TSM_EmbHardw

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

05 – Cache Coherence / Multi Processing Scheduling / Network on Chip / Test

Hans Dermot Doran: Institute of Embedded Systems

V 1.0 23.05.14 New Slides

What you should be able to do after today

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

- The student will understand the usage of layered cache systems
- The student will understand the problems of cache coherence in multi-processor systems
- The student will be able to describe the generalities of snooping and directory coherence protocols. The student will be able to schedule tasks using rate monotonic scheduling
- The student will be able to use Amdahls law to calculate theoretical speed-ups.
- The student will be able to explain the test-and-set as well as derivative instructions and intentions

What you should be able to do after today

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

- The student will be able to explain the SQMS and MQMS scheduling systems and issues
- The student will be able to explain the generalities of Network on Chip technology
- The student will know the difference between white, gray and black box testing.
- The student will be able to explain the generalities of the Open Verification Methodology
- The student will be able to explain the generalities of HW-SW Co-Verification

Digital Signal Processors

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

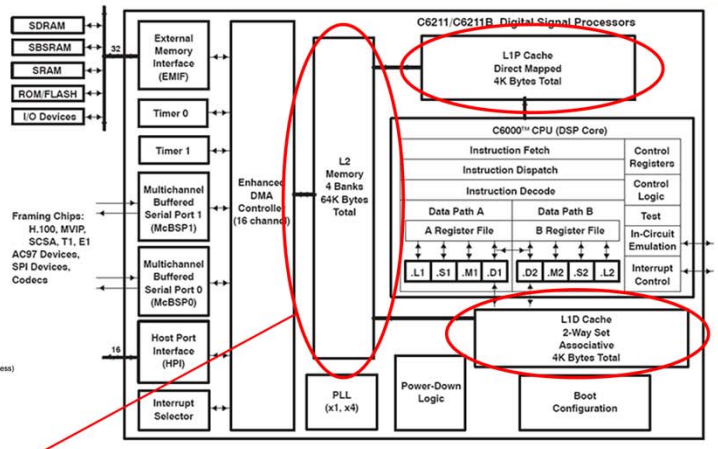
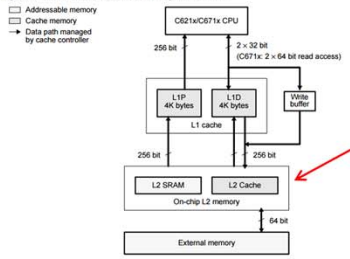


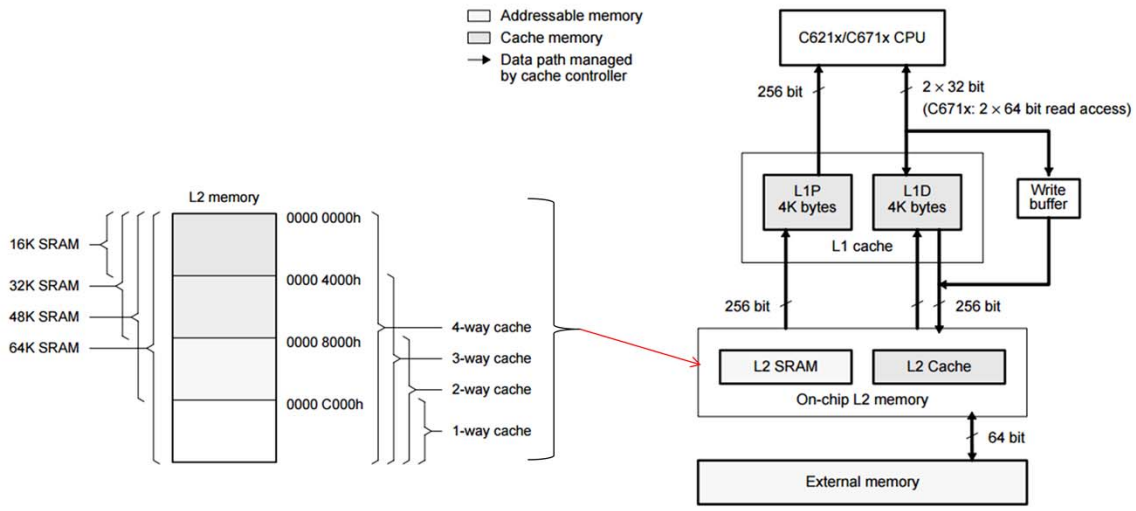
Figure 1–4. C621x/C671x Cache Memory Architecture



Cache Arrangements TI DSP

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

Figure 1–4. C621x/C671x Cache Memory Architecture



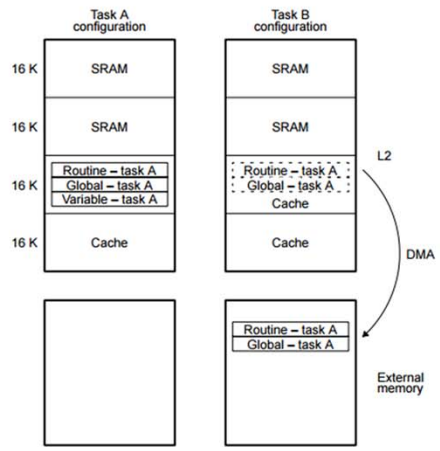
from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf

In the case of a simple co-processor a simple barrier can be used to ensure the code of the main processor doesn't advance till the co-processor has finished its work. The

Dynamic Cache / SPM Management

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

Figure 2–14. Changing L2 Cache Size During Run-Time (C6211/C6711 Devices)



- Cache -> memory switch possible
- Dynamically changing size of cache also possible (see left)
- Memory -> cache possible but cache consistency problems may occur

from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf
https://lagunita.stanford.edu/c4x/Engineering/CS316/asset/A_Primer_on_Memory_Consistency_and_Coherence.pdf

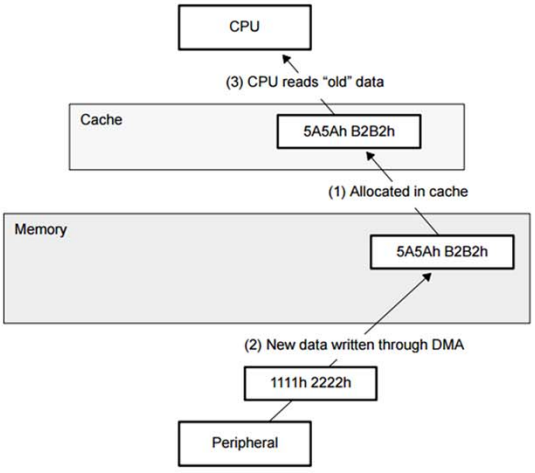
Cache Coherence

Zurich University of Applied Sciences

School of Engineering

InES Institute of Embedded Systems

Figure 2–7. Cache Coherence Problem

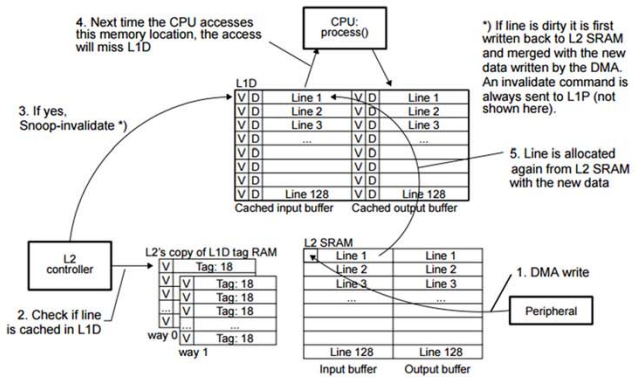


from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf
https://lagunita.stanford.edu/c4x/Engineering/CS316/asset/A_Primer_on_Memory_Consistency_and_Coherence.pdf

Cache Coherence

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

Figure 2–8. DMA Write to L2 SRAM



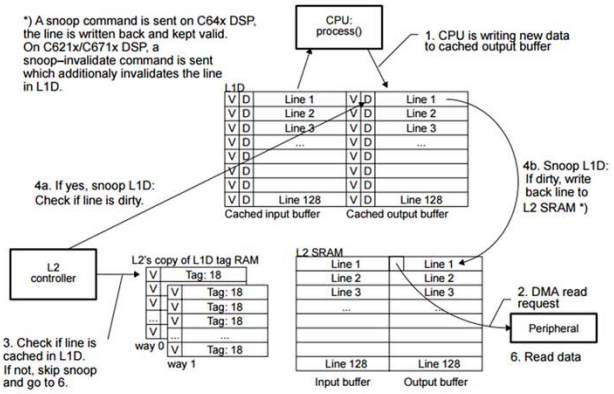
- Peripheral requests write access to line in L2S RAM which maps to set 1 in L1
- L2 Cache controller: is line in L1 cache?
- If cached then snoop-invalidate command to L1D (valid bit = 0). If dirty then writeback to L2. Data written to L2
- On CPU read L1D then cache miss -> Load data from L2

from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf

Cache Coherence

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

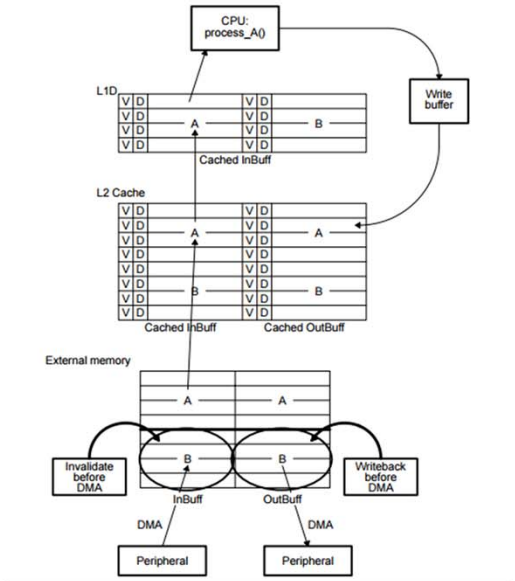
Figure 2–9. DMA Read of L2 SRAM



- Peripheral requests read access to line in L2S RAM which maps to set 1 in L1
- L2 Cache controller: is line in L1 cache?
- If cached then snoop-invalidate command to L1D (valid bit = 0). If dirty then write back to L2. Data written to L2
- Peripheral continues read

from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf

Cache Coherence



- For accesses with external memory where cache coherence protocol does not hold, SW must imitate.
- So InBuff B must be invalidated before DMA write

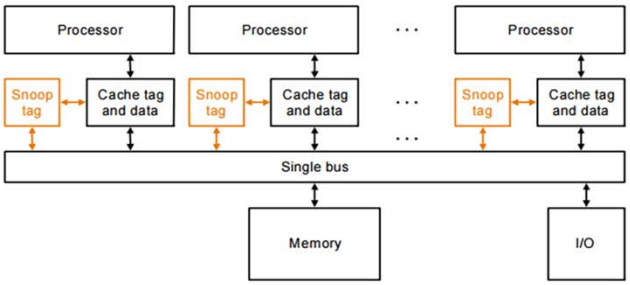
```
CACHE_invL2(InBuffB, BUFSIZE, CACHE_WAIT);
```

- And OutBuff neds to be written back from L+ and L2 before the DMA transfer begins

```
CACHE_wbL2(OutBuffB, BUFSIZE, CACHE_WAIT);
```


from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf

Cache Snooping Protocol



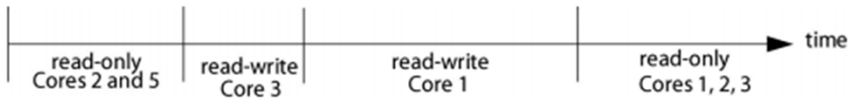
- Cache Controller monitors activity on the bus including its own activity
- Processor initiates a transaction
- If a read transaction on cached line then check for dirty bit and write-back
- If a write transaction then invalidate own cache line.

from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf

Cache Snooping Protocol

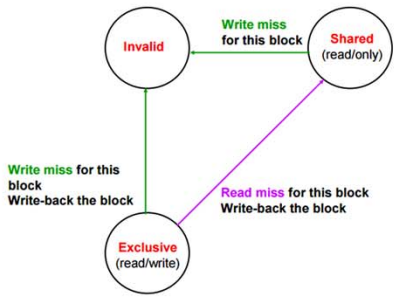
Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

- Single Writer, Multiple Reader
- Time divided into epochs
- In an epoch
  - a single core may write or (rw)
  - Multiple cores may read only (ro)
- Core reads – sends a message to ensure that no other cores are in a rw state
- Messages ends any rw or ro epoch



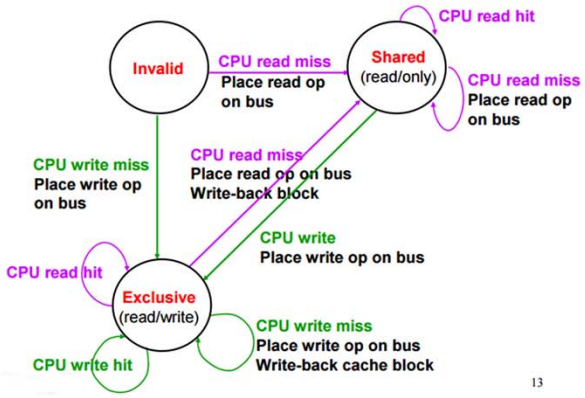
Cache Snooping Protocol MSI

- In Memory a block can be
  - Clean – in one or more caches and up-to-date in memory
  - Dirty – in exactly one cache
  - Uncached – in no caches
- State of cache line is (MSI)
  - Modified – a dirty line
  - Shared – clean and in other caches
  - Invalid – line with no valid date



- MSI protocol makes certain that if a line is dirty then this line is not valid in any other caches and that a read request gets the latest data.

Cache Snooping Protocol MSI

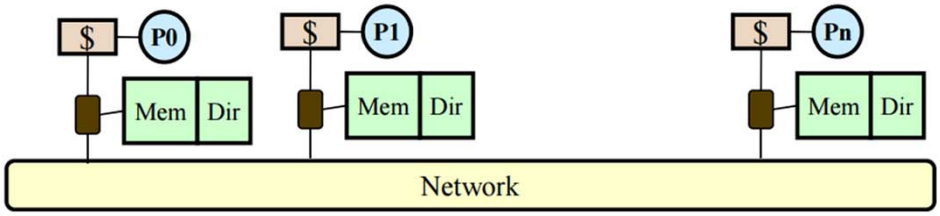


13

- Bus used as broadcast medium
  - Keep-the-bus protocol: Master holds bus until transaction complete
  - Split transaction buses: request/response in different phases, no operation initiated for block already under examination
  - Bandwidth intensive
- Snoop on highest level cache
  - Everything in L1 is also in L2

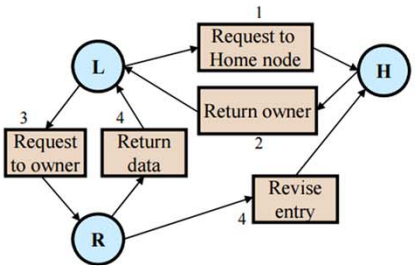
from: http://www.ti.com/lit/ug/spru656a/spru656a.pdf

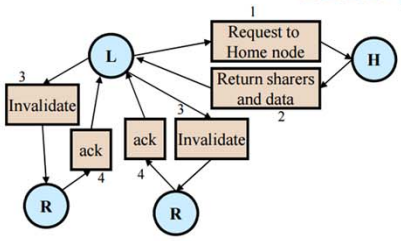
Directory based coherence (1)



- Directory keeps track of state and where each copy of a memory block is cached
- Directory can be distributed or centralised
- Processor consults directory before loading cache lines
- When line is updated then directory is consulted to update or invalidate blocks$
- More scalable than snooping but higher latency

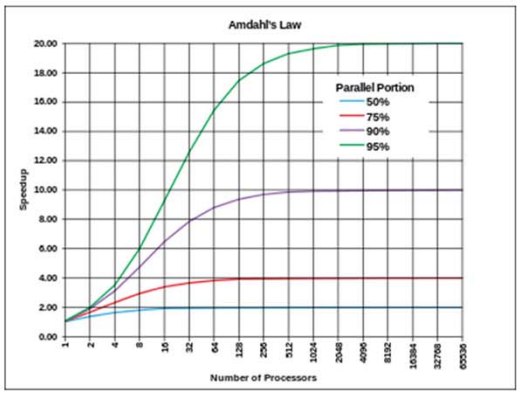
Directory based coherence (2)





- Read miss (left) and write miss (right)
- L(ocal) requestor node – reader/writer of cache block
- H(ome) node – node that stores block in memory
- R(emote) node – other nodes with cached copy
- MSI or MESI protocol is used here as well
  - E in MESI -> split of M into Modified and Exclusive

Other Multi-processor Factors



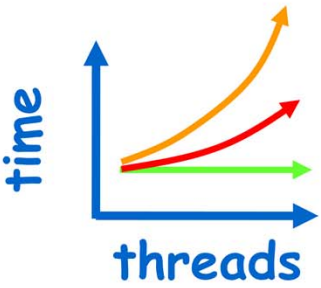
$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

- Multi Processor systems offer speed-up due to multiple processors
- Speed up depends on how much can be parallelized
- Also dependent on degree of coordination between processors
- Amdahl's Law helps predict the theoretical speed-up

from: https://en.wikipedia.org/wiki/Amdahl%27s_law

Other Multi-processor Factors (2)

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems



TAS lock

TTAS lock

Ideal

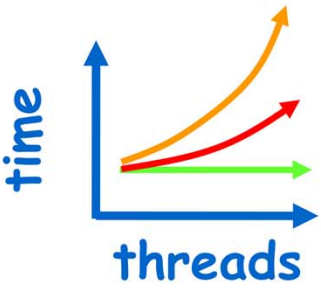
```
TestAndSet:   MOV     AX, 1h
try_CS:               XCHG    AX, gesperrt ; nicht
unterbrechbar
              CMP     AX, 0h
              JNE     try_CS
              RET

Release:      MOV     gesperrt, 0h ; nicht kritisch
              RET
```

- This maximum may not be reached due to:
  - Cache coherence protocols
  - Synchronisation effort
- Synchronisation often required for mutual exclusion
- TAS – Test and Set
  - Atomic instruction – often offered by processor

Other Multi-processor Factors (3)

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems



TAS lock

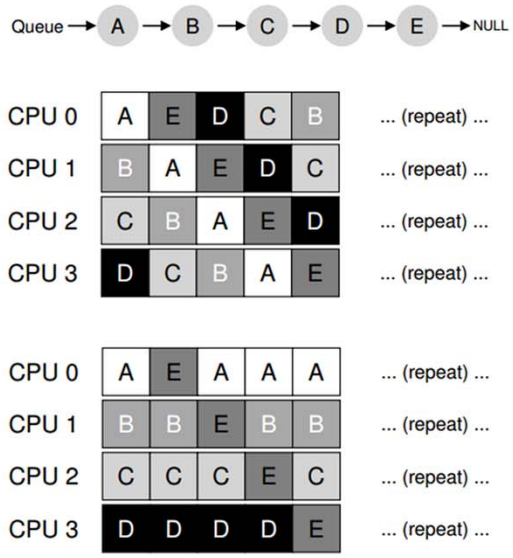
TTAS lock

Ideal

- TAS – Test and Set
  - Slow due to bus locking and cache invalidation
- Test and Test and Set (TTAS) offers improvements

```
boolean locked := false // shared lock variable
procedure EnterCritical() {
        do { while (locked == true) skip // spin until lock seems free
        } while TestAndSet(locked) // actual atomic locking
}
```

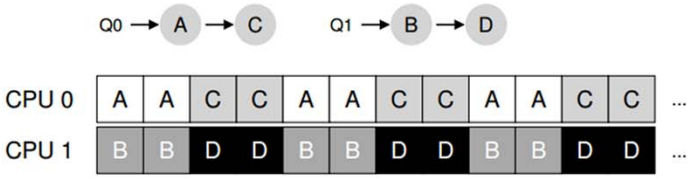
Other Multi-processor Factors (4)



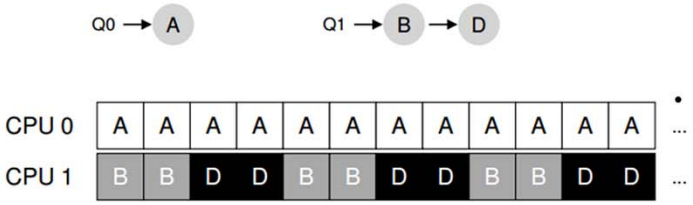
- Scheduling: **SQMS**
  - With a simple task queue (top) and no special arrangements the schedule such as the middle picture will happen
- Every time a task is swapped to another CPU the cache has to be “re-loaded”. With a cache-affinity scheme the tasks are “kept” on a particular CPU to conserve cache states

from: http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-multi.pdf

Other Multi-processor Factors (5)



- Scheduling: **MQMS**
  - SQMS doesn't scale well
  - Multi Queue Multiprocessor Scheduling
  - Scales better as lock and cache contention prevented and cache affinity provided

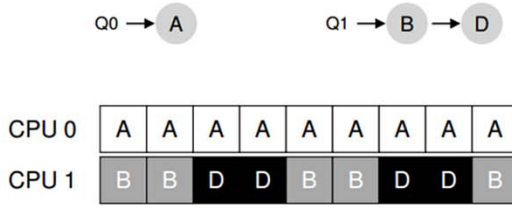


- Suffers from load-imbalancing

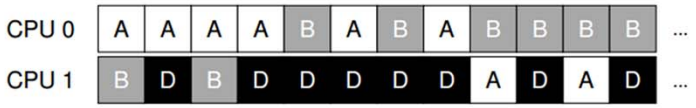
from: http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-multi.pdf

Other Multi-processor Factors (6)

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems



- With load imbalancing A gets too much processing time whereas B and D have to share. If A stops then B and D still share the same CPU and A is idle.



- **Continuous migration** is one solution: Cache affinity is partially kept
- **Work Stealing** also used – a queue will look at other queues to see if they have more tasks, and take them if they do

from: http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-multi.pdf

Network on Chip (1)

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

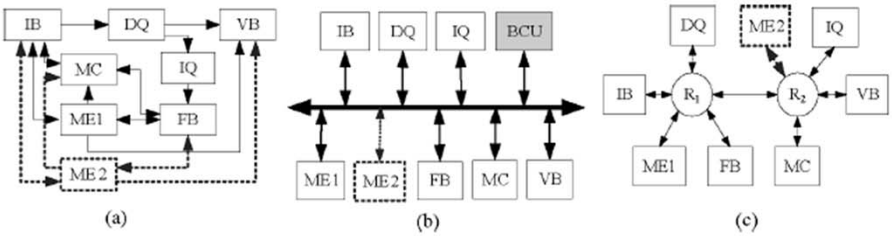


Fig. 5. Implementation of MPEG-2 encoder with 2 MEs using: (a) P2P; (b) bus and (c) NoC communication architectures.

P2P (Hardware acceleration) -> 46.5 frames/sec

Bus -> 36.2 frames/sec

NoC -> 45.6 frames/sec

Zürcher Fachhochschule

23

There are three essential types of on-chip communication systems: Point to Point (P2P), Bus Systems and Network on Chip (NoC).

The diagrams above show three implementations of a MPEG-2 encoder. The units are: (1) input buffer (IB); (2) DCT and quantization (DQ); (3) variable length encoder and output buffer (VE); (4) motion compensation (MC); (5) inverse quantization and inverse DCT (IQ); (6) motion estimation (ME); and (7) reconstructed frame buffer (FB).

The P2P system can offer the highest performance (46.5 frames/sec) but suffers from lack of scalability, higher design effort and high complexity. A similar system can be seen in our discussion on hardware accelleration.

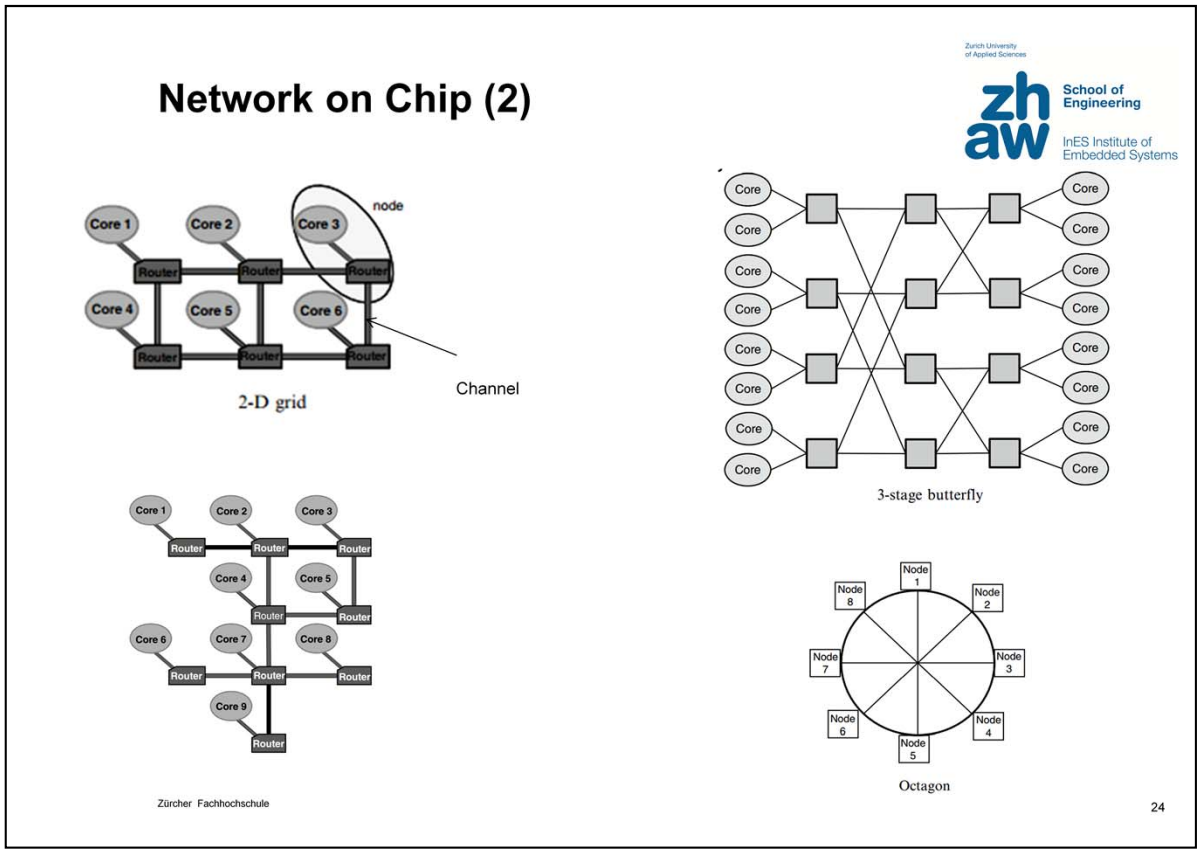
Bus systems are efficient to connectivity of a few tens of cores but suffer from limited bandwidth (36.2 frames/sec) and are not well scaleable. This data was collecting with the cores sending packetised data over both the bus and the NoC whereas the P2P system communicated in streams.

The NoC (45.6 frames/sec) systems offer vastly increased scaleability as well as portability across platforms.

Reference top picture: Hyung Gyu Lee, Naehyuck Chang, Umit Y. Ogras, and Radu Marculescu. 2008. On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. *ACM Trans. Des. Autom. Electron. Syst.* 12, 3, Article 23 (May 2008)

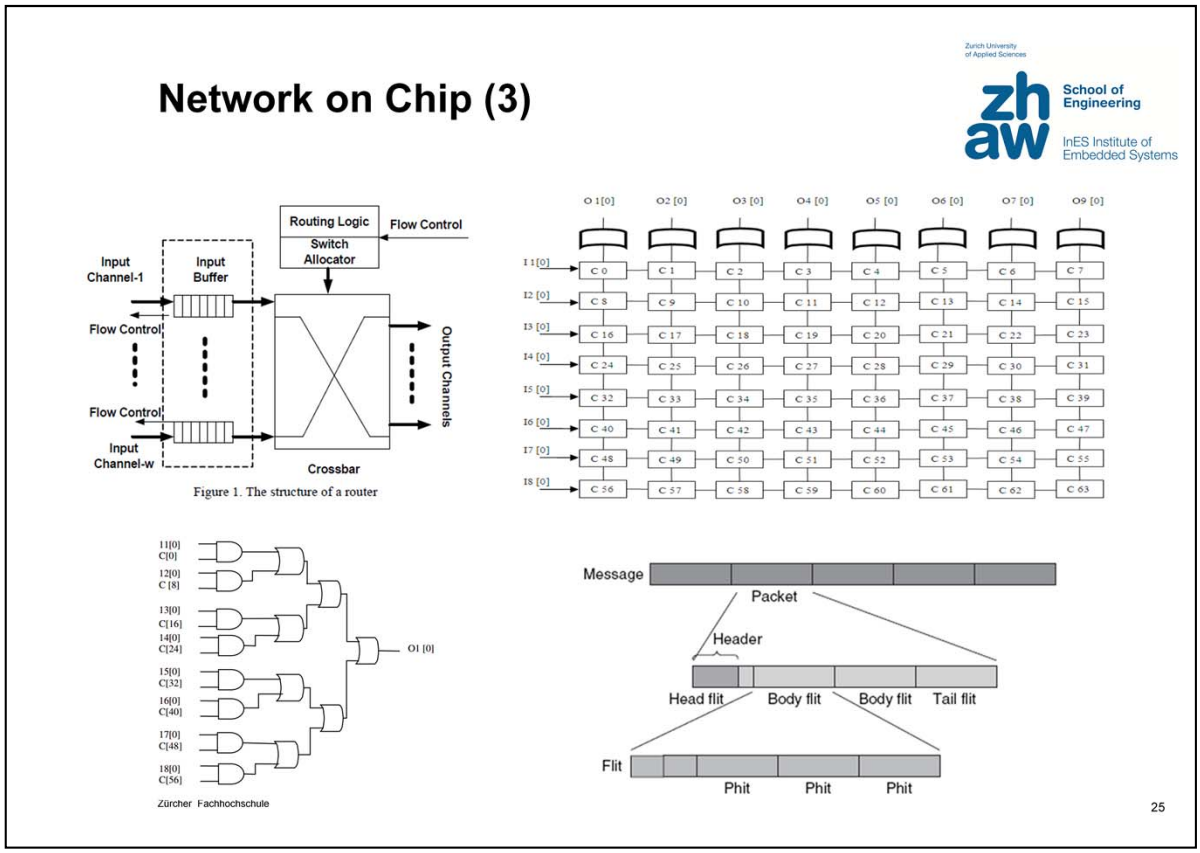
Reference bottom picture: ojs.academypublisher.com index.php jcp article download jcp

Reference Crossbar design: Mohammad Ayoub Khan, Abdul Quaiyum Ansari, Design of 8-Bit Programmable Crossbar Switch for Network-on-Chip Router, Trends in Network and Communications Communications in Computer and Information Science, Volume 197, 2011, pp 526-535



Network on Chip topologies can be very diverse and some ecxamples are shown above : clockwise from left: mesh, butterfly, octagon and reduced mesh.

From: Reliability, Availability and Serviceability of Networks-on-Chip, Cotra, Amory, Lubaszewski, Springer.



Two main modes of transporting flits in a NoC are circuit switching and packet switching

Circuit switching

- physical path between the source and the destination is reserved prior to the transmission of data
- message header flit traverses the network from the source to the destination, reserving links along the way
- Advantage: low latency transfers, once path is reserved
- Disadvantage: pure circuit switching does not scale well with NoC size
  - several links are occupied for the duration of the transmitted data, even when no data is being transmitted for instance in the setup and tear down phases

Packet Switching

- packets are transmitted from source and make their way independently to receiver
  - possibly along different routes and with different delays
- zero start up time, followed by a variable delay due to contention in routers along packet path
- QoS guarantees are harder to make in packet switching than in circuit switching
- three main packet switching scheme variants

SAF switching

- packet is sent from one router to the next only if the receiving router has buffer space for entire packet
- buffer size in the router is at least equal to the size of a packet
- Disadvantage: excessive buffer requirements

Reference: Pictures upper and lower left: Reference Crossbar design: Mohammad Ayoub Khan, Abdul Quaiyum Ansari, Design of 8-Bit Programmable Crossbar Switch for Network-on-Chip Router, Trends in Network and Communications Communications in Computer and Information Science, Volume 197, 2011, pp 526-535

Text and picture lower right: On-Chip Communication Architectures, Sudeep Pasricha & Nikil Dutt, Morgan Kaufmann, ISBN: 978-0-12-373892-9

Network on Chip (4)

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems

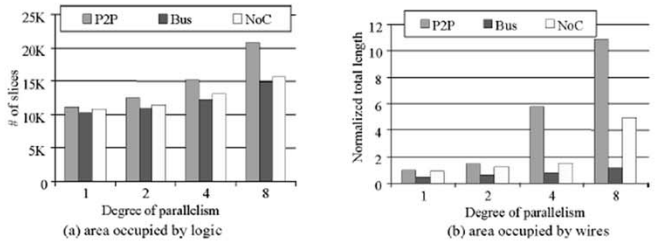


Fig. 6. Area comparisons of the MPEG-2 encoder implemented using P2P, bus, and NoC architectures for increasing levels of parallelism.

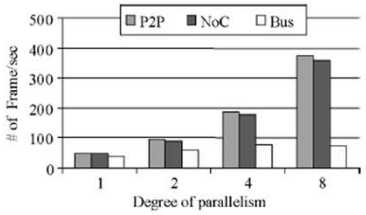


Fig. 8. Throughput comparison of various MPEG-2 encoder implementations.

Zürcher Fachhochschule

26

Cont..

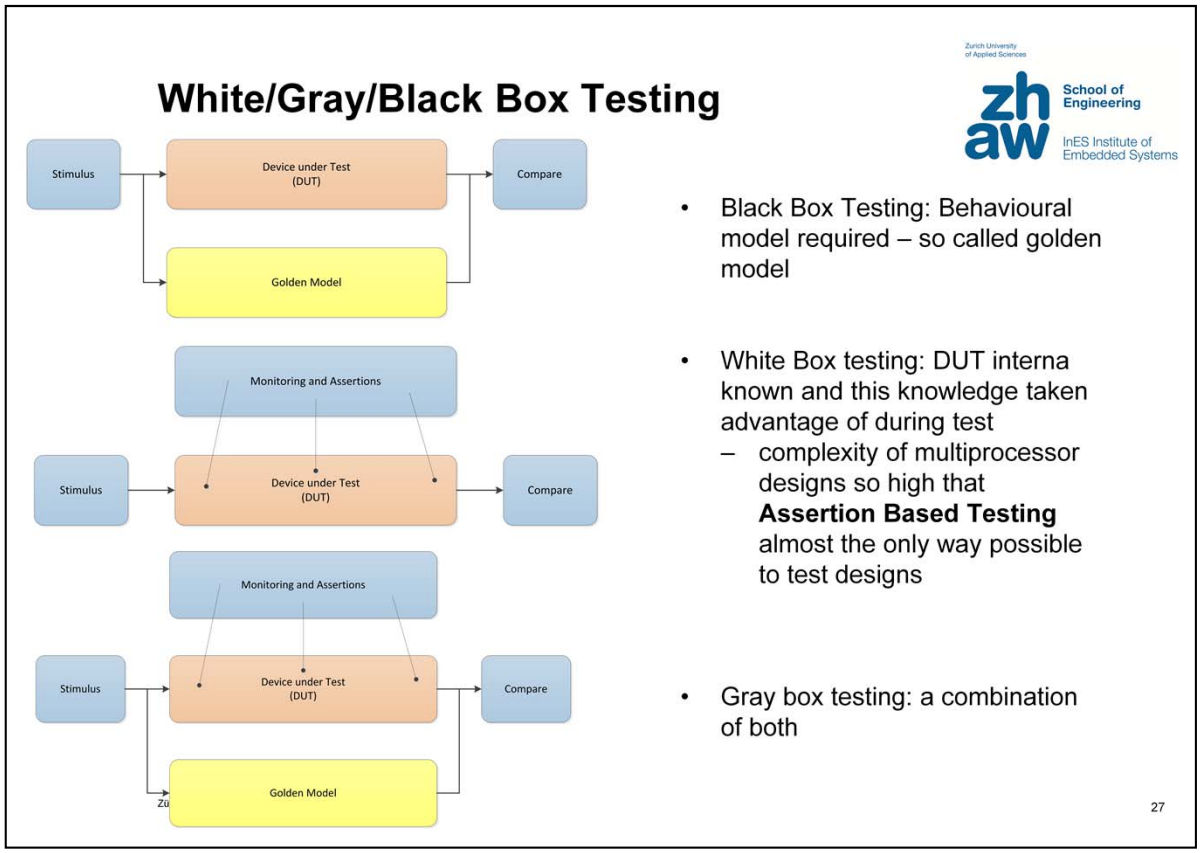
VCT Switching

- reduces router latency over SAF switching by forwarding first flit of a packet as soon as space for the entire packet is available in the next router
- if no space is available in receiving buffer, no flits are sent, and the entire packet is buffered
- same buffering requirements as SAF switching

WH switching

- flit from a packet is forwarded to receiving router if space exists for that flit
- parts of the packet can be distributed among two or more routers
- buffer requirements are reduced to one flit, instead of an entire packet
- more susceptible to deadlocks due to usage dependencies between links

The performance evaluation of the connectivity methods is shown above: It is obvious that the on-chip routing would appear to be a cheap option.



White box testing assumes intimate knowledge of the unit interna which implies that path completeness can be achieved, even though it may be infeasable.

A unit test is an element of White Box testing but white box testing may cover more than a single unit.

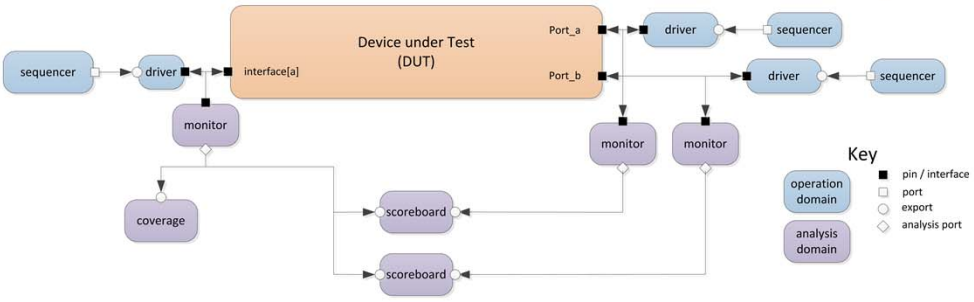
We can reach path completeness (each path through the code is exercised – does not assume that each statement is reached) in many units. It becomes problematic to estimate coverage in a single unit and this becomes more difficult and more complex when considering a chain of units. HW development tools usually supply this information, in software often profiling is required.

Furthermore it might be economically feasible to check some paths against a model and others with assertions -> A combination of the two approaches is called **Gray Box** testing.

The framework with the golden model and without Monitoring and Assertions is known as **Black Box** testing. This tends to be **System-Level** testing.

Experience shows that assertion based checking is useful. Such an architecture is known as **Grey Box** testing

OVM



- Driver converts transaction to bit pattern on ports
- Monitor -> transaction monitor, recognises bit patterns and reports the read/write/idle transactions to scoreboard
- Sequencer coordinates sequences of transactions to the drivers
- Scoreboard contains golden model

Zürcher Fachhochschule

28

We have now reached the level of complexity of a full blown verification system as proposed by the Open Verification Methodology (OVM) as propagated by Cadence Design Systems. OVM defines a terminology, a methodology and an appropriate class library in SystemVerilog, an extension of Verilog (IEEE 1800-2005, IEEE 1364-2005). Proprietary? – possibly but the concepts are universal.

OVM is a complex system and out of the range of this course but – given it's the non plus ultra its useful to know the name and note a few points to get one started.

Notice that the monitors are not assertion checkers but simple transactional monitors. The monitors puts transactional data into both the scoreboard and the coverage counter.

The coverage counter is responsible for counting things such as point coverage; specific values of a variable or a value in a specific range. Transition coverage; sequence of values assigned to a variable or Cross coverage; that is numerical relationships between two independent values.

Examples are; number of bus cycles or range of addresses covered. Also one can create categories of occurrences which are counted, for instance write and read-modify-write cycles may be ignored but read cycles counted. These categories are referred to as bins.

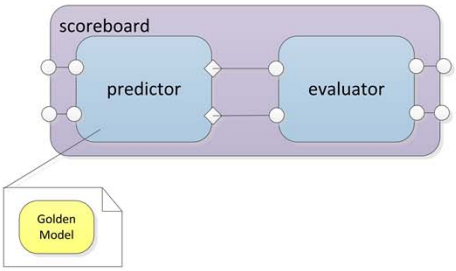
Scoreboarding is where the test takes place. A score boarder takes the model, computes expected output from a stimulated input and compares input and output from the DUT.

Robust scoreboarders should support ordered and unordered checks. Allow for mismatches on start-up and ordered end-of-simulation (any data values in scoreboard an not accounted for)

A full blown test bench as used in FPGA's and ASIC design will implement model based checking, the model is, in the case of OVM, a transactional model. The scoreboard "knows" this model. This is the **golden model** and a black box test.

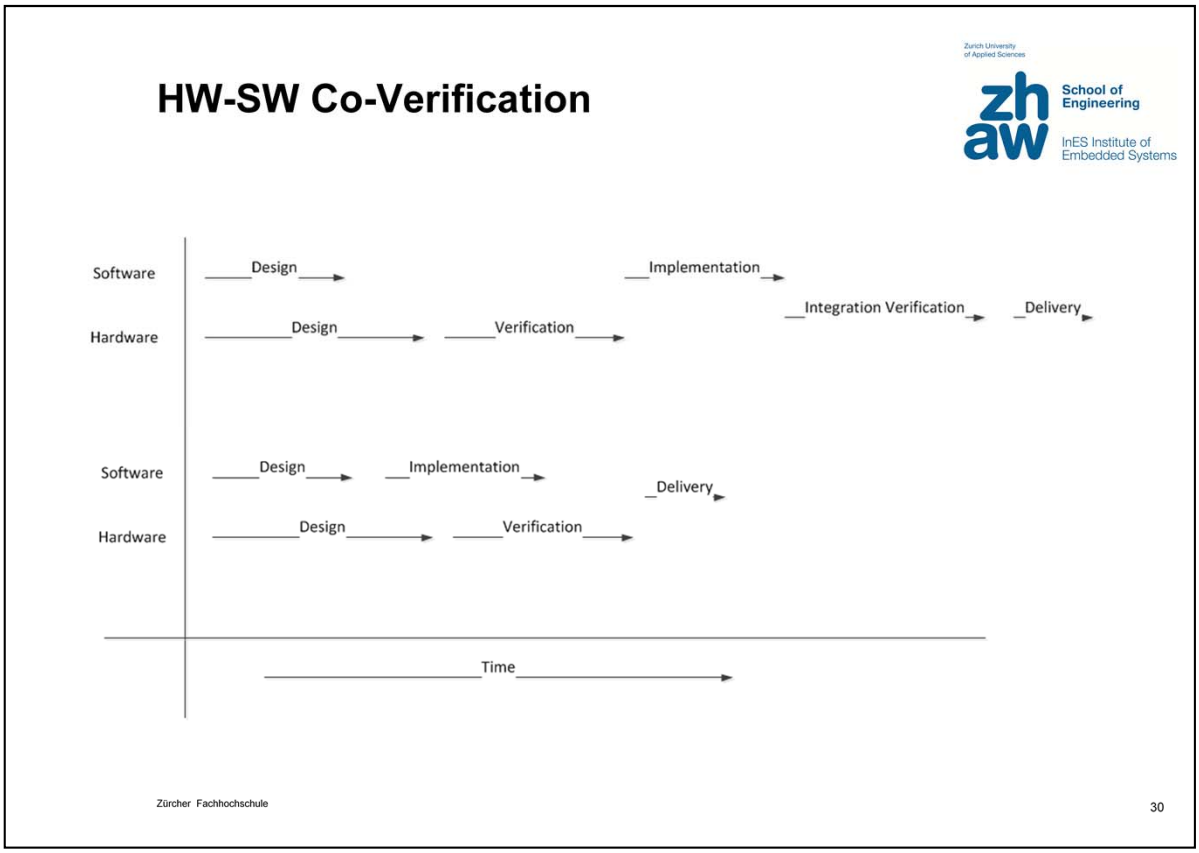
OVM

Zurich University of Applied Sciences
School of Engineering
InES Institute of Embedded Systems



- Code coverage metrics include
- Toggle coverage – each pin must experience at least on H->L and L->H transition
- Line coverage – number of lines of source code have been covered
- Statement coverage – statement based rather than LoC based
- Block coverage – block coverage – the coverage of delimited blocks (i.e. between { })
- Branch coverage – have branches (if, while ..) been evaluated to both true and false
- Expression coverage – conditions evaluated to both true and false
- Finite state machine coverage – the number of times state transitions took place or the number of transition from one state to each of its neighbours.

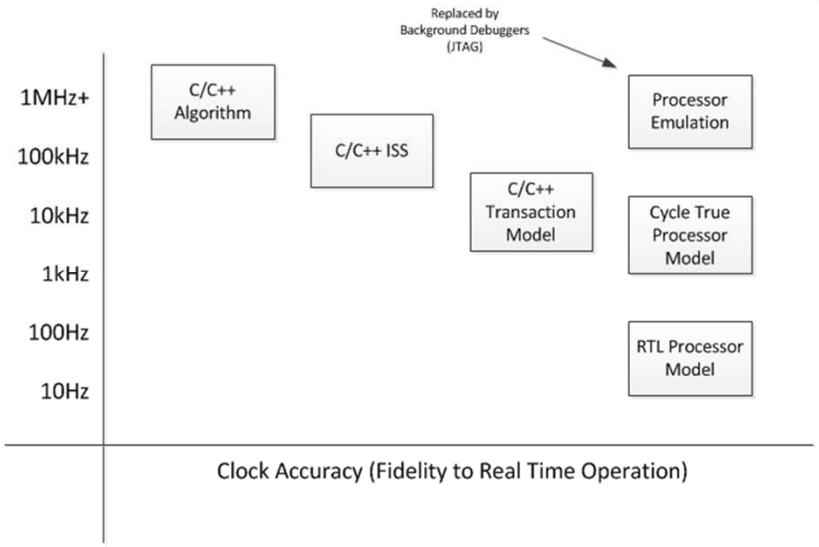
- Scoreboard contains predictor and evaluator
  - Based on transactions the predictor predicts what the result of the transaction should be -> predictor is the golden model
  - The evaluator confirms this



Lets re-cap: HW-SW co-design: from a functional specification a formal model is created (functional model), From this model partitioning and mapping is performed and individual units are defined, Partitioning implies communication between individual units. Design and test process involves bringing these units back together to create the system. This system is in itself a model (->implementation model). Here one should also apply principle of risk reduction by redundancy (Inadvisable to use the same model for implementation and verification) (->verification model). If HW and SW units tested via unit testing then need only verify correct functioning of the interface. If interface previously tested then assume-guarantee principle means no integration testing necessary. If this degree of modularity is not possible then the issue is building a verification model that includes HW, SW and HW-SW communication components .

What makes this difficult is that SW verification may be done on many different platforms whereas HW generally only on development platform so therefore HW-SW Co-Verification usually performed on HW development platform or on the target. The problem is that SW generally runs on microcontroller or DSP. Usually possible to integrate microcontroller into HW development environment and run code on it.

HW-SW Co-Verification



From: Nakamura et. Al „A Fast Hardware/Software Co-Verification Method for System-On-a-Chip by Using a C/C++ Simulator and FPGA Emulator with Shared Register Communication“ DAC 2004, June 7-11

Zürcher Fachhochschule

31

The slide above shows the clock accuracy of the modeling solutions mapped against the speed of execution (and hence time required for test). The method with the slowest execution time is implementing the Microprocessor in **RTL model** this gives cycle true C/C++ performance but simulation time is very high due to simulator executing the entire CPU model in hardware – for each transaction (remember that word?) on system-bus multiple simulation/CPU cycles required.

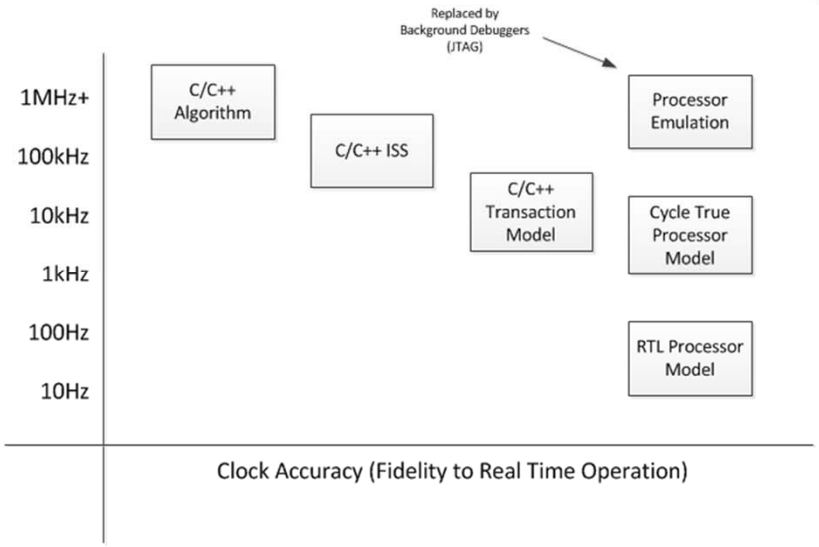
Solution: Either use “real” processor, for instance emulator attached to FPGA or some kind of processor model

Cycle-True CPU modeling: This is a reduced model of processor with, typically, own assembler format. Test/production code is compiled and assembler-output converted to model readable format Code runs on model. Due to simplified model simulation acceleration, but not by much, is achievable.

Emulation: HW is downloaded into FPGA and attached to an external emulator SW orientated test possibility – assumes basic HW works. External connections difficult to manage (100+) in test bench environments. Emulators seriously expensive. No, or little, (FPGA) HW debugging possible. Real-time true

Transaction: As in SystemC, SystemVerilog models. Driver used to create signals from transactions (R, RW, W etc) code functions in background. Simulator orientated system, means HW can be checked (with monitors) simultaneously. No cycle trueness and not real-time capable. Faster due to software algorithms and (internal) memory accesses not being played out on signal bus. Still slower since transactions queued at driver export or stimulator port

HW-SW Co-Verification



From: Nakamura et. Al „A Fast Hardware/Software Co-Verification Method for System-On-a-Chip by Using a C/C++ Simulator and FPGA Emulator with Shared Register Communication“ DAC 2004, June 7-11

Zürcher Fachhochschule

32

Instruction Set Simulator: A model of the processor that executes processor instructions and emulates connection to the hardware. Can be directly attached to HW development environments (modelsim) (and runs on same environment). Excellent (seamless?) integration with development system, good for integrating HW and SW checking. Models available for all processors (NIOS, Microblaze). Reasonable speeds achievable

Algorithm Orientated Verification: A pure instruction simulator – running on a different machine to HW and communicating with HW over registers or serial port. Clock driven by SW. SW orientated – almost HW-in-the-loop like. HW can be in stop-and-go mode – impractical to monitor HW simultaneously. No real-time co-verification possible.

New Trends: Component orientated HW-SW Co-Verification. Comes (obviously) from SW corner of technical world. Model orientated verification. Attempts made to introduce patterns. See publications by Fei Xie et. Al.