Analysis and Estimation

Hardware Software CoDesign

September 2011

Agenda

Analysis and Estimation

1. Basic Themes that came out through the Historical Background (repeat)
2. Analysis and Estimation
3. Continue on "Answering Machine" Assignment Discussions

Analysis and Estimation

Basic Themes

1. **Modeling** :: Because parallelism is important, the choice of an appropriate modeling paradigm is also important. Different modeling formalisms need to be developed that capture the various aspects of system behavior.

2. **Analysis and Estimation** :: Different models of the same system behavior need to be analyzed and estimated for performance. Performance would include tradeoffs for Area, Speed, Power. Cost to build, Time to Market are other factors.

3. **System-level partitioning, synthesis and interfacing** :: The basic steps of co-synthesis. A range of methodologies are applied for this.

4. **Implementation generation** :: Once the architecture has been generated and trade-offs known, the designs for the hardware and software components must be created.

5. **Co-simulation and Emulation** :: This helps designers to evaluate architectures and validate assumptions on implementations. Emulation uses FPGA / other emulation techniques to further speed up execution of system models.

Analysis and Estimation

Introduction

1. **Analysis** :: Assumptions that which are formally dervived. Other than simulation, formal analysis covers all possible behaviors of a system that lead to more reliable verification results. Analysis precision is crucial to avoid costly, overly conservative designs that might not be competitive.

2. **Estimation** :: Assumptions that are based on "educated guesses" possibly based on analysis steps.

Analysis and Estimation

Analysis – Process Path

1. **Process Path** :: A sequence of process statements that is executed for given input data.
   - If there are loops, statements can be executed several times on a path.
   - Data dependent control statements in the process decide which of the possible path is taken.
   - In the most general case :: the determination of the set of possible process paths and the frequency of execution of each single statements is a **known non-computable problem**
   - Practical embedded system processes are bounded in their running time and so, path length.
   - GROUP DISCUSSION How does one determine upper bounds on a set of possible process paths?

Analysis and Estimation

Analysis – Process Path – Path Enumeration Technique

1. GROUP DISCUSSION How does one determine upper bounds on a set of possible process paths?

   One approach was the introduction of **the implicit path enumeration technique.**

   (a) Divide the program into basic blocks :: short sequences of statements in a program that include only one entry point at the first statement and one exit point at the end.

   (b) Determine the running time for these basic blocks (by architecture analysis).

   (c) The path constraints are implicitly given by equations and inequalities that describe the relative frequency of basic block execution.

   (d) Basic block running times and equations / inequalities form an ILP problem (IntegerLinearProgramming).

2. This technique is good for **Single-Processor Systems with Simple Memory Architectures** and is widely used.

3. This technique can be easily extended to cost functions other than timing, by changing the data provided by the model from timing to say power consumption.

4. Besides worst case analysis, it could be possible to add statistical methods that assume branching probabilities to obtain "statistical timing".

5. GROUP DISCUSSION :: What are the scenarios where the process path technique can get complicated.

Analysis and Estimation

Analysis – Process Path – Path Enumeration Technique

1. GROUP DISCUSSION :: What are the scenarios where the process path technique can get complicated.

   (a) The behavior of instruction cache does not fall under the ILP approach since dependencies between statements depend on their memory address and not on the control flow.

   (b) Power analysis is more complicated even for smaller architectures, because in CMOS technology, most of the power consumption is caused by transistor and wire switching that depend on the "sequence of operations" rather than on a single instruction.

   (c) However, for processor architectures like DSP's, the enabling / disabling of datapaths in the instruction sequence can effectively enable power analysis. Even when doing so there is "data dependency" on power analysis.

   (d) A simplified Process path technique can get complicated during analysis of parallel process execution.

Analysis and Estimation

Analysis Of Parallel Process Execution

1. A simplified Process path technique can get complicated during analysis of parallel process execution. While single processes run un-interrupted, the execution of parallel processes can be delayed.

2. Reasons for delay are ::
   (a) Process p_i waits for the output of another process p_j
   (b) Process p_j blocks a shared resource (shared memory, IO channel..) that p_i requires for continue execution.
   (c) Another reason for delay is processes running on one component can interrupt each other.

3. Thus, a problem statement would be :: given a set of tasks, to decide which task gets the CPU time at a given instance in time

4. The Goals are :-
   (a) Minimize turnaround time
   (b) Maximize throughput
   (c) Maximize CPU utilization
   (d) Minimize response time
   (e) Fairness :: avoid starvation

Analysis and Estimation

Analysis Of Parallel Process Execution – Running on One CPU

1. Discuss the following New terms :-

   (a) **Computation Time** :: of a task (or process) – the execution time of a process when it is run without interruption.

   (b) **Response Time** :: of a task – the time between the request for process execution, to the instant when a process has finished execution, including all delays by interrupts and dependencies.

   (c) **Deadline** :: of a task – the instant in which a task must have finished.

   (d) **Overflow** :: of a task – occurs at a time t if t is the deadline of an unfulfilled request.

   (e) **Critical instant** :: of a task – an instant at which a request for the process will have the largest response time.

   (f) **Scheduling** :: of a task – the order in which tasks are executed.

   (g) **Pre-Emptive Scheduling** :: of a task – interrupts running tasks to execute other tasks.

   (h) **Non-Pre-Emptive Scheduling** :: of a task – does not support interrupting executing tasks.

   (i) **Static Scheduling** :: of a task – is when the scheduling algorithm determines a fixed order of process execution.

(j) **Dynamic Scheduling** :: of a task – is when the scheduling is done during run time and requires a scheduler task (itself).

(k) **Scheduling Algorithm** :: is a set of rules that determine the task to be executed at a particular moment.

Analysis and Estimation

Analysis Of Parallel Process Execution – Running on One CPU

1. The timing of parallel processes running on one cpu is a problem in "real time computing"

2. In this problem, many applications (signal processing, control) require "periodic process execution", or they can be mapped to periodic process execution.

3. Assume you know the process deadlines (equal to the individual process periods).

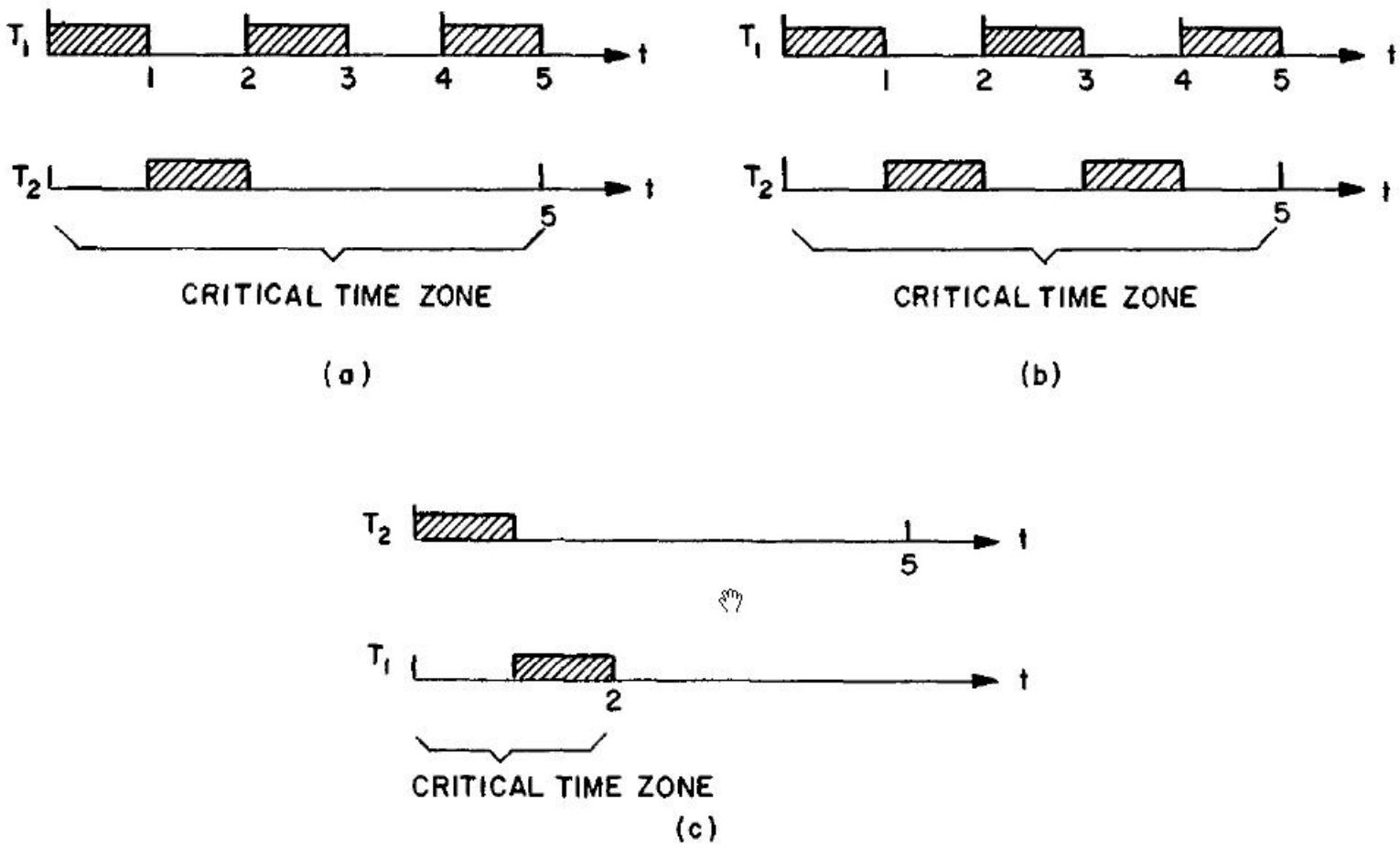
4. And the process maximum computation time.

5. **GROUP DISCUSSION** :: Can you derive an algorithm to show that the system of processes is schedulable?

Analysis and Estimation

Analysis Of Parallel Process Execution – Running on One CPU – Rate Monotic Analysis

1. Let $(T_1, T_2, ...T_m)$ be the request Time periods of processes (tasks) $(p_1, p_2, ...p_m)$
2. Let $(C_1, C_2, ...C_m)$ be their Computation Times.
3. So, the request Rate $= (1/T_n)$ for process n.
4. Take an example where $T_1 = 2, T_2 = 5$ and $C_1 = 1, C_2 = 1$
5. Static prioritization was optimal, with the highest priority going to the process with the shortest period. This is known as **Rate Monotonic Priority Assignment**



(a) (Process p_1 has higher Priority) – Feasible

(b) (Process p_1 has higher Priority) – C_2 can be increased to 2

(c) (Process p_2 has higher Priority) – C_1, C_2 can be at most to 1

6. Processor Utilization $= \sum_{i=1}^{m} (C_i/T_i)$

Analysis and Estimation

Analysis Of Parallel Process Execution – Running on One CPU – Rate Monotic Analysis

Theorem 1

For a set of n independent periodic tasks

if $$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1)$$

where
C_i = execution time
P_i = period of task i

then the set is schedulable by rate monotonic algorithm for all task phasings.

Analysis and Estimation

Analysis Of Parallel Process Execution – Running on One CPU – Rate Monotic Analysis

1. **GROUP DISCUSSION**

Schedulable?

Task Set

Process	C	T	U= C/P
P1	20	100	0.2
P2	40	150	0.267
P3	100	350	0.286
		Total	0.753

Analysis and Estimation

Analysis Of Parallel Process Execution – Running on One CPU – Deadline Driven Analysis

1. While RMA is a static priority scheme, the deadline driven analysis is a dynamic scheme.

2. Priorities are assigned to tasks according to the deadlines of their current requests.

3. A task will be assigned a highest priority if the deadline of its current request is the nearest, and will be assigned the lowest priority if the deadline of its current request is the furthest.

4. At any moment, the task with the highest priority and yet unfulfilled request will be executed.

5. For a given set of m tasks, the deadline driven scheduling algorithm is feasible if and only if $(C_1/T_1) + (C_2/T_2) + ... + (C_m/T_m) \leq 1$

6. Total demand cannot exceed the available processor time.

7. The deadline driven scheduling algorithm is optimum in the sense that if a set of tasks can be scheduled by any algorithm, it can be scheduled by the deadline driven scheduling algorithm.

Analysis and Estimation

Analysis Of Parallel Process Execution – Running on One CPU – Mixed Scheduling Algorithm

1. Implement a deadline driven scheduler for the slower paced tasks.

2. Let tasks $1, 2, ..., k$ be the k tasks of shortest periods. Schedule these according to the fixed priority rate-monotonic scheduling algorithm.

3. Let the remaining tasks $k+1, k+2, ..., m$ be scheduled according to the deadline driven scheduling algorithm when the processor is not occupied by tasks $1, 2, ..., k$

4. **Example** With 3 tasks.
   - $T_1 = 3, T_2 = 4, T_3 = 5$; $C_1 = 1, C_2 = 1, C_3 = 1(ratemonotonic), 2(mixed)$
   - Fixed Priority $U = 1/3 + 1/4 + 1/5 = 78.3\%$
   - Mixed Scheduling $U = 1/3 + 1/4 + 2/5 = 98.3\%$

Analysis and Estimation

Acknowledgements

1. Readings in Hardware / Software Co-design ::G. D. Micheli, Rolf Ernst, Wayne Wolf :: Ch 3

2. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment :: C. L. Liu, James W. Layland :: Journal of ACM (vol 20, January 1973).

3. Hardware / Software Co-Design - Principles and Practice :: J. Staunstrup & W. Wolf :: Ch 2.3