Snoop-based Multiprocessor Design

Topic-3
Chapter-6

Designing SMPs

- Large difference in performance, cost and scale of SMPs
  - Not due to choice of cache-coherence protocol
  - But due to design and implementation of the organisational structure
  - Latency and bandwidth achieved by protocol, depends upon Bus design, Cache design and Integration with Memory
- Three goals of implementation
  - (1) Correctness
  - (2) High Performance
  - (3) Minimal extra hardware

Implementation goals

- Correctness: arises because actions that are considered atomic at abstract-level are not necessarily atomic at hardware level
- High performance: because we want to pipeline memory operations and allow many operations to be outstanding at a time rather than waiting for each operation to complete before we can start the next one
- However, due to numerous complex interactions between events, correctness is a concern

What will we see in this topic?

(1) Correctness requirements

(2) Single-level cache + Single transaction Atomic Bus

(3) Multi-level cache + Single transaction Atomic Bus

(4) Single-level cache + Split transaction Bus

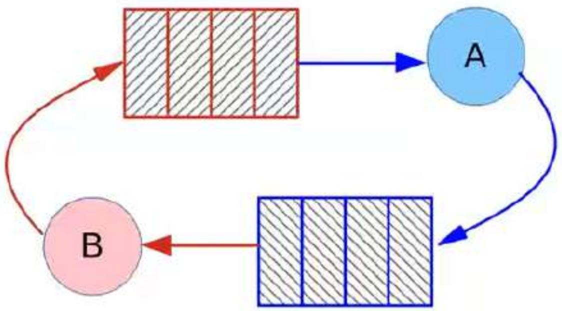
(5) Multi-level cache + Split transaction Bus

Correctness Requirements

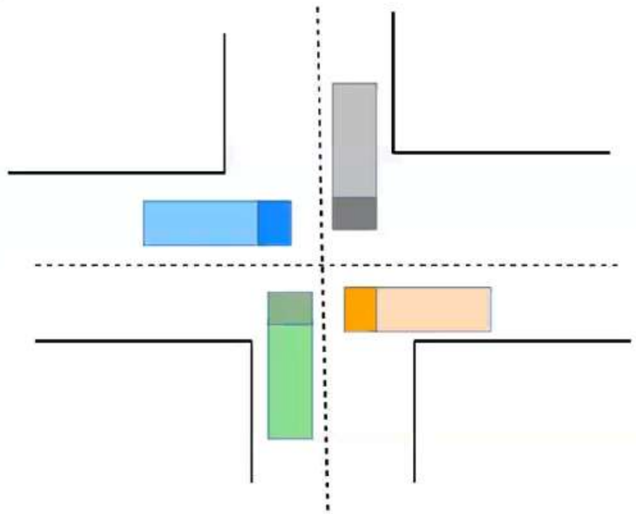
- A cache coherent system must fulfill conditions for coherence and consistency
  - For coherence it should ensure
    - Finding stale copies, update, invalidate on writes
    - Write serialisation
    - Write propagation
  - For consistency
    - Write atomicity
    - Detecting completion of write
- Must be Free from
  - Deadlock
  - Livelock
  - Starvation

Deadlock

- Deadlock
  - All system activity ceases
  - Cycle of resource dependencies



Controllers A and B
With buffers (full)



Traffic light
At intersection

Livelock

- No processor makes forward progress although transactions are performed at hardware level
- Traffic analogy: all cars back-off and try again at the same time, causing deadlock
- e.g. simultaneous writes in invalidation protocol
  - => each requests ownership, invalidating others, but looses ownership before it has finished using the resource and has to release the resource

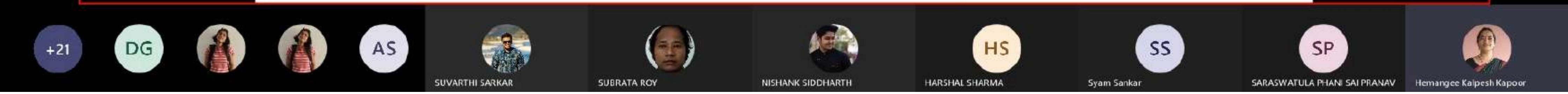
Starvation

- One or more processors make no forward progress, while others do
- Traffic analogy: Busy highway vs Country road
- e.g. Interleaved memory system with NACK on bank being busy
- Often not completely eliminated (not likely, not catastrophic)
  - Because to eliminate it adds a lot of complexity to protocol design

Starvation

- One or more processors make no forward progress, while others do
- Traffic analogy: Busy highway vs Country road
- e.g. Interleaved memory system with NACK on bank being busy
- Often not completely eliminated (not likely, not catastrophic)
  - Because to eliminate it adds a lot of complexity to protocol design

Single-level cache
+
Atomic Bus



Single-level Cache + Atomic Bus (single transaction)

- Here we have some assumptions but they are physically realistic
- List of preliminary design issues
  1) Design of cache controller and tags. Both processor and bus need to lookup
  2) How and when to present snoop results on the bus
  3) Dealing with write-backs, as they may cause race conditions
  4) Overall set of operations for memory access are not atomic. This can introduce race conditions
  5) How to support atomic read-modify-write operations
  6) Any issues that may arise wrt deadlock, livelock, starvation, serialisation, etc.

(i) Cache controller + Tag design

- Coherence protocol defines logical FSM for each cache block
- Implemented as processor-side and Bus-side FSM
- Processor has steps (tag compare, etc.) for cache access
- Snoop also compares tags to take appropriate actions
- Cache controller sends Bus transactions in both cases
- Bus transaction has a sequence of steps:
  - i. Assert request for bus
  - ii. Wait for bus grant
  - iii. Drive address and command lines
  - iv. Wait for command to be accepted by relevant device
  - v. Transfer data

Cache controller + Tag design

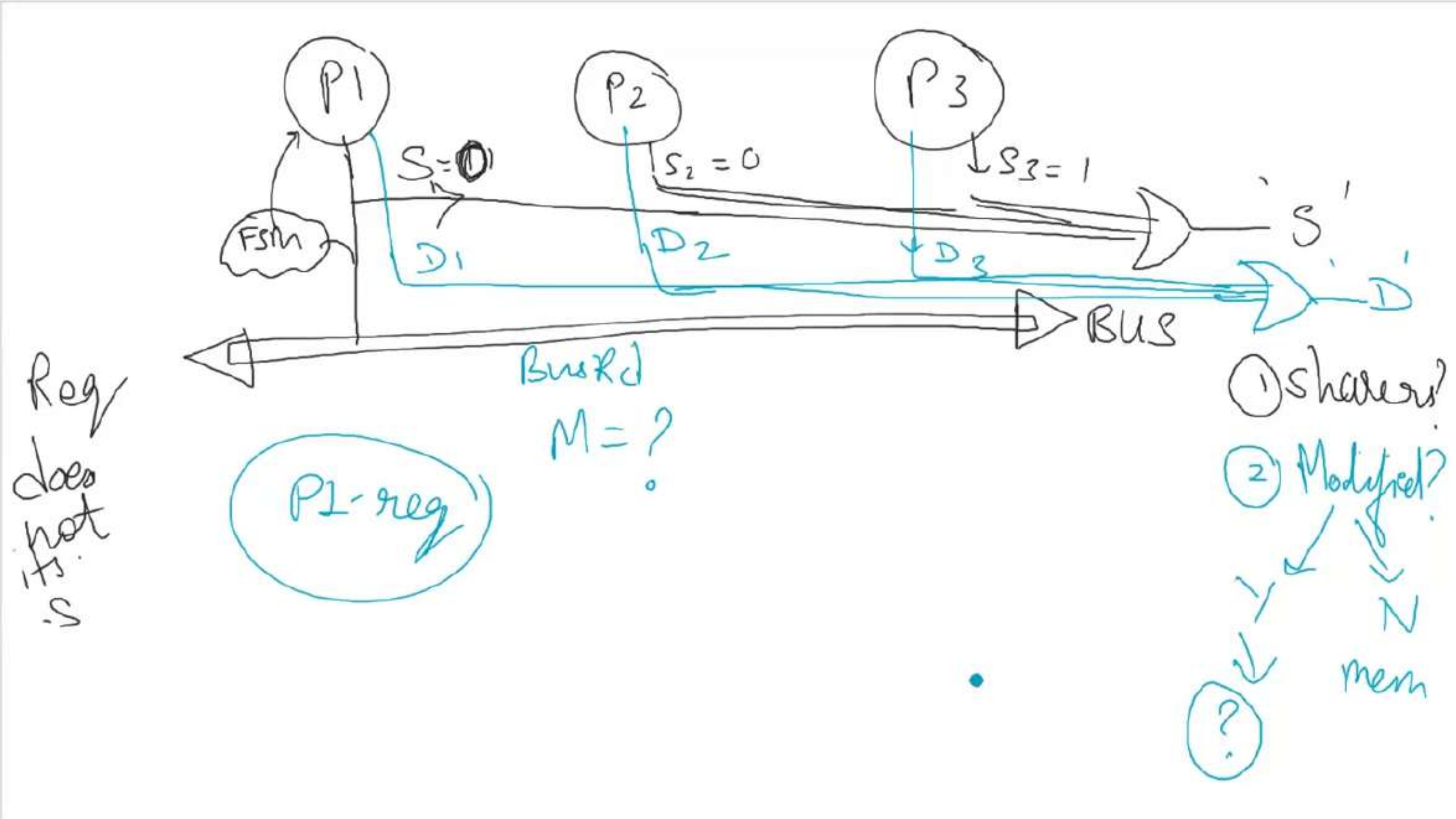
- To implement snooping protocol, cache controller has bus-side and processor-side FSM
- Tag comparison needed by both
- Therefore dual-tags (don't duplicate data) or use dual-ported RAM for tags

- Minimal stall on tag updates
- General operation of tag compare is fast

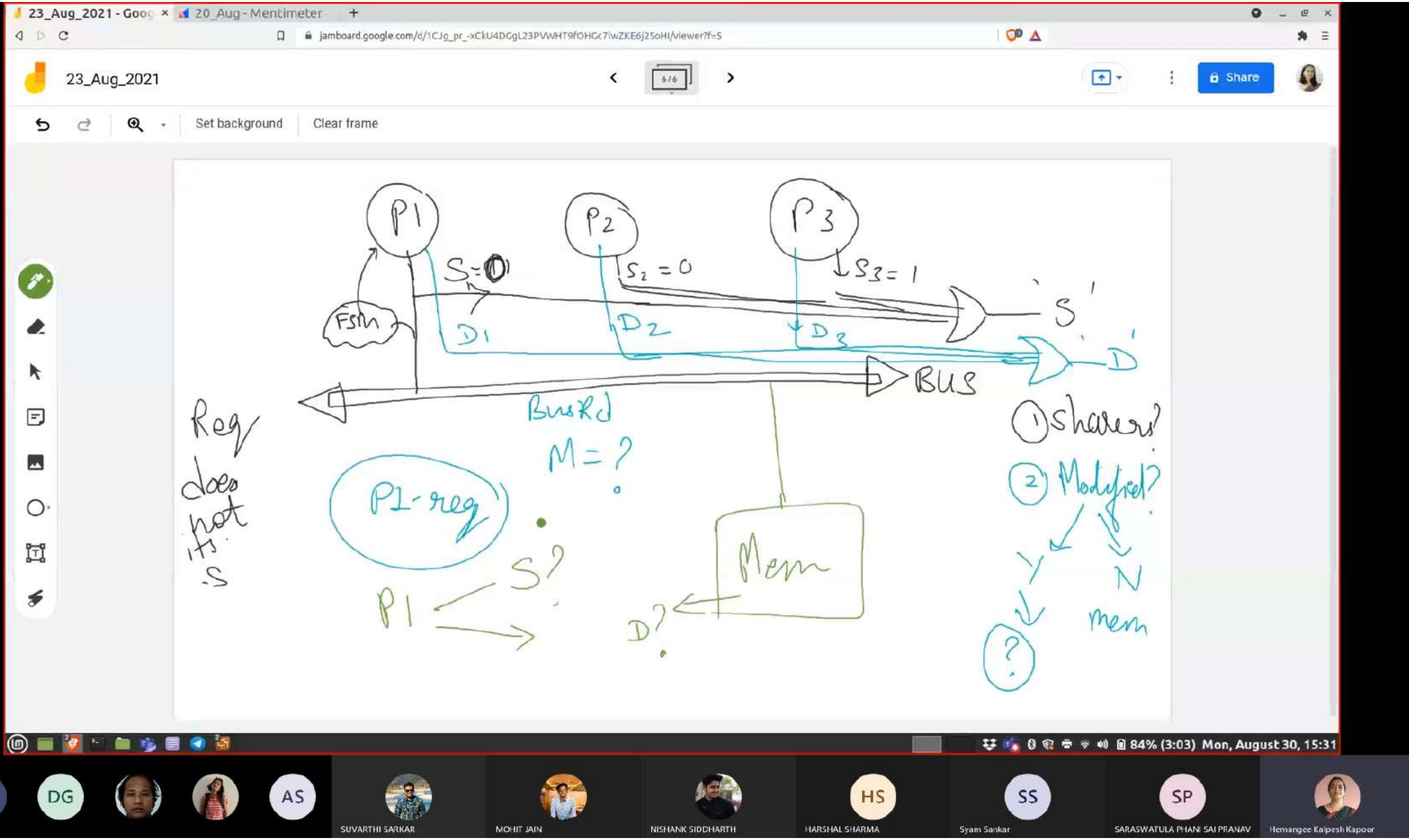
(ii) Reporting Snoop Results: When?

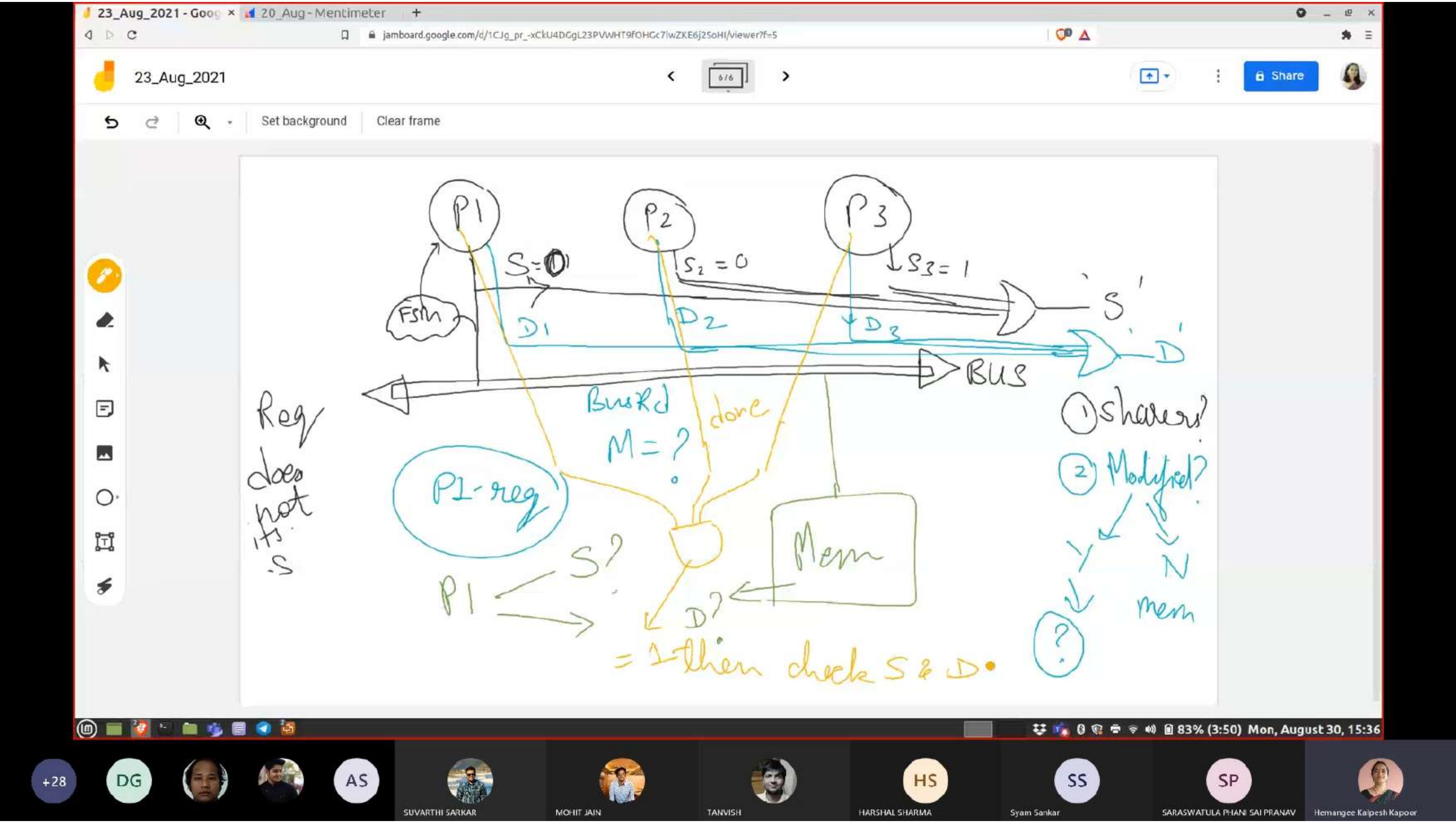
- All caches snoop on bus and do tag compare. Collective result of (tag compare) snoop must be sent on bus before transaction can proceed
- This is required as Memory must know if it should give data. i.e. Memory or cache will give
- Keep decision delay to minimum
- Three options:

23_Aug_2021

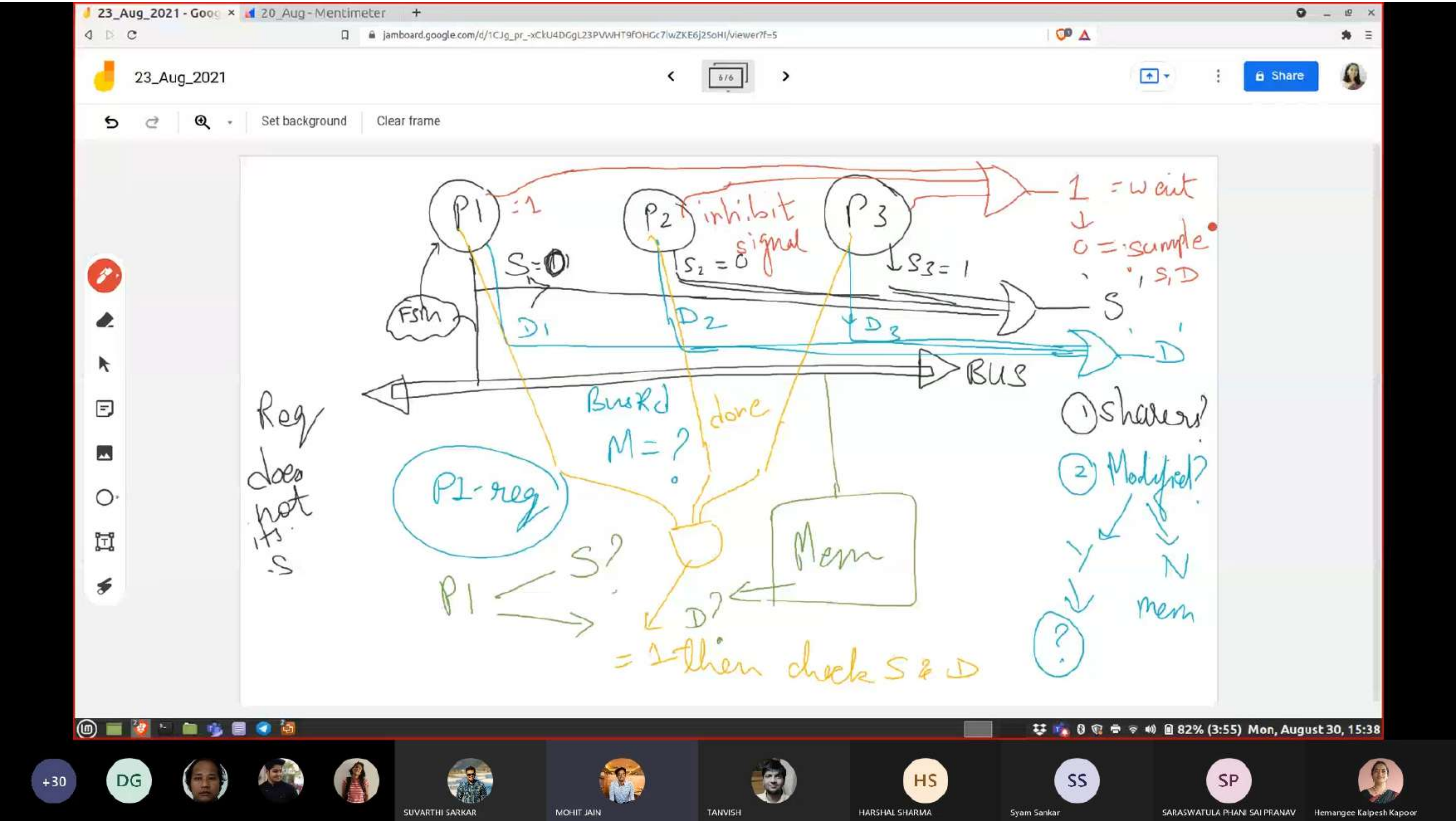


P1
P2
P3
$S_1 = 0$
$S_2 = 0$
$S_3 = 1$
FSM
D_1
D_2
D_3
S'
D'
BUS
Req
Req does not its S
BusRd
M = ?
P1-req
① sharers?
② Modified?
Y
N
?
mem

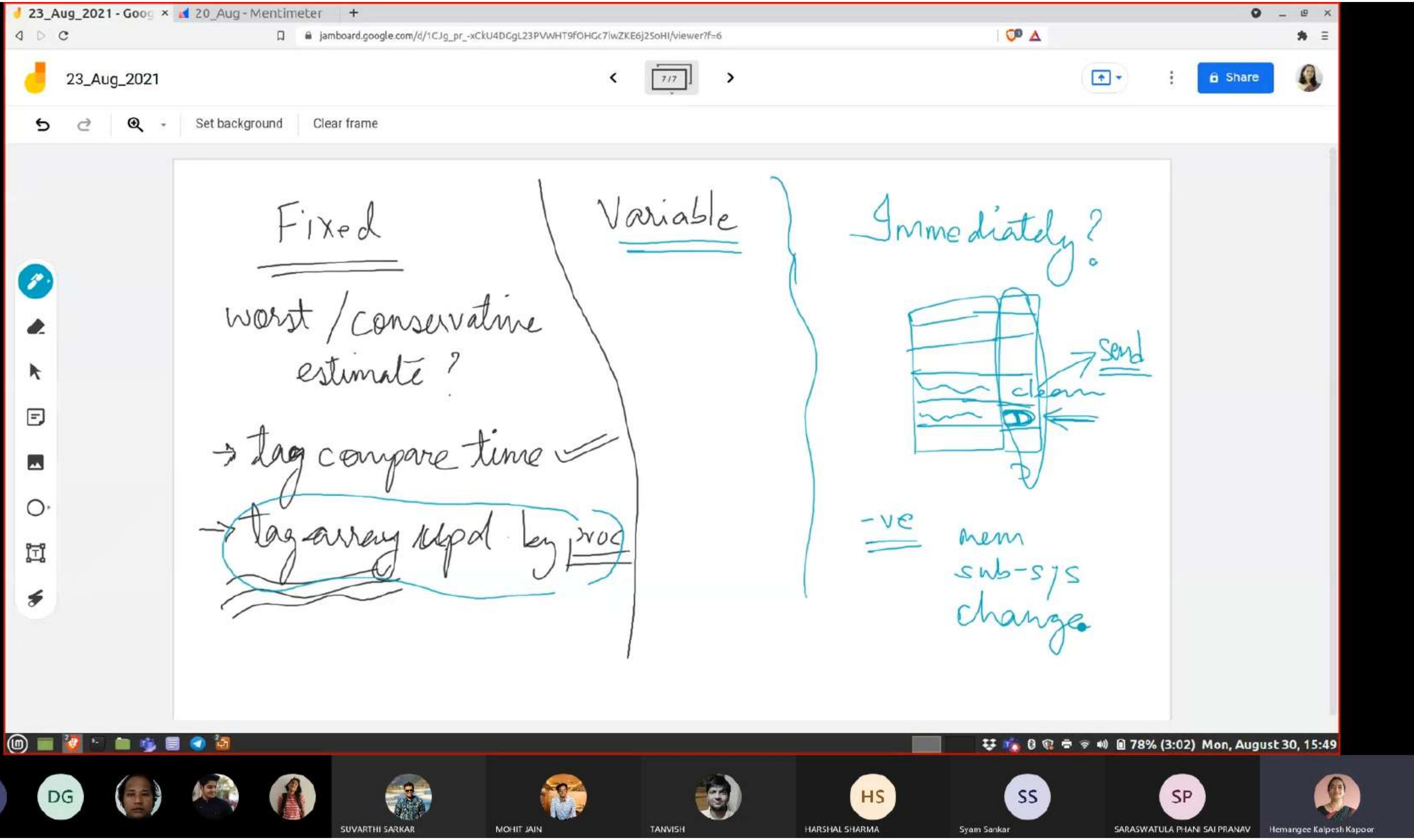

23_Aug_2021 - Goog
20_Aug - Mentimeter
jamboard.google.com/d/1CJg_pr_-xCkU4DGgL23PVWHT9fOHGc7lwZKE6j2SoHI/viewer?f=5
23_Aug_2021
6/6
Share
Set background
Clear frame
P1
P2
P3
S1 = 0
S2 = 0
S3 = 1
FSM
D1
D2
D3
S'
D'
BUS
BusRd
M = ?
Reg does not its S
P1-reg
S?
P1
D?
Mem
1 shares?
2 Modified?
Y
N
?
mem
84% (3:03) Mon, August 30, 15:31
+29
DG
AS
SUVARTHI SARKAR
MOHIT JAIN
NISHANK SIDDHARTH
HS
HARSHAL SHARMA
SS
Syam Sankar
SP
SARASWATULA PHANI SAI PRANAV
Hemangee Kalpesh Kapoor



23_Aug_2021 - Goog
20_Aug - Mentimeter
jamboard.google.com/d/1CJg_pr_-xCkU4DGgL23PVWHT9fOHGc7lwZKE6j2SoHI/viewer?f=5
23_Aug_2021
6/6
Share
Set background
Clear frame
P1
P2
P3
S1 = 0
S2 = 0
S3 = 1
FSM
D1
D2
D3
S'
D'
BUS
Req
Req does not its S
BusRd
M = ?
done
P1-req
Mem
S?
D?
P1
= 1 then check S & D.
1) shared?
2) Modified?
Y
N
mem
?
83% (3:50) Mon, August 30, 15:36
+28
DG
AS
SUVARTHI SARKAR
MOHIT JAIN
TANVISH
HS
HARSHAL SHARMA
SS
Syam Sankar
SP
SARASWATULA PHANI SAI PRANAV
Hemangee Kalpesh Kapoor



23_Aug_2021 - Goog
20_Aug - Mentimeter
jamboard.google.com/d/1CJg_pr_-xCkU4DGgL23PVWHT9fOHGc7lwZKE6j25oHI/viewer?f=5
23_Aug_2021
6/6
Share
Set background
Clear frame
P1
=1
P2
inhibit signal
P3
1 = wait
0 = sample
, S, D
S1 = 0
S2 = 0
S3 = 1
FSM
D1
D2
D3
S
D
BUS
Req
Req does not its S
BusRd
M = ?
done
1 shares?
2 Modified?
Y
N
mem
?
P1-reg
Mem
S?
P1
D?
= 1 then check S & D
82% (3:55) Mon, August 30, 15:38
+30
DG
SUVARTHI SARKAR
MOHIT JAIN
TANVISH
HS
HARSHAL SHARMA
SS
Syam Sankar
SP
SARASWATULA PHANI SAI PRANAV
Hemangee Kalpesh Kapoor


23_Aug_2021 - Goog
20_Aug - Mentimeter
jamboard.google.com/d/1CJg_pr_-xCkU4DGgL23PVWHT9fOHGc7lwZKE6j2SoHI/viewer?f=6
23_Aug_2021
7/7
Share
Set background
Clear frame
Fixed
Variable
Immediately ?
worst / conservative estimate ?
→ tag compare time
→ tag array upd by proc
Send
clean
-ve
mem
sub-sys
change
78% (3:02) Mon, August 30, 15:49
+30
DG
SUVARTHI SARKAR
MOHIT JAIN
TANVISH
HS
HARSHAL SHARMA
SS
Syam Sankar
SP
SARASWATULA PHANI SAI PRANAV
Hemangee Kalpesh Kapoor

(ii) Reporting Snoop Results: When?

- All caches snoop on bus and do tag compare. Collective result of (tag compare) snoop must be sent on bus before transaction can proceed
- This is required as Memory must know if it should give data. i.e. Memory or cache will give
- Keep decision delay to minimum
- Three options:
  - Fixed Delay
  - Variable Delay
  - Immediately

Rep. snoop results.. Fixed Delay

- After fixed number of clock cycles after the address appears on bus
- This needs dual-tags to reduce contention with processor
- Still must be conservative, as CPU may lock both tags arrays when updating state (e.g. ‘E’ -> ‘M’)
- Advantage is main memory design not affected
  - Cache-to-cache handshake is simple
- Dis-advantage is extra hardware, Potentially longer latency
- Ex: Pentium Pro, HP Server, Sun Enterprise

+30 DG SUVARTHI SARKAR MOHIT JAIN TANVISH HS HARSHAL SHARMA SS Syam Sankar SP SARASWATULA PHANI SAI PRANAV Hemangee Kalpesh Kapoor

Rep. snoop results.. Fixed Delay

- After fixed number of clock cycles after the address appears on bus
- This needs dual-tags to reduce contention with processor
- Still must be conservative, as CPU may lock both tags arrays when updating state (e.g. ‘E’ -> ‘M’)
- Advantage is main memory design not affected
  - Cache-to-cache handshake is simple
- Dis-advantage is extra hardware, Potentially longer latency
- Ex: Pentium Pro, HP Server, Sun Enterprise

Rep. snoop results.. Variable Delay

- Memory assumes cache will supply data, until all caches say otherwise
- Less conservative, as we need not assume worst case delay to compute snoop results
- More flexible, more complex (handshakes required between cache <--> memory)
- Memory can however start fetching data. If cache gives data then memory data is not used and memory access stopped
- Ex: SGI Challenge

+30 DG SUVARTHI SARKAR MOHIT JAIN TANVISH HS HARSHAL SHARMA SS Syam Sankar SP SARASWATULA PHANI SAI PRANAV Hemangee Kalpesh Kapoor

Rep. snoop results.. Immediately

- Memory maintains 1-bit per block to indicate if it is being modified or not by any cache
- This bit helps memory to decide if memory should send data. Need not wait for snoop results
- Dis-advantage is extra hardware complexity to memory sub-system

(iii) Reporting Snoop Results: How?

- Collective response from caches must appear on the bus
- Ex: in MESI protocol, we need to know
  - Is block dirty? : should memory respond or not
  - Is block shared? : is block in other caches, to decide if one should load in 'E' or 'S' state

(iii) Reporting Snoop Results: How?

- Collective response from caches must appear on the bus
- Ex: in MESI protocol, we need to know
  - Is block dirty? : should memory respond or not
  - Is block shared? : is block in other caches, to decide if one should load in 'E' or 'S' state
- Use three wired-OR signals
  1) Shared: asserted if any cache (except Requestor) has a copy
  2) Dirty: asserted if any cache has modified copy. Need not know which, since it will know what to do
  3) Snoop-Valid: this is an inhibit signal. Asserted until all caches have completed their snoop. When de-asserted, Memory and Requestor can examine other two signals

Who provides data?

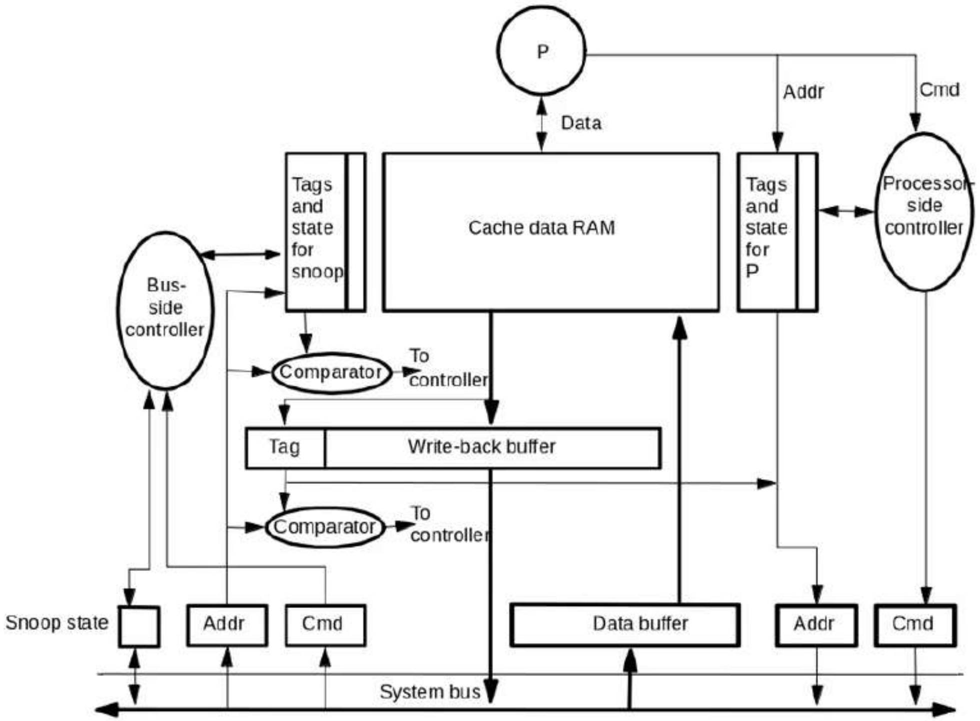
- Full Illinois-MESI is more complex as
  - Block is provided by cache and not Memory
  - Then priority scheme as to which cache gives block if multiple have it
- Therefore commercial systems avoid cache-to-cache transfer
- Ex: SGI Challenge and Sun Enterprise use cache transfers only for modified data
  - SGI Challenge --> updates Memory when cache gives copy
  - Sun Enterprise --> uses 'O' bit. MOESI and Memory does not update when cache gives

+30 DG SUVARTHI SARKAR MOHIT JAIN TANVISH HS HARSHAL SHARMA SS Syam Sankar SP SARASWATULA PHANI SAI PRANAV Hemangee Kalpesh Kapoor

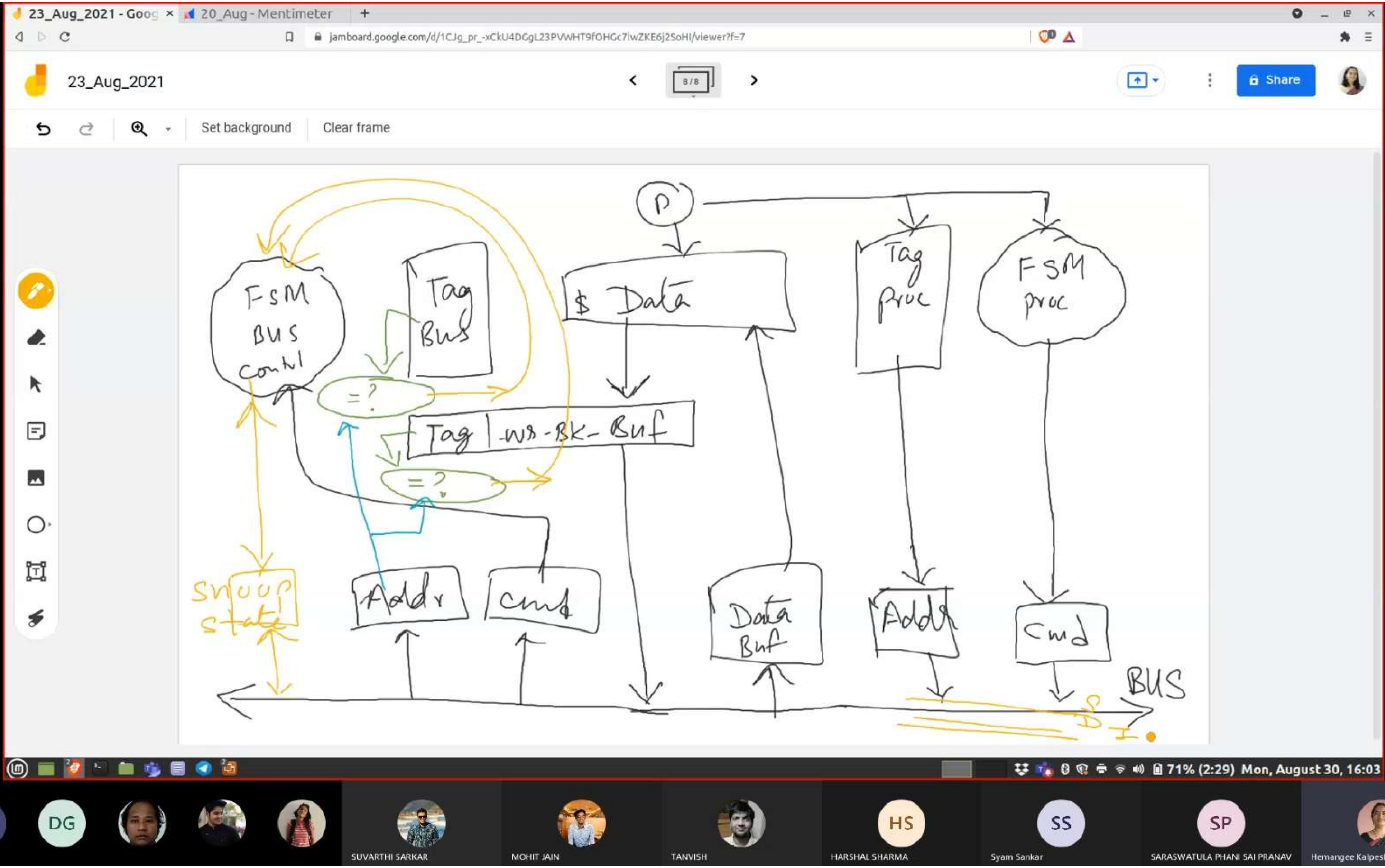
(iii) Dealing with write-backs

- At block replacement => 2 Bus transactions
  - Bring new block + write-back old block
- Want to reduce processor wait time on a cache miss
- Therefore service the miss first and then do write-back asynchronously
- Need additional storage : write-back buffer
- Bus transactions for this block can come. So snooper has to check even the write-back buffer
- Cache controller may have to cancel the write-back, if the block is used from the write-back buffer

Base Organisation



+30 DG SUVARTHI SARKAR MOHIT JAIN TANVISH HS HARSHAL SHARMA SS Syam Sankar SP SARASWATULA PHANI SAI PRANAV Hemangee Kalpesh Kapoor


23_Aug_2021 - Goog
20_Aug - Mentimeter
23_Aug_2021
8/8
Share
Set background
Clear frame
P
$ Data
Tag Proc
FSM Proc
FSM Bus Contl
Tag Bus
=?
Tag
wb-Bk-Buf
=?
Snoop state
Addr
Cmd
Data Buf
Addr
Cmd
BUS
71% (2:29) Mon, August 30, 16:03
+30
DG
SUVARTHI SARKAR
MOHIT JAIN
TANVISH
HS
HARSHAL SHARMA
SS
Syam Sankar
SP
SARASWATULA PHANI SAI PRANAV
Hemangee Kalpesh Kapoor

Base organisation

- Single wr-back cache
- Dual tags: P-side, B-side
- Processor controller places transaction on bus: address + command
- WB-transaction goes via wr-buffer
- Read transaction --> data in data buffer
- B-side controller snoops WB-buffer-tag + cache tages
- Bus arbitration places requests that go on the bus in total order
- Wired-OR snoop results serve as ACK to initiator

  - Using this design, we will see subtle correctness concerns wrt non-atomic state transitions, serialisation (coherence+consistency), deadlock, livelock, starvation

+30 DG HS SS SP

SUVARTHI SARKAR MOHIT JAIN TANVISH HARSHAL SHARMA Syam Sankar SARASWATULA PHANI SAI PRANAV Hemangee Kalpesh Kapoor

(iv) non-atomic state transitions

- In the protocol state diagram, all transitions are assumed to take place instantaneously/atomically
- But each change involves several intermediate actions, even if the bus is atomic
- Actions like:
  - Cache look-up, bus arbitration, actions taken by other controllers at their caches, action of issuing processor's controller, final block write
- Can have race conditions among components of different operations
- Ex: Request for block 'B' may come to some processor, while the processor is still changing state for block 'B'