ECE 762
Theory and Design of Digital Computers, II

(A real course title:

Design and Specification of Digital Components with an HDL)

Lecture Overview

- Course Intro/ Syllabus/ Grading Policy
- General Intro to Digital Design
- Backgound

Syllabus

- The topics list is a guide.
- Note course objective.
- Note grading policy
- There are many, many books. Most are not texts; these books are more for reference.
- Note general policies.

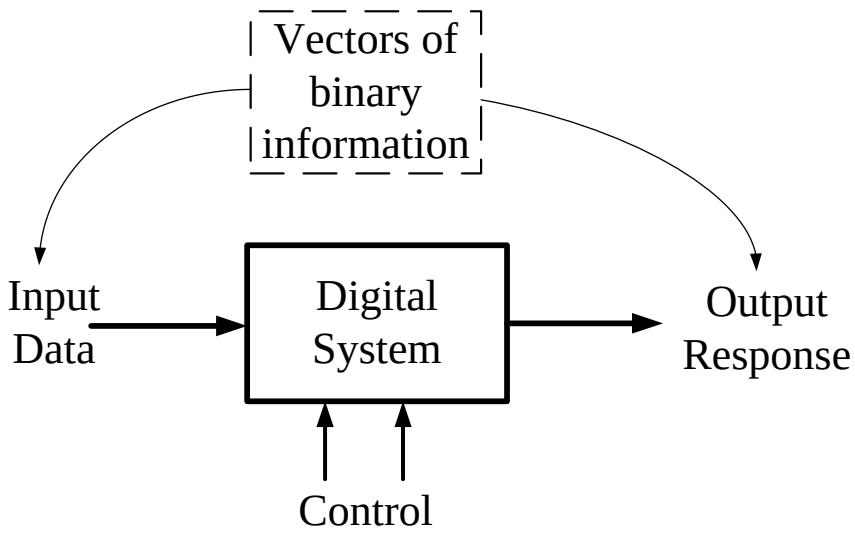
Intro

- What is a digital system?
  - Digital (Webster)
  - System

Intro

- What is a digital system?
- **Digital (Webster)** – Of or relating to the technology of computers and data communications wherein all information is encoded as bits of 1s and 0s that represent on or off states. Contrast with analog. *Digital implies discrete states.*
- System – A composite of equipment, skills, techniques, and information capable of performing and/or supporting an operational role in attaining specified management objectives. A complete system includes related facilities, equipment, material, services, personnel, and information required for its operation to the degree that it can be considered a self-sufficient unit in its intended operational and/or support environment.

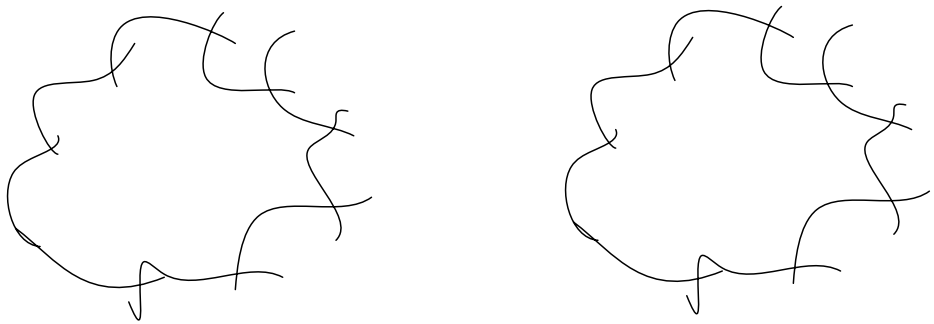
Graphical Perspective



- A Digital System may be an Application Specific IC (ASIC) or a general purpose computer.
- ***"Computers are the most important type of digital system"***
- ***"Virtually every aspect of digital system design is encountered in computer design" (Hill and Peterson)***

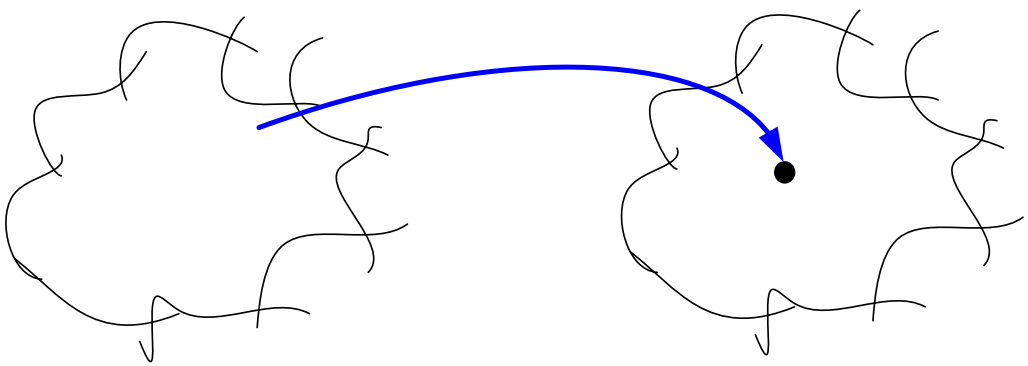
Digital System Design Process

- “Design is a series of transformations.” At each step decisions are made that bind the design, moving it toward an implementation. Design begins at a high level of abstraction and moves to a very detailed level of abstraction.



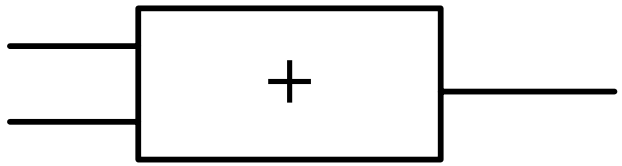
Digital System Design Process

- “Design is a series of transformations.” At each step decisions are made that bind the design, moving it toward an implementation. Design begins at a high level of abstraction and moves to a very detailed level of abstraction.



Example

- Addition of 2 number to produce a sum



- Design Decisions
  - 2 inputs
  - Addition operation
  - A & B format - ?? – Binary numbers, 16 bits each, unsigned
  - Architecture - ?? – ripple adder, carry lookahead
- Design Decisions are significantly impacted by the specifications

B

HDL Design Process

- Start with design idea
- Do a behavioral design for reference
- RTL level design
  - Design data path
  - Design control path
- Use a synthesis tool to produce a gate netlist
- Physical Design – place gates and wire
- Production

An example

- From ASiC Technology & News – “Why ASICs fail in the system.
  - Listen to story
- Key points from story.
  - “Designers knew design was right”
  - “found a functional error”
  - Chips still exploded.
  - Months passed slowly.

HDL motivations

- HDL used to describe hardware for purpose of:
  - Simulation
  - Documentation
  - Modeling
  - Testing
  - Design
- HDLs provide a convenient and compact format for the hierarchical representation of function and wiring details of digital systems.

PAST HDLs

- ISPS – Instruction Set Processor Specification
  - Language for describing the behavior of digital systems
  - Developed at CMU
  - Based on ISP notation

```
mark1 :=
  BEGIN
  ** memory.state **
  m[0:8191]<31:0>,
  ** processor.state **
  pi\present.instruction<15:0>'
    f\function<0:2> := pi<15:13>,
    s<0:12> := pi<12:0>,
  cr\control.register<12:0>,
  acc\accumulator<31:0>,
  ** instruction.execution ** {tc}
MAIN i.cycle :=
  BEGIN
  pi = m[cr]<15:0> NEXT
  DECODE f =>
    BEGIN
  0\jmp   := cr = m[s],
  1\jrp   := cr = cr + m[s],
  2\ldn   := acc = - m[s],
  3\sto   := m[s] = acc,
  4:5\sub := acc = acc - m[s],
  6\cmp   := IF acc LSS 0 => cr = cr + 1,
  7\stp   := STOP(),
    END NEXT
  cr = cr + 1 NEXT
  RESTART i.cycle
  END
```

FIGURE 1.3
An ISPS example, a simple processor. (Source: M. R. Barbacci, The ISPS Computer Description Language, Carnegie-Mellon University, 1981, p. 70.)

PAST HDLs

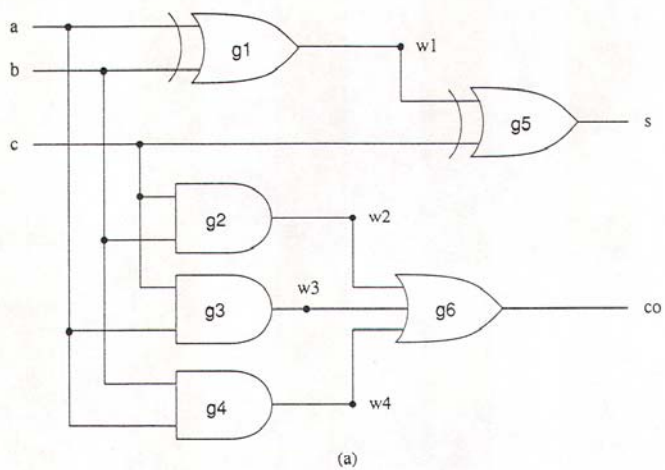
- AHPL – A Hardware Programming Language
  - Designed for representation in an academic environment
  - Developed at the University of Arizona.

```
AHPLMODULE: multiplier.
   MEMORY: ac1[4]; ac2[4]; count[2]; extra[4]; busy.
   EXINPUTS: dataready.
   EXBUSES: inputbus[8].
   OUTPUTS: result[8]; done.
   CLUNITS: INC[2](count); ADD[5](extra; ac2);
1  ac1 <= inputbus[0:3];   ac2 <= inputbus[4:7];
   extra <= 4$0;
   => (~^dataready)/(1).
2  busy <= \1\;
   => (^ac1[3])/(4).
3  extra <= ADD[1:4](extra; ac2).
4  extra, ac1 <= \0\, extra, ac1[0:2];
   count <= INC(count);
   => (^(&/count))/(2).
5  result = extra, ac1;   done = \1\;   busy <= \0\;
   => (1).
   ENDSEQUENCE
   CONTROLRESET(1).
END.
```

FIGURE 1.4
An AHPL example, showing a sequential multiplier.

Other HDLs

- Genrad Hardware Description Language
  - Describes hardware as a netlist of components.
  - Developed by Genrad Corporation, UK



(a)

```
CCT full_adder (a, b, c, s, co)
XOR (RISE = 16, FALL = 12)
   g1 (w1, a, b),
   g5 (s, w1, c);
AND (RISE = 12, FALL = 10)
   g2 (w2, c , b),
   g3 (w3, c, a),
   g4 (w4, b, a);
OR (RISE = 12, FALL = 10)
  g6 (co, w2, w3, w4);
INPUT a, b, c;
WIRE w1, w2, w3, w4;
OUTPUT s, co;
ENDCIRCUIT full_adder
```

(b)

FIGURE 1.5
A Full-Adder, (a) logic diagram, (b) GHDL description.

Other HDLs

- CDL – Computer Design Language
  - A dataflow language – no hierarchy
- CONLAN – Consensus Language
  - Attempt to establish a standard language. Family of languages for describing hardware at various levels of abstraction.
- IDL – Interactive Design Language
  - Internal IBM – Supports Hierarchy – Originally designed for generation of PLAs, then extended
- TEGAS – Texas Instruments Hardware Description Language
  - Internal TI – Multilevel language for design and description – hierarchical

Other HDLs (cont)

- ZEUS
  - GE language – hierarchical – functional descriptions – structural descriptions – No provision for gate delay specification or timing constraints – Does not support asynchronous designs.
- Verilog
  - Hierarchical – Developed by Cadence Design Systems – Procedural descriptions for behavior – Built in features for timing and a fixed logic value system. **Now also a standard. Used by ~60% of market.**
- UDL1
  - Standard language that was developed in Japan – hierarchical – 1 to 1 mapping of language constructs to hardware structures – Designed for synthesis
- System C
  - NEW – now also a standard and supported by tools – had penetrated to about 10% to 15 % of the market.

VHDL

- VHDL – VHSIC Hardware Description Language
- A standard language – the first.
- Development began in 1980.
- Language Requirements set in 1981.
- 1st Version – Version 7.2 with prototype simulation tools – 1985-1987
- 1st Standard – IEEE Standard 1076-1987 approved in 1987.
- New versions in 1993. Also versions in 2000, 2002 and about to be an new version soon???

VHDL features

- Procedural Features
  - Would make a very good concurrent programming language. Up until now file I/O support was poor.
- Dataflow design
- Structural – Hierarchy
- Self defined Value System and capability to design your own if you would need to.
- Semantics and Paradigm formally defined in LRM

In Summary

- There is no way we would have systems of today's complexity without the development and evolution of HDLs.
- HDLs are living languages.
- Today's systems are just too complex to stay with the design methodologies of the 1980s and even to early 1990s.

The Future ????

Processors, Processor Speed, No. of Transistors
Intel Chips

Demel Jan-05

Year	Month	Year	Clock Speed	Year	No. Transistors	Chip
1972	4	1972.3	0.2	1972.3	3500	8008
1974	12	1975	2	1975	6000	8080
1976	8	1976.6	5	1976.6	6500	8085
1978	9	1978.7	10	1978.7	2900	8086
1982	2	1982.1	12	1982.1	134000	286
1985	10	1985.8	16	1985.8	275000	386
1987	2	1987.1	20	1987.1	275000	386
1989	4	1989.3	25	1989.3	1200000	486
1991	6	1991.5	50	1991.5	1200000	486
1993	3	1993.2	60	1993.2	3100000	Pentium
1994	3	1994.2	75	1994.2	3200000	Pentium
1995	3	1995.2	120	1995.2	3200000	Pentium
1995	6	1995.5	133	1995.5	3300000	Pentium
1996	1	1996.1	166	1996.1	3300000	Pentium
1996	6	1996.6	200	1996.6	3300000	Pentium
1997	5	1997.4	300	1997.4	3300000	Pentium II
1998	4	1998.3	400	1998.3	7500000	Pentium II
1998	8	1998.7	450	1998.7	7500000	Pentium II
1999	8	1999.7	600	1999.7	9500000	Pentium III
1999	10	1999.8	733	1999.8	28000000	Pentium III
2000	1	2000.1	800	2000.1	28000000	Pentium III
2000	3	2000.2	1000	2000.2	28000000	Pentium III
2000	11	2000.9	1500	2000.9	42000000	Pentium 4
2001	4	2001.3	1700	2001.3	42000000	Pentium 4
2001	8	2001.7	2000	2001.7	42000000	Pentium 4
2002	1	2002.1	2200	2002.1	42000000	Pentium 4
2002	6	2002.5	2530	2002.5	55000000	Pentium 4
2002	8	2002.7	2800	2002.7	55000000	Pentium 4
2002	11	2002.9	3000	2002.9	55000000	Pentium 4
2003	6	2003.5	3200	2003.5	55000000	Pentium 4
2004	2	2004.1	3400	2004.1	55000000	Pentium 4