Compiling Multimedia Applications for Reconfigurable Architectures

Prith Banerjee

University of Illinois at Chicago
Goals and Objectives

- Develop a compiler to take multimedia processing applications, written in high-level languages such as C/C++/SystemC, and low-level software implementations written in assembly code of embedded processors, and automatically map them onto low-power, high-performance embedded systems consisting of control processors, application specific parallel processors, memories, ASICs and FPGAs.
- The compiler will read in C/C++/SystemC or assembly language code, and automatically generate Register Transfer Level (RTL) VHDL and Verilog code that can be synthesized onto ASICs and FPGAs.
- while optimizing power and energy consumption under performance constraints.
- This research will have the following goals:
  - Design a basic compiler to take high-level applications written in C/C++/SystemC and low-level software implementations written in assembly code for an embedded processor, and automatically generate RTL VHDL and Verilog for mapping to ASICs and FPGAs.
  - Create fast and accurate models for estimating area, delay and power at a system level using macro-modeling techniques.
  - Develop novel high-level synthesis optimizations for hardware synthesis while optimizing power and energy consumption under performance and area constraints.
(a) CURRENT DESIGN FLOW

Manual Design
1. HW/SW Partitioning
2. RTL VHDL Verilog
3. Multiple expensive design iterations

(b) PROPOSED AUTOMATED DESIGN FLOW

New Flow

Automated SW Compilers

Automated CAD tools For FPGA design

Compiled SW Application
1010010
0100100

Hardware Implementation

Satisfied area, delay, power constraints?
If not, repeat design

Reconfigurable Hardware

Automated area, delay, power tradeoffs
Satisfies constraints
Rapid design closure
High Level Synthesis for Low Power

- Given a behavioral description of the target design, in ANSI C or any other high-level language, the goal of behavioral synthesis is to derive a register-transfer level (RTL) description of the circuits.
- Behavioral synthesis is composed of three main interdependent tasks:
  - scheduling, module binding, and resource allocation.
- The scheduling and binding pass will use the data flow graph of each basic block in the program to schedule the computations to clock cycles and bind them to resources, e.g., adders and multipliers.
- Various algorithms for scheduling the CDFG nodes onto architectural resources on the ASICs and FPGAs will be developed.
- Our scheduling algorithms will handle multi-cycle functional operators as well as multi-cycle memory read and write operations.
- Various operator-chaining algorithms will also be developed.
- Investigate ILP formulations for simultaneous module selection, scheduling, allocation, and binding that are based on the high-level area, delay and power estimation methods.
- We will also develop automated methods to compile streaming applications onto SOCs.
High Level Estimation of Area, Delay, Power

• Develop accurate area, delay and power estimates at a system level that will be used in conjunction with the high-level synthesis transformations.
• Area and delay estimation methods will be based on high-level compile time estimations of the areas and delays of the nodes of a Control Data Flow Graph (CDFG).
  – The CDFG nodes will be parameterized with the bit-widths of the inputs (such as N-bit adders and multipliers), and the number of inputs.
  – We will synthesize these RTL operations using commercial synthesis tools and map them onto FPGAs and ASICs.
  – While performing synthesis, we will apply various timing constraints to generate a wide range of component instantiations, yielding different design points in the area-delay-power curve.
  – We will take the area and delay functions that are generated for various bit widths, N, different wireloads, and different input slew rates, and perform a curve fitting using a minimum least squares fit.
  – In addition to such static estimates of area and delay, we will explore the use of fast floor-planning to determine the impact of interconnect on the area and delay.
• Use of power macro-modeling in our compiler framework.
  – Three widely used input parameters for the macro-model follow: average input signal probability, $P_i$, average input transition density, $D_i$, and input spatial correlation, $S_i$.
  – We will combine the above static estimate of power consumption with a dynamic estimation approach that will try to model the actual module switching activities.
Low Power Transformations

- **Loop transformations such as**
  - Loop interchange, loop fusion, loop distribution, array privatization, to modify the loops of a given C, C++, program so as to increase the parallelism that can be extracted by the hardware implementation on an ASIC or FPGA.

- **Clock Gating**
  - Shutting down inactive units is a powerful power reduction technique.
  - During the scheduling, allocation, and binding steps of high-level synthesis, it is clear which functional units (adders, multipliers) will be inactive in which clock cycles.
  - We will develop clock gating techniques that will enable those units to be powered down during specific cycles.

- **Multiple VDD**
  - Dynamic power consumption increases by the square of the supply voltage, and delay decreases linearly with the supply voltage.
  - Trade off power consumption for performance by changing the supply voltage.
  - Operators that can tolerate larger delays in a schedule will be mapped to lower VDD devices.

- **Multiple VTH**
  - Develop high-level synthesis scheduling algorithms that will assign the various RTL operators to high VT and low VT devices.
  - Operators that can tolerate larger delays in a schedule will be mapped to high VT devices.

- **Memory Banking**
  - Memory accesses are responsible for the majority of power consumption in embedded systems.
  - Develop memory banking techniques in which the compiler will partition the data into different memories allowing memory banks that have not recently been accessed to enter low-power self-refresh mode.
  - Develop loop transformation techniques that will rearrange the memory accesses of data to certain memory banks.

- **Other Transformations**