

# Partitioning III

## Extended Partitioning for Embedded Applications

Mahapatra-Texas A&M-Fall'00

1

## Binary partitioning

- Goal: Map each node of a directed acyclic graph (DAG) to hardware or software (binary choice) and to determine the schedule for each node.
- DAG: The task level description of an application is specified as SDF (synchronous data flow) graph, then SDF is translated to DAG representing precedence relationship among the nodes. *A DAG is input to partitioning tool.*
- *Note: For a given mapping of a node (hw or sw), it is possible that the node can be implemented using various algorithms and synthesis mechanisms and they vary by area and delay outcomes. Call this “implementation bins”.*

Mahapatra-Texas A&M-Fall'00

2

# Extended partitioning

Goal: Combine implementation bins with binary partitioning.

- A joint problem of mapping nodes in DAG to hw or sw and within each mapping, select suitable implementation for better results.

Binary Partitioning

Hardware/Software Mapping and Scheduling

Extended Partitioning

Hardware/Software Mapping and Scheduling

+

Implementation-bin selection

Mahapatra-Texas A&M-Fall'00

3

# Assumptions

1. The precedences between the tasks are specified as a DAG ( $G = (N, A)$ ). The throughput constraints on the SDF graph translates to a deadline constraint  $D$ , I.e., the execution time of the DAG should not exceed  $D$  clock cycles.
2. Target architecture: programmable processor and custom datapath. These components have constraints.
  - Software: program and data size,  $AS$  - memory capacity.  
Hardware has maximum size  $AH$ .
3. Communication cost of interface:  $ah_{comm}$  = hardware area such as glue logic interface,  $as_{comm}$  = software area the code size for send/rec,  $t_{comm}$  = #cycles to transfer data.

Mahapatra-Texas A&M-Fall'00

4

## Assumptions

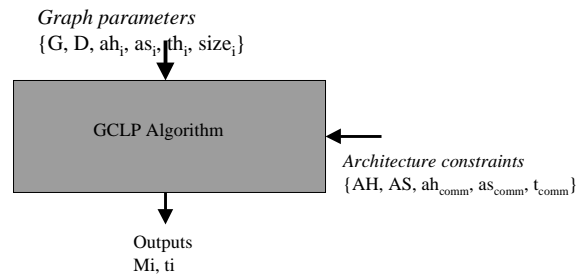
4. Self-timed blocking memory mapped interface.
5. Communication cost of sw-sw and hw-hw neglected.
6. Area and time estimates of each node is known.
7. Nodes mapped to the hw do not share resources.

## Binary partitioning problem

- Given a DAG, area and time estimates for hw and sw mapping of all nodes, and communication cost, subject to resource capacity constraints and deadline  $D$ , determine for each node  $i$ , the hw or sw mapping ( $M_i$ ) and the start time for the execution of the node (schedule  $t_i$ ), such that the total area occupied by the nodes mapped to hardware is minimum.

# Partitioning Algorithm

(with various notations)



$th_i$ : software execution time estimate for node  $i$   
 $size_i$ : size of node  $i$  (number of atomic operation)

Mahapatra-Texas A&M-Fall'00

7

# Foundation

- Uses list scheduling: serial traverse the node list to select a mapping that minimizes objective function
- Objective functions:
  - Minimize finish time of the node or
  - minimize area of the node
- Note that, use of one of the above objective at a time will lead to either reduced optimal or infeasible solutions.
- *the objective function should be adaptive at each node to determine the mapping and schedule. The GCLP algorithm attempts this.*

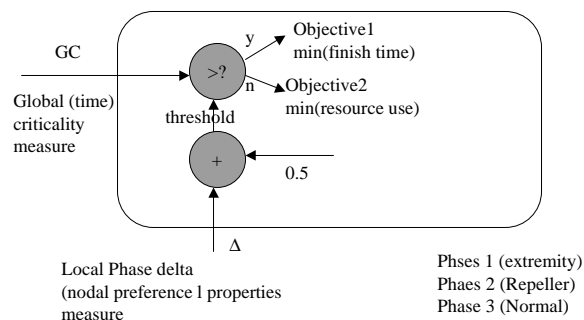
Mahapatra-Texas A&M-Fall'00

8

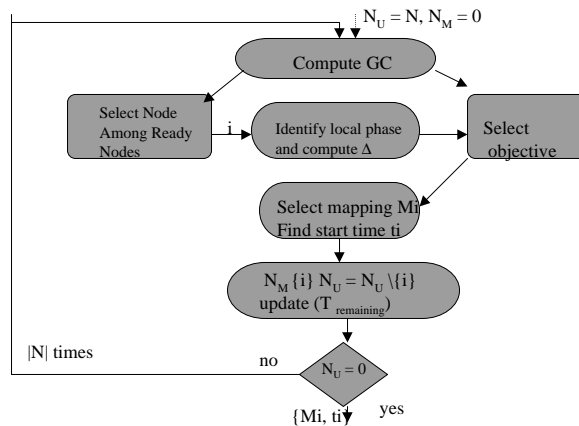
# GC-LP

- GC: *Global Criticality* is a look-ahead method that estimates time criticality of the algorithm. If time is critical, the objective function that minimizes finish time is selected, else the one that minimizes area.
- LP: LP is a classification of nodes based on their heterogeneity and intrinsic properties. Each node is classified as *extremity*, *repeller* or *normal* node. A measure called *local phase delta* quantifies the local mapping preferences of the node and update the threshold.

## Mapping objective at each step



## GCLP flow-graph



Mahapatra-Texas A&M-Fall'00

11

## Global Criticality

- Estimates time criticality at each step in look ahead fashion.
- At a given step, the hw/sw mapping and schedule of already mapped nodes is known
- $T_{rem}$  is determined on the basis of  $D$  and the schedule
- All the unmapped nodes are mapped to software and corresponding finish time  $T^s$  is computed.
- If  $T^s$  exceeds  $D$ , some of the unmapped nodes have to be moved from software to hardware to meet the deadline. Define this to be the set  $N_{S \rightarrow H}$ . The finish time ( $T^H$ ) is recomputed.
  - GC is defined here as fraction of unmapped nodes that have to be moved from software to hardware, to meet the feasibility. High GC  $\Rightarrow$  many as-yet unmapped nodes to be mapped to hw.

Mahapatra-Texas A&M-Fall'00

12

## Illustration

From Kalavade & Lee's paper in Journal of Design Automation of Embedded System.

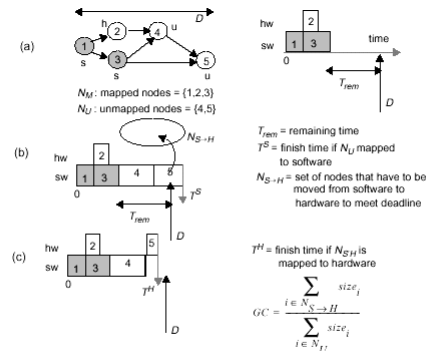


Figure 5. Computation of Global Criticality

## GCLP Procedure

Procedure: Compute \_GC

Input: Mapped ( $N_M$ ) and Unmapped ( $N_U$ ) nodes,  $D$ ,  $ts_p$ ,  $th_p$ ,  $size_p$ ,  $\forall i \in N$

Output: GC

S1. Find the the set  $N_{S \rightarrow H}$  of unmapped nodes that have to be moved from software to hardware to meet the deadline  $D$ .

S1.1. Select a set of node in  $N_U$ , using a priority function  $Pf$ , to move from software to hardware

S1.2. Compute the actual finish time ( $T^H$ ) based on these  $N_{S \rightarrow H}$  nodes being mapped to hardware

S1.3. If  $T^H > D$ , go to S1.1

S2. Calculate GC

## GC procedure explanation

- Priority function( $Pf$ ):
  - rank the nodes in order of decreasing software execution time  $ts_i$  or
  - use  $ts_i/th_i$  as function to rank the nodes. (greatest relative gain in time when moved to hardware) BEST RESULT
  - rank in increasing order of  $ah_i$  (nodes with smaller hardware area are moved out of software first)
- The finish time is computed by an  $O(|A|+|N|)$  algorithm. One can know if the set of nodes are feasible to move to hardware. If not, more nodes are required to move by repeating steps 1.1 to 1.3.
- GC is computed as a ratio of the sum of the sizes of the nodes in  $N_{S \rightarrow H}$  to the sum of the nodes in  $N_U$ . The size of a node is taken as number of *elementary operations* (add,multiply, etc..) in a node.

## Local Phase (LP) classification

- Motivation:
  - Nodes that consume disproportionately large amount of resource on one mapping compared to other are called extremities or LP 1. EX: hardware extremity requires a large area when mapped on to hardware but could be implemented inexpensively in software.
    - *The mapping preference of such nodes are quantified by extremity measure. This measure modifies the threshold used in GC comparison.*
  - Once feasible solutions are obtained, it is possible to swap the nodes to reduce the hardware area. The GCLP uses the concept of repeller or LP 2 nodes to perform on-line swaps. Need to look at nodal property. EX: bit-level versus memory operations. Node with bit-ops is software repeller.
    - *A repeller property is quantified as repeller value. Combined effect of all repeller properties is expressed as repeller measure.*

## Extremity nodes and measure

- Bottleneck resources: hardware  $\rightarrow$  area, Software  $\rightarrow$  time
- Hardware extremity nodes and software extremity nodes
- $E_i$  = extremity measure that is used to modify the threshold to which GC is compared when selecting the mapping objective. ( local phase delta for an extremity node  $i$ )
- Procedure: Compute\_Extremity\_Measure  $E_i$  for such nodes  
Input:  $ts_p, ah_p, \forall_i \in N, \alpha, \beta$  percentiles  
Output:  $E_i, \forall_i \in N, -0.5 \leq E_i \leq 0.5$ 
  - S1. Compute the histograms of all the nodes with respect to their software execution time and hardware areas.

Mahapatra-Texas A&M-Fall'00

17

## Extremity measure

- S2. Determine  $ts(\alpha)$  and  $ah(\beta)$  that corresponds to  $\alpha$  and  $\beta$  percentiles of  $ts$  and  $ah$  histograms respectively
- S3. Classify nodes into software and hardware extremity sets  $EX_s$  and  $EX_h$  respectively:  
if  $(ts_i \geq ts(\alpha) \text{ and } ah_i < ah(\beta))$ ,  $i \in EX_s$  (software extremity)  
if  $(ah_i \geq ah(\beta) \text{ and } ts_i < ts(\alpha))$ ,  $i \in EX_h$  (hardware extremity)
- S4. Determine the extremity value  $x_i$  for node  $i$ :  
  
if  $i \in EX_s$ ,  $x_i$

Mahapatra-Texas A&M-Fall'00

18

## Threshold Modification

- Let  $GC_k$  denotes the value at kth step when an extremity node  $i$  is to be mapped. If  $E_i$  is ignored, the threshold assumes its value of 0.5. Since  $GC_k$  is averaged over all unmapped nodes, mapping of node  $i$  in this case is based on  $GC_k$ . This leads to:
  - Poor mapping: Suppose node  $i$  is hardware extremity. If  $GC_k \geq 0.5$ , Obj1 is selected (minimum time), and  $i$  could get mapped to hardware based on time-criticality. However,  $i$  is a hardware extremity and mapping it to hardware is obviously poor choice for P1.
  - Infeasible mapping: Suppose node  $i$  is software extremity. If  $GC_k < 0.5$ , Obj2 is selected (minimum area) and  $i$  could get mapped to software. Node  $i$  is a software extremity, however, mapping on to software could exceed the deadline.

## Local Phase 2 (Repeller Nodes)

- The use of repellers to effect on-line swaps and reduce the overall hardware area. There are several repeller properties.
  - Bit-level instruction mix (BLIM) : sw repeller
  - Memory intensive instruction mix and look-up table instructions: hw repeller

## Reading Assignments

- 1. Repeller measure procedure
- 2. GCLP algorithm

## GCLP Algorithm

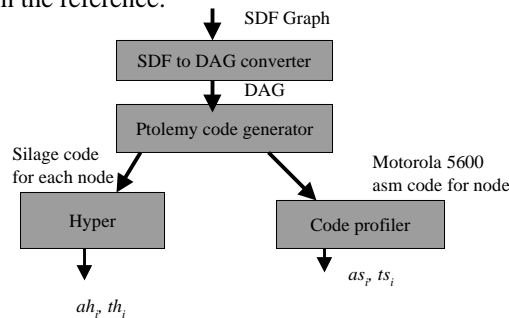
- Step1: GC is computed as the given procedure
- Step2: set of ready nodes computed whose predecessors have been mapped.
- Step3: selection of nodes are made from critical path (step5). Since the execution time is unknown at this point, effective execution time is determined here. It is assumed that a node is mapped to hardware with probability GC and to software with probability (1-GC).
- Step4: compute longest path based on the above effective execution time.
- Step5: select a node from estimated critical path.
- Step6: Mapping and schedule are determined
  - Use of extremity/repeller to modify the threshold. Use of weight factors vary the extremity/repeller measures.

## GCLP contd.

- *Obj1*: Select a mapping that minimizes finish time of a node. A node can begin execution only after all its predecessors have finished execution and data has transferred to it from predecessors. Also, node can not begin execution unless last node mapped to software has finished execution.
- *Obj2*: uses percentage resource consumption measure. It takes account of total cost of communication between node and its predecessors. This favors the software allocation as algorithm proceeds.

## Practical Examples

- 32KHz 2-PSK modem: applications given in SDF in Ptolemy environment. DAG is generated from SDF. Nodes are at task level granularity (carrier recovery, time recovery, equalizer, descrambler etc. 27 nodes). See Fig. 8 in the reference.
- Area time estimates:



## GCLP Versus ILP

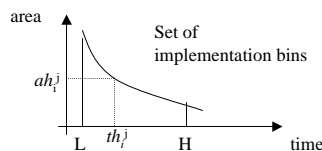
- Random graphs were selected
  - Partitioned using GCLP algorithm. ILP formulation was done using ILP solver CPLEX
  - Refer table for comparison.
  - GCLP is within 30% of optimal solution
  - Examples with more than 20 nodes could not be solved using ILP. Using GCLP, you can exceed 500 nodes.

Mahapatra-Texas A&M-Fall'00

25

## Extended Partitioning

- Implementation-bin curve revisited



- To minimize hardware area, each node is to be mapped towards H, subject to the deadline.
- Extended partitioning is about to choose appropriate implementation bin and mapping for each node so as to yield minimum area and meet the deadline constraint. Complex problem.

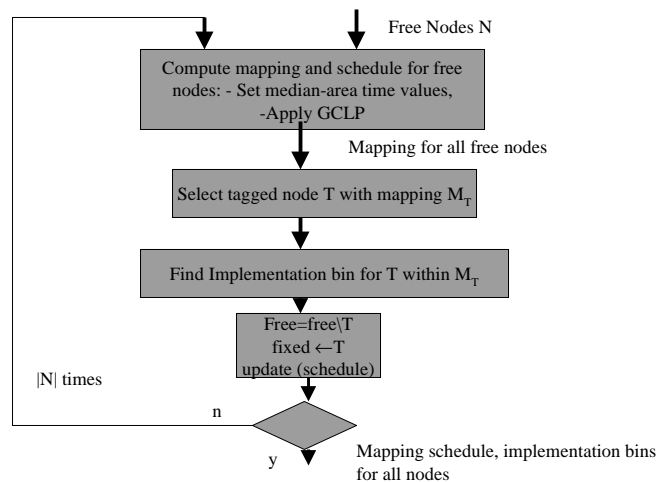
Mahapatra-Texas A&M-Fall'00

26

## Designing Algorithm: Guiding objectives

- Objective 1: (complexity that scales reasonably)
  - Binary partitioning has  $2^{|N|}$  mapping possibilities for  $|N|$  nodes. Given  $B$  implementation bins within a mapping, extended partitioning problem has  $(2B)^{|N|}$  possibilities in the worst-case. The algorithm complexity should not scale with dimensionality of partitioning process ( $|N|^{2B}$ ).
- Objective 2: (Reuse of GCLP)
  - Extended partitioning should decompose into two isolated steps such as mapping and bin selection. Use GCLP for mapping.
  - However, optimization in isolation is ruled out as there is a correlation between implementation bin and mapping.

## MIBS Heuristic

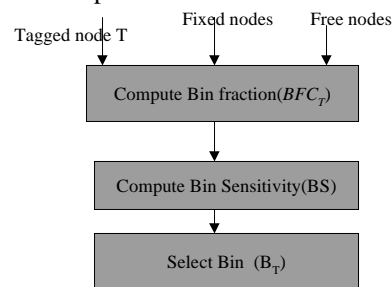


# MIBS Heuristic

- GCLP is used for mapping (design objective 2)
- GCLP and bin selection are applied alternately within each step: hence continuous feedback between mapping and implementation stages.
- ( $O(|N|^3 + B \cdot |N|^2)$ ), where B is number of implementation bins per mapping : scales polynomially (design obj 2).

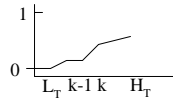
## Implementation-bin selection (Hardware-mapped)

- In MIBS algorithm, GCLP is applied each step to determine revised mapping of free nodes. Let the free nodes mapped to hardware at the current step is  $free^h$  nodes. A tagged node is selected from free nodes.
- Bin selection procedure:



## Bin Selection

- **Key Idea:** Use look-ahead measure to correlate the implementation bin of the tagged node with the hardware area required for the  $free^h$  nodes. It selects most responsive bin in this respect as the implementation bin for the tagged node.
- Assume that  $free^h$  nodes can be either L or H bins. Initially, say H bins.
- Now, for each bin  $j$  of tagged node T, compute the fraction of  $free^h$  nodes that need to be moved from H bins to L bins in order to meet timing constraints ( $BF_T^j$ ).
- The bin fraction curve  $BFC_T$  is the collection of the all bin fraction values of the tagged node T.



Mahapatra-Texas A&M-Fall'00

31

## Bin Selection

- **Bin sensitivity:** is the gradient of  $BFC_T$
- It reflects the responsiveness of the bin fraction to the bin motion of node T.
- **Example:** If maximum slope of bin fraction is between  $k-1$  and  $k$ , moving the tagged node from bin  $k-1$  to  $k$  will shift the largest fraction of free nodes to their L bins. (alternatively, moving  $k$  to  $k-1$  will result largest reduction in area)
- Hence select  $(k-1)$ th bin.

Mahapatra-Texas A&M-Fall'00

32