Partitioning - II

Functional partitioning

Earlier partitioning

• Partition large number of processes among processors
• Partitioning after synthesis
  – Synthesis used to be more time consuming due to non-linear characteristics of its tool heuristics.
  – More power consumption
Partitioning trend

Many applications consist of one or small number of very large processes

- Partitioning before synthesis or compilation has advantages
  - order of magnitude reduction in logic synthesis runtime.
  - Improved system performance as smaller processes can be synthesized with shorter clock period than one large processor.
  - Improved satisfaction of I/O and size capacity constraints on a package, reducing inter-package signals (compared to structural partitioning)

Partitioning approaches

- **Structural**
  - specification
  - Control unit
  - Datapath
  - synthesis
  - partitioning

- **Functional**
  - specification
  - Control unit
  - Datapath
  - synthesis
  - partitioning
Functional Partitioning

- Divides a system’s functional specification into multiple sub-specification.
- Each sub-specification represents the functionality of a system component, such as a custom-hardware or software processor.
- Then the components are synthesized down to gates or compiled to machine codes.

Advantages of FP

- Power reduction due to mutual exclusive components
- smaller board size, lower cost
- increase software speed
- concurrent synthesis and debugging
- less physical design problems
Problem description: Model

- Input: process x (C program or VHDL process)
- A view of the process: set of procedures \( F = \{ f_1, f_2, \ldots f_n \} \) with one as main procedure.
- Variable: simple processor with read and write being the procedure calls.
- Execution of \( F \): procedures executing sequentially, starting with main and that calls other procedures; only one is active at a time.

Problem description: Model

- Functional partitioning creates a partition \( P \) consisting of a set of parts \( \{ p_1, p_2, \ldots p_m \} \), such that every procedure \( f_i \) is assigned to exactly one part \( p_j \), i.e. \( p_1 \cup p_2 \cup \ldots p_m = F \) and \( p_i \cap p_j = 0 \) for all \( i, j, i \neq j \).
- Each \( p_j \) represents the function to be implemented on a single processor. The processors are mutually exclusive.
- Each part \( p_j \) is converted to a single process before synthesis; this process consists of a loop that \textit{detects a request for one of the part’s procedures, receive input parameters, calls the procedure, and sends back output parameters.}
Model contd...

- Function Bus: single bus carries parameter passing between processors
- Protocol: putting destination procedure’s address, pulsing address request, putting parameter, pulsing the data request.
- Process → Synthesis → custom processor component \( C_i \)
- For application we target, \( C_i \) = non-trivial data path and a complex controller with hundreds of states.
- Procedure on \( C_i \) may be implemented either as a control subroutine or datapath component.
- Synthesis may implement process’s procedures in parallel if data dependencies are not violated.
  - While procedures are not mutually exclusive after partitioning, processors are still mutually exclusive.

Five tasks for good partitioning

- Model creation
  - converts input to an internal model (call graph model)
- Allocation
  - Instantiating processors of varying type (done before)
- Partitioning
  - *Dividing input process among allocated processors*
- Transformation
  - modifies the input process into one with different organization but same overall functionality, leading to better partition.
- Estimation
  - provides data used to create values for design metrics. *Pre-estimation and online-estimation.*
Partitioning Methodology

• Three-step method:

Sequence of partitioning steps proposed by Vahid

Step1: Granularity Selection

• **Goal:** Extract procedure from specification, which are to be assigned to processors during N-way assignment.
• Granularity is a measure of complexity
  – Fine: many procedures of low complexity.
    • Little pre-estimation and online-estimation less accurate. Make online-estimation more complex to build higher accuracy.
    • Can be more time consuming and may prohibit the use of assignment heuristics that need many estimations.
  – Course: few procedures of high complexity.
    • many behaviors are grouped together into inseparable unit, so that any possible solution that separate those behavior is excluded.
Granularity

- Procedures are selected very carefully to balance the above effects.
- Each statement is treated as atomic unit.
- Granularity Selection Problem:
  *Partitioning statements into procedures such that, (1) procedures are as course-grained as possible, to enable maximum pre-estimation and application of powerful N-way heuristics and (2) statements are grouped into a procedure only if their separation would yield inferior solution.*

Granularity

- A straightforward heuristic: *choose a specification construct to represent a procedure*. I.e. each statement or block. Also, user defined procedure for partitioning.
- Transformations can be used to improve the above strategy
  - Procedure Inlining: replace procedure call by procedure’s contents making granularity coarser. Inline procedure disappears.
  - Procedure cloning: makes a copy of a procedure for exclusive use by a particular caller. Ex: Multiply-called procedure if inlined might grow excess, and if not-inlined, might needs more communication. Cloning is a compromise.
Illustration

Input specification with many procedures

Access graph

Transformation contd..

• Procedure Exlining: Replaces a subsequences of a procedure’s statements by a call to a new procedure containing only that subsequences. (opposite to inlining). This technique moves towards finer granularity.
  – Redundancy exlining: replaces two or more near-identical sequences of statements by one procedure. (use string matching method: statements are encoded characters)
  – Distinct computation exlining: Divide a large sequence of statements into several smaller procedures such that statements within a procedure are tightly related and would not be separated during N-way assignment solution.
Step 2: Pre-clustering

- **Goal**: Reduce the number of procedures for subsequent N-way assignment by merging procedures whose separation among parts would never represent good solution.
- **Different from granularity step**: procedures being clustered here may not be such that they could exlined into single new procedure. I.e. calls to these procedures are non-adjacent.
- **Different from N-way assignment**: each cluster does not represent a processor and therefore cannot be guided by direct design metrics estimates.
Pre-clustering method

- Uses hierarchical clustering:
- procedures after granularity selection are converted to a graph node and edges are created between every pair weighed by the closeness of the nodes,
- closest pair of nodes are merged to a new node. This is repeated until no nodes are exceeding the threshold weight.[10]

Illustration of pre-clustering

- Two procedures LcdUpdate and LcdSend communicate heavily: 48 times per call.
- These two should never be separated. Since LcdSend appears 48 times inside LcdUpdate, inlining during granularity selection was not reasonable option.
More on pre-clustering

- Can reduce runtime of N-way assignment by 30% or more
- May look at Ethernet example in the reference.

Step 3: N-way assignment

- Goal: Distribute the procedure among given set of processors. Procedures are created after granularity selection and pre-clustering
- Constructive heuristics are used to create initial solution and can include random distribution and clustering.
- There is an additional metric: “Balanced size”. Size of an implementation of both sets of node divided by the size of all nodes. This favors merging small sets over large ones.
- Heuristics applied: Greedy, Simulated Annealing, Hill climbing
N-way assignments

- Greedy algorithm: linear time heuristic that moves nodes that reduce the value of cost function
- Simulated annealing: randomized hill climbing to avoid local minima with long runtime
- Extended hill climbing: with some restrictions and tightly coupled data structure, \( O(n \log(n)) \) runtime

- *cloning* transformation can be applied selectively here
- *port-calling*, another transform: for I/O balance and ease access to shared ports. (I/O procedures are used in place of external port access that take care of send/receive etc.)
Other partitions of operations

- **Aparty**: among datapath modules using multi-stage clustering,
- **Vulcan**: among packages using iterative improvement heuristics
- **Chop**: among packages focusing on providing suite of feasible solutions for each package that would satisfy overall constraints
- **Multipar**: among packages simultaneous with scheduling and allocation, using linear programming
- **SpecPart**: partitioned procedures among packages using clustering and iterative improvements.

Limitation of three-step approach.

- Total hardware increase may be large for examples with small controllers and large datapaths.
- Problems that has large number of small processes - much like a scheduling problem
- Parallel execution on processors