Hardware/Software Co-Design

Shao-Yi Chien

General Co-Design Problems

- Co-design of embedded systems
- Co-design of ISA's
- Co-design of reconfigurable systems

Important Issues in Co-Design

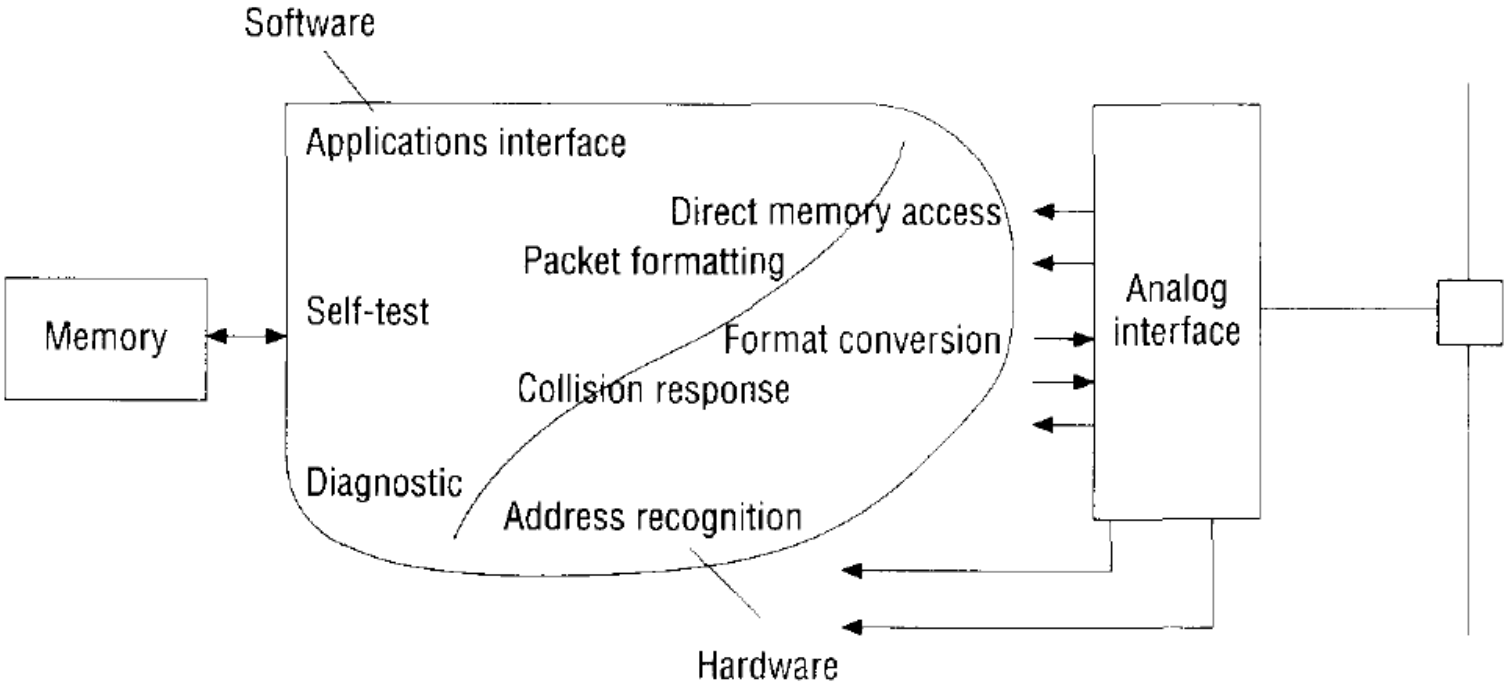
- Hardware/software partition
- Hardware/software co-simulation
- Hardware/software co-verification
- Hardware/software co-synthesis

Hardware-Software Cosynthesis for Digital Systems

R. K. Gupta and G. De Micheli, “Hardware-software cosynthesis for digital systems,” *IEEE Design & Test of Computers*, vol. 10, no. 3, pp. 29—41, Sept. 1993.

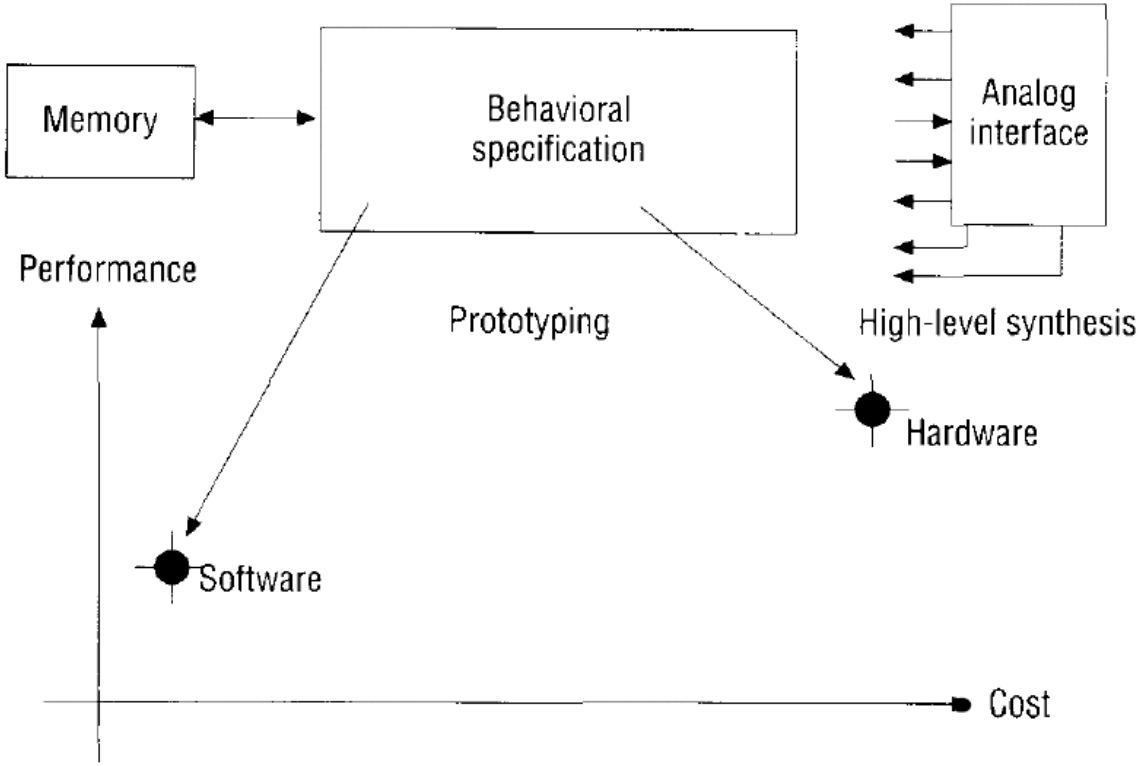
Introduction

- Design-oriented approach to system implementation



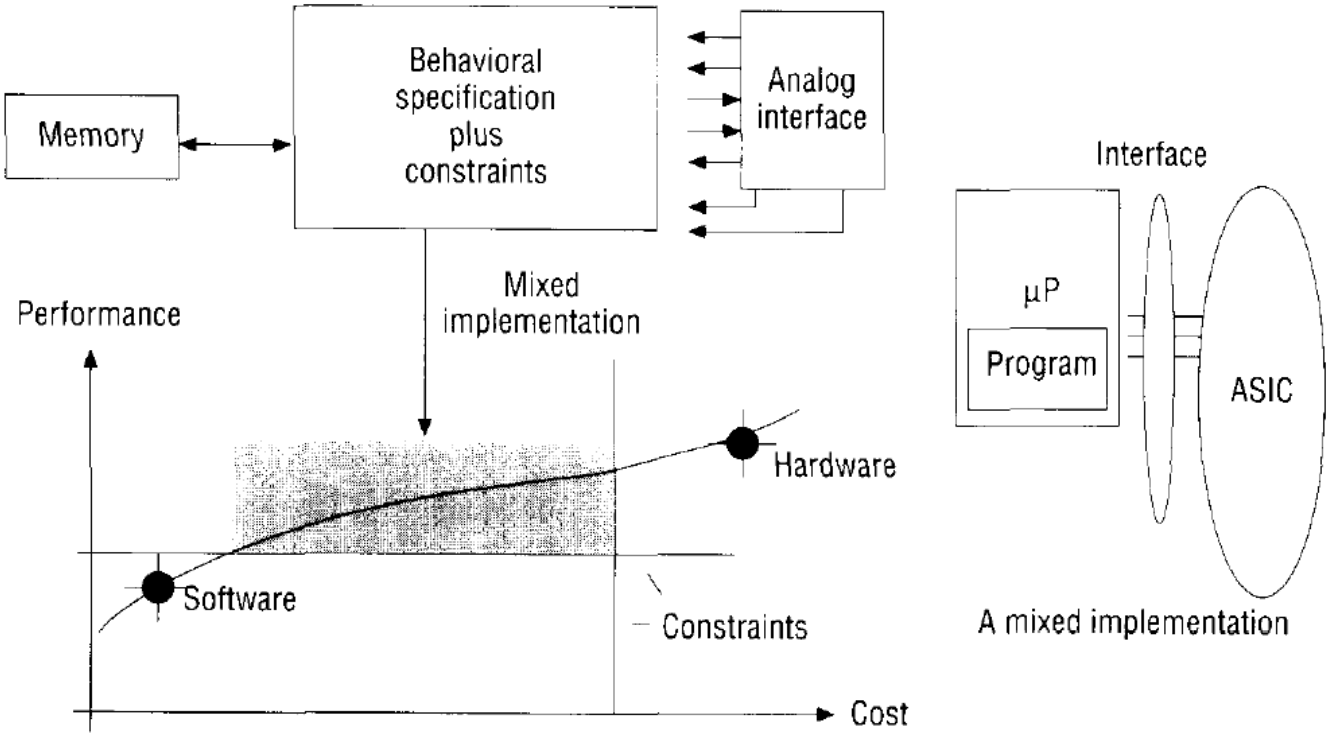
Introduction

- Synthesis-oriented approach

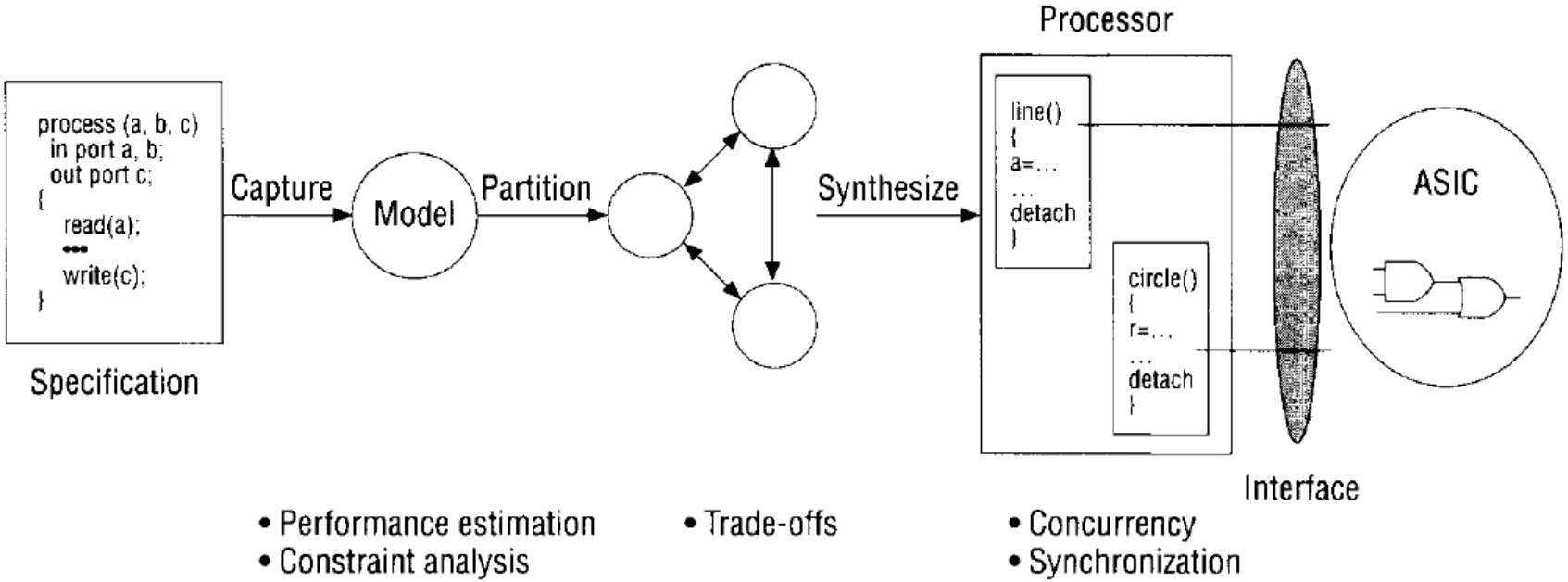


Introduction

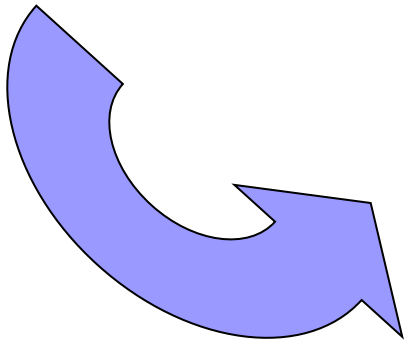
- Proposed method: for mixed implementation

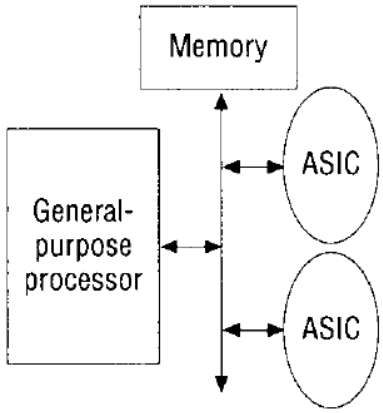


Synthesis Approach Overview



- Performance estimation
- Constraint analysis
- Trade-offs
- Concurrency
- Synchronization

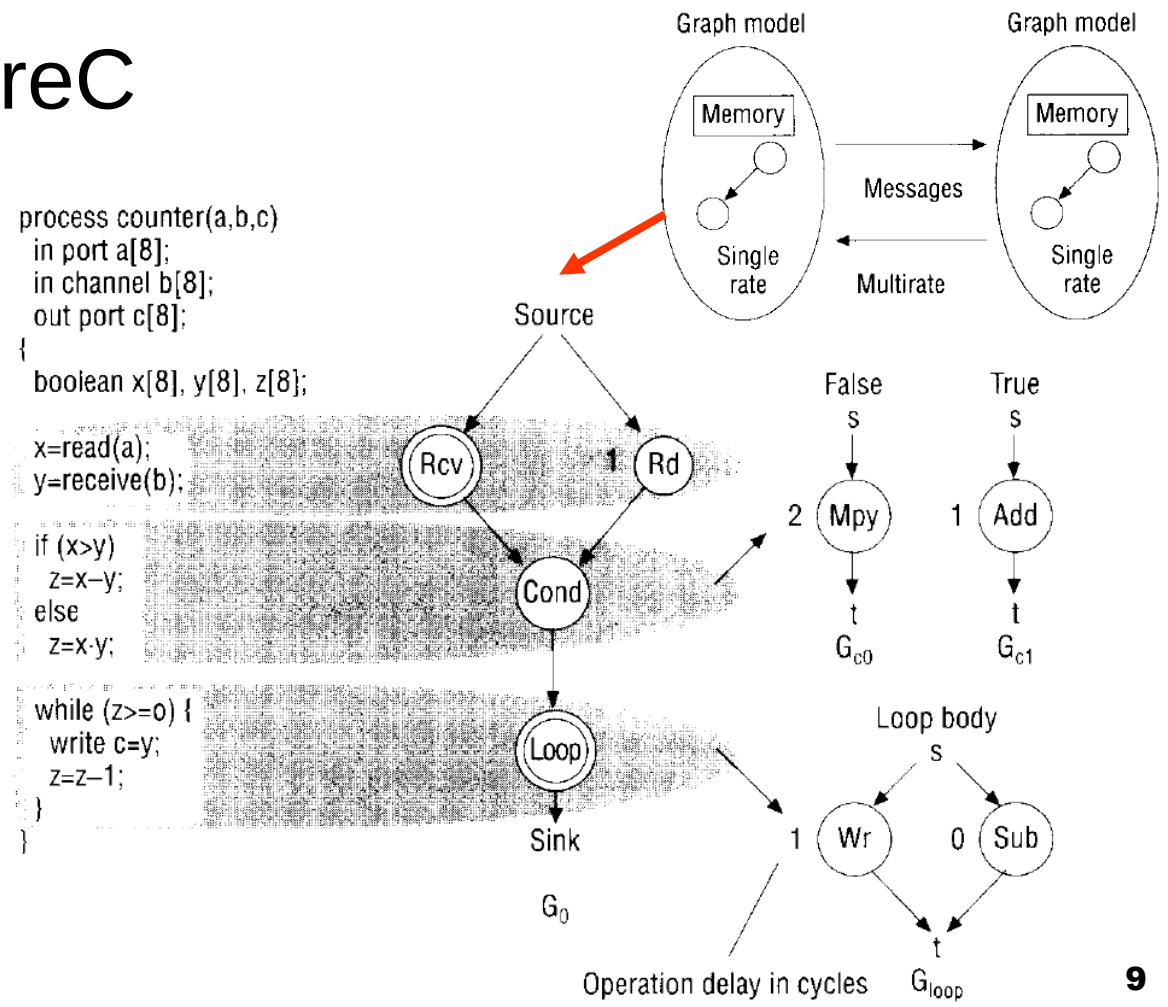




Capturing Specification of System Functionality and Constraints

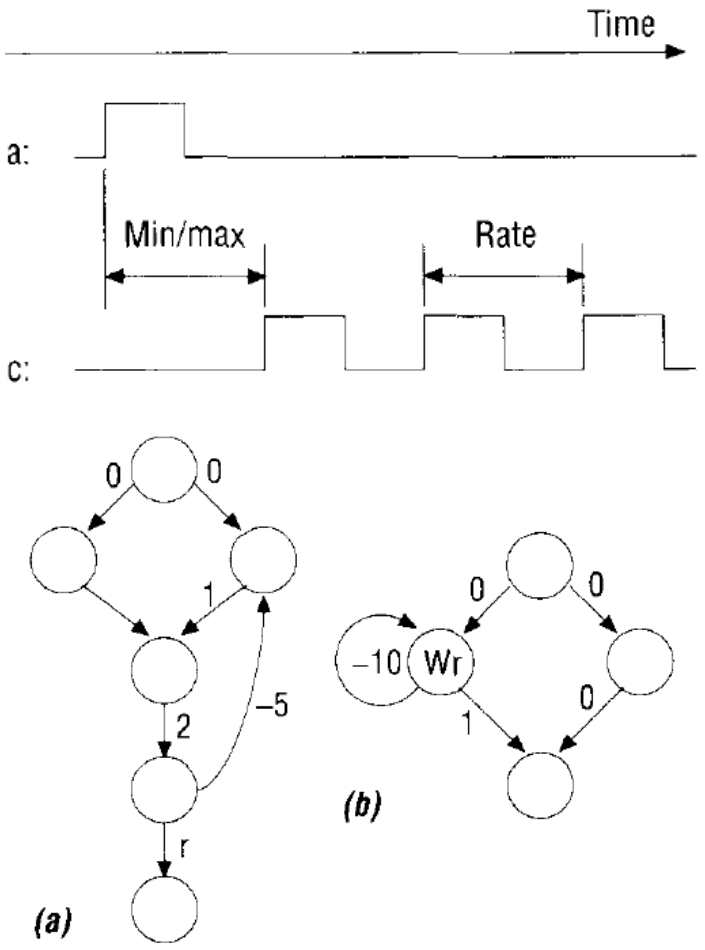
- Use HardwareC

Double circles: nondeterministic (ND) Delay operations
Edge: dependency



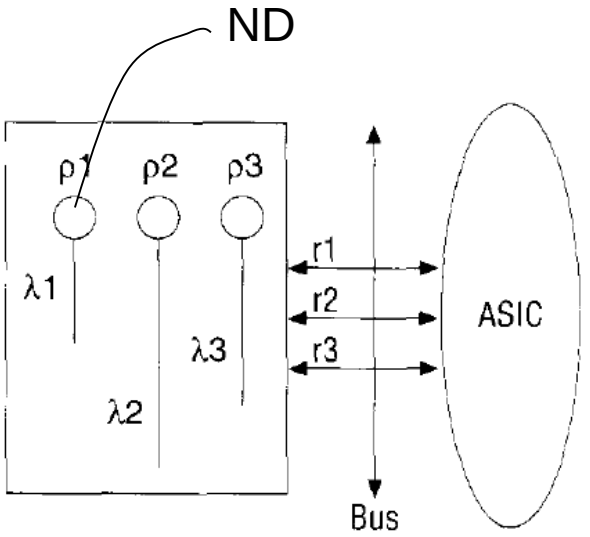
Capturing Specification of System Functionality and Constraints

- Within graph: single rate
- Across graph: multi-rate
- Timing constraints
  - Min/max delay constraints
  - Execution rate constraints



Capturing Specification of System Functionality and Constraints

- Model analysis
  - Processor model (cost, delay of operations, address calculation, …)
  - Assign appropriate delays to the operations with known delays
  - A timing constraint marginally satisfiable if it can be satisfied for all possible value within specified bounds on the delay of the ND operation
  - Think of software as a set of fixed-latency concurrent thread



System Partition

- Assign operations to hardware of software
- The assignment will determine the delay of each operation
- Consider the additional delay due to communication overheads

System Partition

- Properties
  - Thread latency λ_i (seconds)
  - Thread reaction rate ρ_i (per second)
  - Processor utilization $P = \sum_{i=1}^{n} \lambda_i \cdot \rho_i$
  - Bus utilization $B = \sum_{j=1}^{m} r_j$

    j: variable
    r_j:the inverse of the min time interval (second)
  - Hardware size S_H

System Partition

- Target
  - Timing constraints are satisfied for the two sets of graph models
  - Processor utilization $P \leq 1$
  - Bus utilization $B \leq \overline{B}$
  - A partition cost function is minimized $f = f(S_H, B, P^{-1}, m)$

System Partition

- How to find the solution?
  - Exhausted search
  - Use heuristics to find a good solution
  - Start with a constructive initial solution and then improve it iteratively by exchanging operations and paths between partitions

System Synthesis

- Hardware synthesis
- Software synthesis
- Interface synthesis

Hardware/Software Partition

S. Bakshi and D. D. Gajski, “Partitioning and pipelining for performance-constrained hardware/software systems,” *IEEE Trans. on VLSI*, vol. 7, no. 4, Dec 1999.

Introduction

- Hardware/software partition and synthesis of throughput constrained systems
- Not only partition (spatial partition), but also pipelining (temporal partition)

- Develop a design flow to determine
  - Allocation of system-level components
  - Functional partition
  - Pipeline to implement the system at minimal ASIC cost (actually, only consider the ASIC part, and the cost of processor is not included)

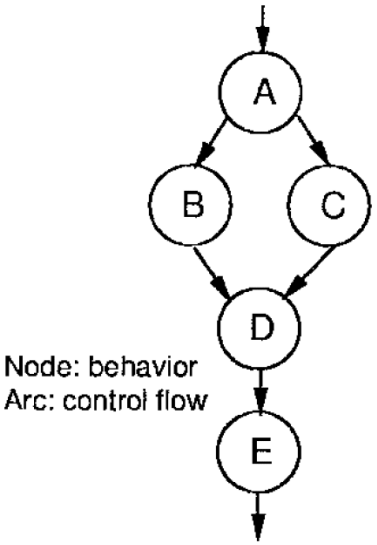
Problem Definition

- Given
  - A specification of the system as a control flow graph (CFG) of behaviors or tasks
  - A hardware library containing functional units characterized by <type, cost, delay>
  - A software/processor library containing a list of processors characterized by <type, clock speed, dollar cost, metrics file>
  - A clock constraints and a throughput constraint for the complete specification

Problem Definition

Input

Control Flow Graph



Hardware Library

Type	Name	Delay *(ns)*	Area *(gates)*
*	Mpy1	30	100
*	Mpy2	50	70
*	Mpy3	70	60
+	Add1	30	45
+	Add2	42	30
>	Cmp1	18	12
=	Cmp2	14	8

Software Library

Type	Clock *(ns)*	$ Cost	Metrics File
Pentium	10	90	pentium.metrics
PowerPC	10	75	powerpc.metrics
68000	50	60	mot68000.metrics

Constraints:

Clock = 10 ns
PS delay = 4000 ns

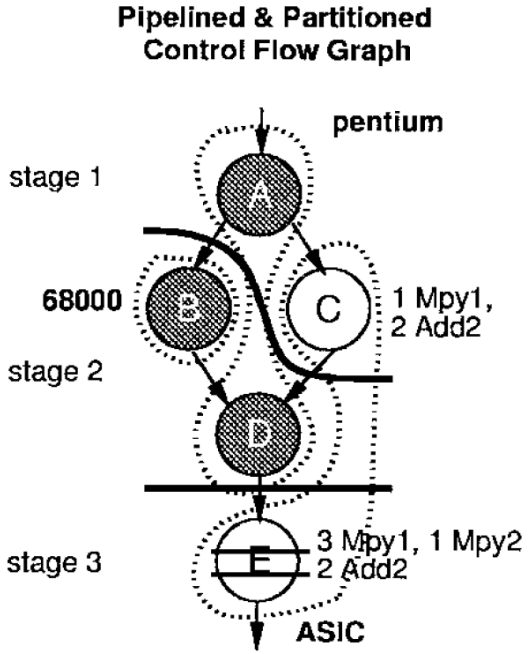
Problem Definition

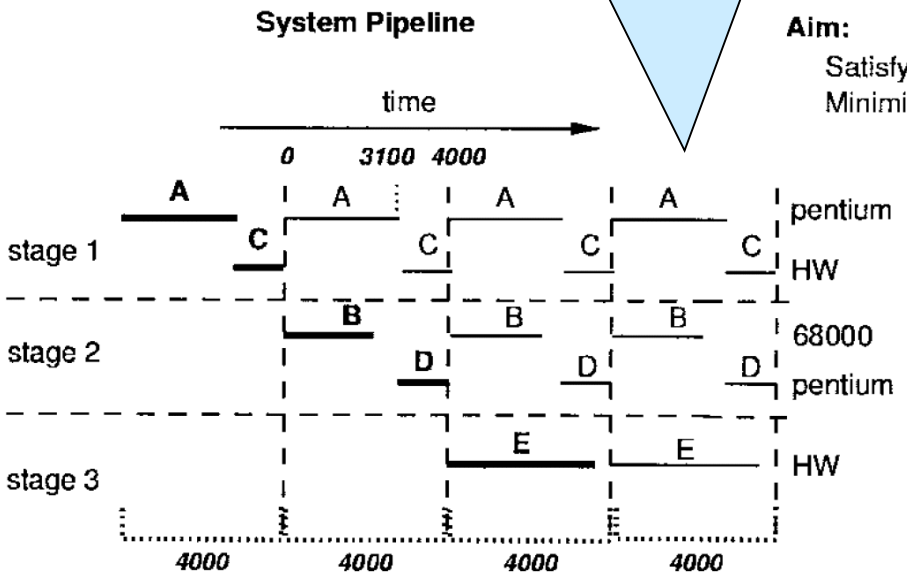
- Determine
  - An implementation type (software or hardware) for every behavior
  - The estimated area for each hardware behavior, as well as the total hardware area for the complete specification
  - The processor to be used for each software behavior, as well as the total number of processors for the complete specification
  - A division of the CFG into pipe stages of delay no more than the given throughput constraint

Problem Definition

A and D can share the same processor because their schedules are non-overlapped

Output





Aim:

Satisfy PS delay constraint

Minimize hardware area

Problem Definition

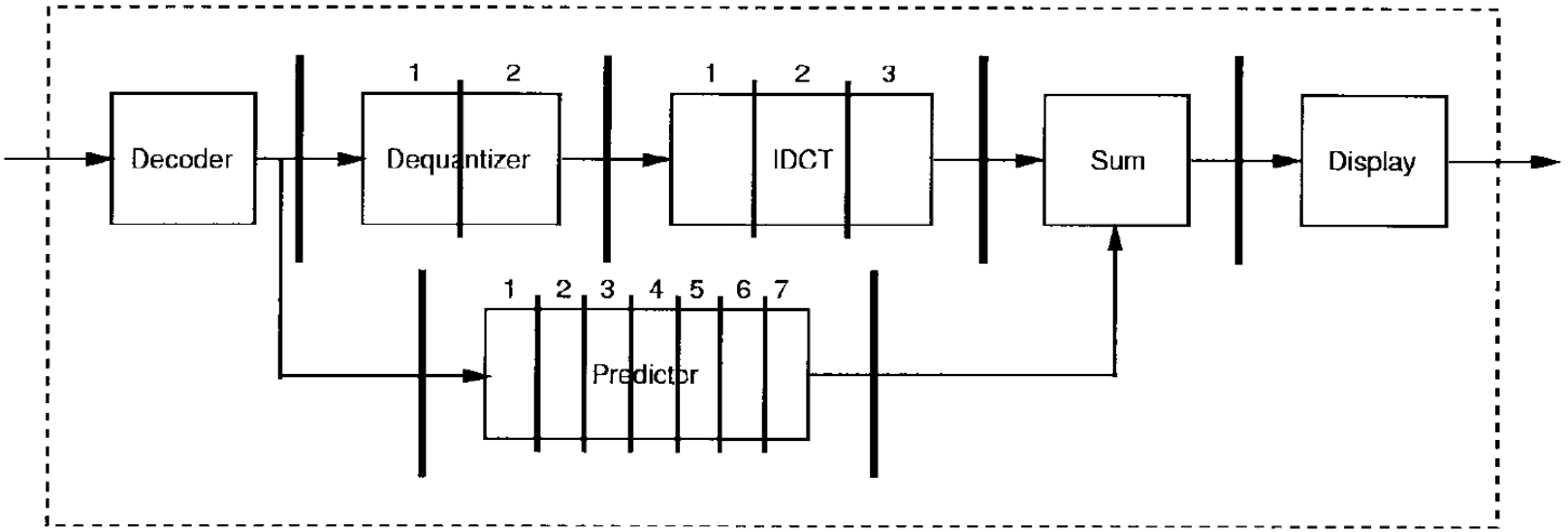
- Such that
  - Constraints on throughput are satisfied
  - Total area of behaviors to be implemented in hardware is minimized

Assumptions

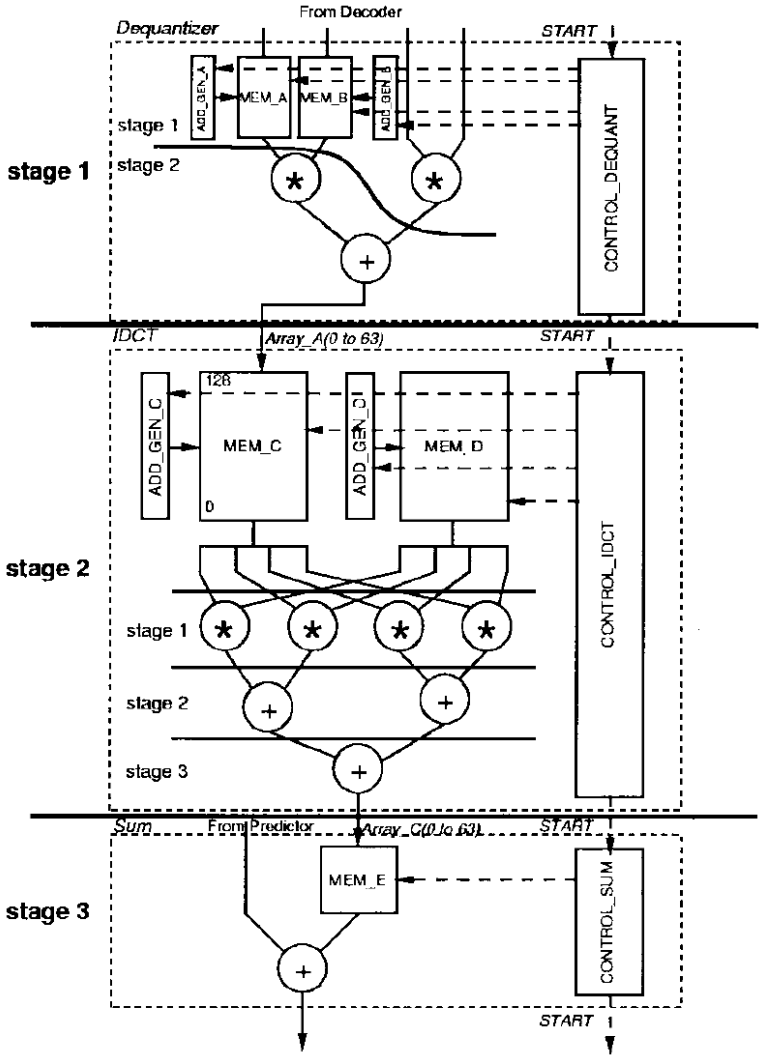
- Two software behaviors may share the same processor
- Every behavior partitioned to hardware will be implemented as a finite-state machine with datapath (FSMD)
- Resource in the datapath may be shared by multiple operations
- Two behaviors executing sequentially in the same pipe stage may share hardware resources, but resource sharing among multiple hardware blocks are not allowed

Pipelined Architecture

- Take MPEG-1 decoder as an example

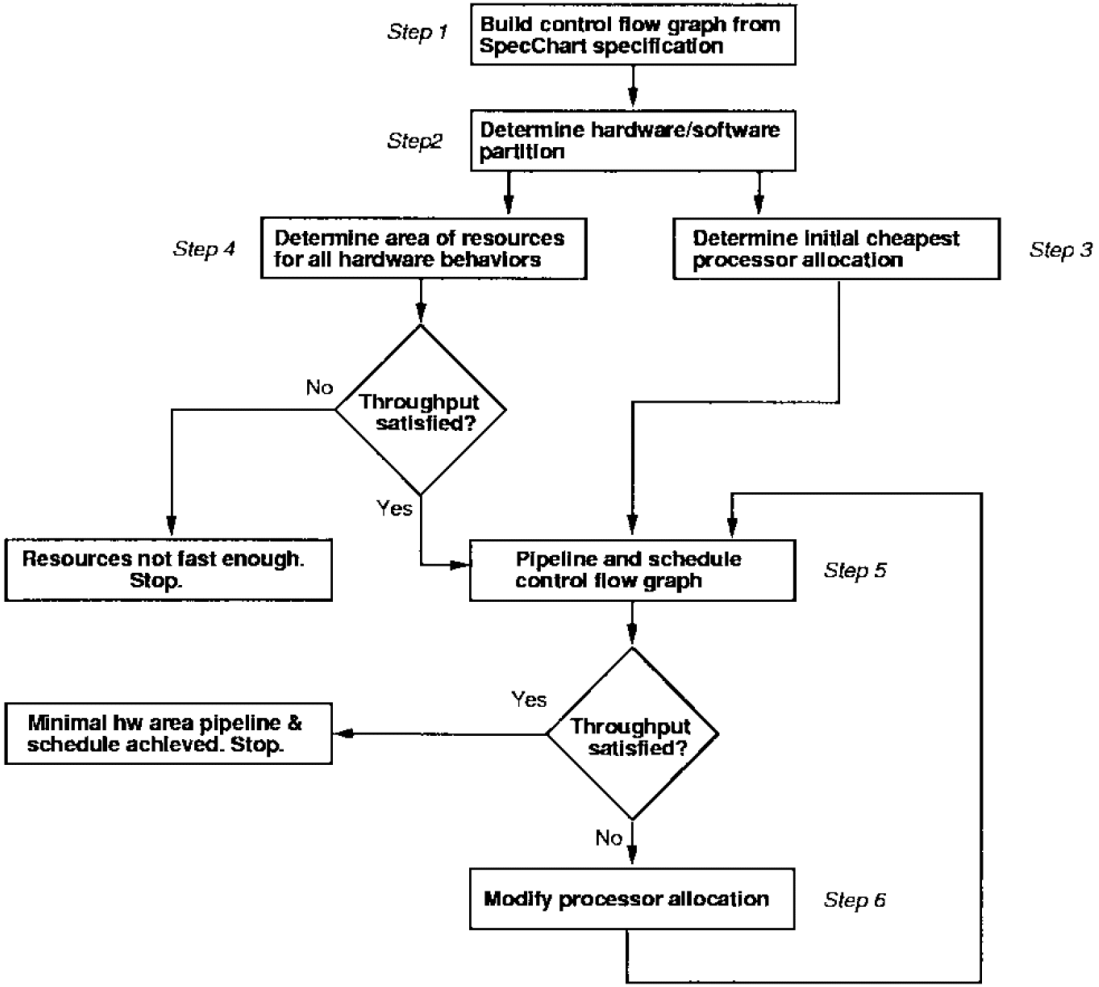


Pipelined Architecture



- Each stage produce/consume one 64-element array every 4000ns
- The fixed number of samples per input is referred to as a **sample set**
  ➔ block pipeline
- Each stage has sufficient memory
  - Ex: double buffer, one for execution, and the other collects samples produced by the preceding FSMD
  - Can we reduce the memory?

Proposed Algorithm



For short design time and high flexibility, it attempts to execute as many as possible behaviors in software.

Step 1, 2, and 3

- First, generate the CFG with *SpecCharts*, their in-house tool with VHDL
- Then …

Software estimation!

```
1.   Build CFG from SpecCharts specification.
2.   Build Processor Execution Time Table for all processors and all behaviors.
3.   For (every behavior in CFG)
4.       If (behavior execution time on fastest processor > throughput constraint)
5.           Behavior Type = Hardware.
6.       Else
7.           Behavior Type = Software.
8.       End If
9.   End Loop
10.  Arrange processors in ascending order of cost.
11.  For (every processor, P, in list)
12.      If (execution time of all software behaviors on P < T)
13.          Initial processor allocation = P.
14.          Exit Loop.
15.      End If
16.  End Loop
```

Step 2

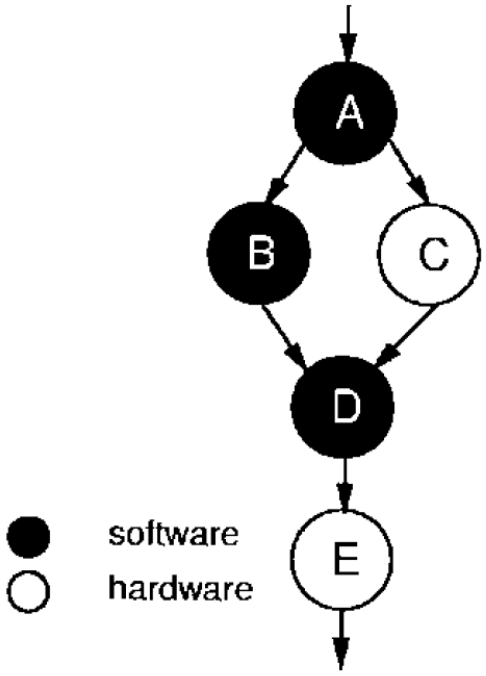
Step 3

Example of Step 1, 2, and 3

Processor Execution–Time Table

Behavior	Processor	Execution Time *(ns)*
A	pentium	3100
A	powerPC	3800
A	68000	6000
B	pentium	1400
B	powerPC	2200
B	68000	2800
C	pentium	8400
C	powerPC	12000
C	68000	18900
D	pentium	900
D	powerPC	1000
D	68000	1200
E	pentium	10230
E	powerPC	14870
E	68000	21080

Hardware/software Partition



Throughput constraint = 4000 ns
Initial processor allocation = powerPC

Step 4: Estimating Hardware Resources

- In order to minimize the cost of all the hardware behaviors, make them execute as slowly as possible
  ➔ attempt to achieve a pipe-stage delay equivalent to the constraint
- Determine the number of pipe stages with scheduling and resource allocation
- The output is a hardware execution table (HET) with area and execution time

Step 5: Scheduling and Pipelining the CFG

- We assign each behavior *v* to a pipe stage and to a time slot within a pipe stage such that **predecessor behaviors of v finish their execution either in a previous stage** or **in the same stage before the behavior *v* begins its execution**
- If *v* is a software behavior, we have to make sure that the selected processor is not used by any other behavior during the time interval that it is executing behavior *v*
- List-scheduling algorithm

Step 5: Scheduling and Pipelining the CFG

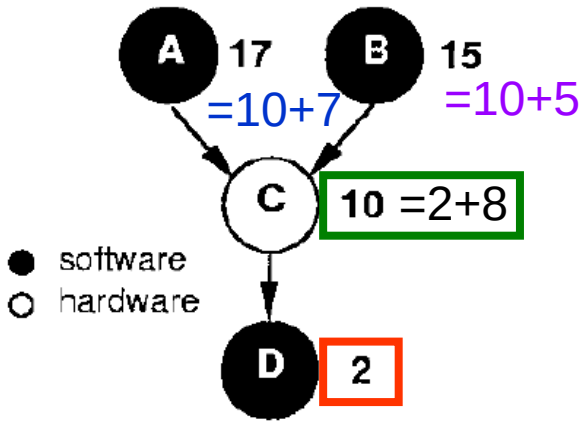
1. For every node in the *CFG*, determine the longest completion time from that node to any output node, and assign node priorities.
2. Initialize the utilization list of all processors.
3. **Loop**
4. Form a new ready list.
5. **Loop**
6. **If** *(node is type software)*
7. Find processor and corresponding time slot that gives earliest completion time.
8. **If** *(no available processor)*
9. Stop. No feasible solution.
10. **Else**
11. Assign node to processor & time slot.
12. Update utilization list of processor.
13. **End if**
14. **Else if** *(node is type hardware)*
15. Assign hardware to earliest feasible time slot.
16. **End if**
17. Mark node as scheduled and remove from ready list.
18. **Until** *(ready list is empty)*
19. **Until** *(all nodes in CFG are scheduled).*

Step 5: Scheduling and Pipelining the CFG

1. For every node in the *CFG*, determine the longest completion time from that node to any output node, and assign node priorities.
2. Initialize the utilization list of all processors.
3. **Loop**
4. Form a new ready list.
5. **Loop**
6. **If** *(node is type software)*
7. Find processor and corresponding time slot that gives earliest completion time.
8. **If** *(no available processor)*
9. Stop. No feasible solution.
10. **Else**
11. Assign node to processor & time slot.
12. Update utilization list of processor.
13. **End if**
14. **Else if** *(node is type hardware)*
15. Assign hardware to earliest feasible time slot
16. **End if**
17. Mark node as scheduled and remove from ready list.
18. **Until** *(ready list is empty)*
19. **Until** *(all nodes in CFG are scheduled).*

From output to input

Control Flow Graph



Processor Execution Time Table

Beh.	Processor	Exec. Time *(ns)*
A	P1	7
A	P2	9
A	P3	11
B	P1	5
B	P2	8
B	P3	9
D	P1	2
D	P2	3
D	P3	4

Thruput Constraint = 10 ns
Processor Allocation = {P1, P2}

Schedule order: A→B→D

Hardware Execution Time Table

Beh.	Area *(gates)*	Exec. Time *(ns)*
C	1200	8

Step 5: Scheduling and Pipelining the CFG

1. For every node in the *CFG*, determine the longest completion time from that node to any output node, and assign node priorities.
2. Initialize the utilization list of all processors.
3. **Loop**
4. Form a new ready list.
5. **Loop**
6. **If** *(node is type software)*
7. Find processor and corresponding time slot that gives earliest completion time.
8. **If** *(no available processor)*
9. Stop. No feasible solution.
10. **Else**
11. Assign node to processor & time slot.
12. Update utilization list of processor.
13. **End if**

...*are)*
...rliest feasible time slot.
...d remove from
...*led).*

Processor Execution Time Table

Beh.	Processor	Exec. Time *(ns)*
A	P1	7
A	P2	9
A	P3	11
B	P1	5
B	P2	8
B	P3	9
D	P1	2
D	P2	3
D	P3	4

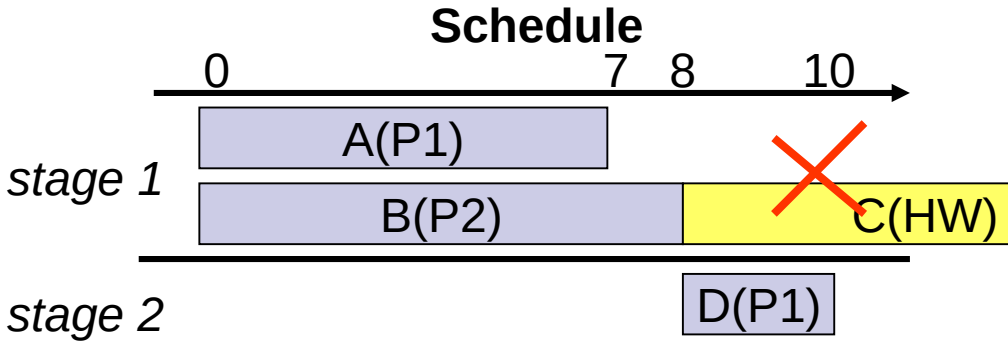
Hardware Execution Time Table

Beh.	Area *(gates)*	Exec. Time *(ns)*
C	1200	8

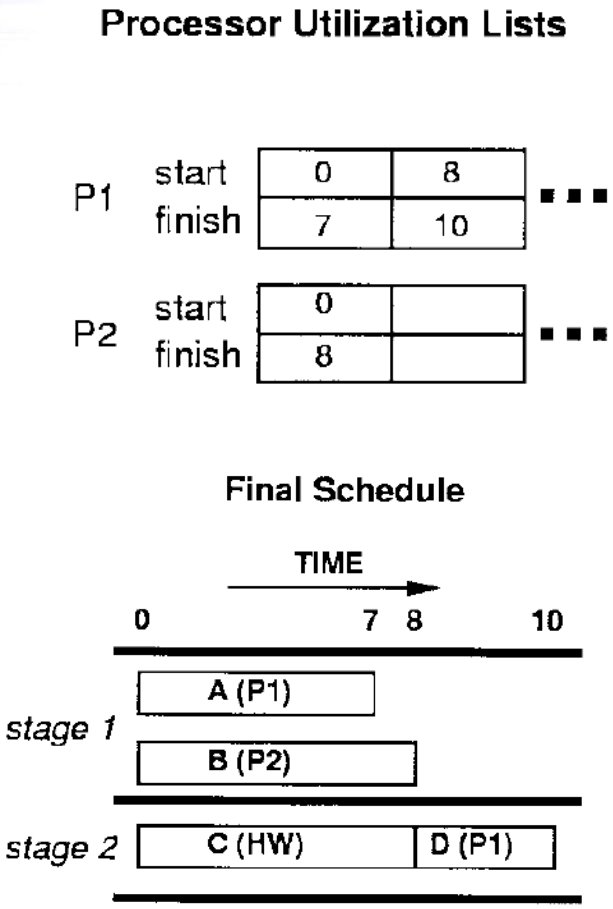
- Software schedule order: A→B→D, assume we now have two processors: {P1, P2}

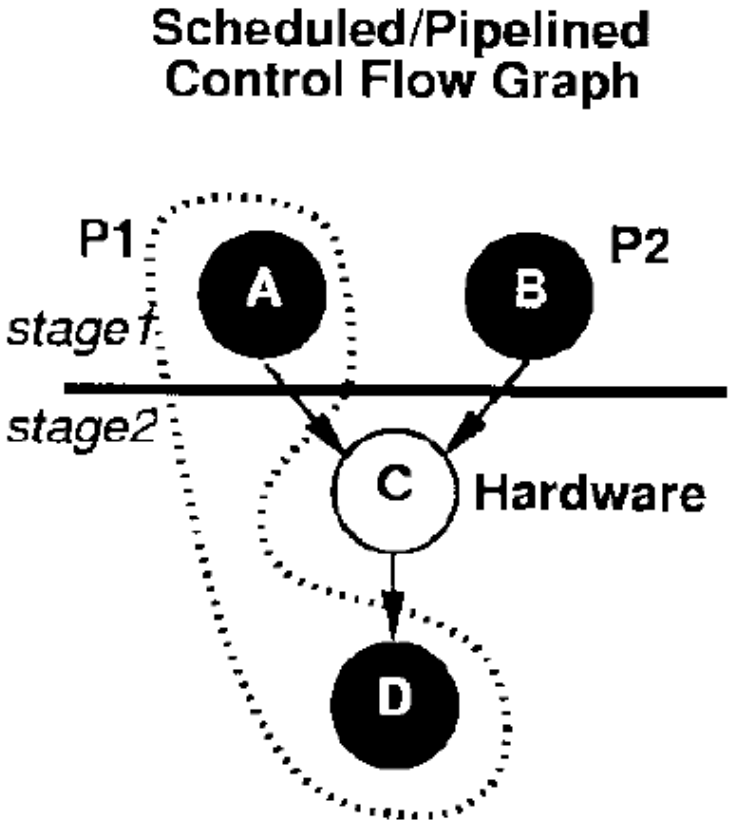
Proc.	Beh. A	Beh. B	Beh. D
P1	7, 1	~~(7+5), 1~~	(8+2), 2
P2	~~9, 1~~	(0+8), 1	~~(8+3), 2~~

Each entry: completion time, pipe stage



Step 5 : Scheduling and Pipelining the CFG





Step 6: Modifying the Processor Allocation

- If we do not find a feasible time slot for a software node ➔ use faster processor or increase the number
- Use a simple almost exhaustive method for the modification
- Stop increasing the number of processors when it equals the number of software nodes

Processor Costs

Processor	$ Cost
P1	50
P2	30
P3	10

Processor Modification Table

Allocation	$ Cost
P3	10
P2	30
P1	50
P3 P3	20
P3 P2	40
P2 P2	60
P2 P1	80
P1 P1	100
P3 P3 P3	30
P3 P3 P2	50
⋮	
P1 P1 P1	150

Order

Possible Extensions

- Consider interface cost and delay
- Partitioning among multiple ASIC's
- Hardware/software cost function
  - Combine all the cost of hardware, processor, and buses

Experimental Results

- MPEG System: precise hardware estimation

TABLE I
COMPARISON OF FUNCTIONAL UNITS USED IN MANUAL DESIGN OF MPEG MODULES VERSUS THOSE ESTIMATED BY OUR ALGORITHM

	Manual Design				*Our Estimation Algortihm*				
	PS delay constraint = 4000 ns, Clock = 25 ns, for all manual and estimated designs								
Behavior Name	**PS delay**	**# Stages**	**# States**	***Component Area (gates)*** **Components**	**PS delay**	**# Stages**	**# States**	***Component Area (gates)*** **Components**	**Comments**
leaf_deq0 *Part of Dequantization*	3325	1	7	***260*** 12–ADD/SUB, 12–CMP>= <=	3300	1	6	***417*** 12–ADD, 11–SUB, 12–CMP >, 8 CMP =	Manual design uses mulit–functional comps; estimated design does not.
leaf_idct1 *Inverse Discrete Cosine Transform*	3325	2	3	***10,384*** (4) 10–MUL, 14–ADD, 13–ADD (2) 12–ADD, 7–ADD, 7–CMP=	3175	2	3	***10,786*** (4) 10–MUL, (5) 16–ADD, 16–CMP =	Manual design uses multiple bitwidth adders; estimated design does not.
leaf_motion *Part of motion prediction*	2025	1	21	***1,472*** 6–MUL, 10–ADD/SUB, 10–CMP = > <	2900	1	29	***2,004*** 6–MUL, 10–ADD, 10–SUB, 10–CMP >, 10–CMP <=, 10–CMP =	Manual design uses mulit–functional comps; estimated design does not.
leaf_addrf *Address calc.*	3425	1	7	***2,015*** 7–MUL, 5–ADD, 10–ADD, (3) 13–ADD, 5–CMP = >	3200	1	6	***1,828*** 7–MUL, (3) 13–ADD, 8–CMP > 8–CMP =	Estimated design performs better resource sharing.
leaf_comp *Part of motion prediction*	3225	1	3	***256*** (2) 8–ADD, 6–CMP =	3100	1	3	***307*** (2) 8–ADD, (3) 8–CMP	Cannot explain.
leaf_plus *Summation block*	3275	1	3	***260*** 10–ADD, 6–ADD, 10–CMP < 10–CMP >, 6–CMP =	3200	3	2	***194*** 10–ADD, 8–CMP >, 8–CMP <	Estimated design performs better resource sharing.

Experimental Results

- MPEG System: larger design space, fewer gate count and fewer of required number of processors

TABLE II
DESIGN EXPLORATION FOR MPEG BY OUR ALGORITHM AND BY MANUAL METHODS

Our Algorithm			Manual Designs		
PS delay	# Pipe Stages	Processors & FU + mem area *(gates)*	PS delay	# Pipe Stages	Processors & FU+mem area *(gates)*
2980	12	462304	–	–	–
3840	12	395838	4000	12	441700
4480	11	376998	4988	12	406882
5760	12	Pentium x 2 335307	5780	12	Pentium x 1 394890
7680	12	Pentium x 4 224455	7172	12	Pentium x 2 302184
17810	10	Pentium x 3 152825	17670	12	Pentium x 4 209776
653970	1	Pentium x 1	661622	1	Pentium x 1

Experimental Results

- Other examples: large design space exploration

TABLE III
DESIGNS EXPLORED BY PARTITIONING AND PIPELINING THE VOLUME. DHRC AND AR EXAMPLES

Volume System			DHRC			AR Filter		
PS delay	# Pipe Stages	Processors & FU area (gates)	PS delay	# Pipe Stages	Processors & FU area (gates)	PS delay	# Pipe Stages	Processors & FU area (gates)
420	6	6541	2580	1	33539	20	1	50688
940	4	6231	3870	1	28131	40	1	31616
1680	3	5715	5180	1	24326	60	1	17664
3470	1	5715	7760	1	13609	80	1	14304
8340	2	Sparc x 1	34850	1	3619	100	1	11104
		Pentium x 1	134620	1	Pentium x 1	150	1	9536
		1497	247850	1	PowerPC x 1	200	1	8160
17010	1	PowerPC x 1	1423840	1	Motorola 68020	500	1	6144
52520	1	68020 x 1	2193250	1	Intel 80286	2190	1	Pentium x 1
						3790	1	PowerPC x 1
						22440	1	68020 x 1
						78240	1	68000 x 1

Architecture Synthesis: Maybe it is the Future?