NOC: Networks on Chip
SoC Interconnection Structures

COE838: Systems-on-Chip Design
http://www.ee.ryerson.ca/~courses/COE838/

Dr. Gul N. Khan
http://www.ee.ryerson.ca/~gnkhan
Electrical and Computer Engineering
Ryerson University

Overview

• Introduction to Networks on a Chip
• Bus and Point-to-point NoC Systems
• Routing Algorithms and Switching Techniques
• Flow Control
• NOC Topology Generation and Analysis

Chapter 5: Computer System Design – System on Chip by M.J. Flynn and W. Luk
Chapter 12: On-Chip Communication Architectures – SoC Interconnect by S. Pasricha & N. Dutt
System-on-Chip and NoC

System-on-Chip --to-- Network-on-Chip

- MPEG CORE
- VGA CORE
- Analog ADC/DAC
- Component
- CPU
- DSP
SoC Structure

A tile of the chip

A computational block

A communication link

Cache L2

Proc

Proc

Proc

Switch Fabric

Control Logic

Network Interface

Data $

Instr$

core

control

data

parity

spare

Switch Fabric

Control Logic

Network Interface

Data $

Instr$

core

bus

p1

p3

p2

p3

p4

p0
Multiple Processor/Core SoC

Inter-node communication between CPU/cores can be performed by message passing or shared memory. Number of processors in the same chip-die increases at each node (CMP and MPSoC).

• Memory sharing will require: **SHARED BUS**
  * Large Multiplexers
  * Cache coherence techniques
  * Not Scalable

• Message Passing: **NOC**
  * Scalable
  * Require data transfer transactions
  * Has overhead of extra communication
NOC: Network-on-Chip

Shared bus is not a long-term solution
• It has poor scalability
On-Chip micro-networks suit the demand of scalability and performance
NOC and Off-Chip Networks

**NOC**
- Sensitive to cost:
  - area and power
- Wires are relatively cheap
- Latency is critical
- Traffic is known a-priori
- Design time specialization
- Custom NoCs are possible

**Off-Chip Networks**
- Cost is in the links
- Latency is tolerable
- Traffic/applications unknown
- Changes at runtime
- Adherence to networking standards
On-Chip Communication Structures

a) bus

b) point-to-point

c) network
On-Chip Bus Interconnection

For highly connected multi-core system
- Communication bottleneck

For multi-master buses
- Arbitration will become a complex problem

Power grows for each communication event as more units attached will increase the capacitive load.

A crossbar switch can overcome some of these problems and limitations of the buses
- Crossbar is not scalable
SOC Communication Structures

Dedicated Point-to-Point

• Advantages
  - Optimal in terms of bandwidth, availability, latency and power usage
  - Simple to design and verify as well as easier to model

• Disadvantages
  - Number of links may increase exponentially with the increase in number of cores
  - Hardware Area
  - Routing Problems
SOC Communication Structures

Network on Chip

Advantages

- Structured architecture – Lower complexity and cost of SOC design
- Reuse of components, architectures, design methods and tools
- Efficient and high performance interconnect.
- Scalability of communication architecture

Disadvantages

- Internal network contention can cause a latency
- Bus oriented IPs need smart wrapping hardware
- Software needs clear synchronization in multiprocessor systems
Networks-on-Chip

• Interconnect for SoCs, CMPs, MPSoC and FPGAs
  ■ Multi-hop, packet-based communication
  ■ Efficient resource sharing

• Scalable communication infrastructure provides scalable performance/efficiency in
  ■ Power
  ■ Hardware Area
  ■ Design productivity
NoC?

A chip-wide network: Processing Elements (PEs) are interconnected via a packet-based network in NoC Architecture.
Network-on-Chip vs. Bus Interconnection

- Total bandwidth grows
- Link speed unaffected
- Concurrent spatial reuse
- Pipelining is built-in
- Separate abstraction layers

However
- No performance guarantee
- Extra delay in routers
- Area and power overhead?
- Modules need NI
- Unfamiliar methodology

BUS inter-connection is fairly simple and familiar

However
- Bandwidth is limited, shared
- Speed goes down as N grows
- No concurrency
- Pipelining is tough
- Central arbitration
- No layers of abstraction (communication and computation are coupled)
NoC Evolution

- Progress of on-chip communication architectures
What is an NoC?

- Network-on-chip (NoC) is a packet switched on-chip communication network designed using a layered methodology
  - “routes packets, not wires”
- NoCs use packets to route data from the source to the destination PE via a network fabric that consists of
  - switches (routers)
  - interconnection links (wires)
NoC

- NoCs are an attempt to scale down the concepts of large-scale networks, and apply them to the embedded system-on-chip (SoC) domain

- NoC Properties
  - Regular geometry that is scalable
  - Flexible QoS guarantees
  - Higher bandwidth
  - Reusable components
    - Buffers, arbiters, routers, protocol stack
  - No long global wires (or global clock tree)
    - No problematic global synchronization
    - GALS: Globally asynchronous, locally synchronous design
  - Reliable and predictable electrical and physical properties
NoC: Buses to Networks

Original Bus Features
- One transaction at a time
- Central Arbiter
- Limited bandwidth
- Synchronous
- Low cost

Shared Bus to Segmented Bus
Advanced Bus

Segmented Bus

- More General/Versatile bus architecture
- Pipelining capability
- Burst transfer
- Split transactions
- Overlapped arbitration
- Transaction preemption, resumption & reordering

Shared Bus to Segmented Bus
Buses to Networks

- Architectural paradigm shift: Replace wire spaghetti by network
- Usage paradigm shift: Pack everything in packets
- Organizational paradigm shift
  - Confiscate communications from logic designers
  - Create a new discipline, a new infrastructure responsibility
NoC Related Main Problems

Global interconnect design problems:
- Delay
- Power
- Noise
- Scalability
- Reliability

System integration
  Productivity problem

Chip Multi Processors
  For power-efficient computing
NoC Wiring Design

- NoC links:
  - Regular
  - Point-to-point -- no fan-out tree (problem)
  - Can use transmission-line layout
  - Well-defined current return path

- Can be optimized for noise / speed / power
  - Low swing, current mode, ….
NoC Scalability

Compare the wire-area for same performance

NoC: $O(n)$

Bus: $O(n^3 \sqrt{n})$

Pt-to-Pt: $O(n^2 \sqrt{n})$

Segmented Bus: $O(n^2 \sqrt{n})$
NoC Topology

Direct Topologies

- each node has direct point-to-point link to a subset of other nodes in the system called neighboring nodes
- nodes consist of computational blocks and/or memories, as well as a NI block that acts as a router e.g. Nostrum, SOCBUS, Proteo, Octagon
- as the number of nodes in the system increases, the total available communication bandwidth also increases
- fundamental trade-off is between connectivity and cost
NoC Topology

• Most direct network topologies have an orthogonal implementation, where nodes can be arranged in an n-dimensional orthogonal space
  ▪ Routing for such networks is fairly simple
  ▪ e.g. n-dimensional mesh, torus, folded torus, hypercube, and octagon

• 2D mesh is most popular topology
  ▪ All links have the same length
    • eases physical design
  ▪ Chip area grows linearly with the number of nodes
  ▪ Must be designed in such a way as to avoid traffic accumulating in the center of the mesh
NoC Topology

Torus topology, also called a k-ary n-cube, is an n-dimensional grid with k nodes in each dimension

- k-ary 1-cube (1-D torus) is essentially a ring network with k nodes
  - Limited scalability as performance decreases when more nodes

- k-ary 2-cube (i.e., 2-D torus) topology is similar to a regular mesh
  - Except that nodes at the edges are connected to switches at the opposite edge via wrap-around channels
  - Long end-around connections can, however, lead to excessive delays
NoC Topology

• Folding torus topology overcomes the long link limitation of a 2-D torus
  ▪ links have the same size

• Meshes and tori can be extended by adding bypass links to increase performance at the cost of higher area
NoC Topology

Octagon topology is another example of a direct network

- messages being sent between any 2 nodes require at most two hops
- more octagons can be tiled together to accommodate larger designs
  - by using one of the nodes is used as a bridge node
NoC Topology

• Indirect Topologies
  ▪ each node is connected to an external switch, and switches have point-to-point links to other switches
  ▪ switches do not perform any information processing, and correspondingly nodes do not perform any packet switching
  ▪ e.g. SPIN, crossbar topologies

• Fat tree topology
  ▪ nodes are connected only to the leaves of the tree
  ▪ more links near root, where bandwidth requirements are higher
Irregular NoC Topologies

- Based on the concept of using only what is necessary.
- Application-specific topologies.
- Eliminate unneeded resources and bandwidth from the system.
- Leads to reduced power and area use.
- Requires additional design work.
NOC Topology

Mesh

Physical implementation
NOC Torus Topology

Torus

Physical implementation
Deadlock, Livelock, and Starvation

**Deadlock:** A packet does not reach its destination, because it is blocked at some intermediate resource.

**Livelock:** A packet does not reach its destination, because it enters a cyclic path.

**Starvation:** A packet does not reach its destination, because some resource does not grant access (while it grants access to other packets).
Definitions and Terminology

- **Switch**: The component of the network that is in charge of flit routing.

- **Flit Latency**: The time needed for a FLIT to reach its target PE from its source PE.

- **Packet Latency**: The time needed for a PACKET to reach its target PE from its source PE.

- **Packet Spread**: The time from the reception of the first flit of a packet to the reception of the last one.
Message Abstraction

- **Packet**: An element of information that a processing element (PE) sends to another PE. A packet may consist of a variable number of flits.”

- **Flit**: The elementary unit of information exchanged in the communication network in a clock cycle.

Message

Packet

Flit

Packet: An element of information that a processing element (PE) sends to another PE. A packet may consist of a variable number of flits.”

Flit: The elementary unit of information exchanged in the communication network in a clock cycle.
Switching Techniques

Two main modes of transporting flits in an 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
Switching Strategies

Virtual Circuit Switching

- creates virtual circuits that are multiplexed on links
- number of virtual links (or virtual channels (VCs)) that can be supported by a physical link depends on buffers allocated to link
- Possible to allocate either one buffer per virtual link or one buffer per physical link
  - Allocating one buffer per virtual link
    - depends on how virtual circuits are spatially distributed in the NoC, routers can have a different number of buffers
    - can be expensive due to the large number of shared buffers
    - multiplexing virtual circuits on a single link also requires scheduling at each router and link (end-to-end schedule)
    - conflicts between different schedules can make it difficult to achieve bandwidth and latency guarantees
Virtual Circuit Switching

- Allocating one buffer per physical link
  - virtual circuits are time multiplexed with a single buffer per link
  - uses time division multiplexing (TDM) to statically schedule the usage of links among virtual circuits
  - flits are typically buffered at the NIs and sent into the NoC according to the TDM schedule
  - global scheduling with TDM makes it easier to achieve end-to-end bandwidth and latency guarantees
  - less expensive router implementation, with fewer buffers
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 (Store and Forward) 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
Packet Switching

VCT (Virtual Cut Through) 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 the receiving buffer, no flits are sent, and the entire packet is buffered
- Same buffering requirements as SAF switching

WH (Wormhole) Switching
- Flit from a packet is forwarded to the 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
- Susceptible to deadlocks due to usage dependencies among links
Routing Algorithms

• Responsible for correctly and efficiently routing packets or circuits from the source to the destination
• Choice of a routing algorithm depends on trade-offs between several potentially conflicting metrics
  ▪ Minimizing power required for routing
  ▪ Minimizing logic & routing tables to achieve lower area footprint
  ▪ Increasing performance by reducing delay and maximizing traffic utilization of the network
  ▪ Improving robustness to better adapt to changing traffic needs
• Routing schemes can be classified into several categories
  ▪ Static or dynamic routing
  ▪ Distributed or source routing
  ▪ Minimal or non-minimal routing
Routing Algorithms

Static and Dynamic routing

- **Static Routing:** fixed paths are used to transfer data between a particular source and destination
  - does not take into account current state of the network
- **Advantages of static routing:**
  - easy to implement, since very little additional router logic is required
  - in-order packet delivery if single path is used
- **Dynamic Routing:** routing decisions are made according to the current state of the network
  - considering factors such as availability and load on links
- Path between source and destination may change over time
  - as traffic conditions and requirements of the application change
- More resources needed to monitor state of the network and dynamically change routing paths
- Able to better distribute traffic in a network
Routing Algorithms

Distributed and Source Routing

- Static and dynamic routing schemes can be further classified depending on where the routing information is stored, and where routing decisions are made.

- Distributed routing: each packet carries the destination address
  - e.g., XY co-ordinates or number identifying destination node/router
  - Routing decisions are made in each router by looking up the destination addresses in a routing table or by executing a hardware function.

- Source routing: packet carries routing information
  - Pre-computed routing tables are stored at a nodes’ NI
  - Routing information is looked up at the source NI and routing information is added to the header of the packet (increasing packet size)
  - When a packet arrives at a router, the routing information is extracted from the routing field in the packet header
  - Does not require a destination address in a packet, any intermediate routing tables, or functions needed to calculate the route.
Routing algorithms

Minimal and Non-minimal routing

- minimal routing: length of the routing path from the source to the destination is the shortest possible length between the two nodes
  - e.g. in a mesh NoC topology (where each node can be identified by its XY co-ordinates in the grid) if source node is at (0, 0) and destination node is at (i, j), then the minimal path length is $|i| + |j|$
  - source does not start sending a packet if minimal path is not available

- Non-minimal routing: can use longer paths if a minimal path is not available.
  - by allowing non-minimal paths, the number of alternative paths is increased, which can be useful for avoiding congestion
  - disadvantage: overhead of additional power consumption
Routing Algorithms

Routing algorithm must ensure freedom from deadlocks

- common in WH switching
- e.g. cyclic dependency shown below

- freedom from deadlocks can be ensured by allocating additional hardware resources or imposing restrictions on the routing
- usually dependency graph of the shared network resources is built and analyzed either statically or dynamically
Routing Algorithms

Routing Algorithm must ensure freedom from Livelocks

- Livelocks are similar to deadlocks, except that states of the resources involved constantly change with regard to one another, without making any progress
  - occurs especially when dynamic (adaptive) routing is used
  - e.g. can occur in a deflective “hot potato” routing if a packet is bounced around over and over again between routers and never reaches its destination
- Livelocks can be avoided with simple priority rules

Routing Algorithm must ensure freedom from starvation

- under scenarios where certain packets are prioritized during routing, some of the low priority packets never reach their intended destination
- can be avoided by using a fair routing algorithm, or reserving some bandwidth for low priority data packets
Flow Control Schemes

• Goal of flow control is to allocate network resources for packets traversing a NoC
  ▪ can also be viewed as a problem of resolving contention during packet traversal
• At the data link-layer level, when transmission errors occur, recovery from the error depends on the support provided by the flow control mechanism
  ▪ e.g. if a corrupted packet needs to be retransmitted, flow of packets from the sender must be stopped, and request signaling must be performed to reallocate buffer and bandwidth resources
• Most flow control techniques can manage link congestion
• But not all schemes can (by themselves) reallocate all the resources required for retransmission when errors occur
  ▪ either error correction or a scheme to handle reliable transfers must be implemented at a higher layer
Flow Control Schemes

STALL/GO

- Low overhead scheme
- Requires only two control wires
  - one going forward and signaling data availability
  - the other going backward and signaling either a condition of buffers filled (STALL) or of buffers free (GO)
- Implement with distributed buffering (pipelining) along link
- good performance – fast recovery from congestion
- does not have any provision for fault handling
  - higher level protocols responsible for handling flit interruption
Flow Control Schemes

T-Error

- More aggressive scheme that can detect faults
  - by making use of a second delayed clock at every buffer stage
- Delayed clock re-samples input data to detect any inconsistencies
  - then emits a VALID control signal
- Re-synchronization stage added between end of link and receiving switch
  - to handle offset between original and delayed clocks
- Timing budget can be used to provide greater reliability by configuring links with appropriate spacing and frequency
- Does not provide a thorough fault handling mechanism
Flow Control Schemes

**ACK/NACK**

- When flits are sent on a link, a local copy is kept in a buffer by sender.
- When ACK received by sender, it deletes copy of flit from its buffer.
- When NACK is received, sender rewinds its output queue and starts resending flits, starting from the corrupted one.
- Implemented either end-to-end or switch-to-switch.
- Sender needs to have a buffer of size $2N + k$.
  - $N$ is number of buffers encountered between source and destination.
  - $k$ depends on latency of logic at the sender and receiver.
- Overall a minimum of $3N + k$ buffers are required.
- Fault handling support comes at cost of greater power, area overhead.
Flow Control Schemes

ACK/NACK

[Diagram showing flow control schemes with routers and acknowledgments]
Flow Control Schemes

Network and Transport-Layer Flow Control

- Flow Control without Resource Reservation
  - Technique #1: drop packets when receiver NI full
    - improves congestion in short term but increases it in long term
  - Technique #2: return packets that do not fit into receiver buffers to sender
    - to avoid deadlock, rejected packets must be accepted by sender
  - Technique #3: deflection routing
    - when packet cannot be accepted at receiver, it is sent back into network
    - packet does not go back to sender, but keeps hopping from router to router till
      it is accepted at receiver

- Flow Control with Resource Reservation
  - credit-based flow control with resource reservation
  - credit counter at sender NI tracks free space available in receiver NI
    buffers
  - credit packets can piggyback on response packets
  - end-to-end or link-to-link
Switching Techniques

Packet Switching – Routing Protocols

**Store and Forward:** Router cost is packet based. Packet size also affects latency and buffering requirements. Stalling happens at two nodes and the link between them.

**Wormhole:** Router cost is based on header. Header can effect latency and buffering at the router is based on the header size. Stalling can happen at all the nodes and links spanned by the packet..

**Virtual Cut-through:** Router cost depends on header and packet size. Stalling at local nodes level.
VCT and Wormhole Routing

**Wormhole**

- Source
- Moving empty slots downstream
- Stalled (NO empty slots downstream)
- Buffer Full
- Packet not consumed
- Sink

**Virtual Cut Through**

- Source
- Moving empty slots downstream
- Moving empty slots downstream
- Free Slots for entire packet
- Packet not consumed
- Sink
Relevant Parameters: Routing

- Minimum latency is of paramount importance in NOCs (inter-process communication).
- Ideally: One clock latency per switch/router (flit enters at time $t$ and exits at $t+1$)
- Maximum switch clock frequency (technology + routing logic limits)
- Deadlock free
- No flits are ever lost; once a flit is injected in the NOC, it must reach to its destination - may be after a long time.
Fixed Shortest Path Routing

Suitable for Regular Topologies
e.g. Mesh, Torus, Tree, etc.

X-Y routing (first x then y direction.

- Simple Router
- No deadlock scenario
- No retransmission
- No reordering of messages
- Power-efficient
Wormhole Routing

- In wormhole routing a header flit “digs” the path and hold.
- Successive flits are routed to the same path or direction
- In case of blocks and loss-less NoC we need:
  - Buffers
  - A back-pressure mechanism if we don’t have infinite size FIFOs…
Wormhole

Src

Dest
Wormhole
Wormhole

TF F4F3F2

Dest
Wormhole
Wormhole

Src

F4

T

F

F3

Dest

F2

HF


Wormhole
Wormhole

![Diagram of Wormhole]

- **Src**
- **F1**
- **F2**
- **F3**
- **F4**
- **Dest**
- **HF**
- **T**

**NOC and SOC Design**
Wormhole
Wormhole
Wormhole

Src

TF

F4
F3
F2
HF
Wormhole
Deflection Routing

**Hot Potato – Deadlock Free Routing**

Every flit can be routed to different directions (no packet notion at the switch level)

- *If the optimal direction is blocked, the flit is “deflected” to another direction*
  - Switch latency of 1 clock cycle whatever the level of congestion
  - Minimum buffer requirements

<table>
<thead>
<tr>
<th>Packets reordering</th>
<th>Adaptive routing</th>
<th>No buffering</th>
<th>No back pressure</th>
<th>Works with Torus/Mesh</th>
</tr>
</thead>
</table>

**Wormhole Routing**

- No packets reordering
- Static routing
- Buffering (≥ 2 flits/port)
- Back pressure
- XY routing needs mesh
Hot-Potato

Src

Dest
Hot-Potato
Hot-Potato

NOC and SOC Design
Hot-Potato

Src

<table>
<thead>
<tr>
<th>T</th>
<th>F</th>
</tr>
</thead>
</table>

F3

Dest

<table>
<thead>
<tr>
<th>F2</th>
</tr>
</thead>
</table>

HF
Hot-Potato

Src -> T -> Dest
F3
F2
HF
Hot-Potato

Src

TF

F3

F2

Dest

H

F
Hot-Potato

Src

Dest

F3

TF

HF

F2
Hot-Potato

Diagram showing flow from Src to Dest with intermediate nodes labeled as F2, HF, TF, F3.
Network-on-Chip

Core

Network Adapter

Routing Node

Link
Core to Network Connection
NOC Switch/Router

Generic Router/Switch

Injection Channel

Ejection Channel

Switch

Routing and Arbitration

Input Channels

Output Channels
VC: Virtual-Channels
A Router Structure

- Flits stored in input ports
- Output port schedules transmission of pending flits according to:
  - Priority (*Service Level*)
  - Buffer space in next router
  - Round-Robin on input ports of same SL
  - Preempt lower priority packets
Virtual Channel 2D Router

VCID (Direction Vector)

From West (W)

Smart Demux

For North-East (NE)

Path Set

For South-East (SE)

W_NE

S_NE

PE_NE

N_SE

W_SE

PE_SE

Mux

Scheduling

Arbiter (VA/SA)

Pre-selection Function (Congestion-Look-Ahead)

For NE

For SE

For SW

For NW

*Decomposed Crossbar (4x4)

N

S

E

W

Credit, Status of Pending Msg Queues

Output VC Resv_State

Pre-selection enable signals
A Typical Router Pipeline

- Routing & Buffers
- VC Allocation
- Arbitration
- Switch Traversal
CAD Problems for NOC

- Application Mapping *(map tasks to cores)*
- Floorplanning/Placement *(within the network)*
- Routing *(of messages)*
- Buffer Sizing *(size of FIFO queues in the routers)*
- Timing Closure *(Link bandwidth capacity allocation)*
- Simulation *(Network simulation for traffic, delay, power modeling)*
- Testing … Combined with problems of designing NOC itself *(topology synthesis, switching, virtual channels, arbitration, flow control,……)*
Topology Generation and Analysis

• **Aim:**
  - Generate a viable network topology.
  - Analyze the generated topology.

• **Targeted Network:**
  - Best-effort, wormhole switched.
  - Lookup table based source routing.
  - No virtual channel support.
  - Round Robin switch output arbitration.
  - One NI per component master or slave interface.
  - All transactions converted to packets of the same length (flit count).
  - Burst beats converted to separate packets.
System Input and Output

- Input:
  - Core Graph
  - Network Parameters

- Output:
  - Topology Graph
  - Route tables
  - Recommended Operating Clock Frequency
Partitioned Crossbar Topologies

• Initial topology: Fully-Connected Crossbar (single switch).

• Ideal latency situation.

• May violate maximum port requirement.

• Partitioning process.
Frequency Selection

- Cyclical relation between contention and frequency.
- Frequency is fixed before contention is analyzed.
- To find minimum valid frequency:
  - Interval halving process.
  - Large number of frequency points.
Results

• Applications and generated topologies.
• Comparative results.
• Resource Usage.
• Accuracy tests.
MPEG4 - Decoder

Clock Frequency: 3.43 GHz
MWD Application

Clock Frequency:
573.4 MHz
AV Benchmark
Comparative Results
Resource Usage

<table>
<thead>
<tr>
<th>Topology</th>
<th>Mesh</th>
<th>Fat Tree</th>
<th>Custom 1</th>
<th>Custom 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG4 Decoder</td>
<td>46</td>
<td>44</td>
<td>22</td>
<td>14</td>
</tr>
<tr>
<td>MWD Application</td>
<td>59</td>
<td>47</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>Av Benchmark</td>
<td>87</td>
<td></td>
<td>67</td>
<td>25</td>
</tr>
</tbody>
</table>
Accuracy Test Results

MPEG4 Transaction Results, 3.43 GHz

MWDO Transaction Results, 573.4 MHz