

# Parallel Processing Interconnect Architectures

*“Topologies”*

[Joseph Wunderlich PhD](#)

For Processors and Cores

- As well as Computer Networks

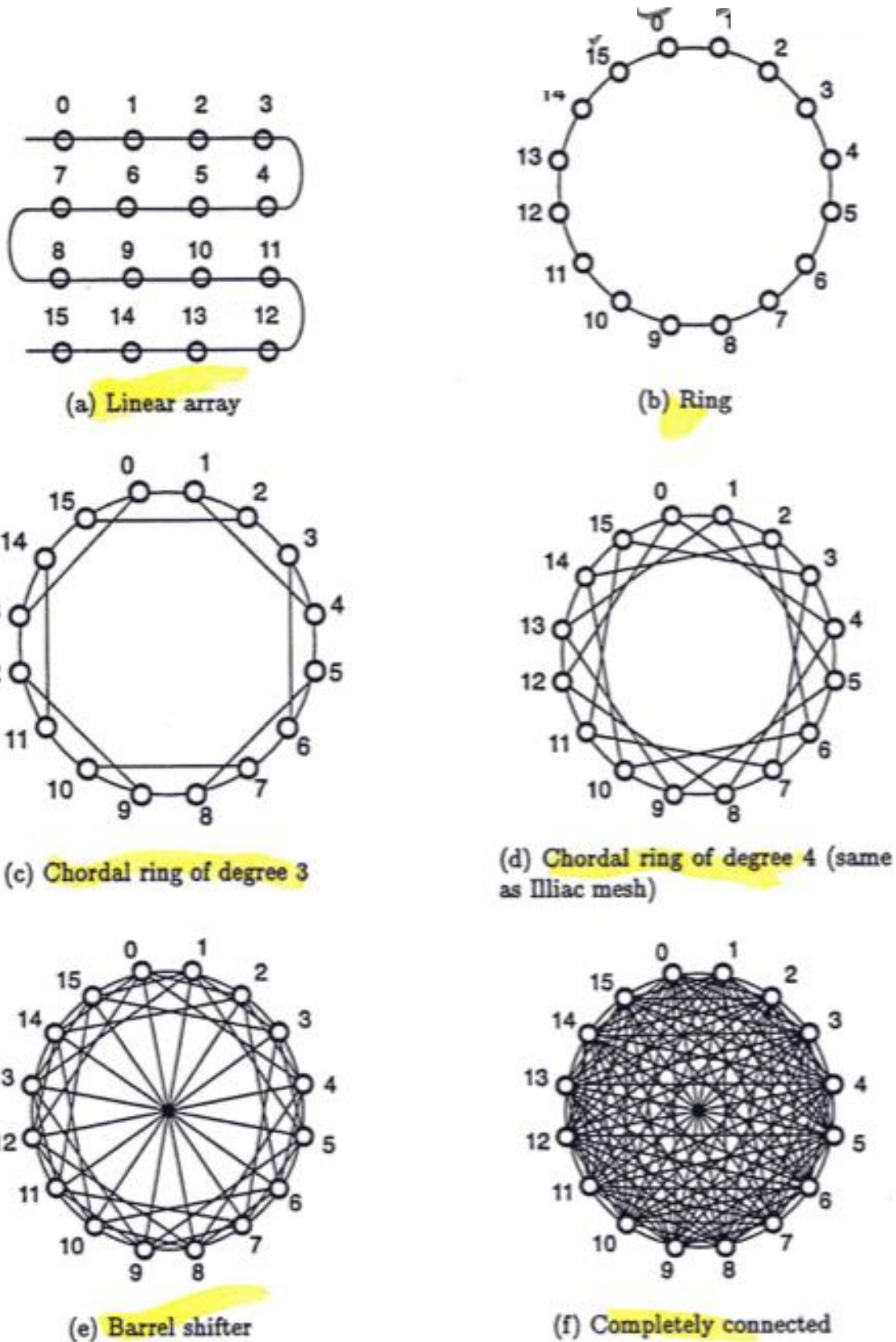


# Parallel Processing Interconnect Architectures

- Static
  - Architectures
  - Network Characteristics
  - Scalability
- Dynamic
  - Architectures
  - Scalability



# STATIC INTERCONNECT ARCHITECTURES



**Figure 2.16** Linear array, ring, chordal rings of degrees 3 and 4, barrel shifter, and completely connected network.



# STATIC INTERCONNECT ARCHITECTURES

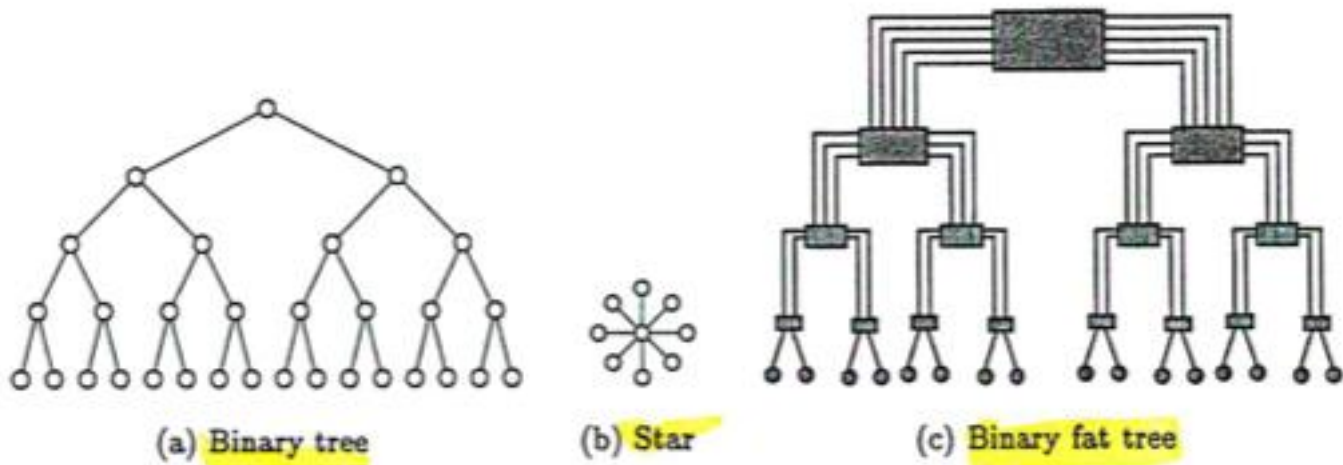
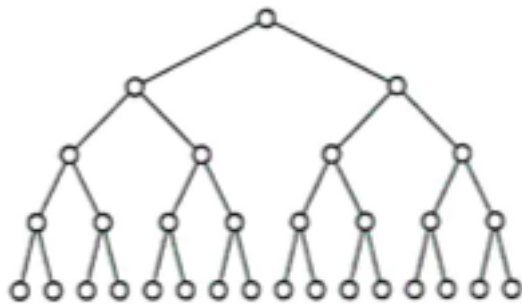


Figure 2.17 Tree, star, and fat tree.



# STATIC INTERCONNECT ARCHITECTURES

J. Wunderlich Neural Network Processor 1992



(a) Binary tree

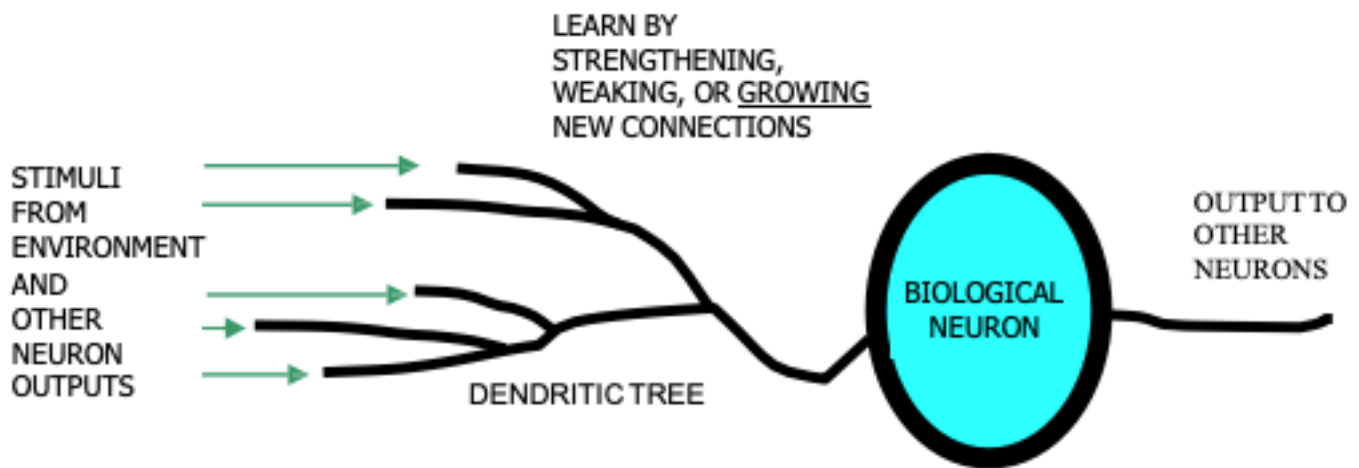
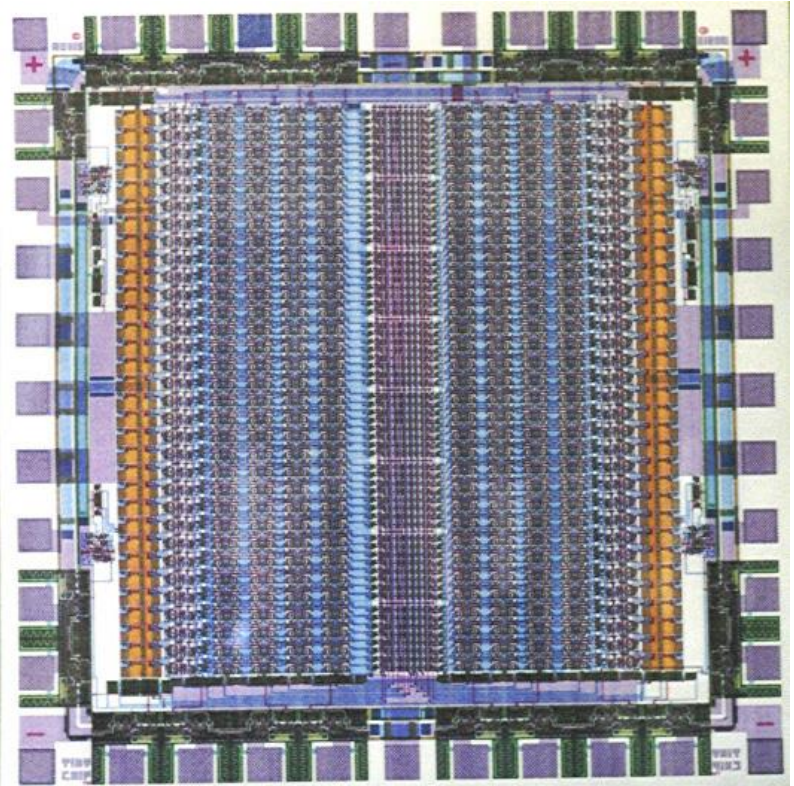


Figure 1. Biological neuron for bottom-up neurocomputer design.

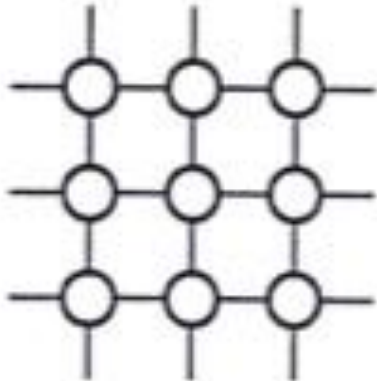
**Hardware for Machine Learning**  
**Two Single-Chip Neurocomputer Designs; One Bottom-Up, One Top-Down**  
**DRAFT BOOK CHAPTER, *Joseph T Wunderlich PhD***

[PPTX-w/audio](#) [PDF](#) [MP4](#) [YouTube](#)

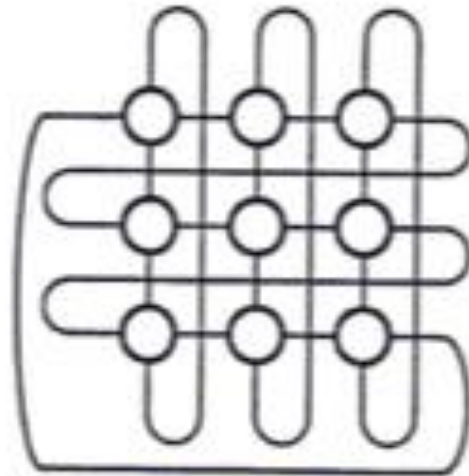




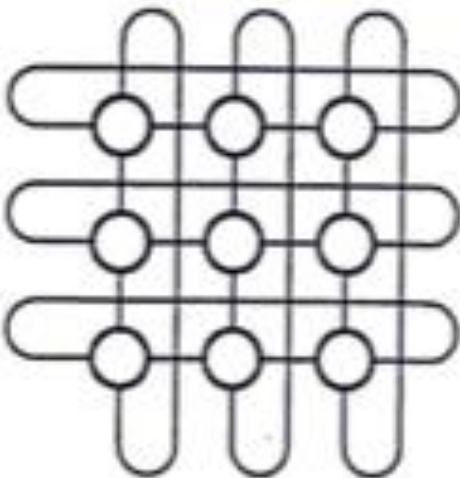
# STATIC INTERCONNECT ARCHITECTURES



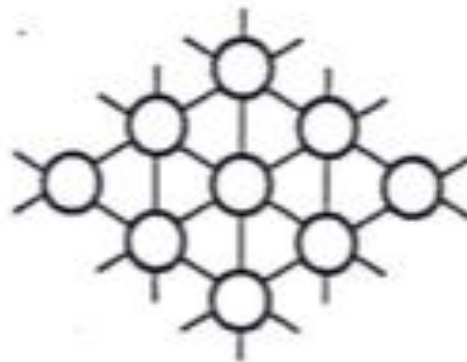
(a) Mesh



(b) Illiac mesh



(c) Torus

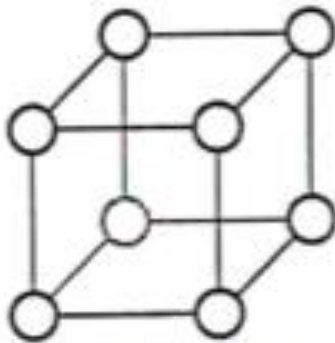


(d) Systolic array

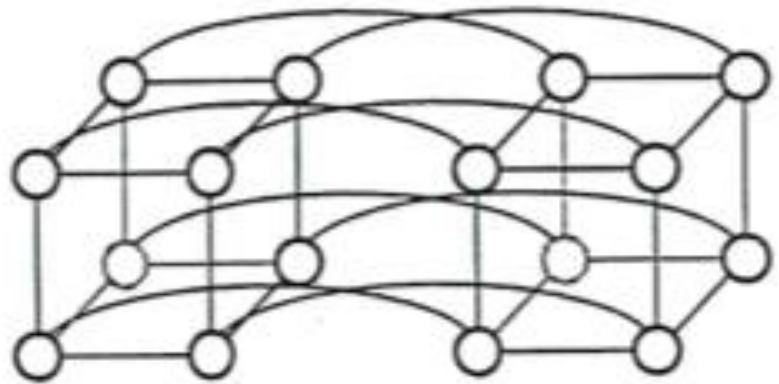
Figure 2.18 Mesh, Illiac mesh, torus, and systolic array.



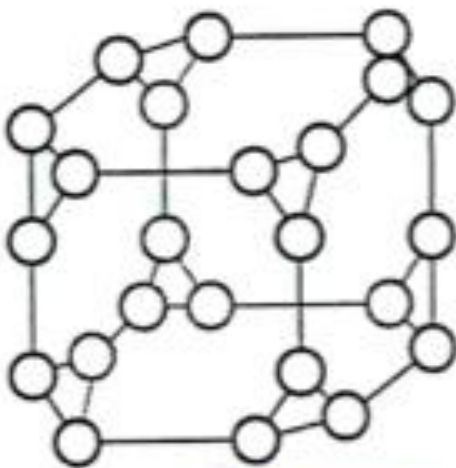
# STATIC INTERCONNECT ARCHITECTURES



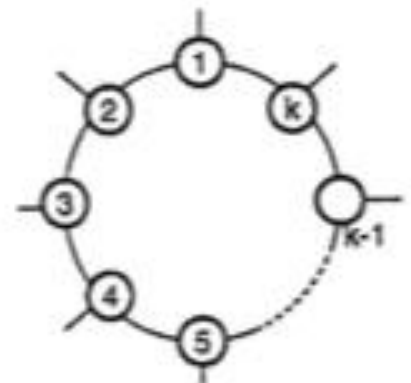
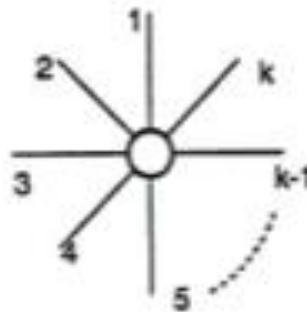
(a) 3-cube



(b) A 4-cube formed by interconnecting two 3-cubes



(c) 3-cube-connected cycles



(d) Replacing each node of a  $k$ -cube by a ring (cycle) of  $k$  nodes to form the  $k$ -cube-connected cycles

Figure 2.19 Hypercubes and cube-connected cycles.



# STATIC INTERCONNECT ARCHITECTURES

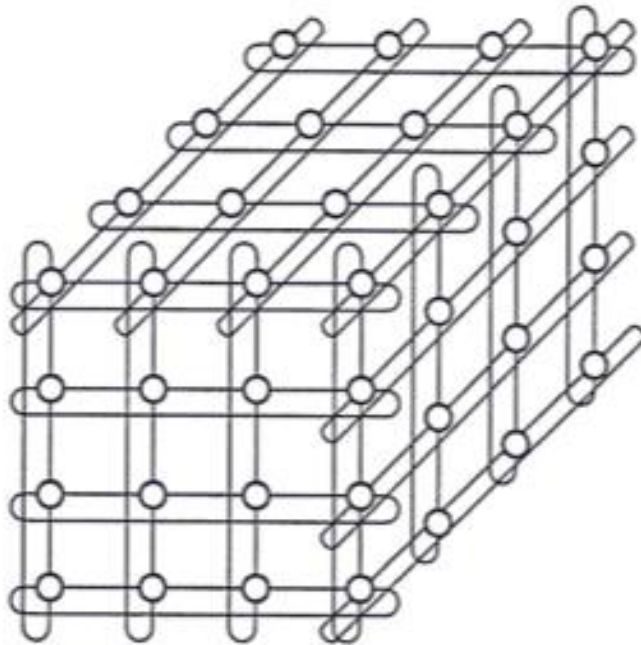


Figure 2.20 The  $k$ -ary  $n$ -cube network shown with  $k = 4$  and  $n = 3$ ; hidden nodes or connections are not shown.

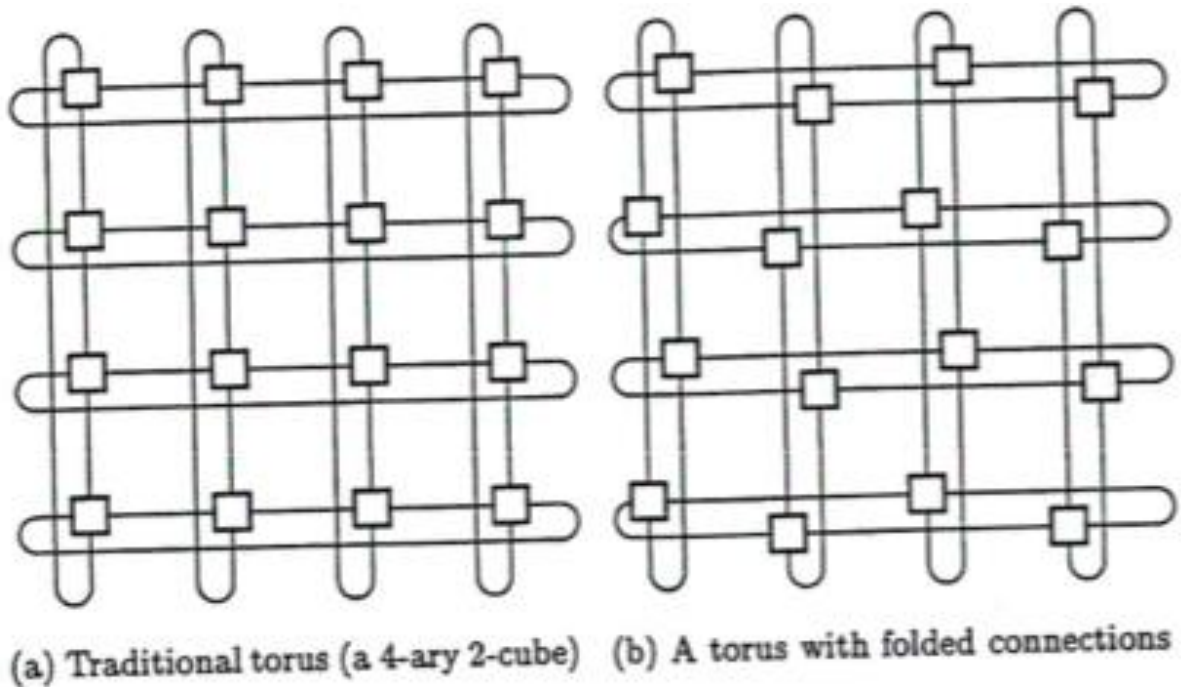
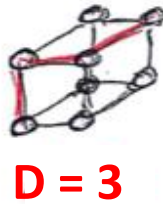


Figure 2.21 Folded connections to equalize the wire length in a torus network. (Courtesy of W. Dally; reprinted with permission from *IEEE Trans. Computers*, June 1990)



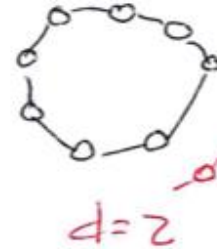
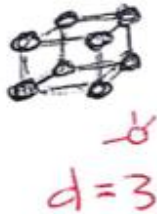
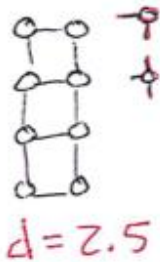
# STATIC INTERCONNECT ARCHITECTURES CHARACTERISTICS

$N$  = # OF NODES  
 $\star$  OPTIMIZE FOR APPLICATION AND TYPICAL PROBLEM GRAIN SIZE  
 WIRE LENGTH = "CHANNEL" LENGTH = "LINK" LENGTH = "GRAPH EDGE" LENGTH  
 $\star$  TRY TO MINIMIZE  
 $D$  = NETWORK DIAMETER (SHORTEST DISTANCE TO FURTHEST NODE)



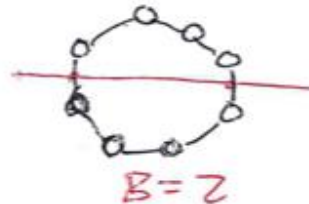
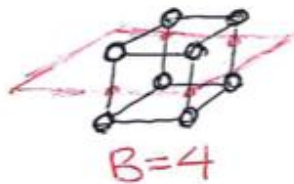
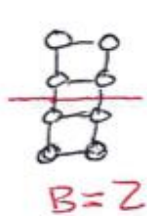
$\star$  MINIMIZE  $D$   
 TO REDUCE COMMUNICATION DELAYS

$d$  = DEGREE (AVE # OF CONNECTION-CHANNELS INTO NODES)



$\star$  A CONSTANT  $d$  IS GOOD FOR MODULARITY AND SCALABILITY

$B$  = BISECTION WIDTH (NUMBER OF LINKS THROUGH SMALLEST CROSS-SECTION)



$\star$  MINIMIZE  $B$  TO MINIMIZE REQUIRED WIRING  
 $\star$  MAXIMIZE  $B$  TO MAX NETWORK COMM BANDWIDTH  
 $\star$  AS  $B \uparrow, D \downarrow$  BUT  $d \uparrow$

$W$  = # OF WIRES IN ONE "CHANNEL", "LINK", "EDGE"

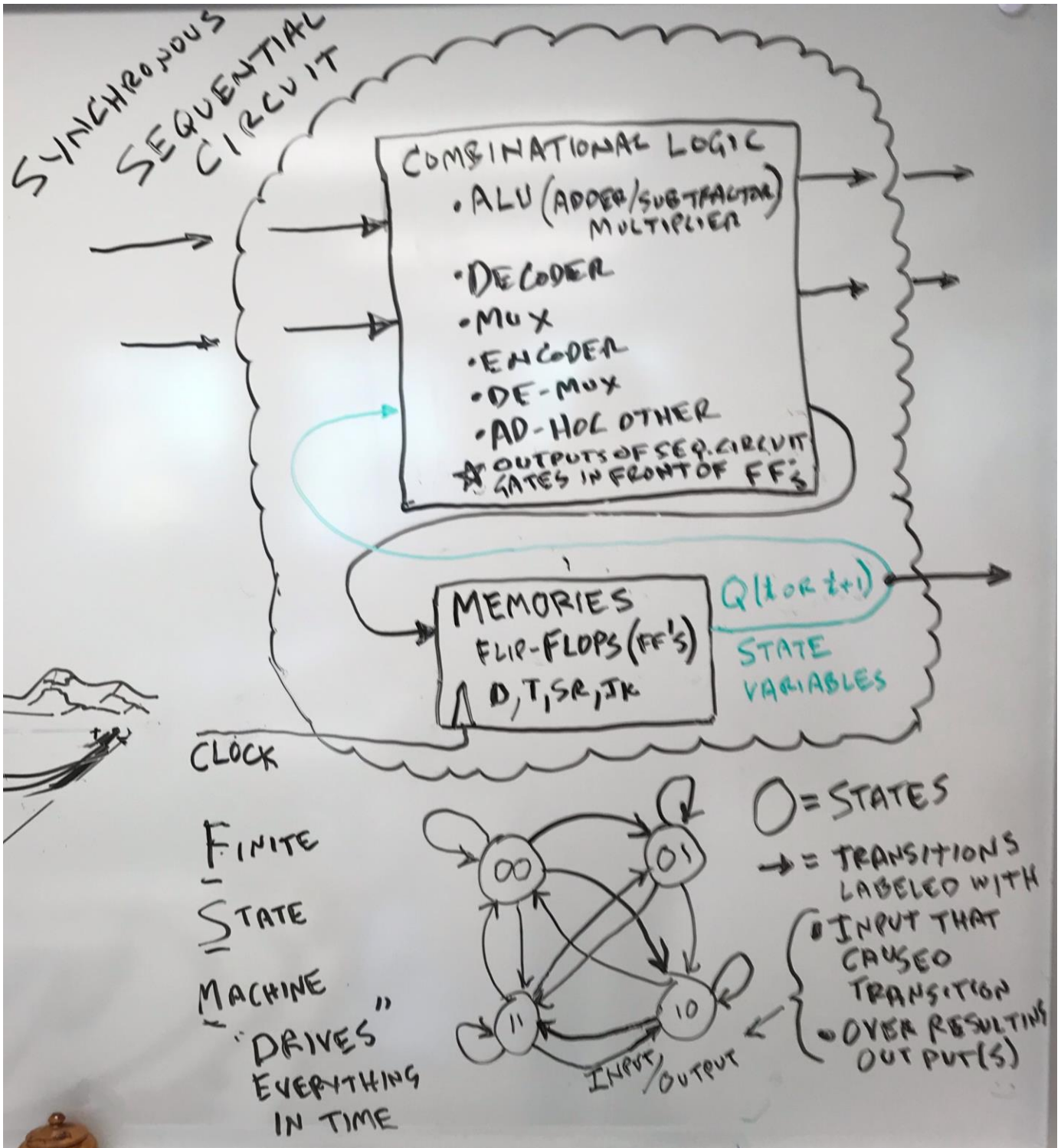


# STATIC INTERCONNECT ARCHITECTURES

## CHARACTERISTICS

$W =$  # OF WIRES  
IN ONE  
"CHANNEL", "LINK", "EDGE"

Recall PARALLEL (Combinational)  
and SERIAL (Sequential)  
Digital Design



# STATIC INTERCONNECT ARCHITECTURES

## PARALLEL (Combinational) Digital Design For COMMUNICATION with external device via INTERRUPTS

**ENCODER**  
EX: 4x2 PRIORITY ENCODER

W, X, Y, Z → VALID → A, B

W	X	Y	Z	A	B	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

OUTPUT CODE

X = DON'T CARE INPUT (REALLY!) (DIFFERENT FROM DON'T CARE OUTPUTS)

W HAS HIGHEST PRIORITY, SO WHEN W=1, AB=11 INDICATING PAY ATTENTION TO W REGARDLESS OF VALUES OF X, Y, OR Z

USEFUL MAPS

EXAMPLE USE:  
COMPUTER HARDWARE AND SYSTEM'S LEVEL SOFTWARE CONTROLLING HOW "INTERUPTS" GENERATED BY EXTERNAL DEVICES ARE HANDELED

Dr. Joseph T Wunderlich  
Edutopia  
Marquette College



# COMMUNICATION with external devices via INTERRUPTS

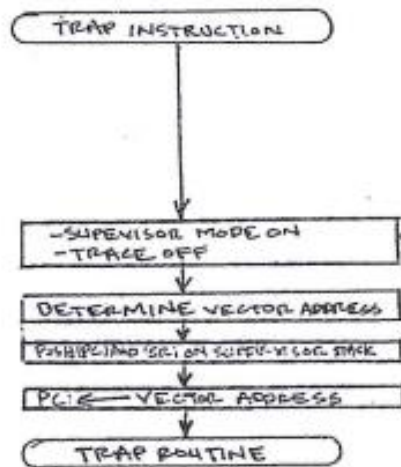
Software "TRAP"

vs.

Hardware "INTERUPT"

- \* OCCUR SYNCHRONOUS WITH NORMAL PROGRAM EXECUTION
- \* SOFTWARE GENERATED AND ARE CAUSED BY EITHER: (a) PROGRAM CONTROL INSTRUCTIONS (b) INSTRUCTIONS TO CHECK PROGRAM CONDITIONS
- \* DESIGNED TO ANTICIPATE AND "TRAP" ERRORS IN EXECUTION

\* TRAP PROCESSING



\* EXAMPLES:

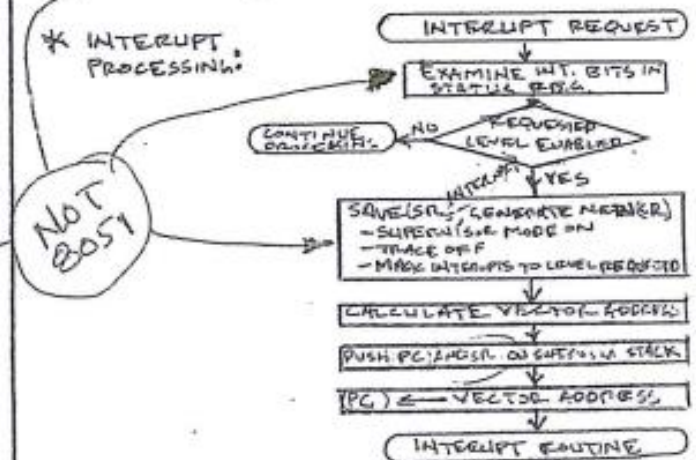
- ATTEMPT TO DIVIDE BY ZERO
- ILLEGAL OR ~~MODE MISMATCHING~~ INSTRUCTION (68000)
- SUPERVISOR MODE PRIVILEGED INSTRUCTION (ATTEMPTED BY USER MODE)
- IN 8051, JUST BY SETTING FLAGS (eg., IN SCON OR TCON)

\* MAY OCCUR AT ANY TIME (ASYNCHRONOUS WITH NORMAL PROGRAM EXECUTION)

- \* HARDWARE GENERATED
- \* PROCESSED ACCORDING TO A PRIORITY SCHEME WHICH IS CALCULATED BY THE PROCESSOR AND/OR BY CIRCUITS EXTERNAL TO THE PROCESSOR.

THE MC68000 CALCULATES 7 PRIORITY LEVELS INTERNALLY (AUTOVECTORS) VIA 3 BITS IN THE STATUS REGISTER; THE PROCESSOR WILL ALSO RESPOND TO 192 OTHER PRIORITY LEVELS GENERATED BY CIRCUITS EXTERNAL TO CPU (USER INTERRUPT VECTORS).

\* INTERRUPT PROCESSING



\* EXAMPLES:

- FACILITY ERROR SIGNAL FROM FACILITY CHECKER CIRCUITS
- WATCH DOG TIMER CIRCUIT (RESPONSE TIME EXCEEDED)
- POWER OUTAGE

HIERARCHY

- 8051 TIMER FLAG INTERRUPT (WHEN TIMER OVERFLOWS)
- 8051 SERIAL PORT INTERRUPT (WHEN DATA BYTE RECEIVED)
- EXTERNAL INTERRUPTS BY DEVICES (AT PINS INT0 AND INT1) <sup>IT'S</sup> <sub>WARNING</sub>
- RESET AT "RESET" PIN <sub>SERVICE</sub>

# STATIC INTERCONNECT ARCHITECTURES CHARACTERISTICS

## SERIAL (Sequential) Digital Design

for IPC (Inter-Processor Communication) BETWEEN PROCESSORS/CORES

$W =$  # OF WIRES IN ONE "CHANNEL", "LINK", "EDGE"

From Prerequisite [EGR/CS332 Digital Design I](#) (before Spring 2022) or [EGR330 Digital & Embedded Systems0](#) (after Spring 2022)

**#1 DEFINE:** DESIGN A CIRCUIT TO DETECT AN INPUT SEQUENCE OF 3 ONES ALSO, DETECT SELF-CORRECTION. *COMPARE ALL FF OUTPUTS*  
 (ASSUME EACH UNUSED STATE TRANSITION DIRECTLY TO A USED STATE THEREAFTER)  
 \* ASSUME INPUT BIT STREAM (X) IS SYNCHRONIZED WITH CLOCK AND USE ONLY POSITIVE-EDGE TRIGGERED FLIP-FLOPS  
 Z=1 WHEN SELF-CORRECTION OCCURS

**2 STATE DIAGRAM:**

**3 STATE TABLE:**

Q(t)	Q(t+1)
AB X	AB Y Z
00	00
01	01
10	10
11	11

**4 FLIP-FLOP INPUTS:**

JK	SR	D	T
JK	SR	D	T
00	00	0	0
01	01	0	1
10	10	1	0
11	11	1	1

**5 EXCITATION TABLES (FOR DESIGN):**

Q(t)Q(t+1)	J	K
00	0	X
01	1	X
10	X	0
11	X	0

Q(t)Q(t+1)	S	R
00	0	X
01	1	X
10	0	0
11	X	0

Q(t)Q(t+1)	D
00	0
01	1
10	1
11	1

Q(t)Q(t+1)	T
00	0
01	1
10	0
11	0

**6 SIMPLIFY OUTPUTS AND FLIP-FLOP INPUTS:**

Y OUTPUT IS SAME REGARDLESS OF FLIP-FLOP TYPE USED:  
 $Y = ABX$

Z OUTPUT IS ALSO SAME REGARDLESS OF FF TYPE:  
 $Z = AB$

FLIP-FLOP INPUTS:  
 $J_A = BX$ ,  $S_A = BX$ ,  $D_A = BX + AX$ ,  $T_A = AX + BX$   
 $K_A = X'$ ,  $R_A = X'$   
 $J_B = AX$ ,  $S_B = ABX$ ,  $D_B = ABX$ ,  $T_B = B + AX$   
 $K_B = 1$ ,  $R_B = B$

**7 LOGIC CIRCUIT IMPLEMENTATIONS:**

**8 SKIP: Converting to NANDS**

**9 ANALYZE CIRCUIT TO SEE WHERE UNUSED STATE GOES (SHORT-CUTS JUST USE FF J/K/T EQUATIONS)**

UNUSED STATE	FLIP-FLOP INPUTS AT TIME t	UNUSED STATE AT TIME t+1
00	JK: 00, SR: 00, D: 0, T: 0	00
01	JK: 10, SR: 01, D: 0, T: 1	01
10	JK: 01, SR: 10, D: 1, T: 0	10
11	JK: 11, SR: 11, D: 1, T: 1	11

**10 DRAW COMPLETE STATE DIAGRAMS**

**11**

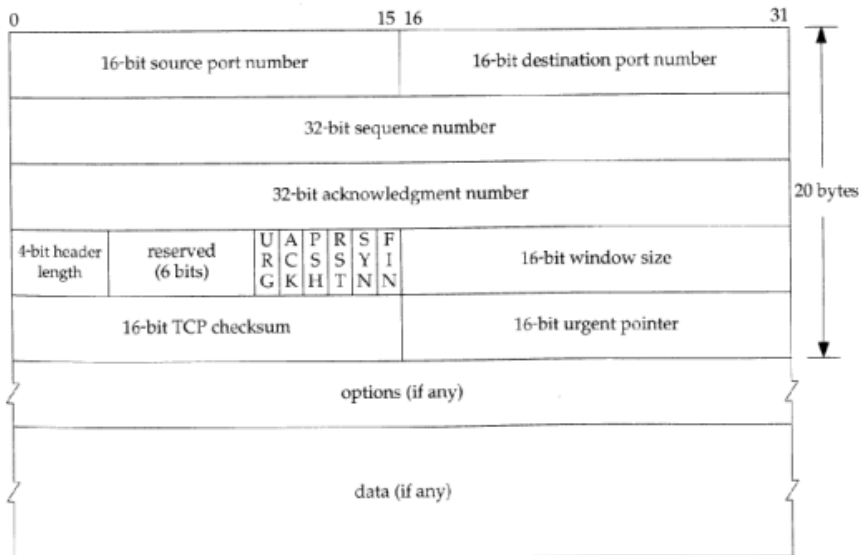
W = # OF WIRES  
IN ONE  
"CHANNEL", "LINK", "EDGE"

# STATIC INTERCONNECT ARCHITECTURES

## CHARACTERISTICS

### SERIAL IPC (Inter-Processor Communication) BETWEEN PROCESSORS/CORES

#### Sometimes PACKETIZED



**SrcPort** and **DstPort** fields identify source and destination ports. These plus source and destination IP addresses combine to identify each TCP connection.

**sequence number** identifies byte in data stream from sending TCP to receiving TCP that the first byte of data in this segment represents.

**acknowledgement number** is next sequence number that sender of acknowledgement expects to receive. i.e., sequence number plus 1 of last successfully received byte of data. This field is valid only if ACK flag is on. Once a connection is established Ack flag is always on.

**Acknowledgement, SequenceNum, and AdvertisedWindow** involved in TCP's sliding window algorithm. The Acknowledgement and AdvertisedWindow field carry info about flow of data going in other direction. In TCP's sliding window algorithm receiver advertises a window size to sender using the AdvertisedWindow field. The sender is then limited to having no more than a value of AdvertisedWindow bytes of unacknowledged data at any given time. The receiver sets a suitable value for the AdvertisedWindow based on the amount of memory allocated to the connection for the purpose of buffering data.

**header length** (in 32-bit words) Required because length of options field is variable.

**6-bit Flags field** used to relay control info between TCP peers. SYN and Fin flags for establishing and terminating a TCP connection, ACK flag is set any time Acknowledgement field is valid, implying that the receiver should pay attention to it. URG flag signifies this segment contains urgent data. When set, UrgPtr indicates where non-urgent data in this segment begins. PUSH flag signifies sender invoked push operation, which indicates to receiving side of TCP that it should notify the receiving process of this. RESET flag signifies receiver has become confused and so wants to abort connection.

**Checksum** (FOR ERROR DETECTION) is a mandatory field calculated by sender, then verified by receiver.

**Option field** is maximum segment size option, called MSS. Each end of connection normally specifies this option on first segment exchanged. It specifies maximum sized segment sender wants to receive.

**Data** portion of TCP segment (optional, but it's the actual data you are most likely trying to send!) i.e., everything else is communication overhead !!

**NOTE: PACKETIZED SERIAL COMMUNICATION between Systems also in Robotics**

SOURCE: : Crouse, J. (2008). The Joint Architecture for Unmanned Systems (JAUS): a subsystem of the wunderbot 4. Elizabethtown College research report.

Wunderbot 4 Wireless JAUS Communication is like TCP/IP, but different. Used for military comm

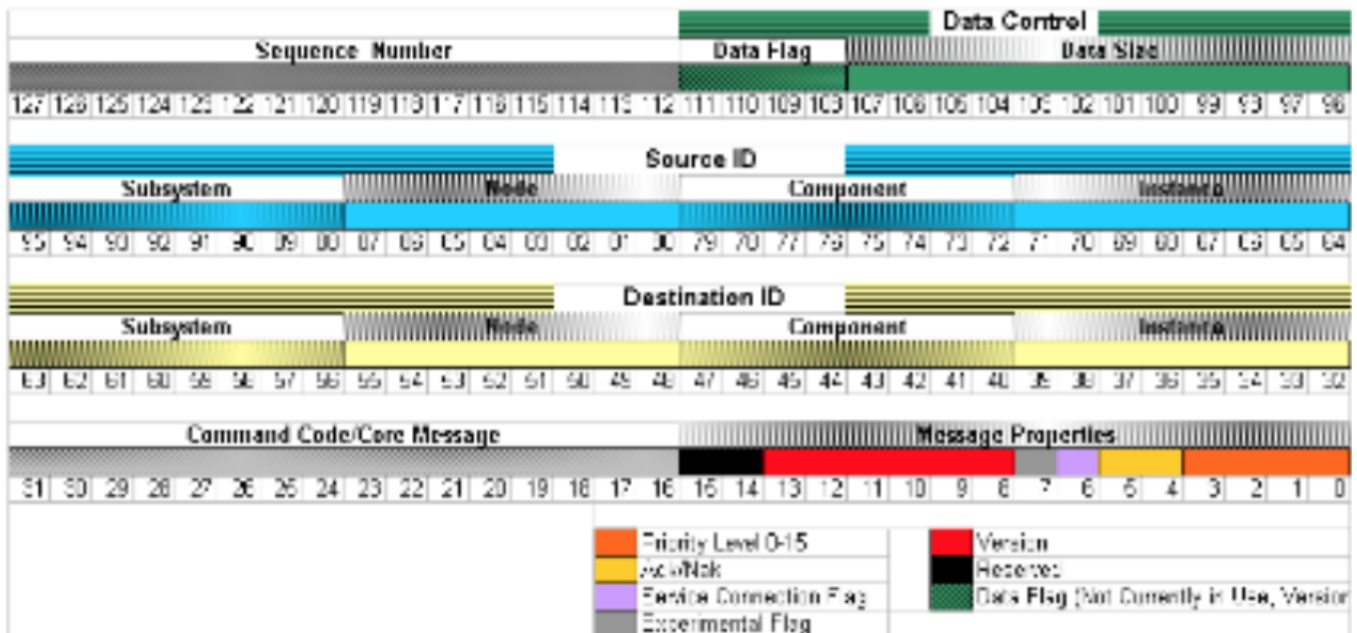
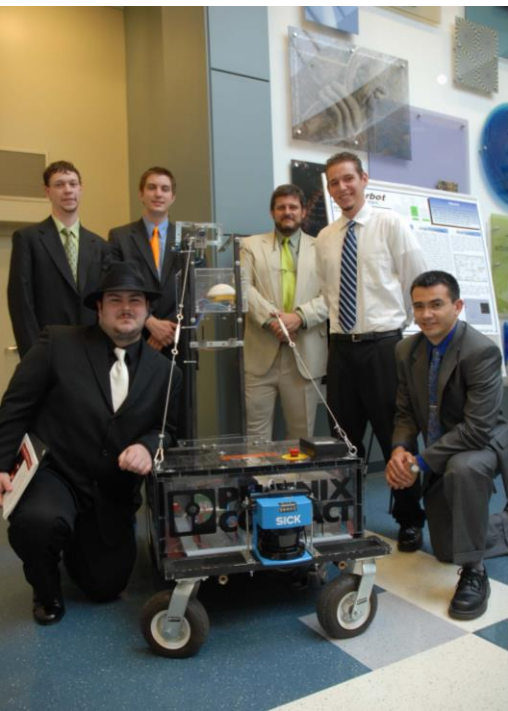


Figure 7: JAUS message header detailed structure [6]




**For communication with other autonomous ground vehicles, like military tanks on Battlefield**



## NOTE: PACKETIZED SERIAL COMMUNICATION between Systems also in Robotics

### Navigation

Implementation and integration of the most recent Wunderbot systems:

 [Wunderbot - Main VI Labview Tutorial](#)  
[Wunderbot - GPS Subsystem Labview Tutorial](#)  
[Wunderbot - LADAR Subsystem Labview Tutorial](#)  
[Wunderbot - JAUS Subsystem Labview Tutorial](#)  
[Wunderbot - Vision Subsystem Labview Tutorial](#)  
[Wunderbot - Motor Control Subsystem Labview Tutorial](#)  
[Wunderbot - Digital Compass Subsystem Labview Tutorial](#)  
[Wunderbot - MCglobal08 Subsystem Labview Tutorial](#)  
[nanoLC Robot Simulation](#)

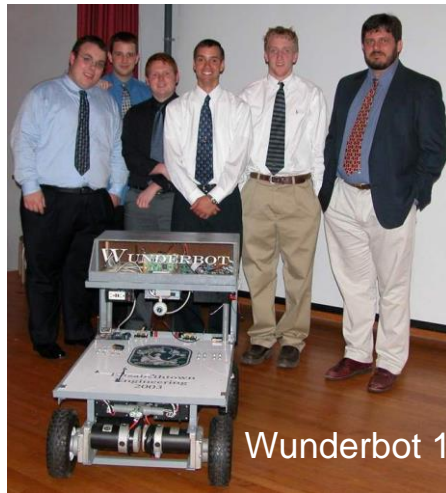
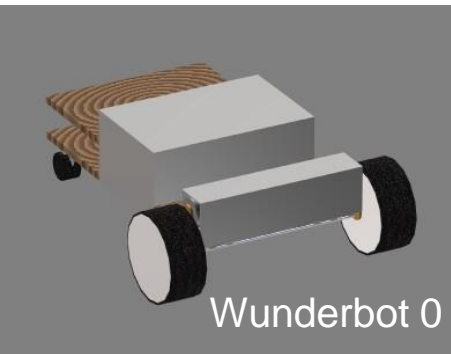


And theory and design decisions here:

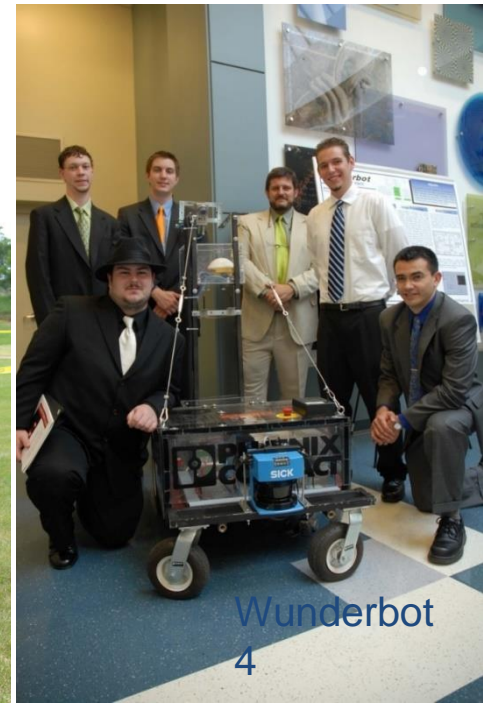
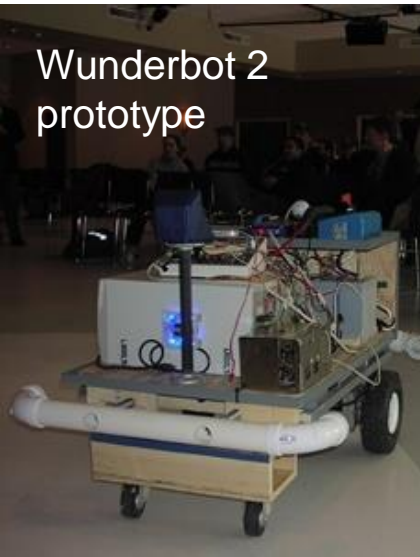
- [1] Painter, J. and Wunderlich, J.T. (2008). [Wunderbot IV: autonomous robot for international competition](#). In *Proceedings of the 12th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2008, Orlando, FL*: (pp. 62-67). And [HERE](#)
- [2] Coleman, D. and Wunderlich, J.T. (2008). [O<sup>3</sup>: an optimal and opportunistic path planner \(with obstacle avoidance\) using voronoi polygons](#). In *Proceedings of IEEE the 10th international Workshop on Advanced Motion Control, Trento, Italy*. vol. 1, (pp. 371-376). IEEE Press.
- [3] [JAUS wireless packetized communication by Jeremy Crouse](#)



**NOTE: PACKETIZED SERIAL COMMUNICATION between Systems  
also in Robotics**



All recent Wunderbots  
share website:  
<http://www2.etown.edu/wunderbot/>



See evolution 1999:

[http://users.etown.edu/w/wunderjt/home\\_researchMINE.html](http://users.etown.edu/w/wunderjt/home_researchMINE.html)

[http://users.etown.edu/w/wunderjt/Weblab\\_archive.htm](http://users.etown.edu/w/wunderjt/Weblab_archive.htm)



# STATIC INTERCONNECT ARCHITECTURES

## SCALABILITY

Table 2.2 Summary of Static Network Characteristics

Network type	Node degree, $d$	Network diameter, $D$	No. of links, $l$	Bisection width, $B$	Symmetry	Remarks on network size
Linear Array	2	$N - 1$	$N - 1$	1	No	$N$ nodes
Ring	2	$\lfloor N/2 \rfloor$	$N$	2	Yes	$N$ nodes
Completely Connected	$N - 1$	1	$N(N - 1)/2$	$(N/2)^2$	Yes	$N$ nodes
Binary Tree	3	$2(h - 1)$	$N - 1$	1	No	Tree height $h = \lceil \log_2 N \rceil$
Star	$N - 1$	2	$N - 1$	$\lfloor N/2 \rfloor$	No	$N$ nodes
2D-Mesh	4	$2(r - 1)$	$2N - 2r$	$r$	No	$r \times r$ mesh where $r = \sqrt{N}$
Illiac Mesh	4	$r - 1$	$2N$	$2r$	No	Equivalent to a chordal ring of $r = \sqrt{N}$
2D-Torus	4	$2\lfloor r/2 \rfloor$	$2N$	$2r$	Yes	$r \times r$ torus where $r = \sqrt{N}$
Hypercube	$n$	$n$	$nN/2$	$N/2$	Yes	$N$ nodes, $n = \log_2 N$ (dimension)
CCC	3	$2k - 1 + \lfloor k/2 \rfloor$	$3N/2$	$N/(2k)$	Yes	$N = k \times 2^k$ nodes with a cycle length $k \geq$
$k$ -ary $n$ -cube	$2n$	$n\lfloor k/2 \rfloor$	$nN$	$2k^{n-1}$	Yes	$N = k^n$ nodes



# DYNAMIC INTERCONNECT ARCHITECTURES

## BUS

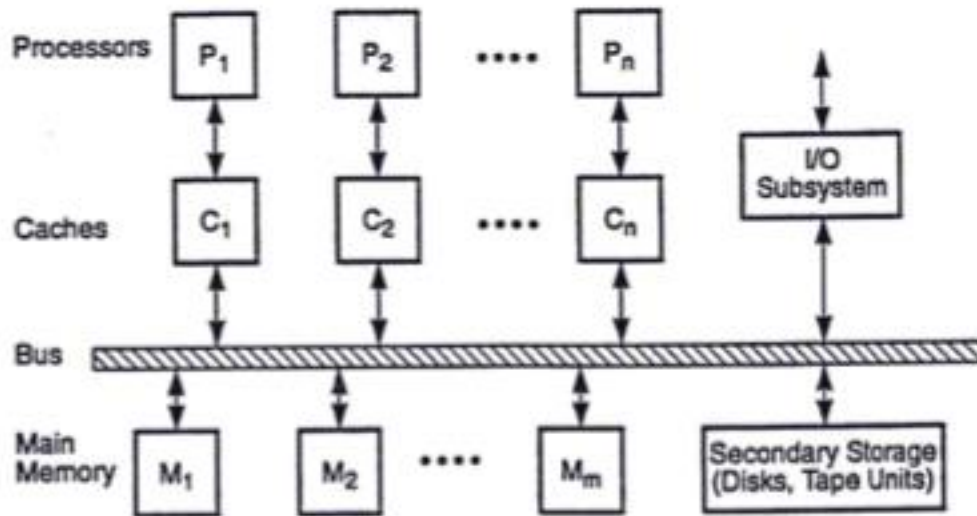


Figure 2.22 A bus-connected multiprocessor system, such as the Sequent Symmetry S1.



# DYNAMIC INTERCONNECT ARCHITECTURES

## PC (Personal Computer) BUS's

### Conceptual Computer Architecture

Dr. J. T. Wunderlich  
Elizabethtown College

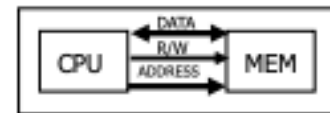
### Conceptual Computer Architecture for typical PC

#### ■ Mother Board:



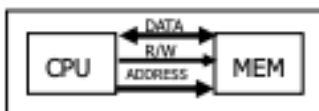
#### ■ CPU (Central Processing Unit):

- Processes all Machine Instructions
- Contains some "fast-access" memory (usually static-RAM or "Latches" made from "D type Flip-Flops")



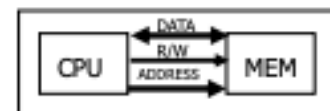
#### ■ MEMORY ("Main Memory"):

- Stores Machine instructions
- Stores Data (i.e., "operands" for instructions)
- Typically made from Dynamic RAM (Random Access Memory)



#### ■ DATA BUS:

- 8 to 128 bit wide electrical pathway
- Exchange data between CPU and "Memory"
- Exchange machine instructions from CPU to "Memory"



#### ■ ADDRESS BUS:

- 16 to 40 bit wide electrical pathway
- Takes Memory addresses generated by CPU to address memory locations

# DYNAMIC INTERCONNECT ARCHITECTURES

## BUS's

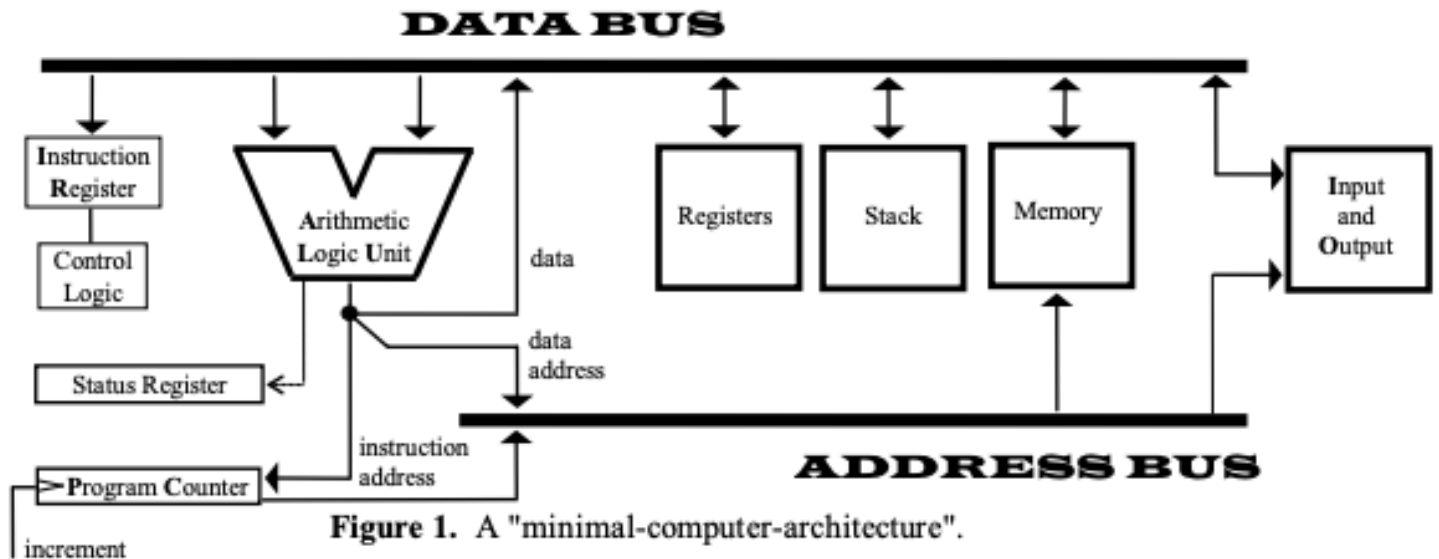


Figure 1. A "minimal-computer-architecture".

As shown in Fig. 1, each device contains:

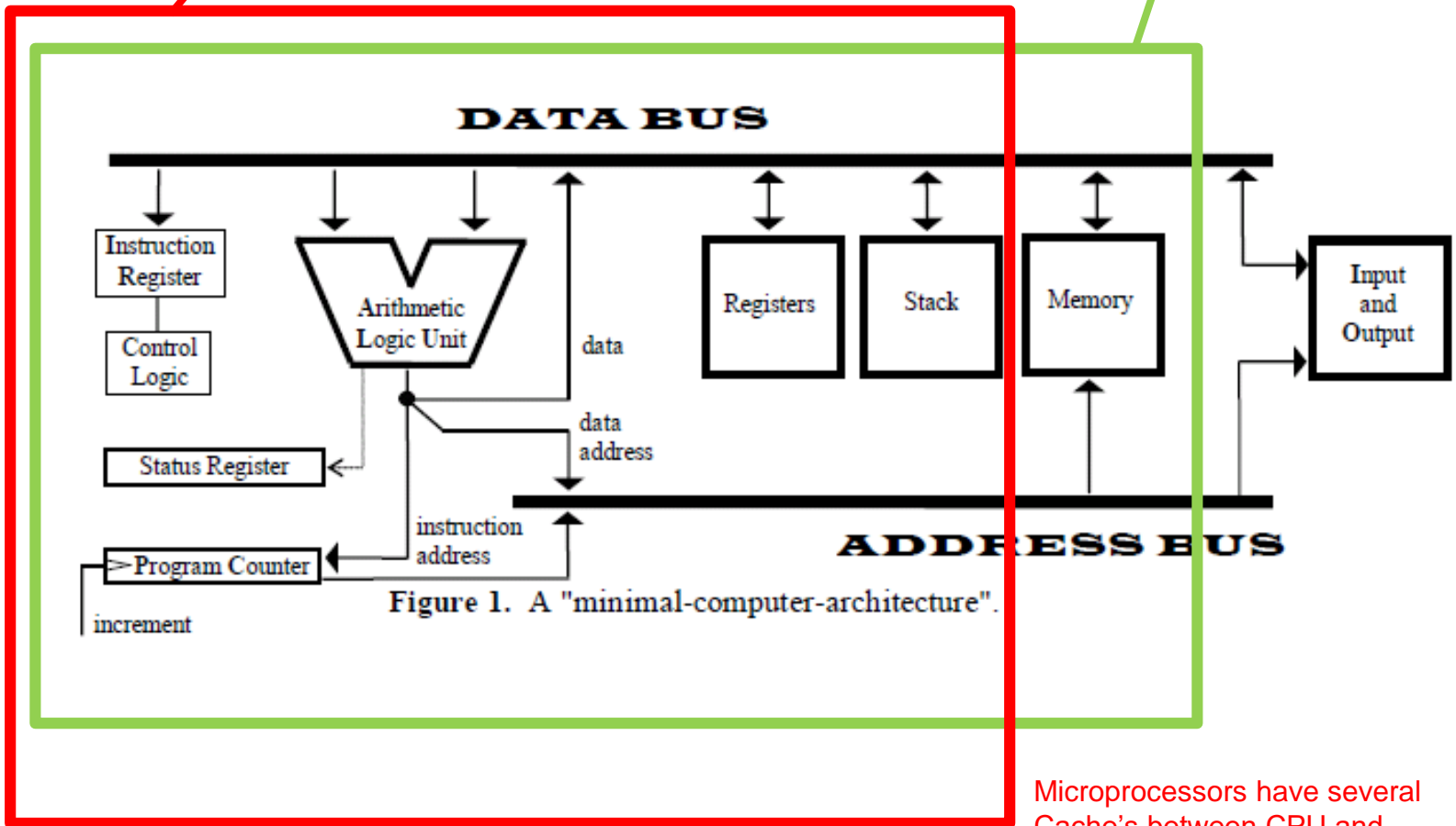
- A program counter to address instructions to be fetched from memory.
- An instruction register to put the fetched instruction in.
- Control logic to create all routing signals after decoding the fetched instruction.
- An ALU for arithmetic and logical manipulation of data and addresses.
- Registers for storing intermediate results of calculations.
- A status register for status flags and condition codes.
- Memory for storing data and instructions.
- A stack for storing addresses (or processor status) for returning from program-calls (or interrupts).
- I/O which is addressed as memory (i.e., memory-mapped I/O).

Wunderlich, J.T. (1999). [Focusing on the blurry distinction between microprocessors and microcontrollers.](#) In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.



# DYNAMIC INTERCONNECT ARCHITECTURES BUS's

## Microprocessors vs. Microcontrollers



Microprocessors have several Cache's between CPU and "Memory"

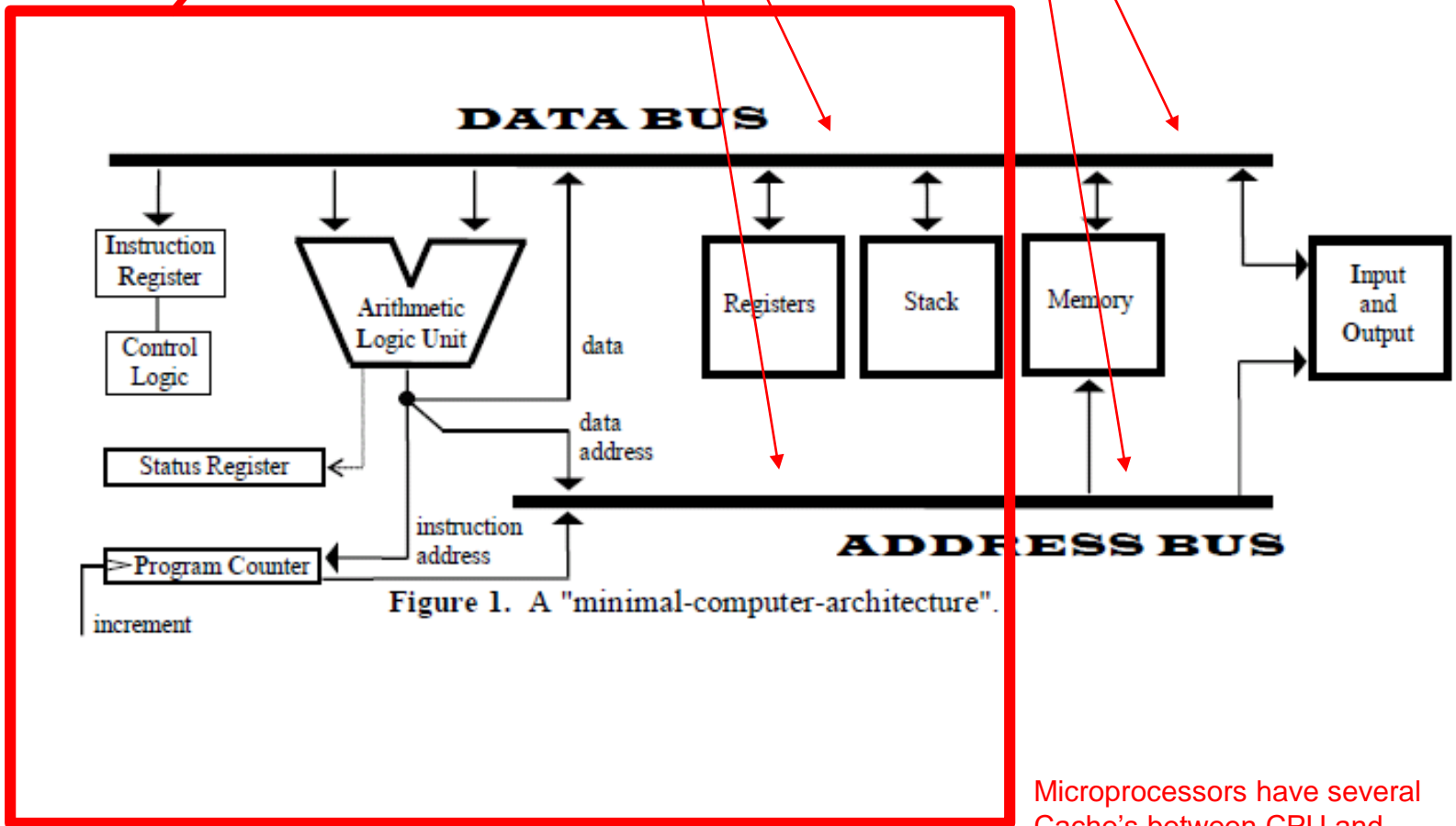
Wunderlich, J.T. (1999). [Focusing on the blurry distinction between microprocessors and microcontrollers](#). In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.



# DYNAMIC INTERCONNECT ARCHITECTURES BUS's

In PC's: **Back-Side Bus, Front-Side-Bus (FSB)**

Microprocessors



Microprocessors have several Cache's between CPU and "Memory"

Wunderlich, J.T. (1999). [Focusing on the blurry distinction between microprocessors and microcontrollers](#). In *Proceedings of 1999 ASEE Annual Conference & Exposition, Charlotte, NC*: (session 3547), [CD-ROM]. ASEE Publications.



# DYNAMIC INTERCONNECT ARCHITECTURES

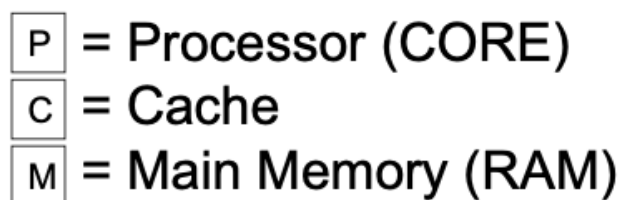
## PC (Personal Computer) BUS's

### PC Design Steps (Pick CPU)

- Multi-core, (multiple CPU's in chip package)

- Don't assume speed-up proportional, yet! – there is recent research on treating Cores like network nodes where performance of overall system actually increases with more nodes, unlike the decrease in performance when many more cores share one memory

- SMP (Symmetric Multi-Processing)



- Cache(s) common “on-chip” (usually now in same chip package, and on same **silicon chip** -- i.e., same piece of silicon chipped off of wafer that is sliced off of big cylindrical ingot)

- Typically L1 cache now integrated into CPU's (“CORE's”)
- Typically L2 cache connected to CPU's via “Back-Side-Bus” (in same chip-package!) connecting everything together on the chip (unlike the **FSB** Front Side Bus which connects the CPU chip package to the main memory RAM on the motherboard)

- Cooling requirements (e.g., heat-sink, dedicated-fan, heat-pipe, liquid)

- Especially if you plan on overclocking CPU



From Lecture on “Design a PC”  
[PDF](#) [PPTX-w/audio](#) [MP4](#) [YouTube](#)

in [EGR/CS230 Computer Architecture](#)



# DYNAMIC INTERCONNECT ARCHITECTURES

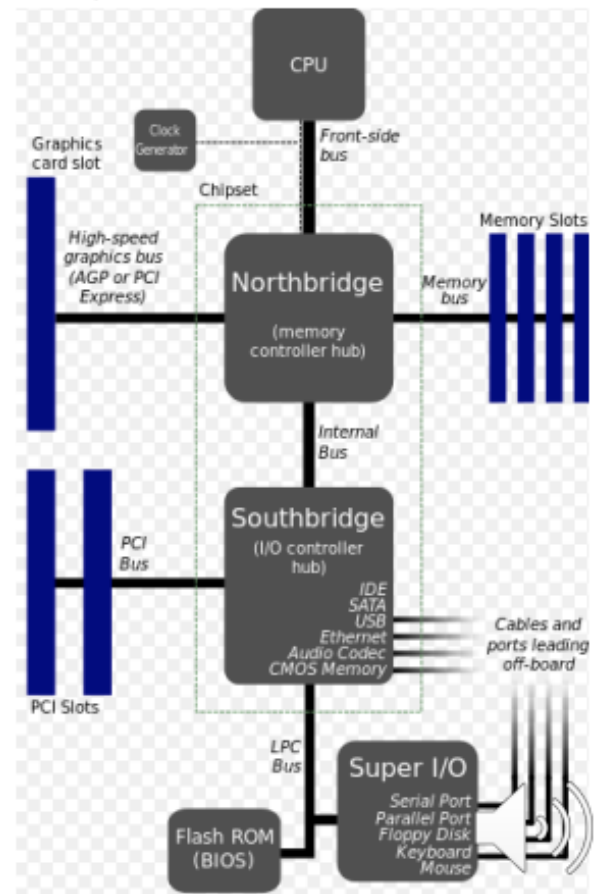
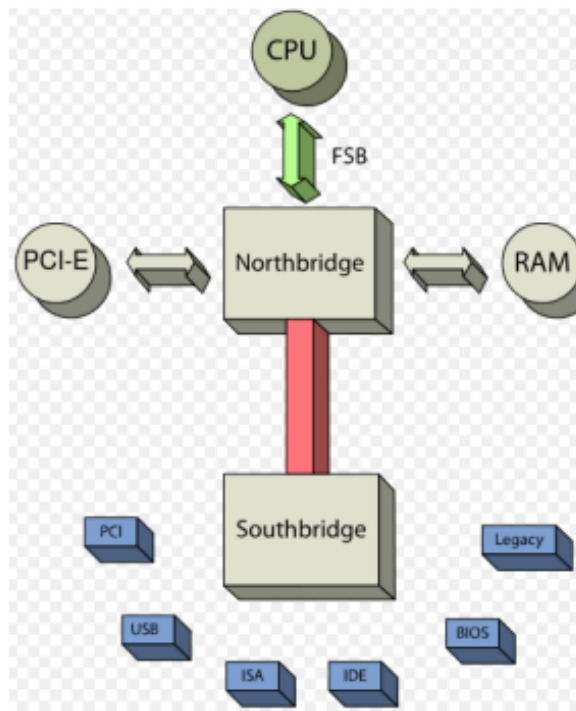
## PC (Personal Computer) BUS's

### PC Design Steps (Pick Motherboard)

- Match speed of CPU, RAM, and Motherboard Front-Side-Bus (FSB) which connects CPU and RAM
- Make sure it has socket to plug in your CPU (i.e. Intel or AMD)
- Make sure it has correct chip-set to handle your CPU, RAM, Graphics Card, and other I/O needs

Northbridge for RAM and video card control,  
and restricts overclocking

Southbridge for power, clock, and other I/O control



Images from:  
[https://en.wikipedia.org/wiki/Northbridge\\_\(computer\)](https://en.wikipedia.org/wiki/Northbridge_(computer))

From Lecture on “Design a PC”  
[PDF](#) [PPTX-w/audio](#) [MP4](#) [YouTube](#)

in [EGR/CS230 Computer Architecture](#)



# DYNAMIC INTERCONNECT ARCHITECTURES

## PC (Personal Computer) BUS's

### PC Design Steps (Pick Motherboard)

- May want **Dual-Channel** capability (can handle two banks of RAM concurrently)
- Make sure Motherboard can **handle your Graphics Card** (NVIDIA, etc)

OLD:

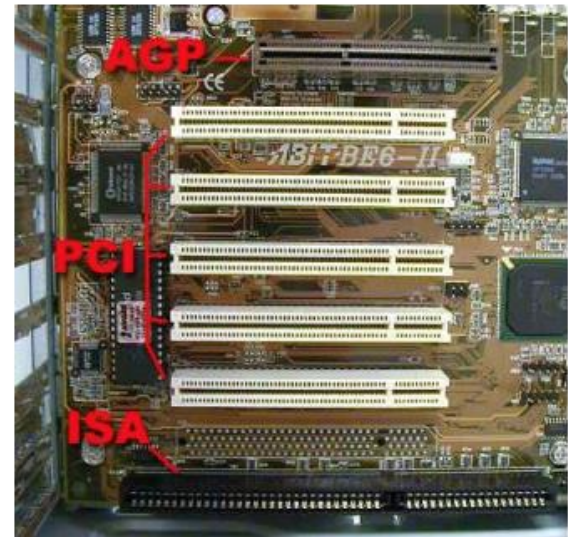
- **AGP** ("Accelerated Graphics Port")
- **PCI**
- **ISA**

RECENT:

**PCIexpress** (not a bus protocol)

*ISA, PCI, and AGP use PARALLEL communication of data*

*PCIexpress uses a packetizing **SERIAL** protocol like that used for Ethernet TCP/IP, and then many serial lines are implemented in parallel*



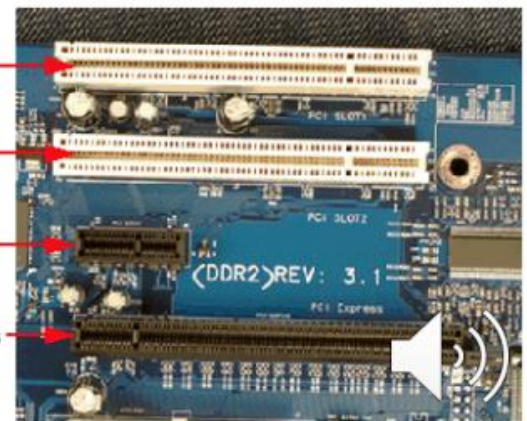
Slots:

PCI 2.0 32-bit

PCI 2.0 32-bit

PCI Express x1

PCI Express x16



From Lecture on "Design a PC"  
[PDF](#) [PPTX-w/audio](#) [MP4](#) [YouTube](#)

in [EGR/CS230 Computer Architecture](#)



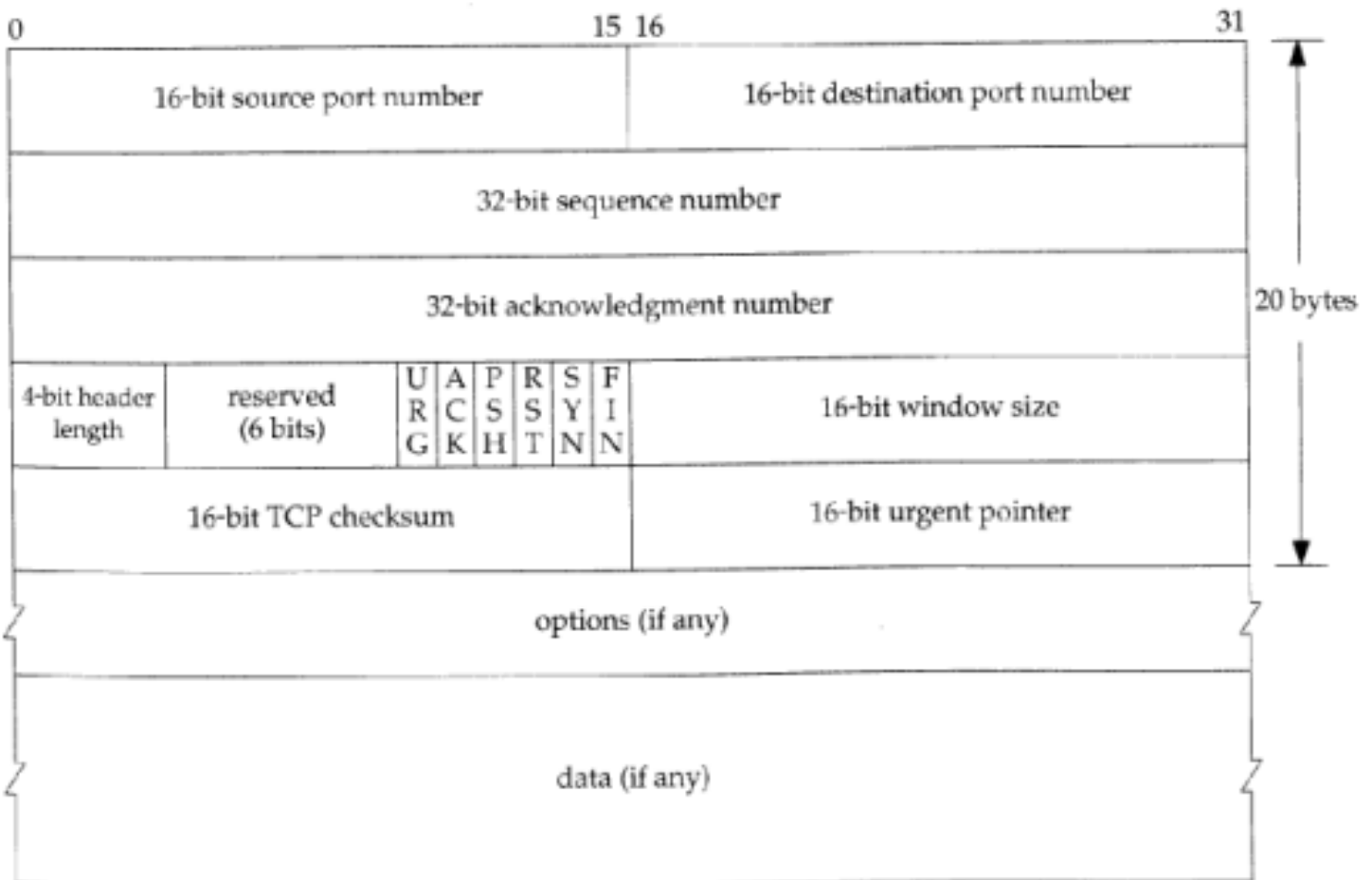
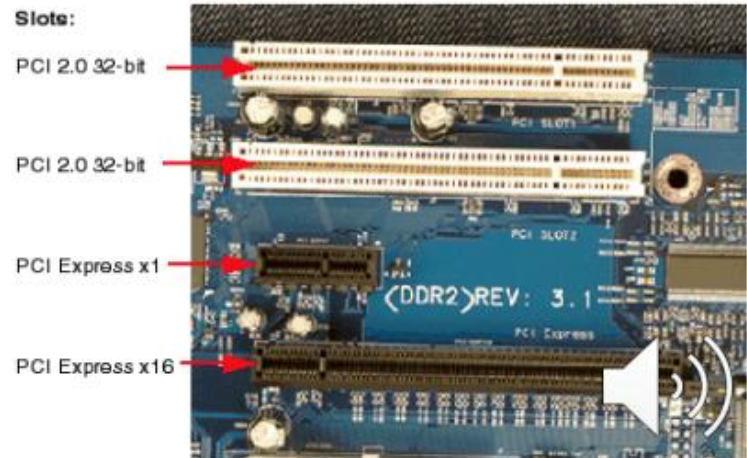
# DYNAMIC INTERCONNECT ARCHITECTURES

## PC (Personal Computer) BUS's

**PCIexpress** (not a bus protocol)

*ISA, PCI, and AGP use PARALLEL communication of data*

*PCIexpress uses a packetizing **SERIAL** protocol like that used for Ethernet TCP/IP, and then many serial lines are implemented in parallel*



From Lecture on "Design a PC"  
[PDF](#) [PPTX-w/audio](#) [MP4](#) [YouTube](#)  
in [EGR/CS230 Computer Architecture](#)



# DYNAMIC INTERCONNECT ARCHITECTURES

## BUS

### Design-Lab of a SERIAL BUS

in [EGR/CS333 Digital Design II, Assembly Language, & Interfacing](#)

Create a Serial BUS and your own communication protocol (like packetized TCP/IP, but simpler) to connect previous lab projects.

Review Course Lecture: <http://users.etown.edu/w/wunderjt/Communications.pdf>

And our custom Lab Manuals:

[2018 IC's, Circuit Trainer, and Power Supply](#)

[2019 Relays](#)

[2019 NanoLC PLC\(Programmable Logic Controller\)](#)

[2019 Rasberry Pi and ARM Microcontroller](#)

[2019 Intel 8051 Microcontroller](#) ([2015](#), [2014](#), [Pre-2013](#))

[2019 AXIOLINE PLC](#)

[2021 PLC-NEXT](#) (*in development during this course*)

### REAL-TIME

1. Make a NanolineLC (or PLC NEXT or Axioline PLC,) implement shifting in a nibble; And, shifting out a nibble; shift it in, store it, and then shift it back back out of PLC. Deal with the parity of bits coming into, or leaving PLC (i.e., do error-detection/parity-check)
2. Using tri-state buffers and appropriate relays between devices (Phoenix Contact, Arduino, or others located in lab), and needed control lines communicating within and between devices, create a BIDIRECTIONAL SINGLE WIRE SERIAL communication DATA BUS shared by one circuit trainer (with circuits from a previous lab) and the PLC. You can have as many control lines as you wish, but only one shared bidirectional serial single-wire data bus.
3. Demonstrate bidirectional serial communication with parity checking
4. Use your circuit trainer clock generator to see how fast you can communicate to the PLC (i.e., if you can communicate as fast as the fastest clock generator available)

### SIMULATION

1. Create a complete simulation in Logisim of everything above including forced error in data transmission, clock generators, and a logic facsimile of what your PLC is actually doing (i.e., translate your PLC logical decisions and actions into equivalent logic gates in Logisim)
2. Demonstrate simulated versions of control schemes of other devices when available in development system (e.g., NanoLC)
3. Simulate all above communications in 8051 Assembly Language (just some approximation of functionality)

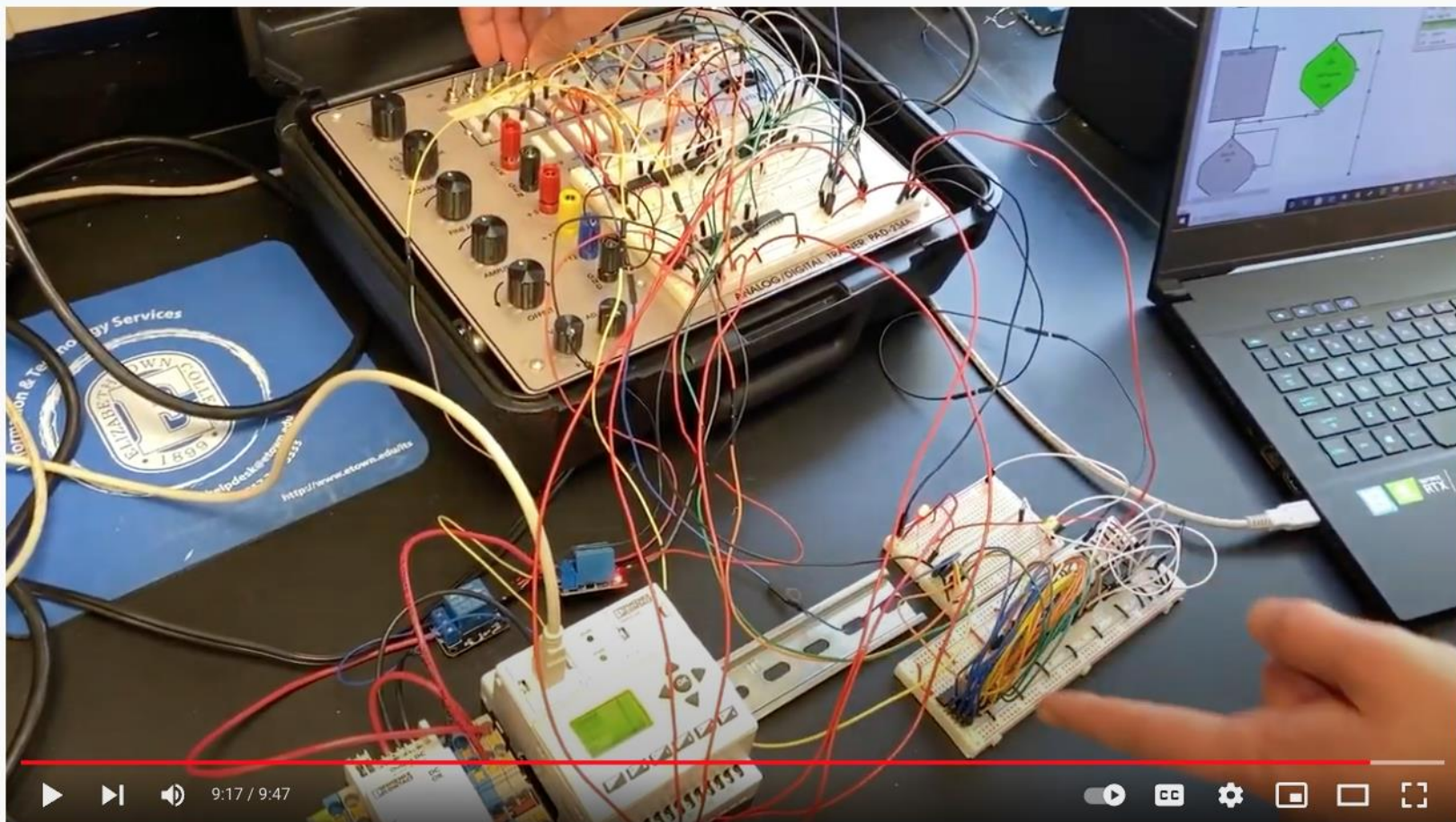
**GRADE:** (75%) 8 to 10 minute (total) video of all Simulation and Real-Time Circuits; (25%) Lab Report [PER SYLLABUS SPECIFICATIONS!](#) of entire Design process Just one person submit both of these, and the same grade will be given to each team member.



# DYNAMIC INTERCONNECT ARCHITECTURES BUS

Design-Lab of a **SERIAL BUS**

in [EGR/CS333 Digital Design II, Assembly Language, & Interfacing](#)



Digital Design II Lab 4 Simulations and Real Time Circuit Demonstration

<https://www.youtube.com/watch?v=2csorAkpjyg>



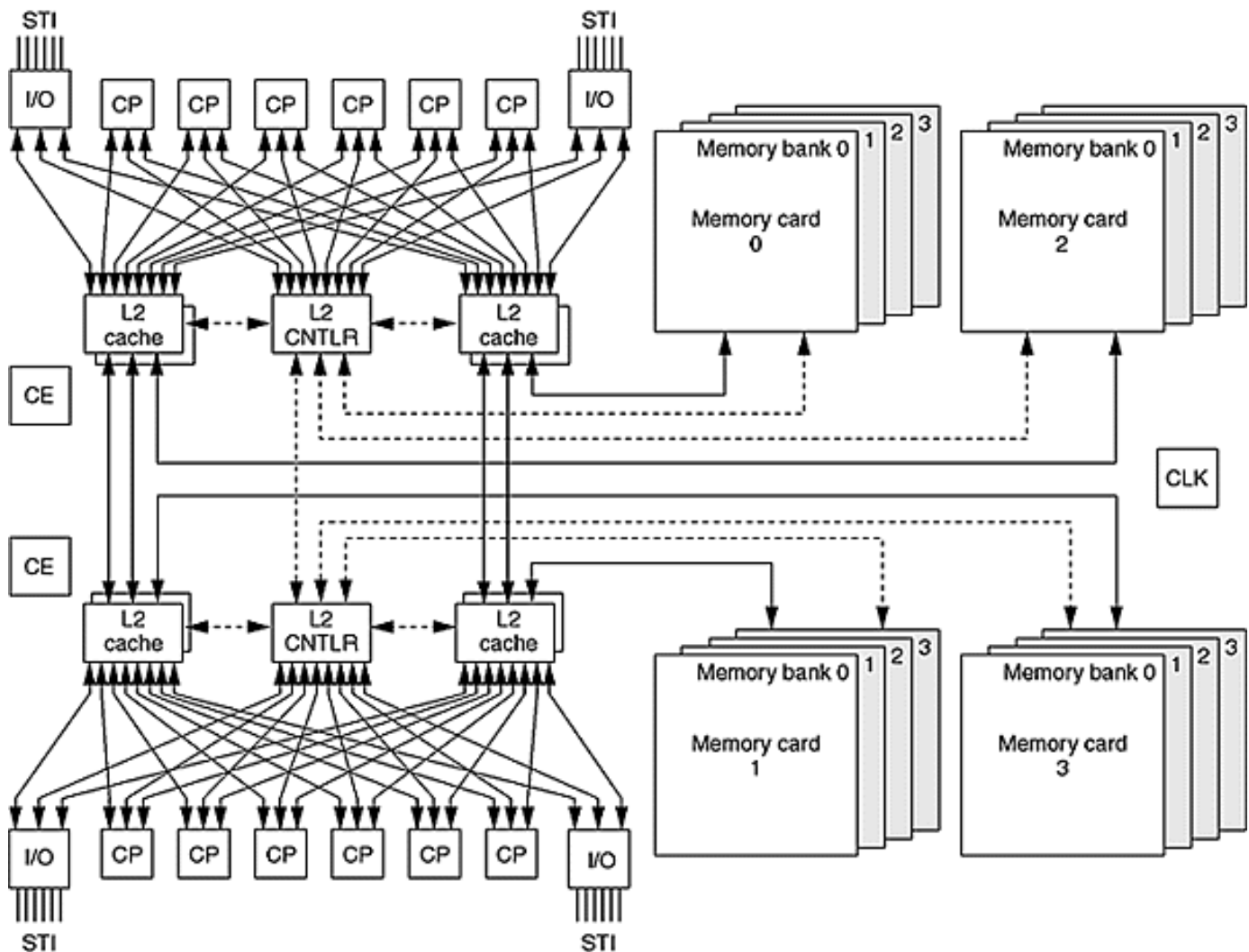
# DYNAMIC INTERCONNECT ARCHITECTURES

## BUS



IBM S/390 of 1990's  
(IBM Z-Series 2000 to 2020's)

*J Wunderlich PhD.  
Advisory-Level Engineer, & Researcher*



# DYNAMIC INTERCONNECT ARCHITECTURES

## BUS

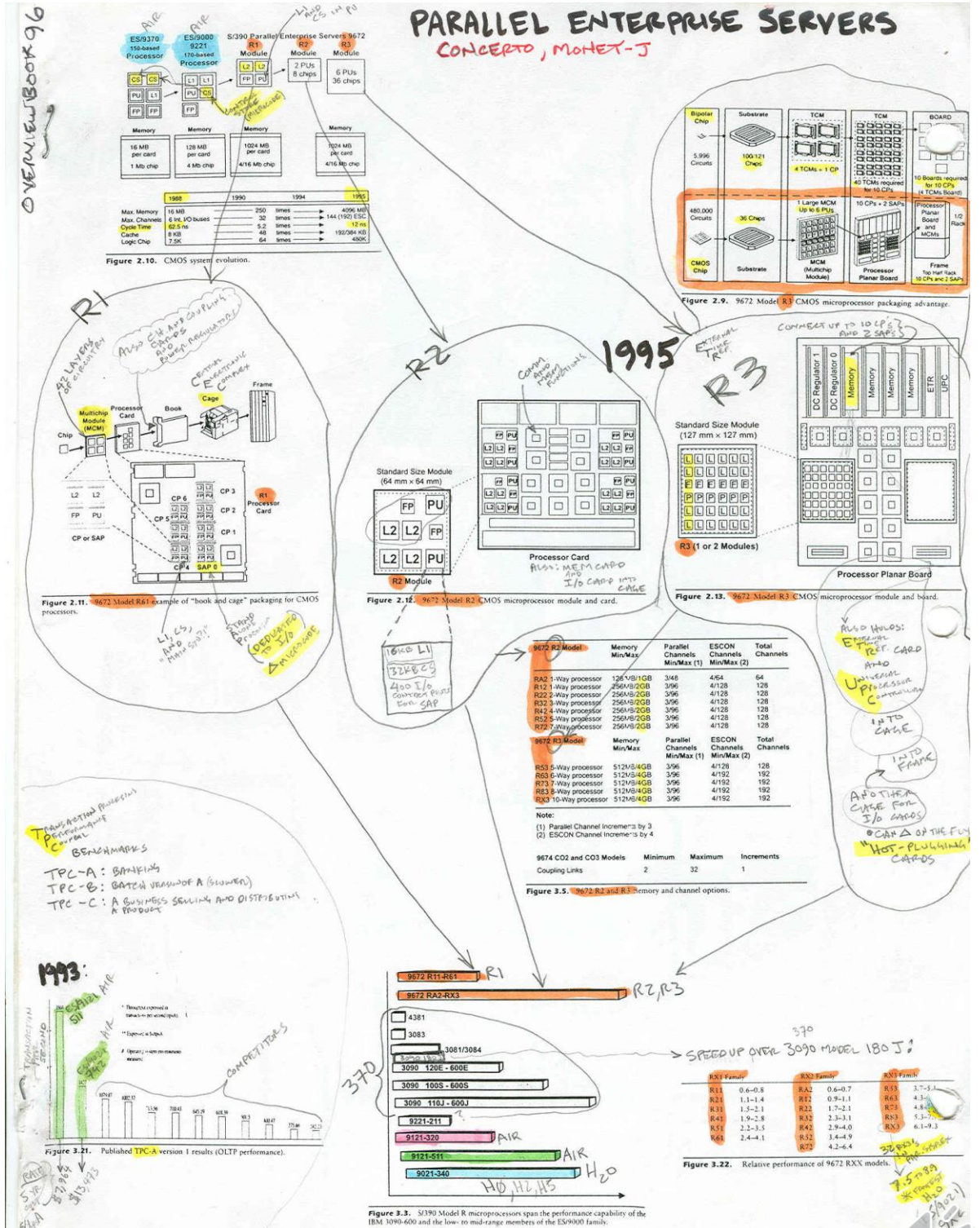
IBM S/390 of 1990's (IBM Z-Series 2000 to 2020's)

*J Wunderlich PhD.*

*Advisory-Level Engineer, & Researcher*



<http://users.etoyn.edu/w/wunderlit/home IBM.html>



# DYNAMIC INTERCONNECT ARCHITECTURES

## BUS

IBM S/390 of 1990's (IBM Z-Series 2000 to 2020's)

*J Wunderlich PhD.*

*Advisory-Level Engineer, & Researcher*



<http://users.etsown.edu/w/wunderlit/home IBM.html>

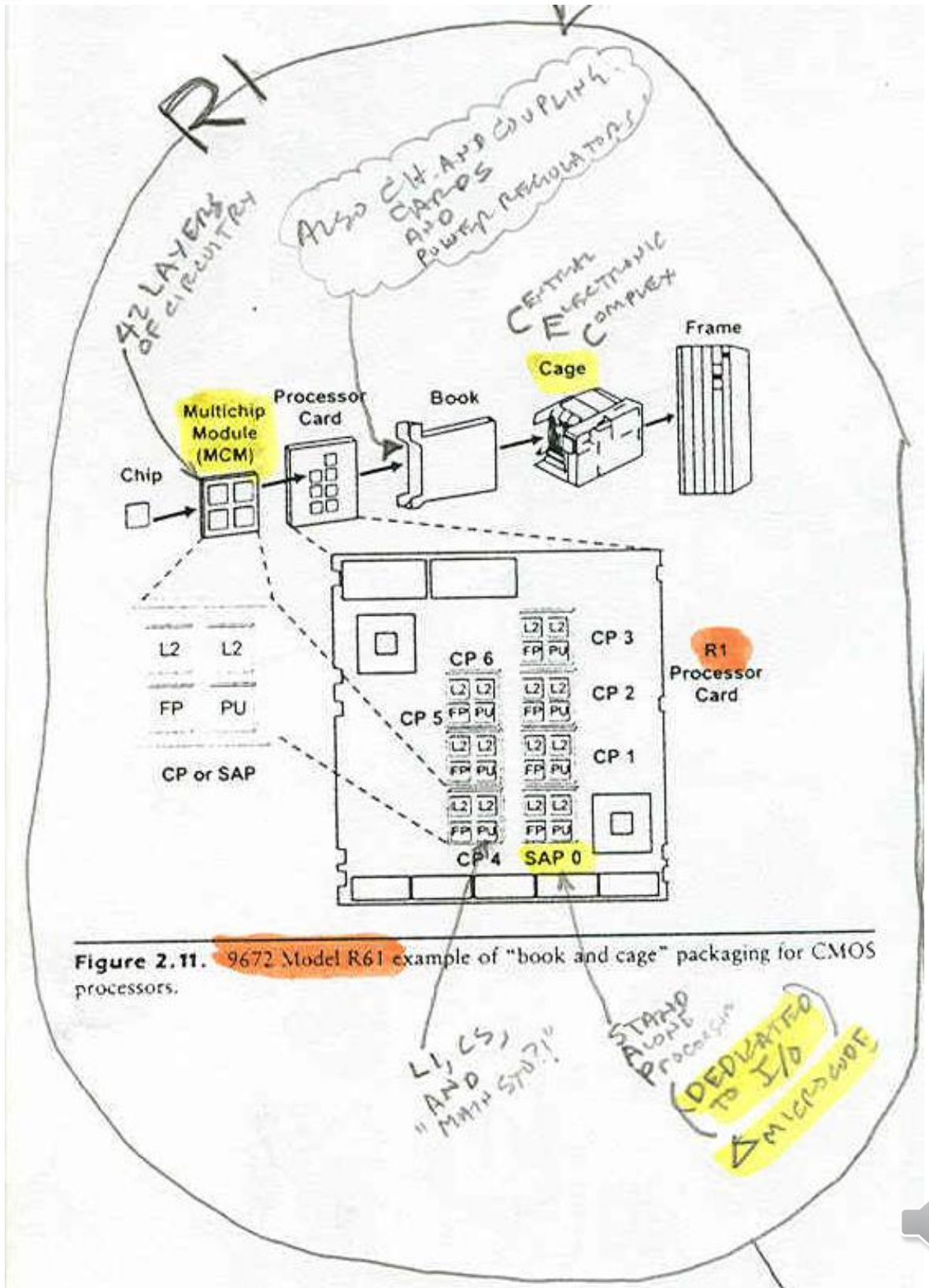


Figure 2.11. 9672 Model R61 example of "book and cage" packaging for CMOS processors.



# DYNAMIC INTERCONNECT ARCHITECTURES

## BUS

IBM S/390 of 1990's (IBM Z-Series 2000 to 2020's)

*J Wunderlich PhD.*

*Advisory-Level Engineer, & Researcher*

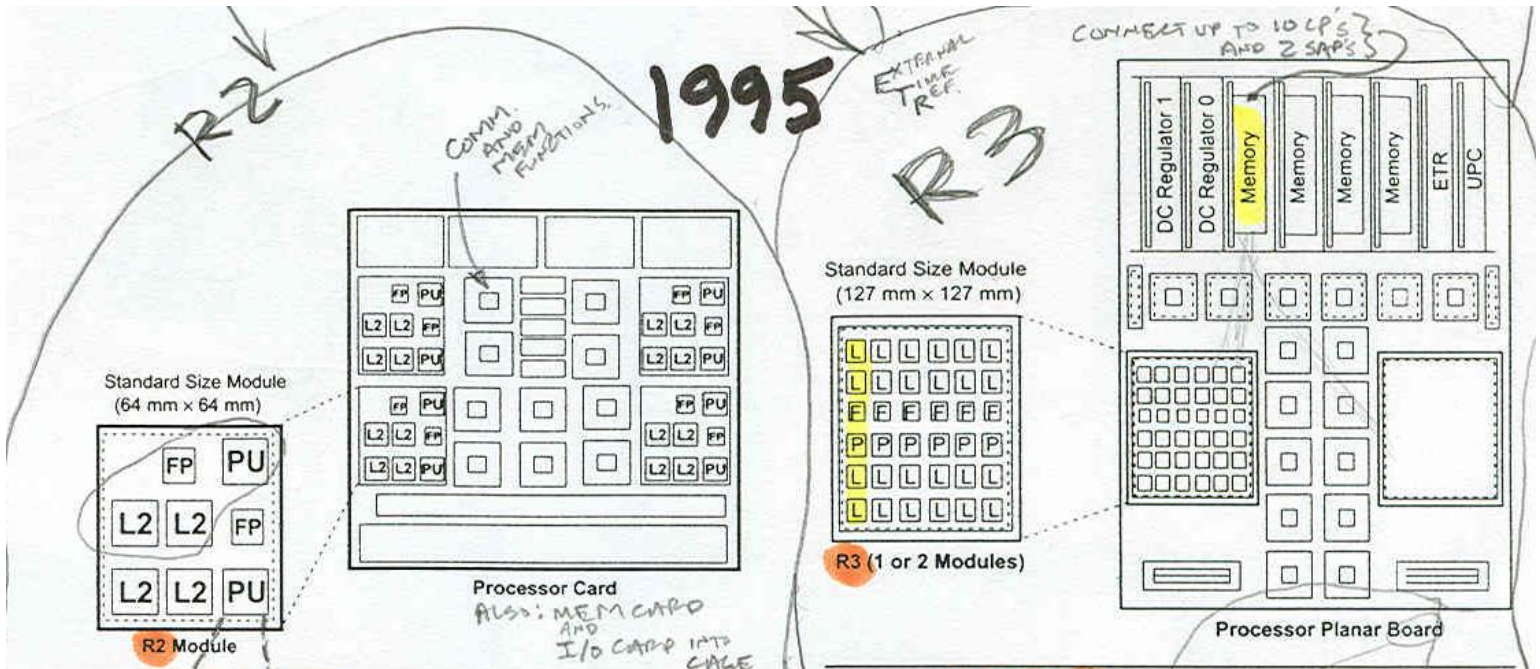


Figure 2.12. 9672 Model R2 CMOS microprocessor module and card.

Figure 2.13. 9672 Model R3 CMOS microprocessor module and board.

16KB LI  
32KB CS  
400 I/O CONTACT POINTS FOR SAP

9672 R2 Model	Memory Min/Max	Parallel Channels Min/Max (1)	ESCON Channels Min/Max (2)	Total Channels
RA2 1-Way processor	128MB/1GB	3/48	4/64	64
R12 1-Way processor	256MB/2GB	3/96	4/128	128
R22 2-Way processor	256MB/2GB	3/96	4/128	128
R32 3-Way processor	256MB/2GB	3/96	4/128	128
R42 4-Way processor	256MB/2GB	3/96	4/128	128
R52 5-Way processor	256MB/2GB	3/96	4/128	128
R72 7-Way processor	256MB/2GB	3/96	4/128	128
9672 R3 Model	Memory Min/Max	Parallel Channels Min/Max (1)	ESCON Channels Min/Max (2)	Total Channels
R53 5-Way processor	512MB/4GB	3/96	4/128	128
R63 6-Way processor	512MB/4GB	3/96	4/192	192
R73 7-Way processor	512MB/4GB	3/96	4/192	192
R83 8-Way processor	512MB/4GB	3/96	4/192	192
RX3 10-Way processor	512MB/4GB	3/96	4/192	192

Note:  
(1) Parallel Channel Increments by 3  
(2) ESCON Channel Increments by 4

9674 CO2 and CO3 Models	Minimum	Maximum	Increments
Coupling Links	2	32	1

Figure 3.5. 9672 R2 and R3 memory and channel options.

# DYNAMIC INTERCONNECT ARCHITECTURES

## MULTISTAGE INTERCONNECTION NETWORK

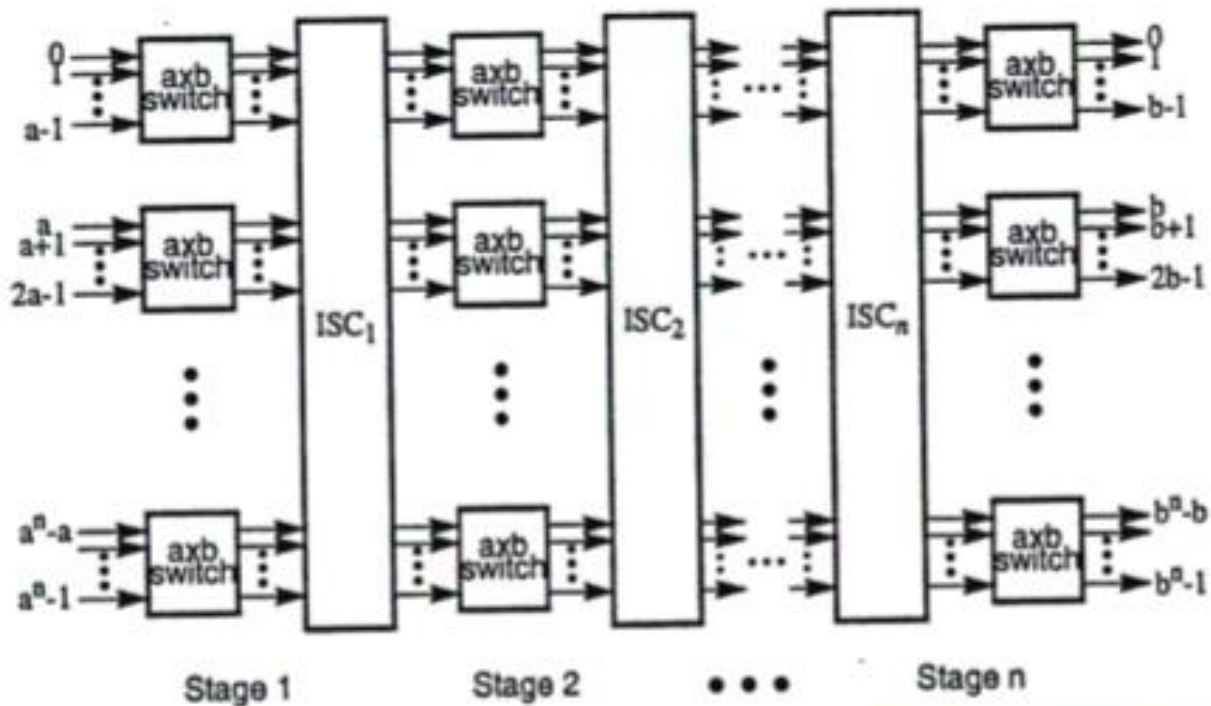
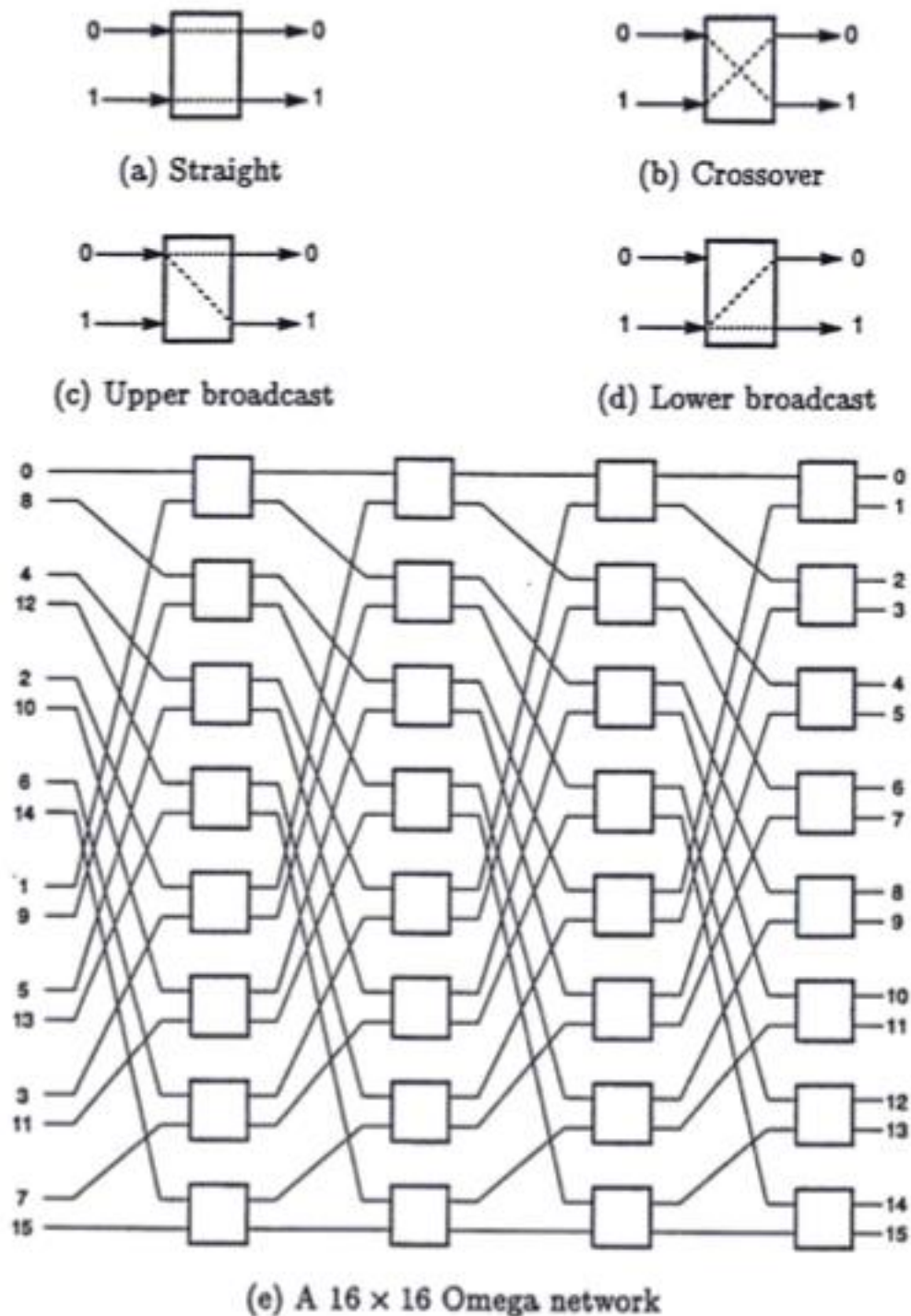


Figure 2.23 A generalized structure of a multistage interconnection network (MIN) built with  $a \times b$  switch modules and interstage connection patterns  $ISC_1, ISC_2, \dots, ISC_n$ .



# DYNAMIC INTERCONNECT ARCHITECTURES

## OMEGA NETWORK

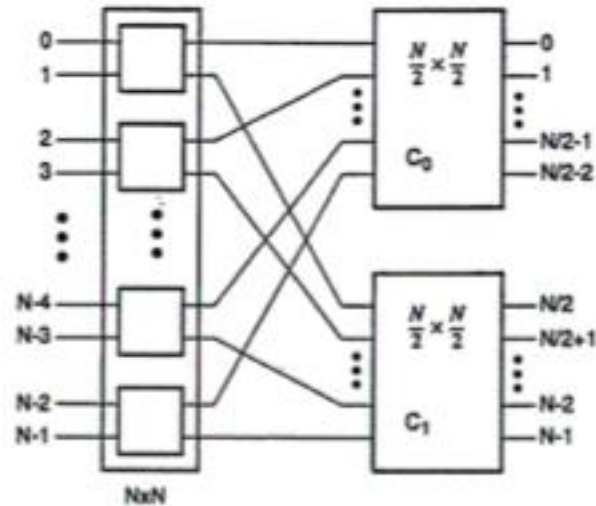


**Figure 2.24** The use of  $2 \times 2$  switches and perfect shuffle as an interstage connection pattern to construct a  $16 \times 16$  Omega network. (Courtesy of Duncan Lawrie; reprinted with permission from *IEEE Trans. Computers*, Dec. 1975)

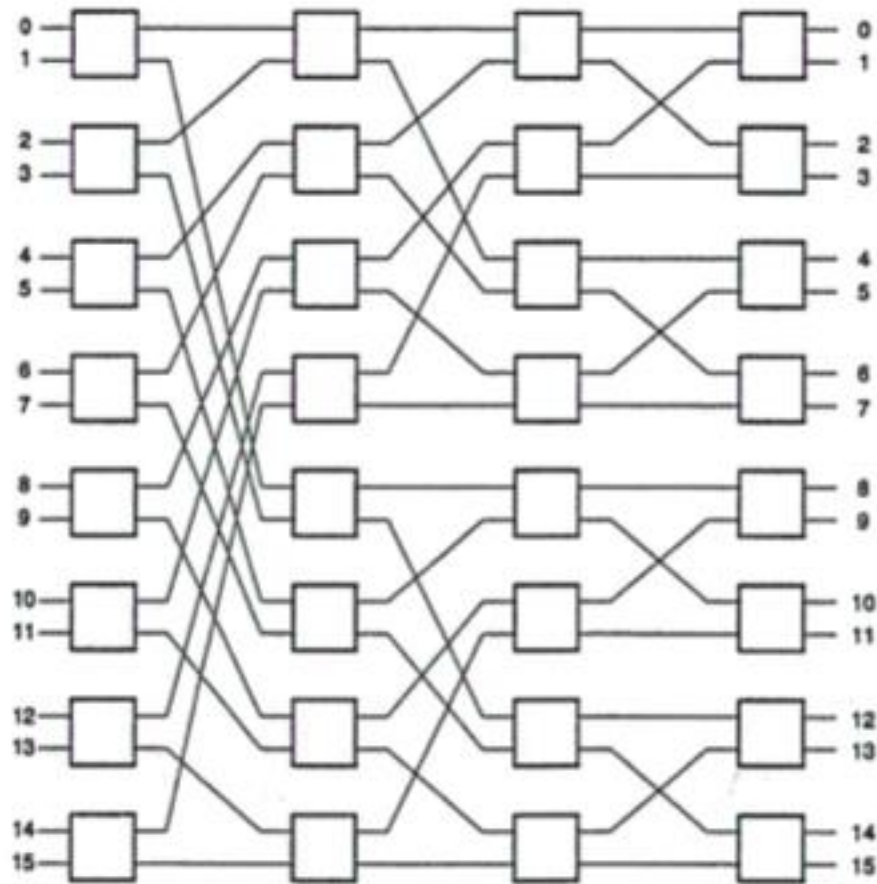


# DYNAMIC INTERCONNECT ARCHITECTURES

## BASELINE NETWORK



(a) Recursive construction



(b) A 16 x 16 Baseline network

**Figure 2.25** Recursive construction of a Baseline network. (Courtesy of Wu and Feng; reprinted with permission from *IEEE Trans. Computers*, August 1980)



# DYNAMIC INTERCONNECT ARCHITECTURES

## CROSSBAR SWITCH

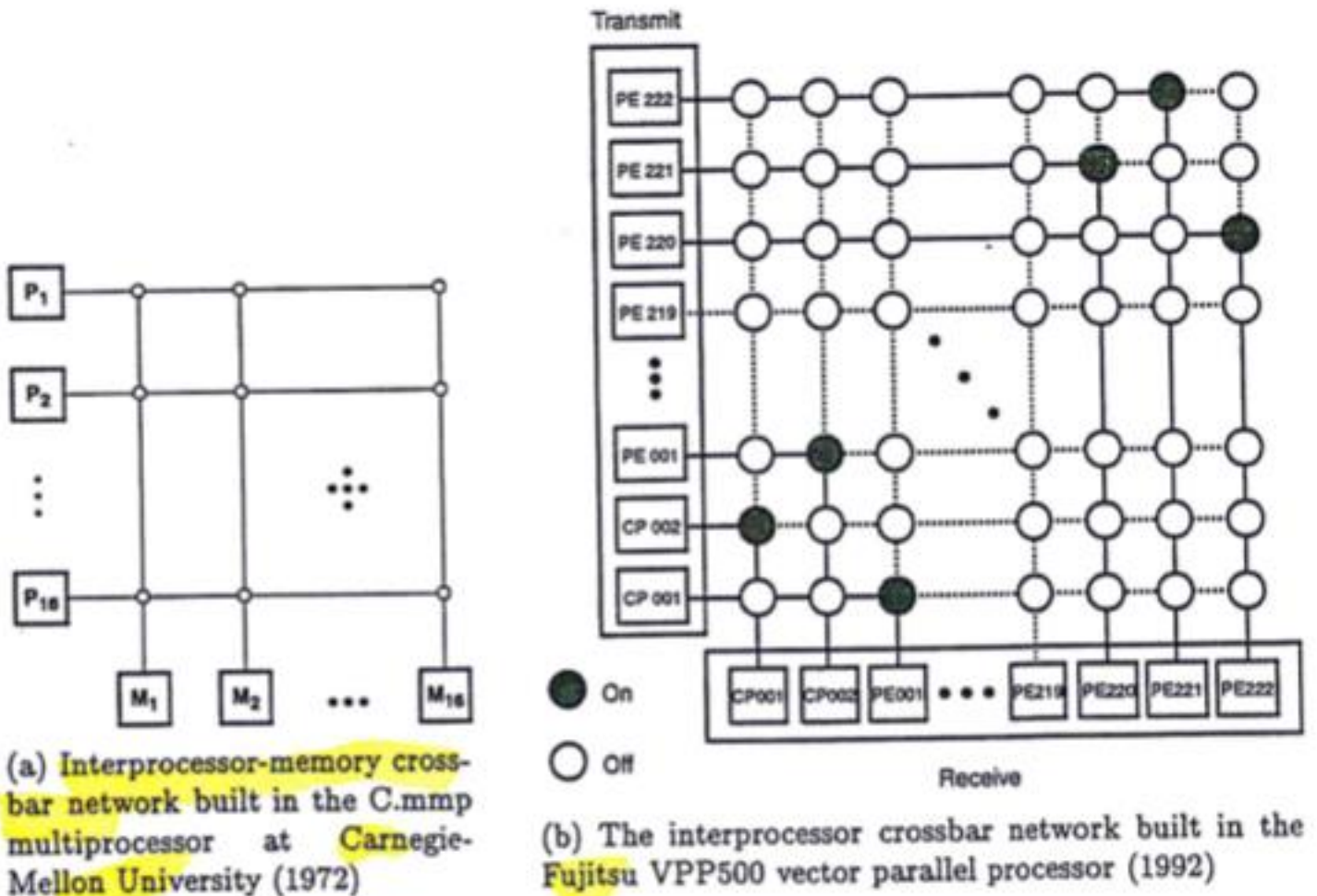


Figure 2.26 Two crossbar switch network configurations.



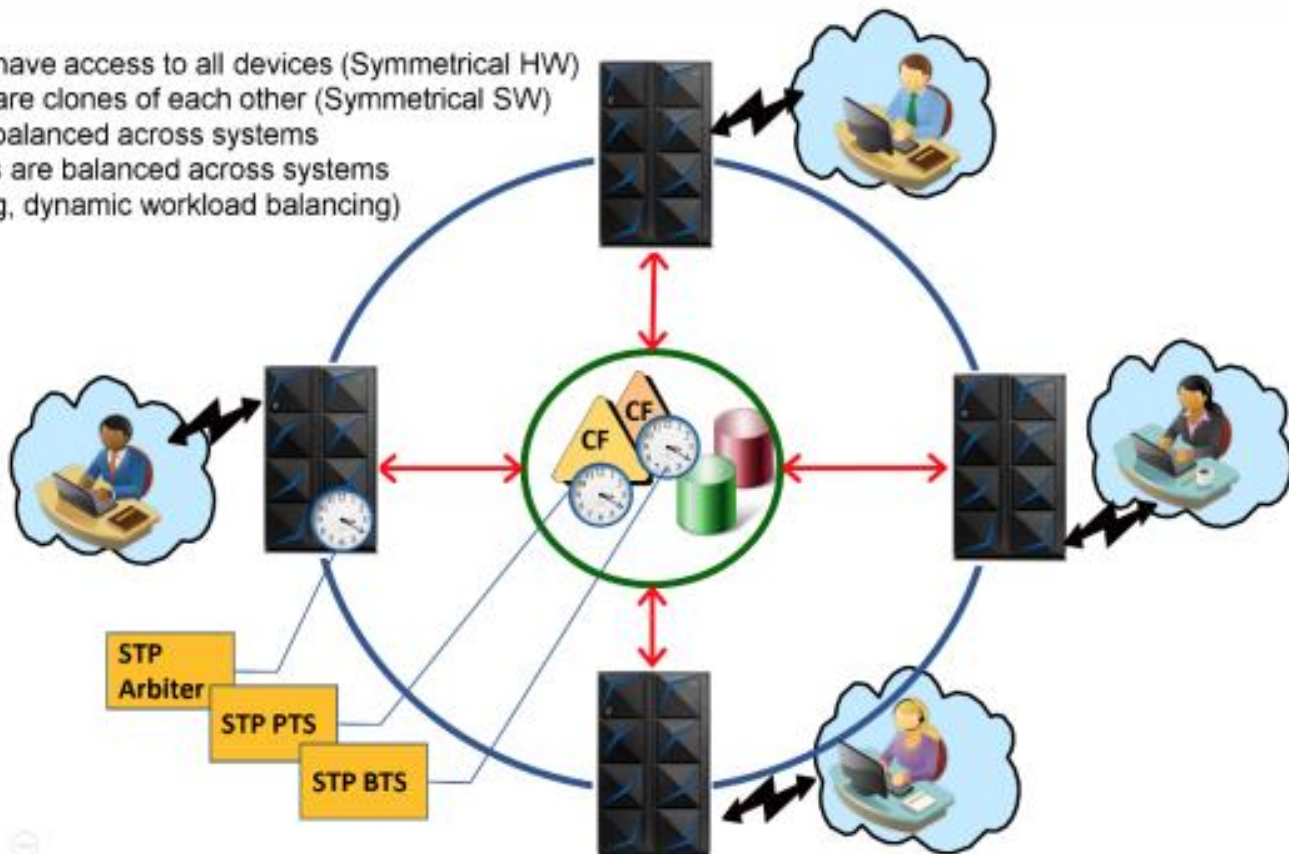
# DYNAMIC INTERCONNECT ARCHITECTURES

## CROSSBAR SWITCH

### IBM PARALLEL SYSPLEX 1990's to 2020's

Example of a four-way Parallel Sysplex:

- All systems have access to all devices (Symmetrical HW)
- All systems are clones of each other (Symmetrical SW)
- Logons are balanced across systems
- Transactions are balanced across systems (data sharing, dynamic workload balancing)



<https://www.ibm.com/downloads/cas/W54PP4LN>

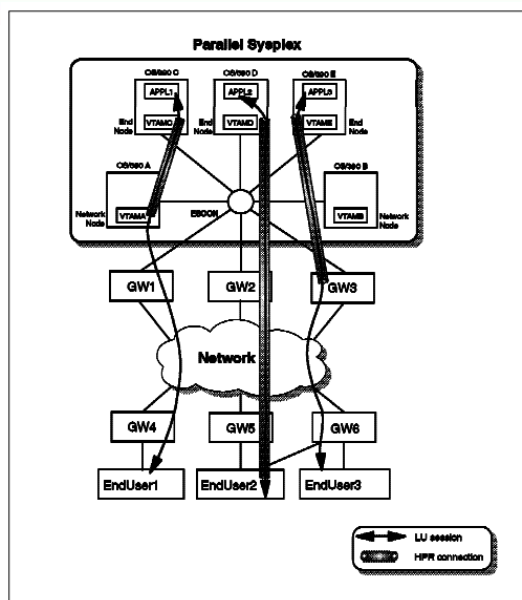


Figure 5. HPR Connections in a Parallel Sysplex

<https://www.redbooks.ibm.com/redbooks/pdfs/sg245639.pdf>



# DYNAMIC INTERCONNECT ARCHITECTURES

## CROSSBAR SWITCH IBM PARALLEL SYSPLEX 1990's to 2020's

### PARALLEL SYSPLEX

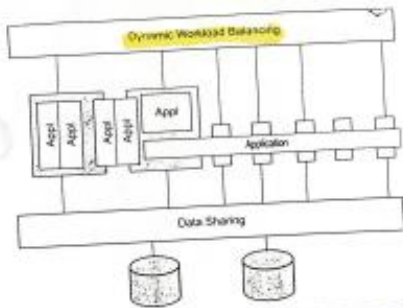


Figure 2.10. Interrelationship of parallel sysplex features.

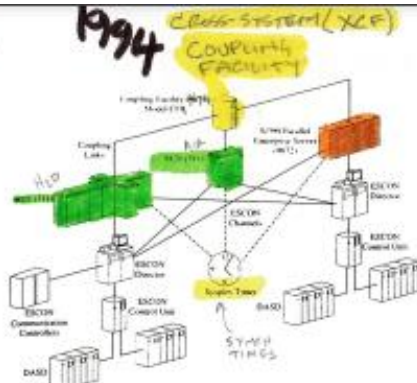


Figure 3.1. A parallel sysplex configuration requires several components.

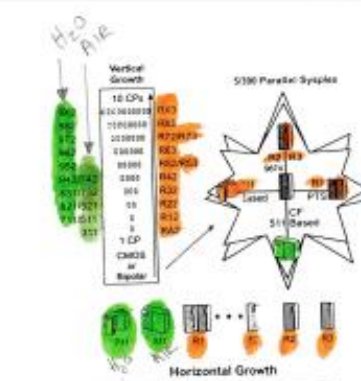


Figure 3.2. Parallel sysplex upgrade flexibility.

**PARALLEL SYSPLEX**  
 FROM MVS IS LIKE RS/6000 FOR UNIX

**SP**  
 • SCALABLE  
 • MANAGEABLE  
 • PARTITIONED FROM A SINGLE POINT OF CONTROL  
 • 2 TO 32 NODES

BY DEFINING MULTIPLE IMAGES UNDER MVS IN A SYSTEM COMPLEX SYSPLEX USING COUPLING FACILITY → A SINGLE IMAGE, DATA-SHARING MVS OS.

UP TO 32 COMPUTERS (~10,000 MIPS)

TRANSPARENT TO USERS, NETWORKS, APPS, AND OPERATIONS MANAGER.

COMPATIBLE W/ EXISTING APPS

SINGLE IMAGE OPERATIONS

DYNAMIC WORKLOAD BALANCING

DATA SHARING (W/INTegrity)

ACCESS MULTIPLE PROCESSORS

OPEN CLIENT/SERVER ENVIRONMENT

IBM'S FUTURE

BETTER THAN JUST CLUSTERING (IE AFTER WITH 77MIP MANUAL REPLY)

CONTINUOUS AVAILABILITY (COMPONENTS CAN BE REMOVED)

COMPLEX MANAGING OF CHANNELS (NO SLOW MESSAGE PASSING FOR REQ DATA/WAIT)

SCALE DATA BETWEEN SYSTEMS

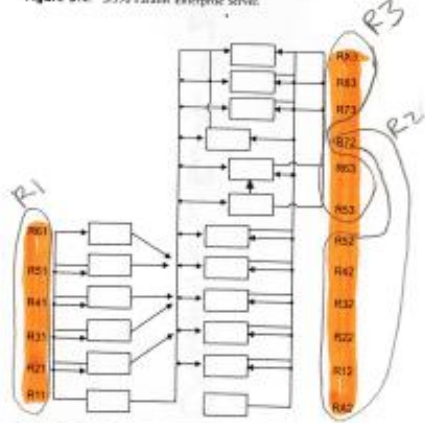
SHARED QUEUES AND STOPS

PHYSICAL SEPARATION OF SYSTEMS TO DO VLS-PROOF DATA-INTENSIVE APP FAILURE RECOVERY

- 1 OPEN BLUEPRINT
  - NETWORK OF OXIS ACT AS SINGLE OS
- 2 OBJECTS: BANK ACCOUNTS, CUSTOMER ORDERS, EMPLOYEE RECORDS
- 3 ENTRY SERVER OFFERING (ESD)
  - CUSTOM HW/SW SERVICES
- 4 PC SERVER 500 SYSTEM/390

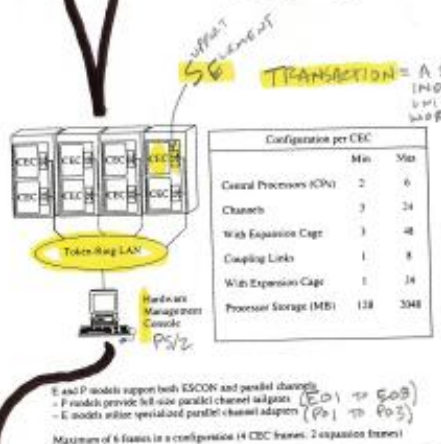


Figure 3.4. S/390 Parallel Enterprise Server.



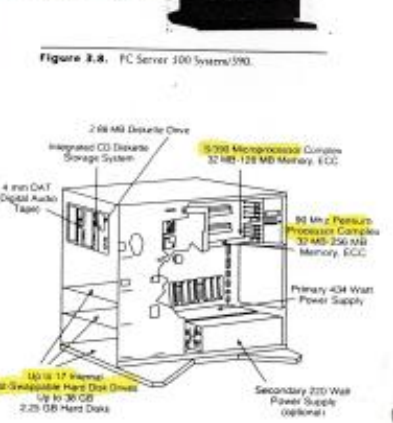
- 3.6. Parallel Enterprise Server upgrades.
- NO MORE GENERAL PURPOSE
  - NOW FOR SPECIFICALLY DEFINED WORK ENVIRONMENTS
  - NO MORE VECTOR
    - PERFORMANCE, RELIABILITY, CRITICAL
  - HIGH LEVEL OF SCALABILITY (FINE GRAIN)
- \$500,000 MIN

### PARALLEL TRANSACTION SERVER



- 3.7. S/390 Parallel Transaction Server (9472) specifics.
- DEDICATED COMPUTERS FOR TRANSACTIONS
  - REMOVED FROM MAINFRAME
  - RE-DESIGNED TO EXISTING MAINFRAME PROC.
  - NO SIGNIFICANT DEVELOPMENT WHEN COMPARING TO PARALLEL

### PC SERVER 500 SYSTEM/390



- 3.8. PC Server 500 System/390.
- 3.9. IBM PC Server 500 System/390 (Linux-optimized view).
- 3/390 EISA SYSTEM SET (PROCESSOR)
  - BUS: VM/ESA, VSC/ESA, AND MVS/ESA OS
  - FEMTOUM PROCESSOR SUPPORTS LAN, OS/2
  - CAN BE A FILE, PRINT, OR APP SERVER
  - 60NS SHIELD AND 3/390 SW
  - DEVELOPMENT WORKBENCH SERVER (EVS) IS AN OPEN/OS SERVER
  - NO CLIENT
  - 4.5 TDS MIPS
- \$100,000



# DYNAMIC INTERCONNECT ARCHITECTURES ("TOPOLOGIES")

## SCALABILITY



Table 2.4 Summary of Dynamic Network Characteristics

Network Characteristics	Bus System	Multistage Network	Crossbar Switch
Minimum latency for unit data transfer	Constant $\times$	$O(\log_k n)$	Constant $\times$
Bandwidth per processor	$O(w/n)$ to $O(w)$	$O(w)$ to $O(nw)$	$O(w)$ to $O(nw)$
Wiring Complexity	$O(w)$ $\times$	$O(nw \log_k n)$	$O(n^2 w)$
Switching Complexity	$O(n)$ $\times$	$O(n \log_k n)$	$O(n^2)$
Connectivity and routing capability	Only one to one at a time.	Some permutations and broadcast, if network unblocked	All permutations, one at a time.
Representative computers	Symmetry S-1, Encore Multimax	BBN TC-2000, IBM RP3	Cray Y-MP/816, Fujitsu VPP500
Remarks	Assume $n$ processors on the bus; bus width is $w$ bits.	$n \times n$ MIN using $k \times k$ switches with line width of $w$ bits.	Assume $n \times n$ crossbar with line width of $w$ bits.

IBM S/390,  
Z--Series

IBM Parallel  
Sysplex

