CS-417
COMPUTER SYSTEMS MODELING

Spring Semester 2020

Batch: 2016-17

(LECTURE # 5)

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

Recap of Lecture # 4

Benchmarking
Purpose of benchmarking
Types of benchmarks
Precautions & List of Challenges
Benchmark Strategies

Chapter # 2

MEASUREMENT TECHNIQUES

Measurement Techniques

- *How does one measure a system or component performance?*

- The analyst must first know
  - all of the events of interest in the system and
  - the relationship these events have with each other.

- These events form a *hierarchy of relationships*, where
  - the finer, granular events are used to construct the coarse-grained events in the system.

The state of system: defined by values contained in various storage elements, i.e. memory locations, registers or flip flops.

- Depending on measurement objectives, some of these may be used to define *relevant states* and others to provide further information about happenings in those states.
- We call the former *primary* variables and others *auxiliary*.
- e.g., the relevant states may be only the *busy/idle* condition of the paging disk, but we may be interested in knowing the *no. of free blocks* whenever a page I/O is initiated.

Event: refers to a change in relevant state variable.

- The events could also be classified as primary or auxiliary depending on what type of state variables are involved.

Three Primary Types of Measures

- ***Type A*** looks to <u>count the number of times</u> a given state is visited during a given time period.

- Some of the examples are:
  - the number of times a data structure is referenced,
  - relative frequency of executing a given instruction,
  - No. of times I/O done from cylinder 0 of some disk etc.
  - In the last example, the relevant state is defined by the *situation where I/O is in progress on cylinder 0* and the primary event is the *initiation of such an I/O operation*.

Three Primary Types of Measures

- ***Type B*** measures all state variables.
  - e.g. to extract all of the values for all internal registers and devices at the beginning of an instruction execution cycle.
  - Another example is the number of processes in the ready list whenever an I/O operation is initiated.

- ***Type C*** measures the fraction of time the system is within a state.
  - e.g. the fraction of time the system is executing load instructions versus all other kinds of instructions during the measured period of time.
  - Another example, what fraction of the time the disk head stays on cylinder 0.

Issues in Measurement

Two issues in measurement:

- How do we recognize conditions for measurement?
- How do we actually measure the quantity of interest?

Conditions for Measurement

The *condition for measurement* can be recognized in two ways:

1) As being in a relevant state:

- A natural way to do this is to sample the system and check if the primary state variables have the desired values.
- For example, to check if the control is inside a given procedure, we sample the program counter and see if contains an address that belongs to that procedure.
- This leads to *sample monitoring*.

***2) As an event that brings the system to a relevant state*:**

- To check if control is inside a given procedure, we explicitly look for events of entry to and exit from the procedure.
- This leads to *trace monitoring*.

Remarks

- It is easy to see that we can do measurements of all three types using trace monitoring.

- Type A measurements cannot be done using sampling.

- Similar comments apply to type B measurements.

- Type C measurements are possible by sampling since the computed relative frequency really gives an estimate of the fraction of time spent in the desired state.

- It may appear, from the discussion, that trace monitoring is always preferable to sampling, but this is not true.

Classification of Instrumentation

- Hardware Monitoring,
- Software Monitoring,
- Hybrid monitoring

"A monitor is a tool used to observe the activities on a system. In general, monitors observe the performance of systems, collect performance statistics, analyze data, and display results"

Monitors are used not only by performance analysts but also by programmers and systems managers.

Some of the reasons to monitor a system are as follows:

- To find frequently used segments of software and optimize their performance.
- To measure resource utilizations and to find performance bottlenecks.
- To tune the system. The system parameters can be adjusted to improve the performance.
- To characterize the workload. The results may be used for capacity planning and for creating test workloads.
- To find model parameters, to validate models, and to develop inputs for models.

Hardware Monitoring

- Employs additional monitoring hardware that is interfaced with the system under measurement in a nonintrusive way.
- For example:
  - a *logic analyzer* to measure the signals within the system or
  - insert a *specially designed hardware card* to extract some signals from a system.
- We can only measure what is exposed and available to be attached to for monitoring.
- Hardware monitors consist of:
  - a set of probes or sensors,
  - a logic-sensing device,
  - a set of counters, and
  - a display or recording unit.

- The *probes* monitor the state of the chosen system points.
    - Typically, probes can be programmed to trigger on a specific event, thereby providing the ability to trace specific occurrences within a system.

- The *logic-sensing subsystem* is used to interpret the raw input data being probed into meaningful information items.

- The *counters* are used to set sampling rates on other activities requiring timed intervals.

- The last component *records and displays* the information as it is sensed and reduced.

- The ability to perform *effective operational analysis* is directly dependent on the hardware and software monitors' ability to extract information.
- Another form of hardware monitoring uses *integral test hardware*, designed into the system being monitored during systems design.
- Many VLSI devices are designed so that:
  - all data items of interest can be tested in the device itself, or,
  - at a minimum, the test data points are brought outside of the chip
  - so additional devices can be used to gather this information and compute the health of the device.
- In all of these cases it is imperative that the hardware monitoring be designed as an integral component of the system, so it will not interfere with the operational system.

Requirements for Hardware Monitoring

- It is not desirable for the monitoring equipment to interfere with the system being monitored.

- The determination of *sampling sites* and *the frequency of measurements* must be designed ahead of time, not after the monitor has been put in place.

- The *monitoring method* has to be set up ahead of time also.

- We must determine and define all aspects of the monitor's existence in the measured system.

Detailed Mechanism - Hardware Monitoring

- The *conditions for measurement* are indicated by a *logic signal S*, synthesized using more primitive logic signals available from the machine backplane.
- S may be synthesized as both *single-bit* and *multi-bit* signals.
- In addition to normal boolean functions, the synthesizer should provide functions like *comparison of multi-bit signals*, AND of all bits, OR of all bits, etc.

Assumptions:

- a ***0 to 1*** transition in S takes to a relevant state.
- a ***1 to 0*** transition takes out of that state.
- e.g., suppose the condition of interest is the simultaneous operation of two devices, say D_1 and D_2.
- Busy condition for D_i indicated by boolean flag F_i in the backplane.

  Then: $\mathbf{S = F_1 \wedge F_2}$
- If there is more than one such state, a multi-bit signal, **S'**, synthesized using auxiliary state variables.

<u>Relative Frequency of executing each instruction:</u>

- a 0 to 1 transition in S may indicate a new instruction has been fetched into the instruction register, and
- S' is the set of bits that represent the opcode.

<u>Type A</u>

- increment a counter whenever S makes a 0 to 1 transition.
- The counter is chosen from an array indexed by S'.
- The signal S' may be valid only when S = 1 as in the instruction frequency example above.

<u>Type B</u>

- leading edge of S used to transfer auxiliary state information from the backplane to a memory module part of the monitor.

Type C

- In discussing type *C* measurement, assume that S is already synthesized appropriately.
- The measurement can be done using either the trace or the sampling technique.
- In the trace technique, we can accumulate the duration for which S = 1 by keeping track of the times when S makes 0 to 1 and 1 to 0 transitions but that's nearly impossible considering rapid rate at which hardware signals change.
- Because of this, *type C measurements in hardware are almost invariably done* by *sampling*.
- The idea is to *use S to gate clock pulses*, which are then counted by a hardware counter.

Sampled monitoring of type *C*

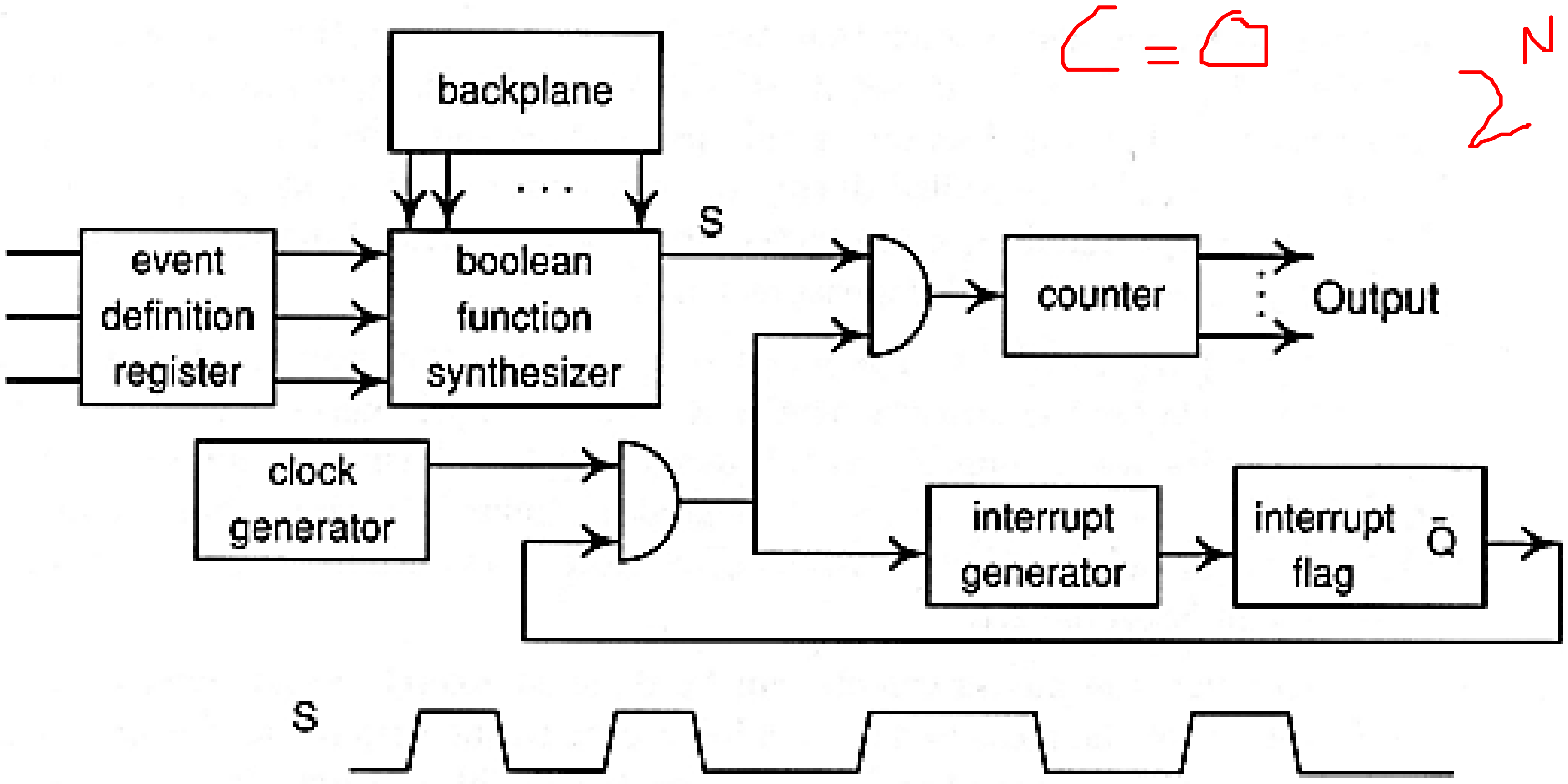

Fig 1: Instrumentation for sampled hardware monitoring