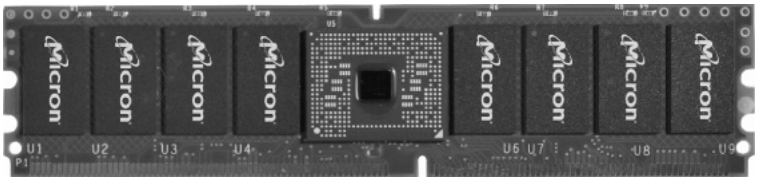
# FB Dimm's




Reference: "Memory Systems: Cache, DRAM, Disk

Bruce Jacob, Spencer Ng, & David Wang

Today's material & any uncredited diagram came from Chapter 14



# The Problem

- Multi-drop busses don't scale well
  - demand for higher memory bandwidth continues
  - traditional memory architecture scaling
    - » can happen in speed or capacity but not both
    - » market expectation
      - more for the same price
      - OR more for less
- Intel's idea but solution is a common one
  - replace with point to point signaling
    - » mitigates signal integrity problem
    - » introduces the multi-hop problem
- Short story
  - already has proven to be problematic
  - Intel now moving to BoB (Buffer on Board)
    - » essentially a multi-spigot FB Dimm idea

# ITRS Comparison Humor

**Remember in 2001: frequency wars were still ongoing, and power worries were just starting to peek over the horizon**

| 2001 ITRS | | | | | |
|---|---|---|---|---|---|
| | 2004 | 2007 | 2010 | 2013 | 2016 |
| Process (nm) | 90 | 65 | 45 | 32 | 22 |
| CPU GHz | 3.99 | 6.74 | 12 | 19 | 29 |
| Mlogic T/CM² | 77.2 | 154.3 | 309 | 617 | 1235 |
| Hperf pin ct. | 2263 | 3012 | 4009 | 5335 | 7100 |
| HP cents/pin | 1.88 | 1.61 | 1.68 | 1.44 | 1.22 |
| Mem cents/pin | 0.34-1.39 | 0.27-0.84 | 0.22-0.34 | 0.19-0.39 | 0.19-0.33 |
| Mem pin ct. | 48-160 | 48-160 | 62-208 | 81-270 | 105-351 |
| CPU cost/pkg | 42.5444 | 48.4932 | 67.3512 | 76.824 | 86.62 |
| Max Mem cost/ pkg | 2.224 | 1.344 | 0.7072 | 1.053 | 1.1583 |
| Min Mem cost/ pkg | 0.1632 | 0.1296 | 0.1364 | 0.1539 | 0.1995 |

| 2008 ITRS | | | | | |
|---|---|---|---|---|---|
| | | 2007 | 2010 | 2013 | 2016 |
| Process (nm) | | 68 | 45 | 32 | 22.5 |
| CPU GHz | 12 inv delays | 4.7 | 5.8 | 7.3 | 9.1 |
| Mlogic T/CM² | | 154 | 309 | 617 | 1235 |
| Hperf pin ct. | 33% P & G | 3072 | 3072 | 3072 | 3072 |
| HP cents/pin | | .69-1.13 | .60-1.20 | .51-.87 | .44-.75 |
| Mem cents/pin | | .27-.5 | .23-.44 | .20-.38 | .20-.32 |
| Mem pin ct. | Unspecified in 2008 ITRS Update | | | | |
| CPU cost/pkg | | | | | |
| Max Mem cost/ pkg | | | | | |
| Min Mem cost/ pkg | | | | | |

**Note in 2008:**
- lack of specification on memory packaging and cost
- no frequency prediction other than 12-inverter delay

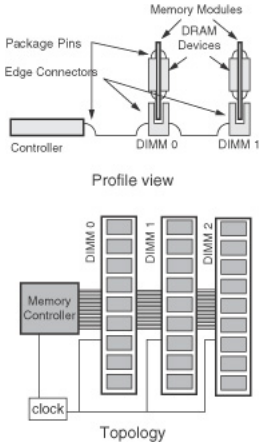


# FB Dimm Idea

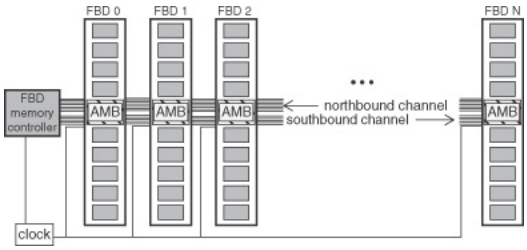
- **Need to create higher bandwidth**
  - **DDR2**
    - » **400 MT/s configured**
      - **up to 4 two-rank DIMM's**
  - **DDR3**
    - » **800 MT/s configured**
      - **1 2-rank Dimm**
- **Move multi-drop bus to the DIMM**
  - **daisy chain the DIMMs using an AMB ASIC**
    - » **AMB ::= Advanced Memory Buffer**
      - **actually much more than a buffer**
      - **also does bit-lane retiming**
      - **packetized frame-relay protocol**
    - » **AMD duties**
      - **extract DRAM commands from frame**
      - **control DRAM devices (2ndary mem.ctlr)**

# Replacement Strategy

THIS




WITH THIS






# FB Dimm Problems

- **Device compatibility (the usual boat anchor)**
  - **still need**
    - » **JEDEC standard – adopted after Intel push**
    - » **use commodity DDR2&3 components**
    - » **retain user configuration flexibility**
- **However**
  - **significant increase in idle system latency**
  - **increased power consumption**
    - » **big problem**
  - **AMB cost adder**
    - » **incompatible with DRAM market economics**
- **Result**
  - **lots of resistance from system manufacturers**
    - » **remember Intel makes the parts not the system**

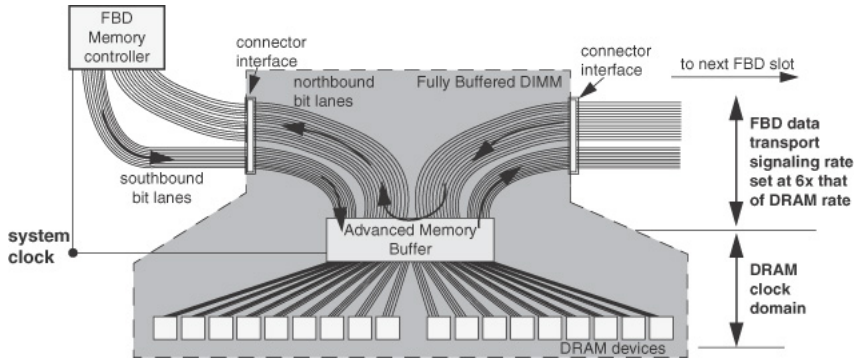
# The AMB ASIC

- **One hop in a daisy chain**
  - **role (incoming side = southbound channel)**
    - » **examine frame contents**
    - » **is it for me?**
      - **If so broadcast to the DRAM's on the appropriate rank**
        - **marshal write data to the DRAM's**
        - **read data converted into frames to place on northbound channel**
      - **If not – recondition signals and pass it on**
        - **classic store and forward network with a wrinkle**
        - **wrinkle is essentially cut-through routing**
          - **forward before digest and check**
          - **reduces latency but increases affected scope of errors**
  - **role (outbound = northbound side)**
    - » **encapsulate data burst into frames**
      - **this involves a significant amount of bit-lane retiming**
- **Benefit**
  - **6x FBD signaling rate over DRAM devices for DDR2**



# 2 Rank FB-Dimm Diagram

FBD clock rate = 4 GT/s




DDR2 clock rate = 667 MT/s

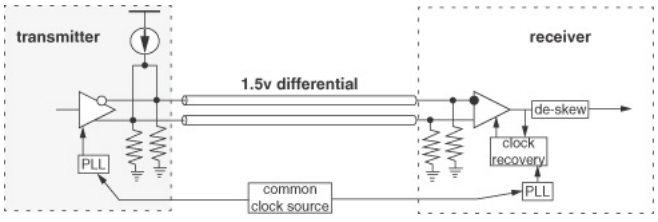
# FB Dimm Mesochronous Timing

- **Multiple clock domains**
  - **synchronous w.r.t. each other**
  - **BUT – phase relationships are not strictly defined**
    - » **hence the need for bit-lane retiming**
      - **skew and jitter exceed bit cycle times**
      - **removes need for trace length equality**
        - **"stub electronics" problem**
        - **a major stumbling block in non-FB DDR2 memory systems**
        - **simplifies interconnect design**
        - **at the expense of active "correct it" silicon**
    - » **north- and south-bound lanes designed to be timing independent**
- **As always**
  - **the devil is in the details**
    - » **so let's look at some of them**



# Signaling & Timing

- **Not all that different – borrowed technology**
  - **Northbridge**
    - » **likely contains both the PCIe and Mem_Ctlr**
    - » **so use PCIe style signaling**
      - **well understood technology**
      - **1.5v differential signaling**
        - **optimized for FR4 PCB's**



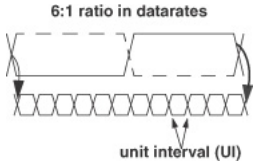
# Clock Data Recovery

- **Common problem**
  - clock doesn't have known phase relationship with data
    - » one known technique
      - recover clock from the data signal
      - but this requires a known number of signal transitions
      - real data doesn't look this way so encoding is required
        - 8b/10b Fibre Channel or HyperTransport scrambling models for example
          - provided DC balance – electrically important
          - simplifies clock recovery by insuring that enough transitions occur per some unit of time
    - » result
      - use transitions to recover clock
      - use recovered clock to determine data
        - implied: clock skew+jitter doesn't change wildly in short time frame
- **Actual FB-Dimm standard uses a simpler approach**
  - no inter-lane phase relationships specified
    - » does specify transition density
      - 6 transition minimum in a 512 bit frame



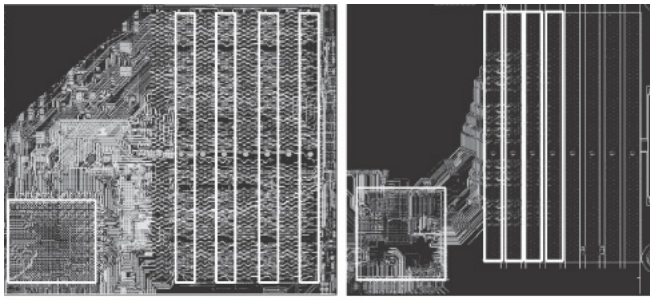
# Unit Interval

- **DDR & 6:1**
  - 12 UI's/DRAM clock cycle form the FB-Dimm "frame"




- **Bit lane independence**
  - cause: latency and path length variations
  - result: several UI difference in lane burst arrival at an AMB
- **FB-Dimm and AMB requirements**
  - logic to deskew the data across the independent bit lanes
  - danger: increased latency = de-skew-time*hop-count

## Benefit: Less Routing Restriction



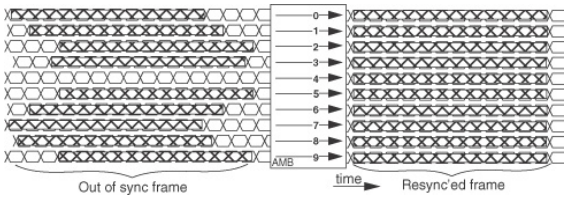
DDR2 SDRAM channel routing | FB-DIMM channel routing

Source: Intel



## Resync Latency Cost

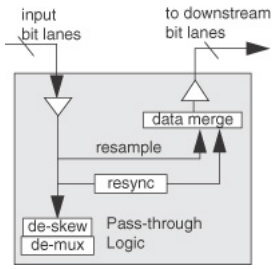
- **Forwarding delay dominated by slowest lane**




- **Too slow if resync is done on every hop**
  - **hence 2 southbound frame relay modes**
    - » **resample**
      - **clock recovery removes bit jitter in a lane**
      - **does not correct lane UI skew**
        - **spec allows a maximum of 46 UI difference between lanes**
    - » **resync**
      - **delay retransmit until all lanes are collected**
      - **then drive resynchronized frame**

# 3 AMB Datapaths

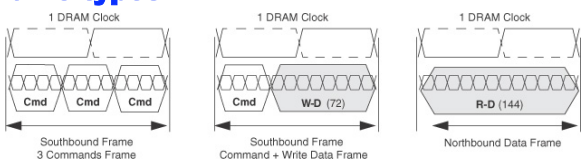
- Resample and Resync
- Plus need to extract southbound command
  - in case target is this DIMM
  - note forward anyway style
    - » decode and forward if it's not for me option is intractable
      - since decode time would have to be added to each southbound hop






# Protocol

- Asymmetric channels
  - southbound
    - » 10 bit lanes * 12 UI's = 120 bits/frame
      - half peak write bandwidth
      - 4 UI's for command – hence 80 write data bits/frame
  - northbound
    - » 14 bit lanes * 12 UI's = 168 bits/frame
      - full peak read bandwidth of a target rank
  - both contain CRC info for data recovery at receiver
    - » and actual data/frame is less: 72 (64+8) & 144 (128+16)
      - to support fail over mechanism (more on this soon)
- 3 common frame types



## Frame Formats

- Southbound command only
  - » 3 commands/frame
    - sent to independent DIMMs or ranks
      - – improves parallelism
      - – can also allow certain modules to be moved to a lower power state
    - nops or platform specific debug patterns pad frame when 3 commands aren't needed
- Southbound command and write data
  - » command, 64 data, and 8 check bits
    - 8 bits can be used as a byte mask if DIMM doesn't support ECC
  - » weirdness
    - multiple frames are needed for a full write burst
      - – they do not need to be contiguous (indicates read priority model)
    - each write-data subframe only contains 1 bit of the target AMB address
      - – 3 subframes needed to form full address (8 DIMM max spec)
      - – implies ALL AMB's must buffer write data before destination is known
      - – energy cost of writes exacerbated
- Northbound read – 1 DIMM cycle read return 128 + 16



## Commands

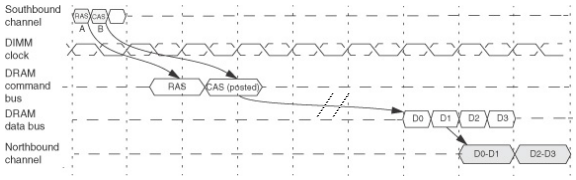
- 2 types
  - channel
    - » manage the AMB's
      - debug
      - read and write configuration register
      - clock enable management
      - soft channel reset
        - – recover when a transmission error is detected
          - – mem_ctlr detects CRC error or AMB signals via an alert frame
        - – reset and then retry all writes that weren't committed
      - channel sync
        - – insure that AMB clock recovery circuits see the min. # of transitions
          - – southbound – transitions provided by mem_ctlr as fake write data
          - – northbound response – last DIMM sends fake read return
        - – must be inserted once every 42-46 frames (JEDEC standard)
        - – implies channel can't be powered down easily (another power defect)
  - DRAM
    - » AMB's decode and send to DRAM devices on the DIMM

# Frame and Command Scheduling

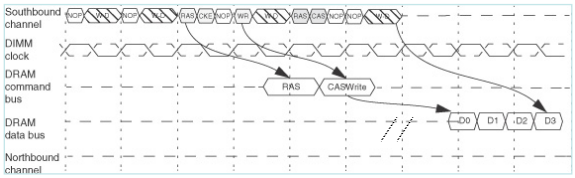
- **Interesting set of choices**
  - master to multiple slave controllers (obvious)
  - FB mem_ctlr still maintains total control of:
    - » DRAM and frame scheduling
      - minimizes logic in AMB's
      - AMB's respond to channel commands w/ predictable timing
        - also translates channel to DRAM commands but w/o additional scheduling
      - AMB's do not
        - check for DRAM protocol compliance
        - does not protect against northbound frame collision
    - » apparent strategy
      - minimize additional latency hit in the AMB daisy chain
        - already problematic due to the resync issue
      - maintain centralized control over scheduling and DRAM timing
        - AMB is less specialized for the DDRx DRAM component flavor
        - AMB predictable timing response is required for this to happen anyway
- **Result**
  - improve capacity & bandwidth, sacrifice latency



# Sample Read & Write Transactions




**A: RAS B:CAS and precharge**
**DRAM RAS and posted CAS scheduled to different DRAM clocks**
**Latency critical commands should be posted in slot A**




**Write data does not need to be contiguous – allows read returns to be interleaved in a write burst, write command can precede completion of write data delivery**

# AMB Asic

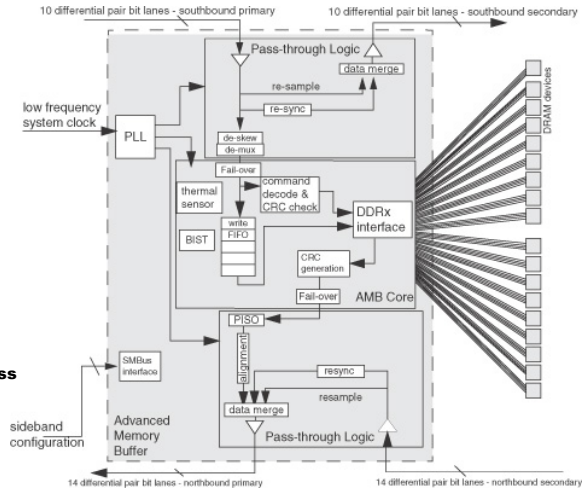
- 3 logic blocks
  - northbound pass-through
  - southbound pass-through
    - » all commands must be partially decoded
  - core
    - » current write buffer design
      - buffer 32 72-bit write data frames
        - allows priority for read returns
      - plus buffer the 3 write data frames that must be speculatively stored
        - since only 1-bit of the target AMB address is contained in each frame
    - » CRC check & generate logic
    - » PISO (parallel in serial out)
      - serializes read returns into proper frame format on northbound lane
    - » read return data is sync'd for seamless entry onto northbound lanes
      - removes rank switching overhead seen in conventional DDRx
      - maximizes read bandwidth



# Typical AMB Block Diagram

SMBus: Mem_Ctlr R/W access to configuration registers. Independent of high-speed N & S lanes.

DOES NOT – allow data access if northbound lanes fail



## Additional Features

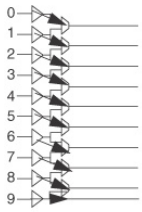
- BIST
  - for large capacity sequential testing is prohibitive
  - BIST feature allows parallel test
  - what is it really?
    - » several autonomous FSM's configured via the SMBus
- Thermal sensor
  - 2-rank FB-Dimm and AMB consumes up to 20 watts
    - » hence thermals can change rapidly
    - » need
      - protect the devices
      - keep the thermal sensitive electrical properties in "open eye" status
  - FB mem_ctlr periodically reads the thermal sensor
    - » throttles commands as necessary
    - » more centralized control



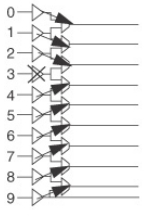
## RAS Features

- Reliability, Availability, Serviceability
- Checksum in the transport layer
  - CRC
    - » particularly needed due to timing uncertainty
      - correct when a single bit lane loses phase
      - resulting in burst loss on a single lane
- Bit lane steering
  - lane failure happens
    - » most commonly caused by DIMM socket interconnect failure
      - users put DIMMs in sockets
      - uneven or ham-fisted pressure causes metal fatigue
      - repeated thermal variations subsequently cause permanent failure
  - cure
    - » for single lane failure steer remaining 9 lanes to the working lanes

## Steering Example




Bit steering logic passes through in functional case




Bit steering logic merges around failed bit lane

- South lane failure example
  - alert frame sent north
  - enters error wait
  - FB ctlr sends soft reset
    - » hence must keep copies of commands and data in flight
    - » run training sequence to discover faulty lane
    - » reconfigure registers via SMBus
  - failed lane does reduce CRC protection
  - note
    - » top and bottom lanes are not protected



## Southbound Fail Over Mode

- Command and write data example
  - normal 10 lanes & 120 bit frames
    - » 2 bits: frame type
    - » 24 bits: command
    - » 8 check or mask bits
    - » 22 bits of CRC
    - » 64 bits of data
  - 9 bit lanes due to lane failure
    - » 22 bit CRC reduced to 10 bits
      - remember 12 bits per frame per lane
    - » good enough to detect
      - 1, 2, & 3 bit faults
      - continuous faults in another lane

## Northbound Fail Over Mode

- 14 lanes – 168 bits/frame
  - 128 data
    - » split into two groups
  - 16 check
  - 24 CRC
    - » also split into two groups to match data split
- 13 lane failover
  - CRC becomes 2 6-bit groups
- 12 lane failover
  - lose the check bits
  - however
    - » in the standard
    - » not currently supported by AMB ASIC
  - so if 2nd lane fails
    - » use 13 lane to remove data (corrected by CRC) & quarantine

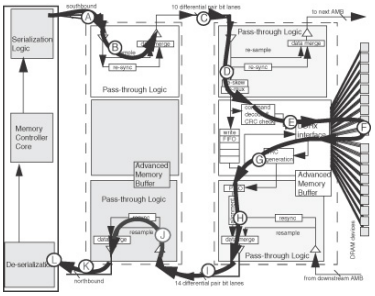


## Hot Add and Replace

- Point to point signaling
  - lends itself to fault isolation
  - connectors are pass through if no DIMM
  - UI timing slack already built into the protocol
- Error log kept
  - sysadmin notified
    - » direct data removal and quarantine if possible
    - » under quarantine power removed from faulty socket
      - replace faulty DIMM
    - » unquarantine brings new DIMM back online

# FB Dimm Performance

| Component | Min (ps) | Max (ps) | Notes |
|---|---|---|---|
| A: Ctrlr to DIMM flight | 800 | 1200 | routing distance dependent |
| B: SB frame resample | 900 | 1600 | process dependent |
| C: SB DIMM-DIMM flight | 600 | 900 | routing distance dependent |
| D: Freme de-skew & parallize | 5000 | 5900 | realign independent bit-lanes |
| E: Cmd check & decode | 3000 | 3000 | AMB specific |
| F: DRAM access | 25200 | 25200 | tRCD+tCAS+tDQSCK+CLK_Delay |
| G: Data serialization | 4500 | 4500 | includes CRC generation |
| H: Data merge w/ NB traffic | 1800 | 2800 | time to wait for frame alignment |
| I: NB DIMM2DIMM flight | 600 | 900 | routing distance dependent |
| J: NB frame resync | 2000 | 3200 | may need to remerge on NB lanes |
| K: DIMM2CTLR flight | 800 | 1200 | routing distance dependent |
| L: Frame-into-CTLR | 3000 | 3000 | deserialization delay |




**Basis: 667 MT/s DDR2 Dram**

**2 AMB example – actual latency increases w/ capacity e.g. # of FB-DIMMs**

**Typical – 1st FB-Dimm operates in resync – rest in resample**



# Fixed vs. Variable Latency Scheduling

- **More FB-DIMM complexity**
  - **Actual latency depends on where the FB-DIMM sits**
    - » **closest is fastest**
    - » **BUT different DRAM speeds are also allowed**
      - **CTLR samples to determine properties**
  - **Fixed**
    - » **base all timing schedules on slowest return**
      - **each AMB responsible for placing their return to match slowest**
  - **Variable**
    - » **DIMM puts return on as soon as it is available**
    - » **problems – you bet!!**
      - **northbound collisions could occur**
      - **hence limited to short channel configurations (presently)**
- **Extensions being studied**
  - **there are obvious flaws in the current standard**

# Conclusions

- Not clear if FB-DIMM is a good idea
  - improves bandwidth but additional cost and latency
    - » DRAM system cost is a huge concern for platform builders
  - allows more capacity but w/ capacity dependent latency
    - » BoB designed to mitigate this
      - but higher cost due to more pins
- Will Intel cut and run
  - TBD
- Personal conclusion
  - there just has to be a better way
    - » reluctance of system builders to adopt is a strong signal
  - DRAM by nature is hairy
    - » FB just made it worse
- Phew!!