AHIR: from program to hardware

Sameer Sahasrabuddhe, K. Arya, M. P. Desai

Concept

- Start with a program written in a “standard programming language” such as C.
  - Use software compiler to optimize program to extract parallelism etc.
- Transform the program into an intermediate representation (a virtual circuit)
  - AHIR
- Map the virtual circuit to an actual logic circuit
  - Optimize hardware by sharing resources etc.

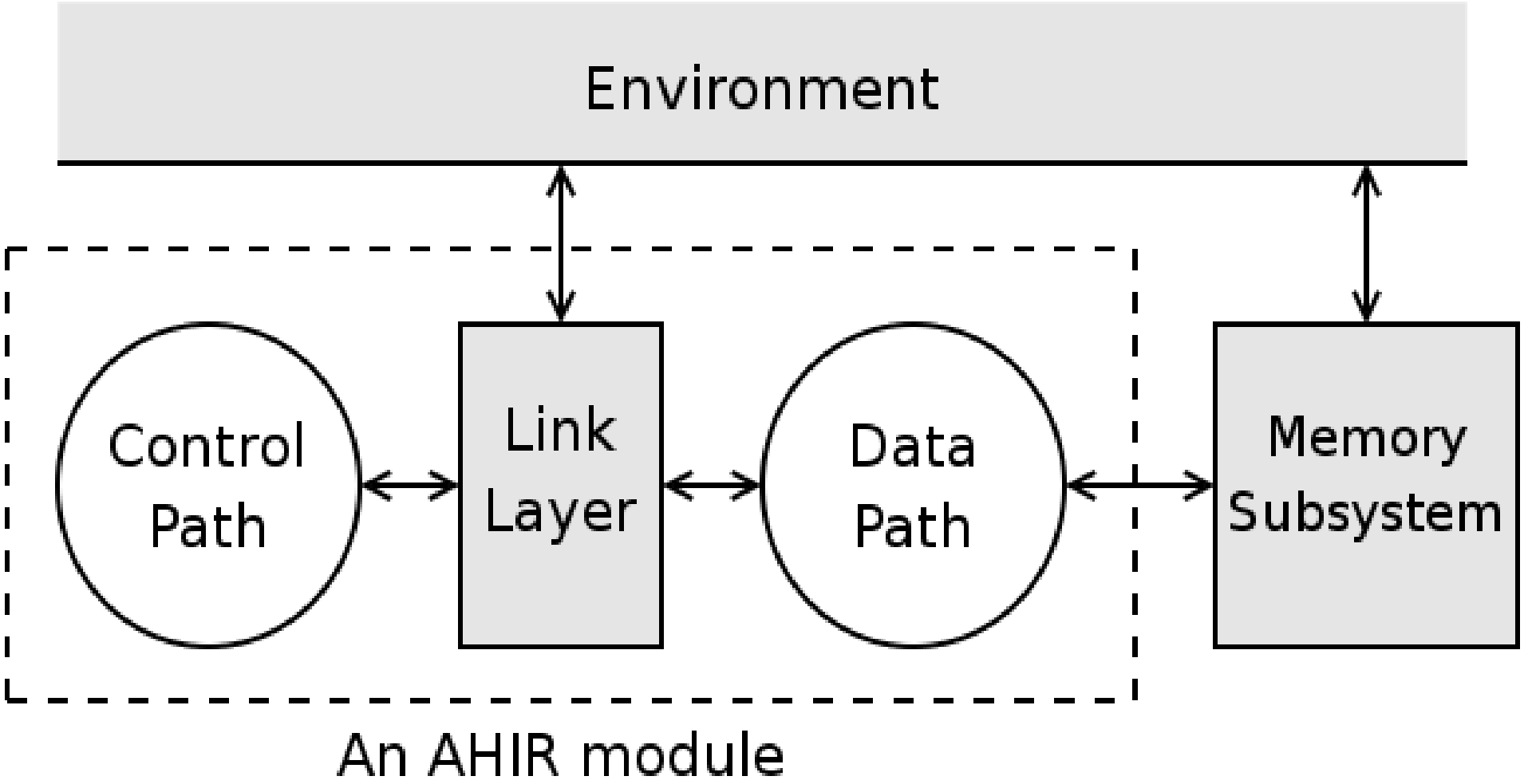
Potential Value

- Replace use of embedded processors in SoCs
  - Provided that generated circuits are 10X superior in energy-delay product and/or throughput/area metrics
- Use as part of VLSI Design Flow
  - Provided that generated circuits are competitive with conventionally designed circuits.
- Simplify the verification process considerably
  - verification/validation at the program (specification) level rather than the RTL level.

C to CDFG to AHIR

- C program translated to CDFG by compiler tools
  - LLVM tool-suite
- CDFG transformed to AHIR
  - Provably correct transformation
- AHIR representation
  - Orthogonal representation: CP X DP X Memory
  - Each factor can be optimized without affecting the others.

AHIR


Environment
Control Path
Link Layer
Data Path
Memory Subsystem
An AHIR module

AHIR

- Orthogonal representation
  - CP X DP X Memory
- Amenable to static analysis
  - e.g. Arbiterless sharing of operators in the datapath.
- Information for memory optimizations
  - Data placement
  - Hazard-free scheduling.

Throughput/Area metrics

- Ahir RTL mapped to FPGA vs. P-4 processor
  - 2X (Linpack) to 500X (A5/1-stream cipher) better in throughput/area metric. Typical improvement 10X over range of applications: FFT,Red-black-tree,AES, Linpack, A5/1.
- Ahir RTL vs. handcrafted RTL
  - 360X worse for AES
  - 900X worse for A5/1
  - Room for improvement

Looking ahead

- Trying to quantify energy-delay benefits of using AHIR
- Optimized memory subsystem to reduce run-times
  - Using more exact dependency rules.
- Exploiting compiler-level optimizations
- Parallel-programming languages as starting points?
  - e.g. Esterel?

Conclusion

- A complete flow from C to hardware
  - Scalable, applicable to complex software.
  - Competitive with processor implementations in throughput/area
  - Expected to be even more competitive if energy-delay metric is used for comparison
- Use of parallel programming language as starting point will get us closer to hand-crafted RTL.