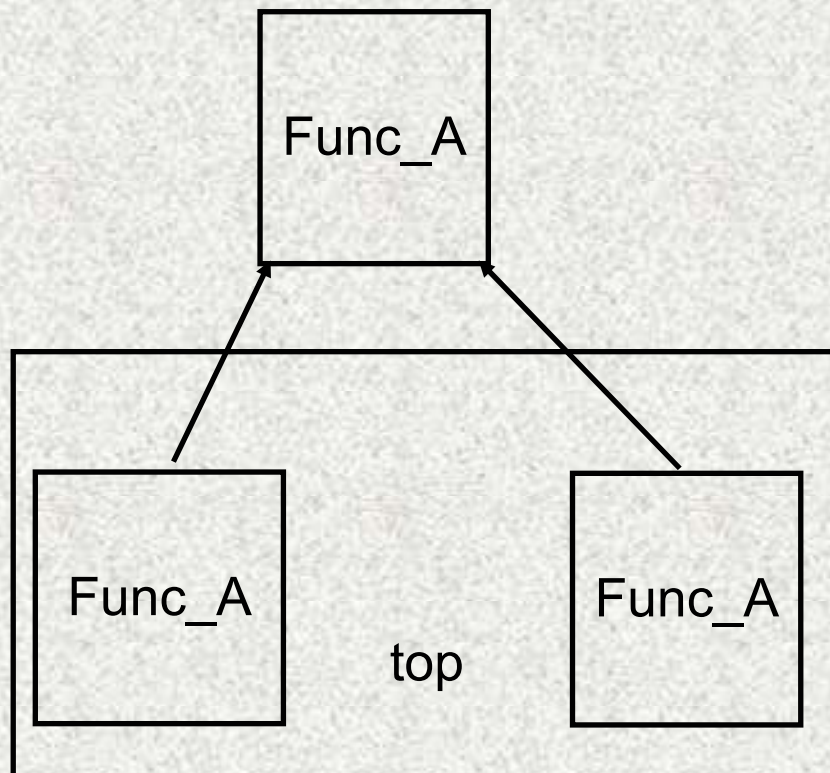
- Scheduler decides the fine-grain parallelism
- Port limitations of on-chip memory blocks
- Memory partitioning pragma
 - Used in IDCT so that unrolling can be more effective



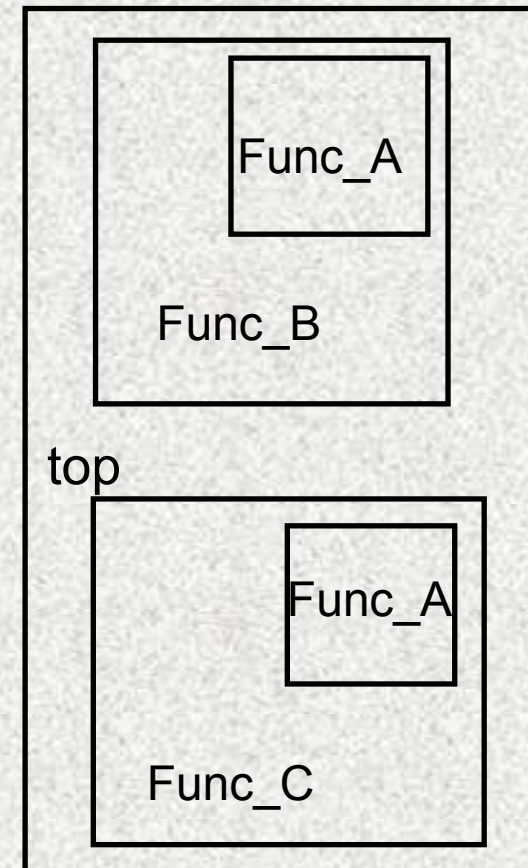
Memory partitioning in matrix-vector multiplication

Tuning Hierarchy for Sharing

- Function hierarchy also provides coarse-grain sharing opportunity



Inlining Func_A will destroy the coarse-grain sharing opportunity



Inlining Func_B and Func_C will enable the coarse-grain sharing opportunity

Memory Optimizations

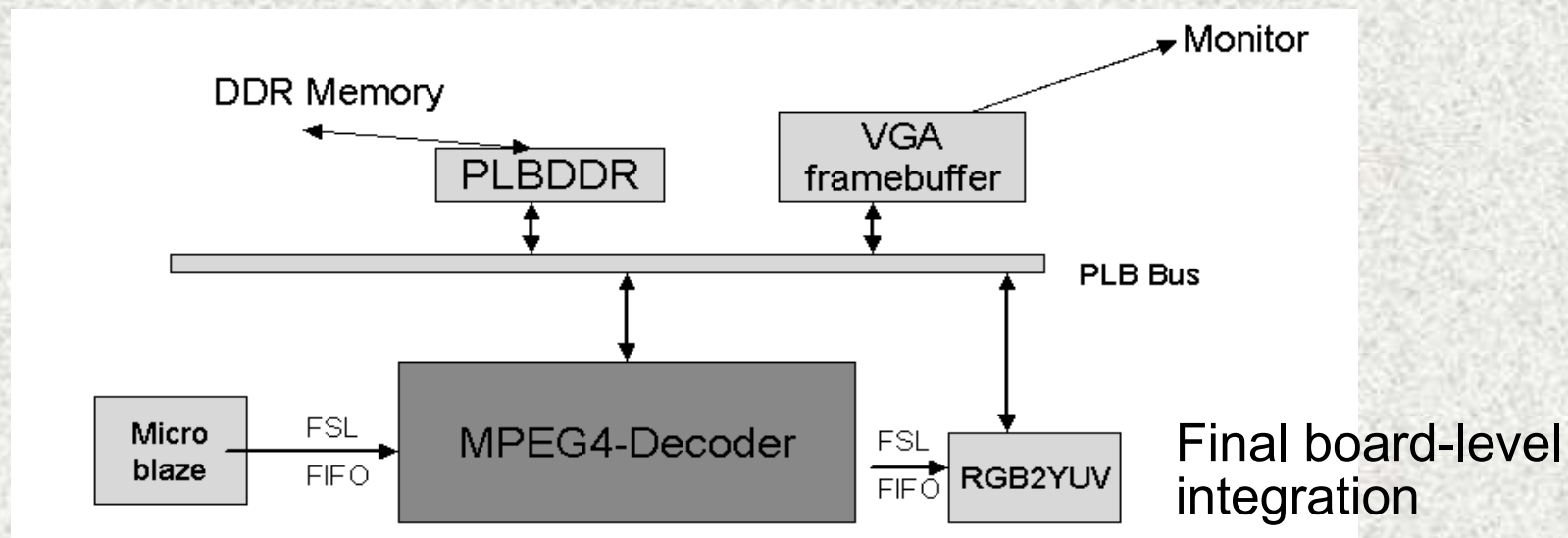
- Memory merging
 - Vertical merging
 - Enlarge the addressing range; generate decoder and mux logic automatically
 - Horizontal merging
 - A wider bitwidth
- Memory module selection
 - Selection of memory implementation from FF, LUT (distributed RAM), block RAM and external RAM

Control Structure Optimization and Bitwidth Optimization

- Control structure optimization
 - Automated transform
 - If-conversions
 - Loop fusions
 - Manual code refinement
 - Reverse of loop switching
- Bitwidth optimization
 - Automatic bitwidth optimization
 - Manually specify bitwidth through arbitrary precision integer (APint)

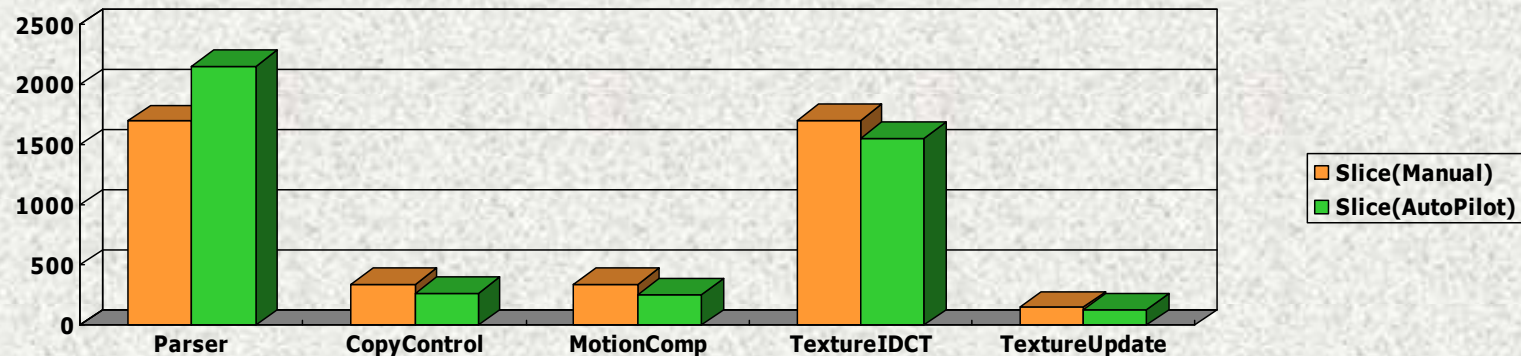
Verifications and Integrations

- C-level verification
- SystemC simulation
- RTL simulation
- Board level simulation
 - Vendor tool with automatically generated interfaces



Experiment Results and Conclusion

- 60fps@4CIF; area comparable with manual design
 - Most modules outperforming manual one
 - Area of Parser module is still off



- In conclusion, modern ESL tools are capable to handle complex designs such as the whole MPEG-4 decoder
 - Various tuning and optimization is performed to achieve good performance and area