Logic Design

With Simulation Exercises

Philosophy

- All Theoretical Subjects Should Be Backed Up With Practical Exercises
- Students Should Verify All “Paper” Designs to Verify Their Correctness
- Students Should Verify Exhaustive Proofs by Using an Exhaustive Simulation
- Students Should be Exposed to the Principles of Testing

Fundamental Laws

Equation	Law
a•1=a	And Identity Law
a+0=a	Or Identity Law
ab = ba	Commutativity of and
a+b = b+a	Commutativity of or
(ab)c = a(bc)	Associativity of and
(a+b)+c = a+(b+c)	Associativity of or
a(b+c) = ab+bc	And distributes over or
a+bc = (a+b)(a+c)	Or distributes over and
aa’=0	And inverse law
a+a’=1	Or inverse law

Identity Laws

FHDL

```
Ident:  circuit
        inputs  a
        outputs xa,xo

        one     i1
        zero    i0
g1:     and     (a,i1),xa
g2:     or      (a,i0),xo
        endcircuit
```

VECTORS

```
0
1
```

Associativity of AND

FHDL

```
aa:     circuit
        inputs  a,b,c
        outputs left,right

g1:     and     (a,b),l1
g2:     and     (l1,c),left

h1:     and     (b,c),r1
h2:     and     (a,r1),right
        endcircuit
```

VECTORS

0,0,0
0,0,1
0,1,0
0,1,1
1,0,0
1,0,1
1,1,0
1,1,1

Karnaugh Maps

- Start with A Truth Table
- Construct A Boolean Function
- Translate the Boolean Function into FHDL
- Simulate The FHDL to Reconstruct the Truth Table

Design With Gates

- Determine Number of Inputs & Outputs
- Derive Truth-Tables for Outputs
- Derive Boolean Functions for each Output
- Draw Logic Diagram
- *Encode Logic Diagram in FHDL*
- *Simulate FHDL to Verify Correct Operation*

Design with MSI Functions

- Implement Example MSI Functions
- Simulate to Verify Correctness
- Design using MSI Components Directly
- Implement Using FHDL Functional Blocks and Hierarchical Design
- Simulate to Verify Correctness

Sequential Circuits

- ★ Design and Simulate RS, D, and Edge-Triggered D flip-flops
- ★ Illustrate Oscillations using the RS flip-flop
- ★ Illustrate The Need for Edge-Triggering, or Master/Slave Synchronization
- ★ Illustrate Logic Hazards, and the Danger They Pose to Asynchronous Circuits

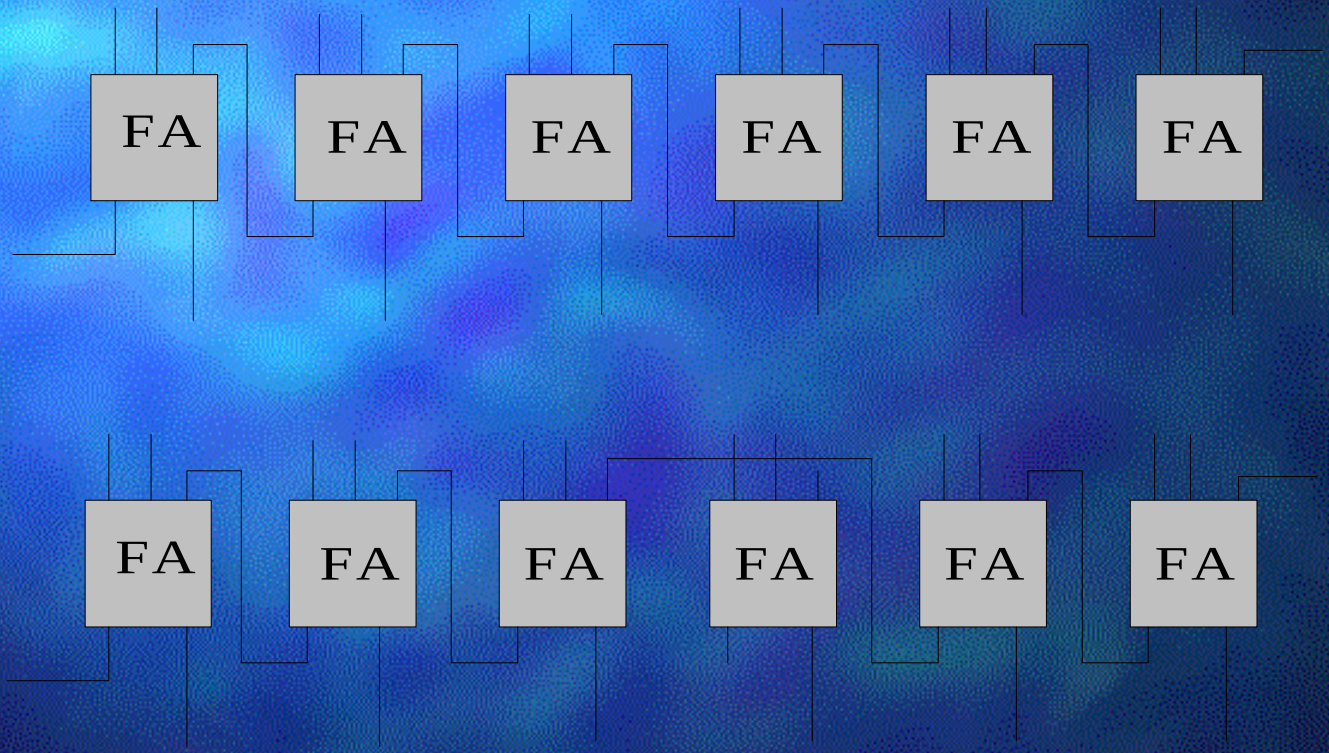
Registers And Memory

- Registers, with Variations are Directly Simulated by FHDL
- Registers Act as Collections of Unsynchronized D Flip-Flops
- RAM Can be Simulated Directly
- Simple ROMs and PLAs are Easy to Construct

Sequential Control

- State Machines Can be Constructed from Basic FHDL Components
- The ROM Preprocessor Allows RTL-Like Specifications
- The PLA Preprocessor Can Be Used To Create More Exotic State Machines
- ASMs can be specified and simulated directly

Testing: An Example



Distinguish Between The Two

How To Do It

Test 1

0,0,1,0,0,0
0,0,0,0,0,0

Presence or Absence of Carry Must Be Detectable

Test 2

0,0,1,1,0,0
0,0,0,1,0,0

Carry Must Be Tested In Both States

Unacceptable Substitute For Test 2!

0,0,1,1,1,0
0,0,0,1,1,0

Error Is Masked, Because C2 and C3 are Identical in Both Tests

Future Development

Fault Simulation in FHDL

Logic Design

With Simulation Exercises

Philosophy

- All Theoretical Subjects Should Be Backed Up With Practical Exercises
- Students Should Verify All "Paper" Designs to Verify Their Correctness
- Students Should Verify Exhaustive Proofs by Using an Exhaustive Simulation
- Students Should be Exposed to the Principles of Testing

Fundamental Laws

Equation	Law
a•1=a	And Identity Law
a+0=a	Or Identity Law
ab = ba	Commutativity of and
a+b = b+a	Commutativity of or
(ab)c = a(bc)	Associativity of and
(a+b)+c = a+(b+c)	Associativity of or
a(b+c) = ab+bc	And distributes over or
a+bc = (a+b)(a+c)	Or distributes over and
aa’=0	And inverse law
a+a’=1	Or inverse law

Identity Laws

FHDL

```
Ident:  circuit
        inputs  a
        outputs xa,xo

        one     i1
        zero    i0
g1:     and     (a,i1),xa
g2:     or      (a,i0),xo
        endcircuit
```

VECTORS

```
0
1
```

Associativity of AND

FHDL

```
aa:     circuit
        inputs  a,b,c
        outputs left,right

g1:     and     (a,b),l1
g2:     and     (l1,c),left

h1:     and     (b,c),r1
h2:     and     (a,r1),right
        endcircuit
```

VECTORS

```
0,0,0
0,0,1
0,1,0
0,1,1
1,0,0
1,0,1
1,1,0
1,1,1
```

Karnaugh Maps

- Start with A Truth Table
- Construct A Boolean Function
- Translate the Boolean Function into FHDL
- Simulate The FHDL to Reconstruct the Truth Table

Design With Gates

- Determine Number of Inputs & Outputs
- Derive Truth-Tables for Outputs
- Derive Boolean Functions for each Output
- Draw Logic Diagram
- *Encode Logic Diagram in FHDL*
- *Simulate FHDL to Verify Correct Operation*

Design with MSI Functions

- Implement Example MSI Functions
- Simulate to Verify Correctness
- Design using MSI Components Directly
- Implement Using FHDL Functional Blocks and Hierarchical Design
- Simulate to Verify Correctness

Sequential Circuits

- Design and Simulate RS, D, and Edge-Triggered D flip-flops
- Illustrate Oscillations using the RS flip-flop
- Illustrate The Need for Edge-Triggering, or Master/Slave Synchronization
- Illustrate Logic Hazards, and the Danger They Pose to Asynchronous Circuits

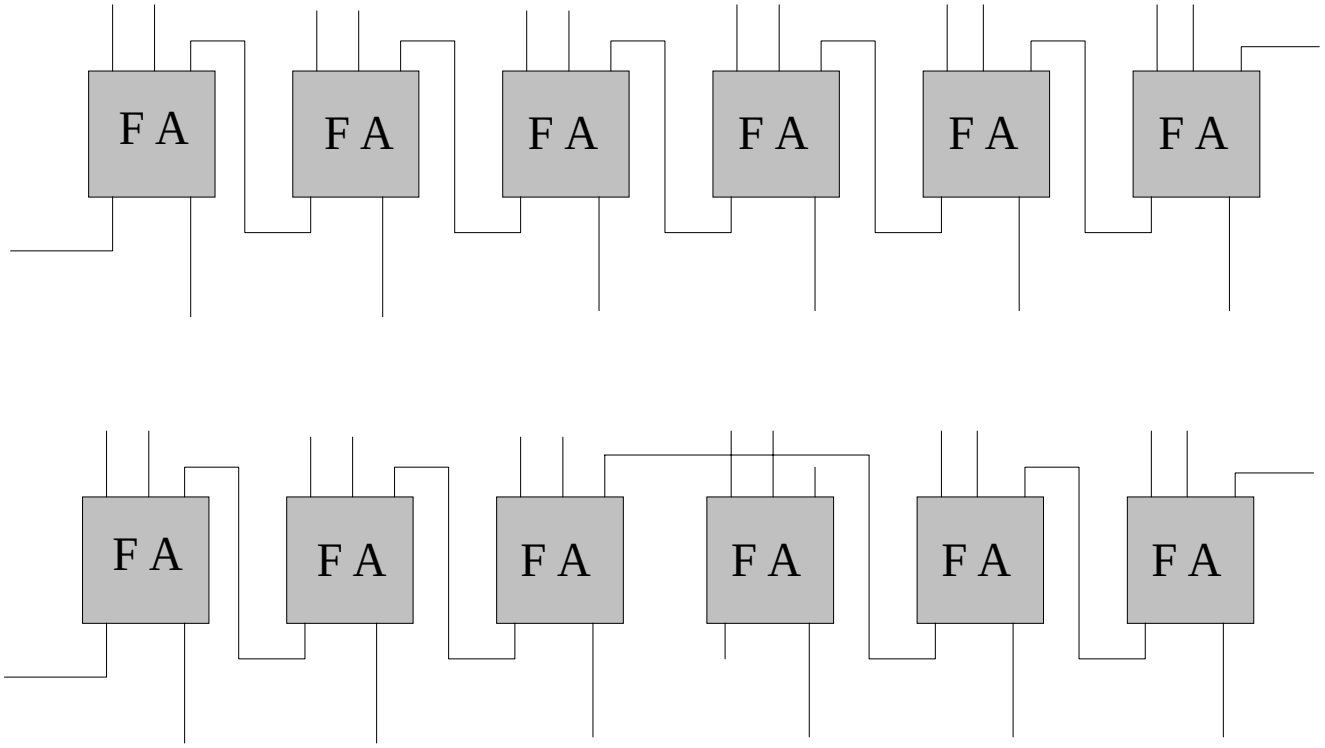
Registers And Memory

- Registers, with Variations are Directly Simulated by FHDL
- Registers Act as Collections of Unsynchronized D Flip-Flops
- RAM Can be Simulated Directly
- Simple ROMs and PLAs are Easy to Construct

Sequential Control

- State Machines Can be Constructed from Basic FHDL Components
- The ROM Preprocessor Allows RTL-Like Specifications
- The PLA Preprocessor Can Be Used To Create More Exotic State Machines
- ASMs can be specified and simulated directly

Testing: An Example



Distinguish Between The Two

How To Do It

Test 1

0,0,1,0,0,0
0,0,0,0,0,0

Presence or Absence of Carry Must Be Detectable

Test 2

0,0,1,1,0,0
0,0,0,1,0,0

Carry Must Be Tested In Both States

Unacceptable Substitute For Test 2!

0,0,1,1,1,0
0,0,0,1,1,0

Error Is Masked, Because C2 and C3 are Identical in Both Tests

Future Development

Fault Simulation in FHDL