Code Generation on embedded processors

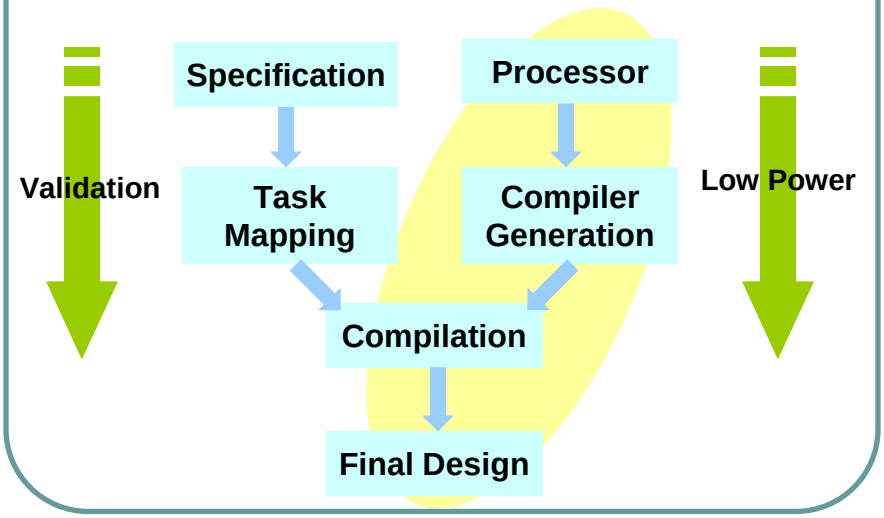
11. 12. 2004.

Eun Kyong Seo

Key Requirements

- Code quality in size and performance
- Chip area constraints
- Real time constraints
- Efficient power consumption

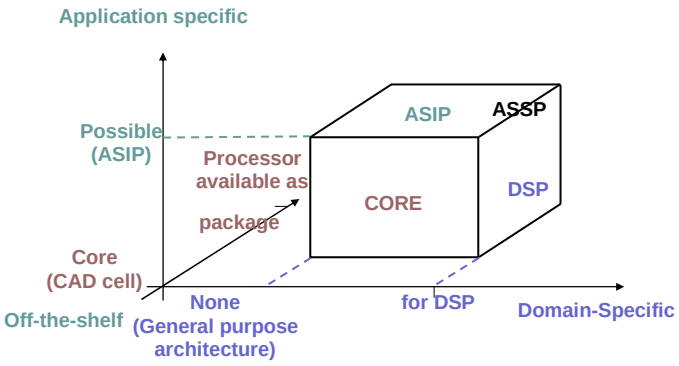
Simplified Design Flow



Architectural Features Of Embedded Processors

- Designed for efficiency
- Huge variety of processors
- Harvard architecture
- Heterogeneous register sets
- Limited instruction-level parallelism or VLIW ISA
- Different operation modes (saturating arithmetic, fixed point)

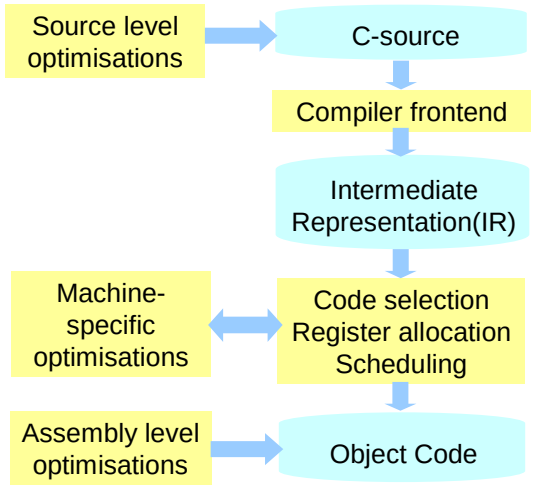
Processor classification



Requirements for compilers

- Demand for compact code
- Demand for extremely fast code
  - Prior to short compilation times
- Need for high dependability
- Constraints for real-time response
  - Smarter compiler to calculate the speed of the code
- Support for DSP algorithm
- Support for DSP architectures

Compiler Structure



Source level optimization *(cont')*

- Standard optimizations *(also at IR level)*
  - Constant folding
  - Common subexpression elimination
  - Jump optimization
- Address code transformation
  - Simplification of array index expressions for memory-intensive applications

Source level optimization *(cont')*

- Loop transformations
  - Loop unrolling
  - Loop folding, Software pipelining
- Function inlining
  - To reduce the calling overhead, replacing function call with body

Optimized Instruction Set Mapping

- Machine-specific features
  - Makes efficient code generation difficult
    - Special-purpose registers
    - Complex instruction patterns
    - Inter-instruction constraints
  - Tree pattern matching
  - Phase-coupled code generation
  - Multimedia instruction sets

Optimized instruction set mapping

- Tree pattern matching
  - Instruction set mapping or code selection
  - Nodes: variables, constants & operations
  - Edges: data dependencies
  - Minimum cover of a DFTs (data flow trees)

Optimized instruction set mapping

- Phase-coupled code generation
  - Code Selection, Register allocation & instruction scheduling (NP hard)
- Multimedia instruction sets
  - VLIW/RISC-like architecture
  - SIMD
    - Faster execution on short values
    - Exploited as compiler-known functions
    - Non-portable → special language constructs or libraries needed

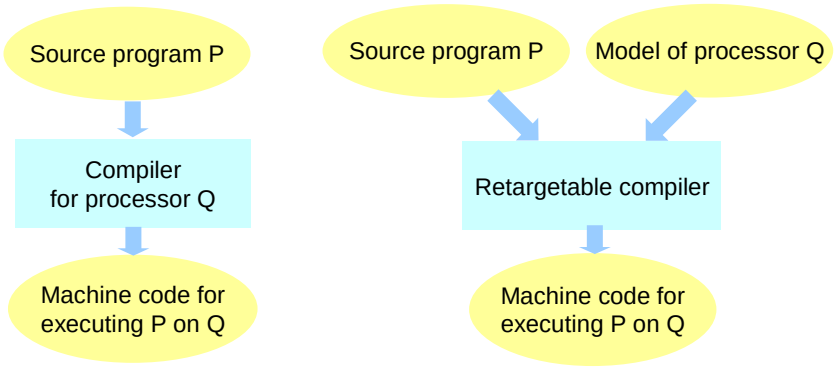
Assembly level optimization

- Memory access optimization
  - Two memory banks accessible in parallel
  - Partitioning based on pre-scheduled assembly code
  - Memory interface & fast access mode, processor pipeline timing information
- Address code optimization
  - Dedicated AGU (Address Generation Unit)

Assembly level optimization

- Instruction scheduling
  - Local scheduling (code compaction) algorithms limited to a basic block
  - Global techniques for global critical paths with larger code size

Retargetability

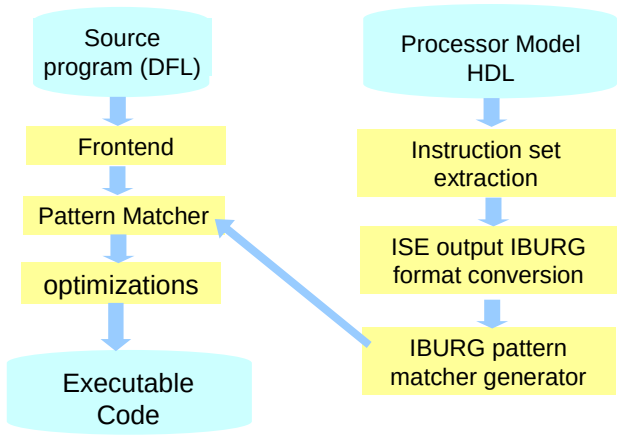


Traditional vs. Retargetable compiler

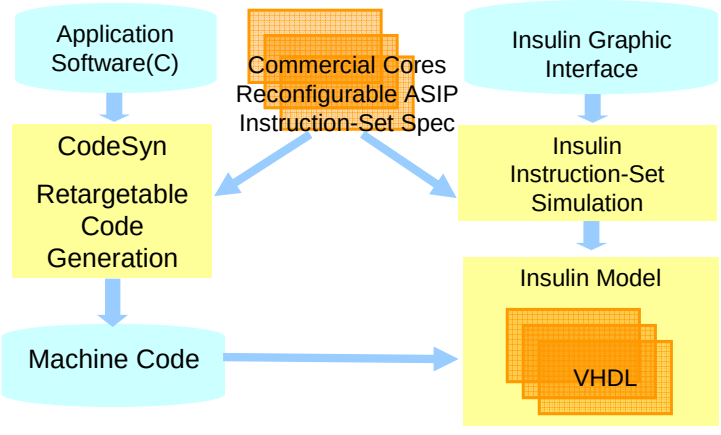
Retargetability

- Generates code for different target processors, based on external (e.g. HDL) machine models.
- Application area
  - Parameterizable ASIPs
    - To explore different configurations of a given ASIP
  - Architecture exploration at system level
    - To estimate the performance of different target processors for a given application early
- Portability, target independence

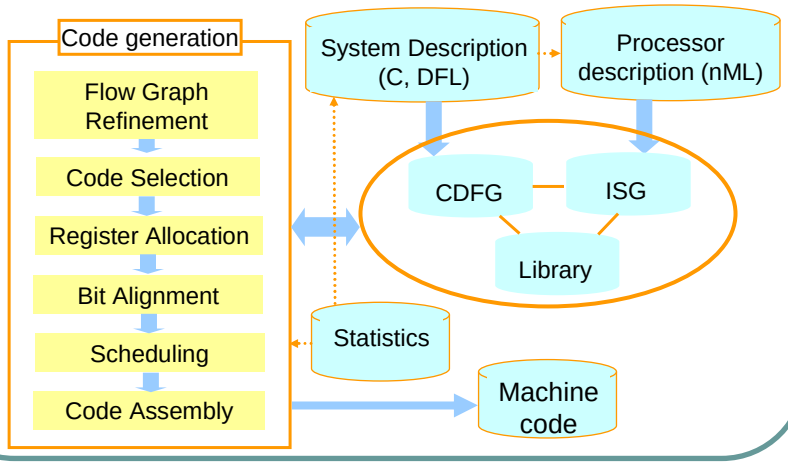
Retargetability - RECORD



Retargetability - Flexware



Retargetability - CHESS



Low Power

- May be contrary to code size and performance
- Instruction selection & scheduling of machine code
- Alternative instruction, data encoding and power-efficient on-chip memories

Processor Classes *(con't)*

- Microcontrollers
  - Control-intensive applications
  - CISC architecture
  - High code density
  - Limited computational resources

Processor Classes *(con't)*

- RISC processors
  - Load-store architecture
  - Large file of general-purpose registers
  - Global register allocation technique

Processor Classes *(cont')*

- DSP processors
  - Arithmetic intensive applications
  - Fast execution of DSP routines by dedicated HW support
    (e.g. multipliers and address generation units)
  - DSP-specific data path architectures

Processor Classes *(cont')*

- Application-specific processors
  - A compromise between off-the-shelf processors and ASICs
  - Application-specific data paths
  - retargetable compilers

Processor Classes

- Multimedia processors
  - Recent mixture of RISC and DSP processors
  - VLIW programming paradigm
  - Very high performance
  - Low code density and high power consumption
  - Support for vectorized (SIMD) instructions.

Cores for reuse

	Soft cores	Hard cores
Models	**Synthesizable Behavioral**	**Layout Datasheet**
Portability	**Yes**	**No**
Design Time	**High**	**Low**
Optimization	**Low**	**High**
Timing Issues	**Full timing verification after synthesis**	**Timing characteristics provided**

Frontend issues

- Lexical, syntactical, and semantical analysis of source programs,

  IR generation
- Machine independence