CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020
Batch: 2016-17
(LECTURE # 6)

FAKHRA AFTAB
LECTURER
DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Recap of Lecture # 5

State of the System & Events

Three primary types of measures

Issues in Measurement

Classification of Measurement

Hardware Monitoring

Chapter # 2 (Cont'd)

MEASUREMENT TECHNIQUES

Example 1
(Illustrating the measurement procedure)

Question: Consider a system with one CPU and *n* channels. Show the setup for measuring the fraction of time the CPU and *k* channels (for a given $k \in 1 \ldots n$) are busy simultaneously.

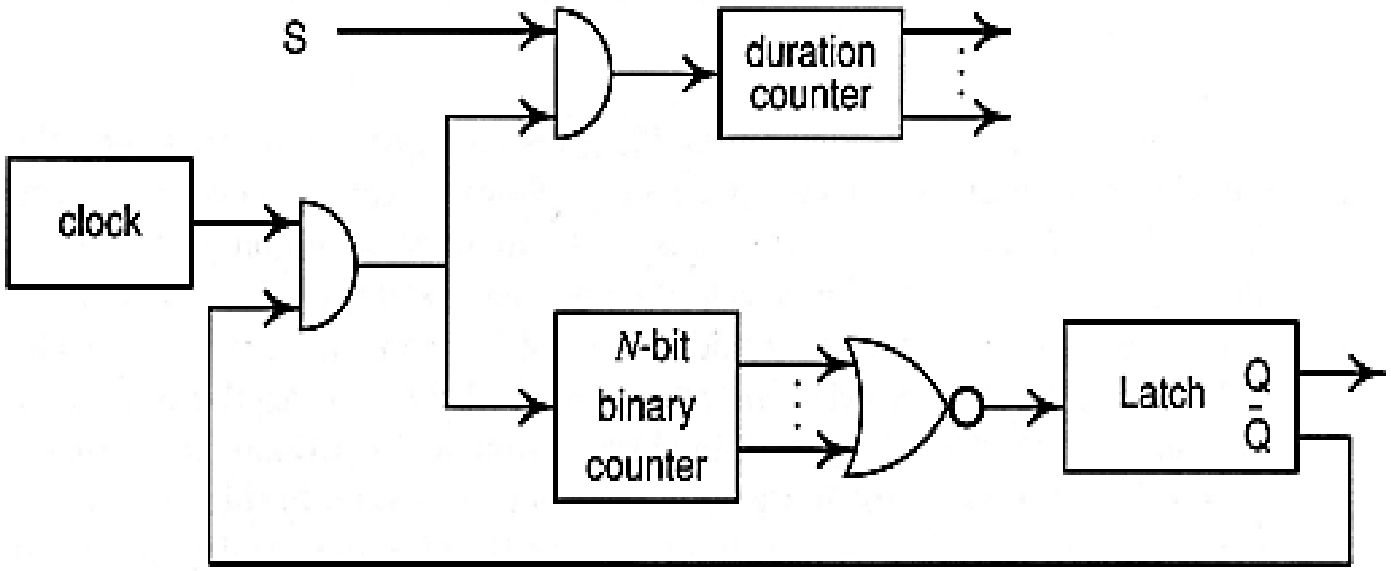

Fig 1: Setup for CPU-Channel overlap measurement

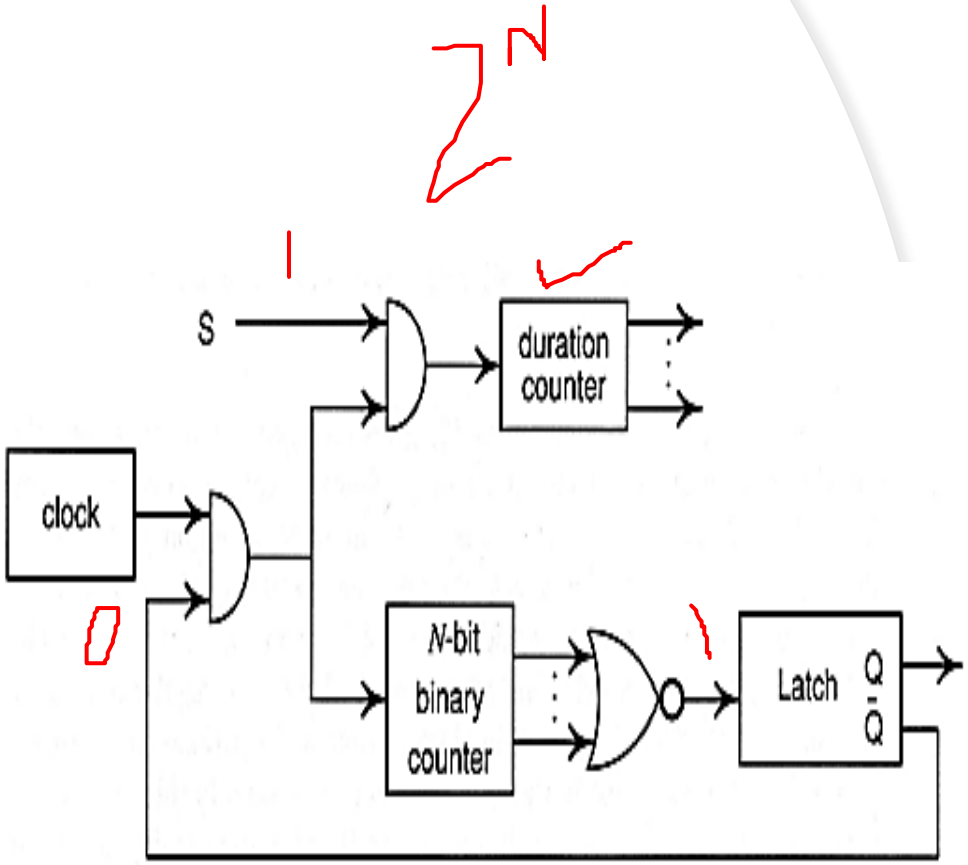

Fig 1: Setup for CPU-Channel Overlap Measurement

Solution:

- The required input signals are *CPU-busy bit* (*CpB)* and the *channel-busy bits* (*ChB*$_i$ for channel *i).*
- Assume these are available from the backplane.
- Let $f_{k/n}$ denotes the boolean function that generates the output 1 if exactly *k* out of its *n* boolean inputs at logic 1.
- The boolean function S computed as follows:
- $S = CpB \wedge f_{k/n}(ChB_1, \ldots, ChB_n)$ ---------- (1)
- Refer to Fig 1, for each clock pulse, if S is high and the measurement interval is not over, the duration counter will be incremented by 1.
- The N-bit binary counter along with the NOR gate acts as an interrupt generator.
- Obviously, the NOR gate output will make a 0-to-1 transition when the counter overflows.
- This, in turn, will set the latch and freeze the duration counter.

- The measurement can be controlled at the program level by providing a system call such as IO_OVERLAP(k, X)
- where X indicates the memory location in which the measurement result is returned.
- The system call could spawn a surrogate process as follows:
  - Determine the function code to load into the EDR using the parameter k.
  - Initialize the EDR, clear both counters, and reset the interrupt latch (should be done last).
  - The monitoring would then start automatically.
  - Block until the monitor posts an interrupt. Then read the value from the duration counter, convert it to a real number, put it in location X, and exit.
- Hardware monitoring may be inconvenient for measuring software-level characteristics.

SOFTWARE MONITORING

- Used to monitor operating system and higher level software such as networks and databases.
- At each activation, several instructions are executed therefore are suitable only if input rate is low.

Example:

- On a 1-MIPS machine, if a monitor executes 100 instructions per event then each activation would take 0.1 millisecond.
- To limit the overhead to 1%, it should be activated at intervals of 10 milliseconds or more.
- That is, the input rate should be less than 100 events per second.

SOFTWARE MONITORING DESIGN ISSUES

S. No.	Design Issue	Description
1.	*Activation Mechanism*	a) Trap: The trap instruction is a software interrupt mechanism that transfers control to a data collection routine. The monitor measures elapsed time for various operating system services using trap instructions at the beginning and at the end of the service code.
		b) Trace Mode: the instruction execution is interrupted after every instruction, and the control is passed on to a data collection routine. It is used only for those monitoring applications where time between events is not to be measured.
		c) Timer Interrupt: A timer-interrupt service provided by the operating system is used to transfer control to a data collection routine at fixed intervals. This mechanism is called sampling.
2.	*Buffer Size*	Optimal buffer size is a function of the input rate, input width, and emptying rate.
3.	*Number of Buffers*	A minimum of two buffers is required for continuous, simultaneous operation.
4.	*Buffer Overflow*	The monitoring process is required to either overwrite a previously written buffer or stop monitoring until a buffer becomes available.

SOFTWARE MONITORING DESIGN ISSUES (Cont'd)

S. No.	*Design Issue*	*Description*
5.	***Data Compression or Analysis***	Monitor processes the data as it is observed. This helps reduce the storage space required.
6.	***On/Off Switch***	Most hardware monitors have an on/off switch that enables/disables the monitoring operation. A software monitor should similarly have conditional (IF ... THEN ...) statements so that the monitoring can be enabled/disabled easily.
7.	***Language***	A software monitor is usually a part of the system being monitored, it is better to write both in the same programming language.
8.	***Priority***	If the monitor runs asynchronously, its priority should be low so that key system operations are least affected.
9.	***Abnormal-Events Monitoring***	A monitor should be able to observe normal as well as abnormal events on the system. Examples of abnormal events include system initialization, device failures, and program failures.

Task

A newly developed compiler for C was found to be much slower than expected. After some preliminary investigation, it was suspected that the compiler was spending too much time manipulating the symbol table.

Show how the fraction of time used for symbol table manipulations, denoted *frac,* can be measured accurately using *software monitoring.*

HYBRID MONITORING

- A monitor using a combination of software & hardware is a hybrid monitor.
- Involves transferring relevant information under software control from system under measurement to a set of interface registers where it can be picked up by a measuring system.

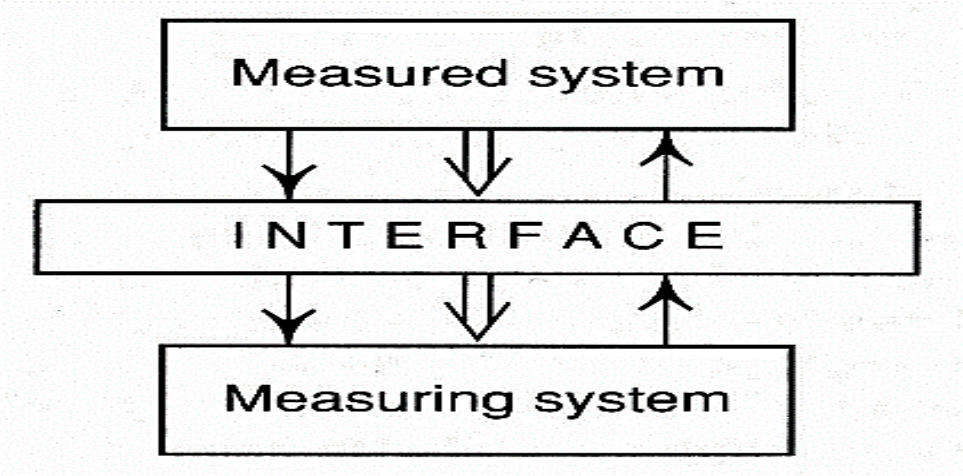

Fig 2: Setup for hybrid measurement

Task

Consider the measurement of the relative frequency of instruction usage.

a) Discuss pure hardware monitoring and software monitoring solutions for the given problem.
b) Discuss the drawbacks (if any) of both of these methods.
c) Suggest the solution using hybrid monitoring.
d) Highlight the advantages of hybrid monitoring for the given problem.

Solution (Part c)

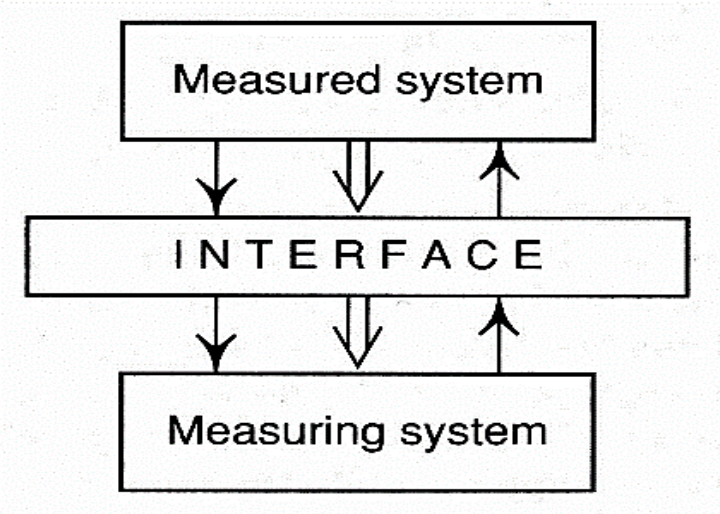

Fig 3: Setup for hybrid measurement

- Hybrid monitoring suggests to do half of the job in hardware and half in software.
- For that scheme, we will use two general purpose computers: one under measurement and one to measure.
- The first system outputs the opcode to an interface register.
- The second system picks up the op-code from this register and uses it to increment the **COUNT** array, which is now stored in the main memory of the measuring system.

Solution Part (d)

Advantages of Hybrid Monitoring:

1) One clear advantage over the hardware technique

- we have a full-fledged system to do the measurement;
- change in measurement objectives will require no new hardware.

2) A second advantage:

- The control software on measured system can be designed to place any relevant information on system bus, from where it can be picked up by the interface.
- In the context of hardware monitoring, this approach can be used to expose information in main memory, machine registers, and even devices.
- It is also possible to provide DMA capability in the interface.

3) Advantage over software technique: the measurement now has very little effect on functioning of the measured system; and is fast, hardware level signals can be captured.

Set-up of Hybrid Measurement

1) The control program(s), if any, to be run on the measured system.

2) What information the monitoring interface should capture and when?

- A number of general purpose registers and flags, and
- the setup phase will decide what information goes into what register/flag, when it is written and read, etc.

3) How the measuring system should retrieve information from the interface and what to do with it (display, store, etc)?

4) How the measured and measuring system would synchronize?

- Use some flags in the interface and
- a protocol be set up for setting and clearing these flags.

Set-up of Hybrid Measurement (Cont'd)

Measured System	Measuring system
loop Process next instruction; busy-wait until $F_2 = 1$; Reset F_2; R ← instruction opcode; Set F_1; forever;	loop Wait until $F_1 = 1$; Reset F_1; Read opcode from R; Increment COUNT[<opcode>]; Set F_2; forever;

Fig 4: Two-way synchronization in hybrid measurement