

# [ Embedded Software in Real -Time Signal Processing Systems: Design Technologies. ]

Presented by Suvidehan Dhirakaosal

## [ Introduction. ]

- Software is becoming big role in embedded systems design.
- Upto 60% of development time spent on software coding.
- Software design phase is becoming bottleneck in system design process.

## [ Paradigm shift from HW to SW ]

- Possible to make changes late in design cycle.
- Easy to add new features to.
- Three types of embedded processors: general purpose, ASIP, and parameterizable processors (Tensilica).

## [ Software Bottleneck. ]

- Parameterizable processors and ASIPs are hard to develop compilers for as the target architecture is not fixed.
- Developers have to go to low level (assembly coding) which runs into legacy code problems.
- Hard to transfer to new processors.
- Also a design usually does not last more than 2 generations.

## [ Software Bottleneck ...continued ]

- Compiled code has to be optimized for each processor architecture.
- Therefore development of compilation tools with architectural retargetability.

## [ Compilation View of Processor Architectures ]

- Classification of processor based on:
  - Arithmetic specialization.
  - Datatype.
  - Codetype.
  - Instruction format.
  - Memory structure.
  - Register structure.
  - Control flow capabilities.

## [ Arithmetic Specialization. ]

- Use of parallel multiplier/accumulator unit for DSPs to implement digital filters, cross correlation, sampling, etc.
- Specialized arithmetic units are implemented in ASIPs to guarantee execution of critical sections.

## [ Data Type. ]

- Most embedded processors implement fixed point arithmetic.
- Floating point arithmetic requires extra silicon area and power.
- DSPs usually have multiple fixed point data types. (16-bit for ALU ops, 32-bit shifters, 40-bit accumulators.)

## [ Code Type. ]

- Presence of data pipelining?
- If yes, data-stationary coding or time-stationary coding?
  - Data-stationary coding: every instruction controls a sequence of operations to execute on a specific data item in the pipeline.
  - Time-stationary coding: every instruction controls a complete set that has to be executed in a single machine cycle (ASIP mostly).

## [ Instruction Format. ]

- Orthogonal format:
  - Fixed control field that can be set independently from each other.
- Encoded format:
  - Interpretation of the instruction bits as control fields may be different from instruction to instruction.
- Encoding restricts instruction level parallelism. (Not good for multimedia, image processing.)
- Most ASIPs have orthogonal instruction format.

## [ Memory Structure. ]

- Von Neuman architecture:
  - Processor has a single memory space for data and program (older processors).
- Harvard architecture:
  - Data and program accessible through separate hardware (current DSPs and ASIPs).
- Addressing modes:
  - Multiple addressing: immediate, direct, and indirect.
- Operand location:
  - Load-store (register to register) architecture.
  - Memory-memory and memory-register architecture.

## [ Register Structure. ]

- Homogeneous set: all registers are interchangeable.
- Heterogeneous set: contains special purpose registers.

## Control Flow.

- Small branch penalties.
- Zero overhead loops instruction. (Execute repetitive algorithm without cycles for loop control.)

## Issues in Software compilation.

- Research on compilers for DSP and ASIP has been ignored until early 1990's.
- GCC is the most popular ported compiler for embedded processors.
- C language is often augmented to support user-definable data types.

## The software compiler.

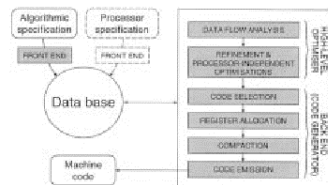


Fig. 5. Anatomy of a software compiler.

## Additions to compiler for embedded processors.

- Phase coupling (Dependencies caused from instruction-level parallelism and presence of heterogeneous register structures).
- Need of more specialized compiler algorithm to optimize heterogeneous register structures.

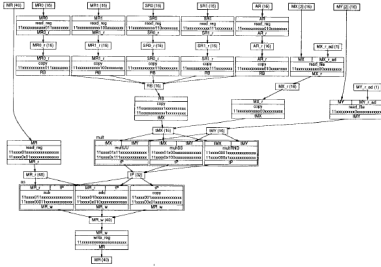
## Processor Specification Languages.

- Efficient and powerful models of processor is the key in making software compilation retargetable.
- Netlist-Based Languages:
  - Describe processor as a netlist of HW building blocks.
  - Disadvantage: design has to be completed prior.
- High-Level Languages:
  - Use high level language to capture information that is available in a programmer's manual of a processor.

## Processor Models for Compilation.

- 1) Template Pattern Base: enumerates the different partial instructions available in the instruction set. Generates a tree grammar.
- 2) Graph Models: derived from detailed processor netlist.
  - Captures both behavioral and structural information.

## [ ISG representation of ADSP -21xx processor. ]



## [ Code Selection. ]

- Code generation is NP -complete.
  - But optimal vertical code can be constructed in polynomial time if:
    - 1) Intermediate representation is an expression tree.
    - 2) Template pattern base is restricted to only regular tree grammar.
    - 3) Processor has homogeneous register structure.

## [ Dynamic Programming ]

- Stepwise partitioning of the code selection problem.
- *Tree pattern matching* : locate parts of expression tree that correspond to available tree patterns in pattern base.
- *Tree covering* : find a complete cover of the subject tree with available patterns.
- Can be adapted to work with heterogeneous register structures.
- Used in recent compilers for embedded processors.

## [ LRParsing. ]

- If the processor model is a *regular tree grammar*, code selection can be seen as parsing the subject tree using specified grammar.
- When generating code for subtree, code for the left operand of the root node is selected without considering the right operand.
- May generate inferior results compared to dynamic programming.

## [ Graph Matching. ]

- Pattern matching algorithm to generate optimal vertical code for directed acyclic graph structure.

## [ Bundling. ]

- This code selection technique constructs required patterns on the fly during the traversal of the intermediate representation.
- Advantage:
  - Support intermediate representation and partial instructions that are graphs rather than trees.
  - Useful for DSP and ASIP.
- Disadvantage:
  - Increased algorithmic complexity.

## [ Rule-Driven Code Selection. ]

- A set of rules which guide each transformation at each phase of compilation.
- For code selection phase, virtual machine that resembles the actual instruction set of the real machine. (w/o parallelism).
- User then specifies rules to map operation patterns to virtual machine instructions.

## [ Graph matching example. ]

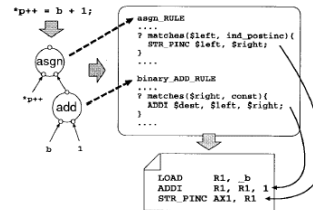


Fig. 10. Virtual code selection.

## [ Register Allocation ]

- Graph coloring can be used.
  - Must have homogenous register structure.
  - Code selection has been done.
  - We know the execution ordering.
  - Interference graph is constructed based on live range for every intermediate computation value.
  - Heuristic graph coloring algorithms are used to color the nodes.

## [ Register Allocation ...continued ]

- Data routing is used for heterogeneous register structures. (DSPs, ASIPs)
- Uses local greedy search techniques to determine data routes.

## [ Memory Allocation and Address Generation. ]

- Graph coloring technique like register allocation.
- *Postmodification*: next address can be calculated by adding a modifier value to the current address while the current memory access is taking place.

## [ Scheduling. ]

- Scheduling is essential for DSP and ASIP processors because they are generally pipelined or contain instruction level parallelism.

## [ Scheduling ...continued ]

- *Global scheduling*: partial instructions can be moved across basic block boundaries.
- Takes care of pipelining and instruction level parallelism.
- *Software pipelining*: does loop unrolling to pipeline loops.