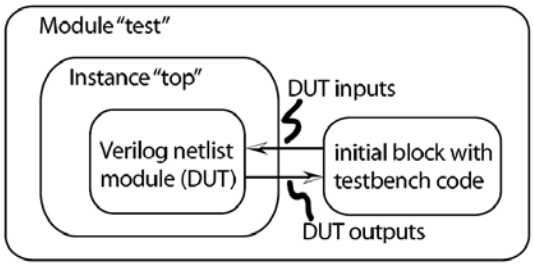
TESTER HARDWARE OVERVIEW

CS/ECE 6712

What's an ASIC Tester?

- Like a logic analyzer
  - But with a pattern generator so that you can program in the sequence of signals to send to your chip
  - With a comparison module that compares the outputs from your chip to a list of what you expected to see
  - Similar to building a self-checking testbench in Verilog…



What's an ASIC Tester?

- Like a logic analyzer
  - With programmable electronics so that you can modify the voltages that get sent to your chip (Vdd, Vih, Vil), and the voltages the tester uses for comparison (Vol, Vil).
  - With detailed timing for when signals are sent, and when comparisons are made
  - With programmable "schmoo" capability
    - The ability to define a series of tests where a set of different timing or voltage parameters are used

A Tale of Two Testers

- LV500
  - Ancient (1990-vintage) 25MHz tester
  - I know a lot about how to use it
  - But, it may not be as healthy as we'd like…
- Verigy 93000
  - Relatively modern (2005-vintage) 100MHz tester
  - But, I know very little about how to use it
  - And it may not be completely healthy either…

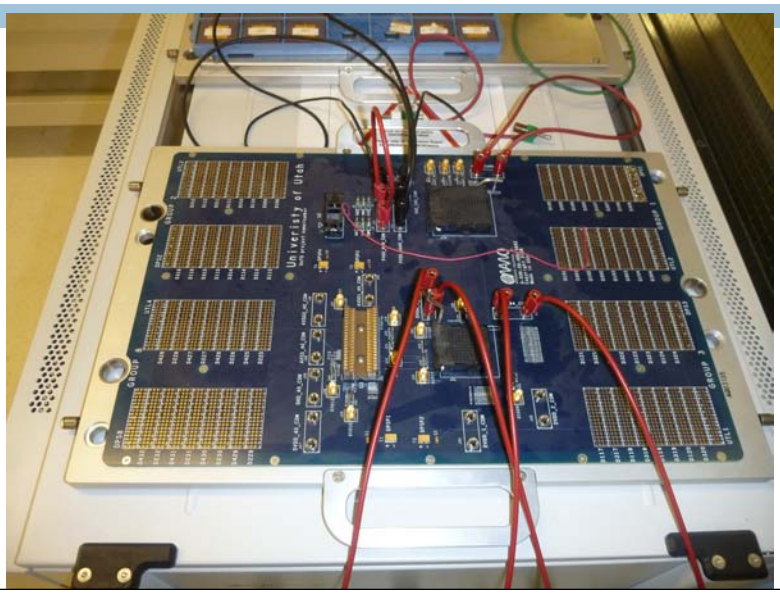
Verigy 93000

- Originally HP, then Agilent, now Verigy
- Scalable tester architecture
  - This means there are different versions with different speeds and capabilities
  - We have a fairly low-end version, but it's still pretty fancy
  - "Breaks the $1000/pin cost barrier"
    - Yikes! Consider that a small test head may have 128 to 256 pins…
  - Complex beast – we don't have good, simple tutorials yet…

Verigy 93000

Verigy 93000

Verigy 93000



Verigy 93000

Verigy 93000

Verigy Bottom Line

- Ken Stevens is working on a tutorial example
  - We're trying to understand some issues related to pattern generation for the test files
  - We're trying to understand some issues with how signals are mapped to the test head
  - We're trying to understand some issues with the thermal circuit breaker that seems to keep flipping
- We probably won't be able to use it this semester, but there's always a chance!

Testing with the LV-500

- Tektronix LV-500
- Built in 1989-1991
  - i.e. Ancient technology!
  - eBay is a source for spare parts these days…
- Specifically designed to be a stand-alone tester for ASICs
  - Based on a Tektronix DAS 9200 logic analyzer
    - Which in turn is based on a Motorola 68000 processor…
  - LV500 version has more testing features than a basic logic analyzer

LV500

- The main differences from DAS 9200 are in the test head, the pattern/error cards, and the Schmoo
  - The test head has up to 256 bi-directional pins where each pin has programmable electronics
    - voltage drive, current drive, voltage sense, etc.
  - The pattern/error cards store and compare the test vectors at up to 50MHz
    - fast for 1989!
  - A Schmoo lets you run repeated tests while the tester alters one or two independent variables like threshold, delay, cycle length, voltage, etc.

Flavors of LV500s

- Common Features
  - Based on a Motorola 68000 microprocessor
  - Test speeds up to 50MHz
  - Up to 64,000 unique test vectors
  - Network connection for uploading tests
    - Thinlan ethernet
  - 8 Meg of RAM
  - 21 or 43 Meg hard drive
  - 5.25 floppy (1.2M floppy)

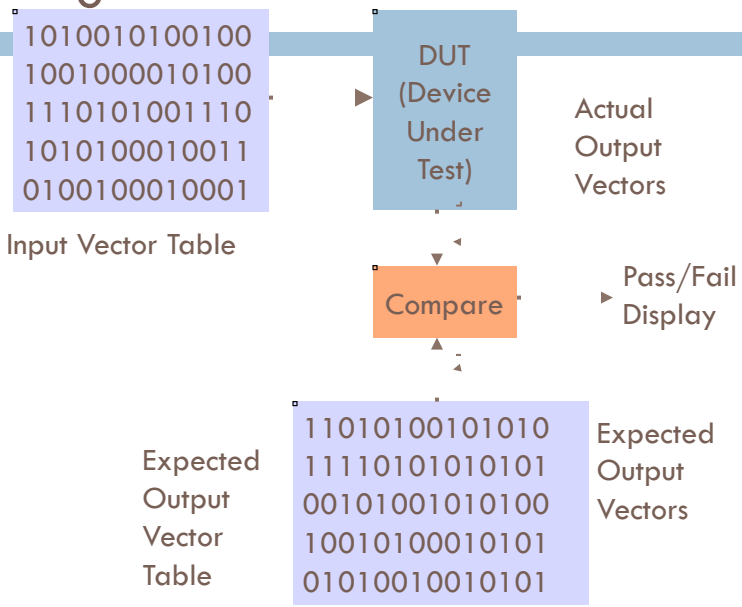
Flavors of LV500s

- LV514
  - 192 test channels (12 sectors)
    - 160 are usable (two sectors may be bad)
  - Pre-wired test card for class chips
  - *(should really be called LV513, but that's a long story)*
- LV512
  - 128 test channels (8 sectors)
    - All channels are usable
  - Used mostly for tutorial purposes
  - Also has a (different) pre-wired DUT card

LV514

LV512

The Big Picture



The More Detailed Picture

- Conceptually this is simple, in practice there are lots of details…
  - Define the input and expected-output vectors
    - Can do this using your Verilog simulations
  - Define which signals are inputs and outputs on your chip
  - Define how those signals are mapped to tester channels
  - Wire up the DUT card so that those channels map to your chip pins
  - Define the timing and electrical characteristics of your test

Three Essential Parts of a Test

1. A properly wired DUT (Device Under Test) card
   - This electrically connects each of your chip pins to the correct tester channels
2. A properly configured LV-500
   - Configure the timing of when inputs are applied, when outputs are checked, what the voltages and currents are, etc.
3. A complete set of test vectors
   - Vectors are applied and checked on each cycle
   - "Force data" are inputs to your chip
   - "Compare data" are expected outputs from your chip
   - "Mask data" are pins that are ignored (not driven or checked)

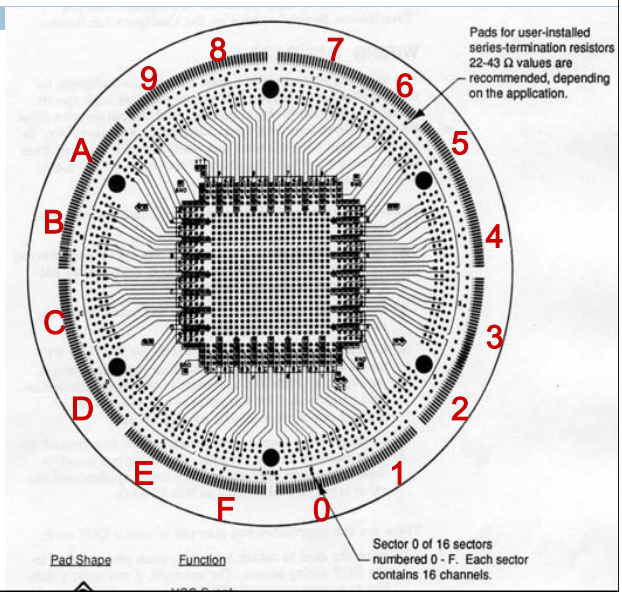
More Details

- Pin mapping
  - Describe how your chip pins map to tester channels
- Tester config
  - Describe the different voltages, etc
- Timing description
  - Describe when to apply inputs and when to check for outputs
- Test patterns
  - Describe what to send to the DUT on each cycle, and what to check for on the DUT outputs

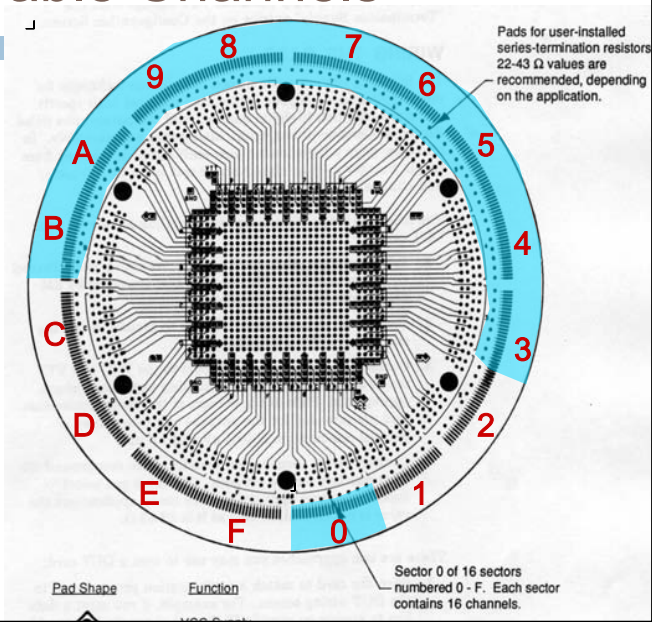
Tester Channels

- 256 possible pins on the test head
  - There are 16 "sectors" labeled 0-f
  - Each sector has 16 "channels" labeled 0-f
  - Each pin is defined as sector.channel (i.e. 0.2, d.3, a.c)
- On each cycle, each pin may be either a "force" channel or a "compare" channel, but not both
  - If you have bi-directional pins on your chip, you need to define which are inputs and which are outputs on each cycle!
  - Or a pin can be "mask" and thus be ignored

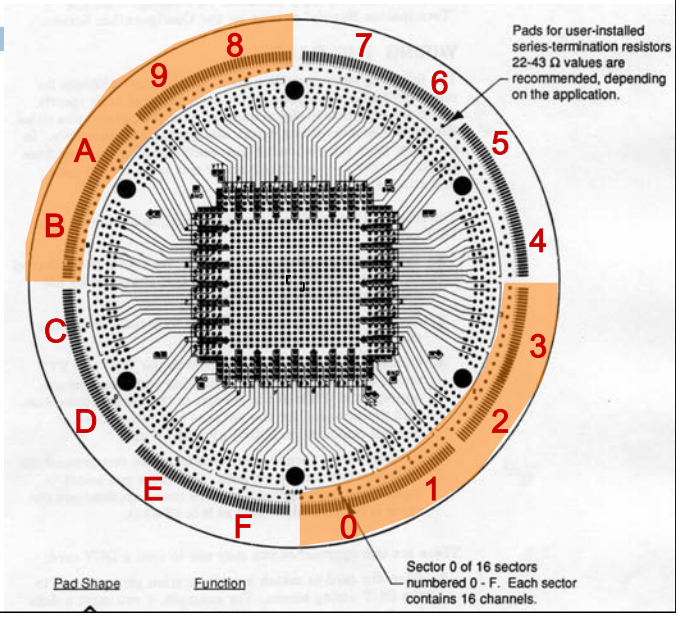
DUT Card with Sectors Marked



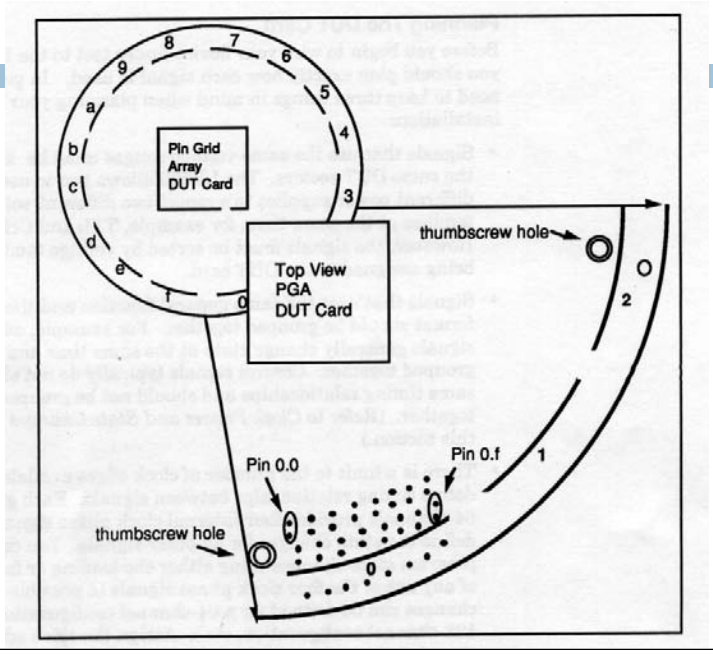
LV514 Usable Channels



LV512 Usable Channels

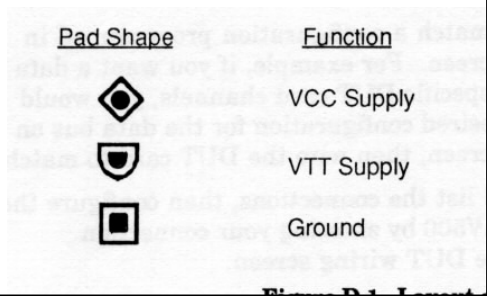


DUT Card Sectors & Channels



DUT Cards

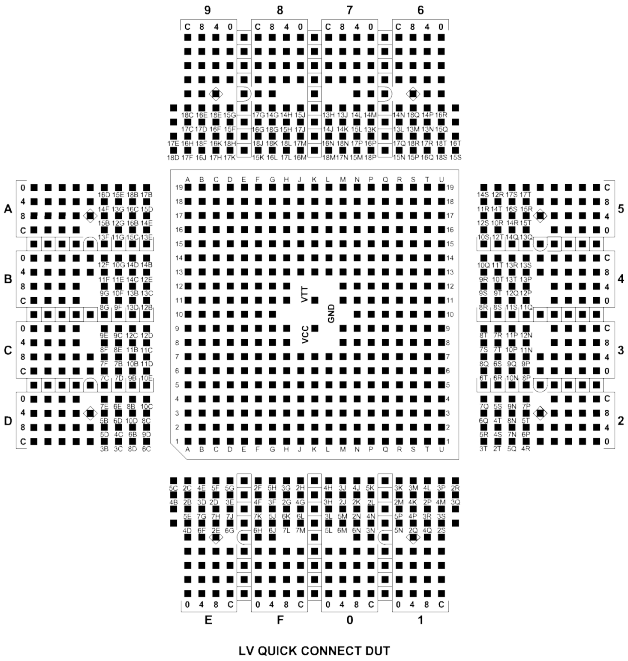
- The DUT cards are how you wire from tester channels to chip pins
- These cards also have VDD, VTT and GND power supply connections
  - VDD and VTT are two independently controllable power supply voltages



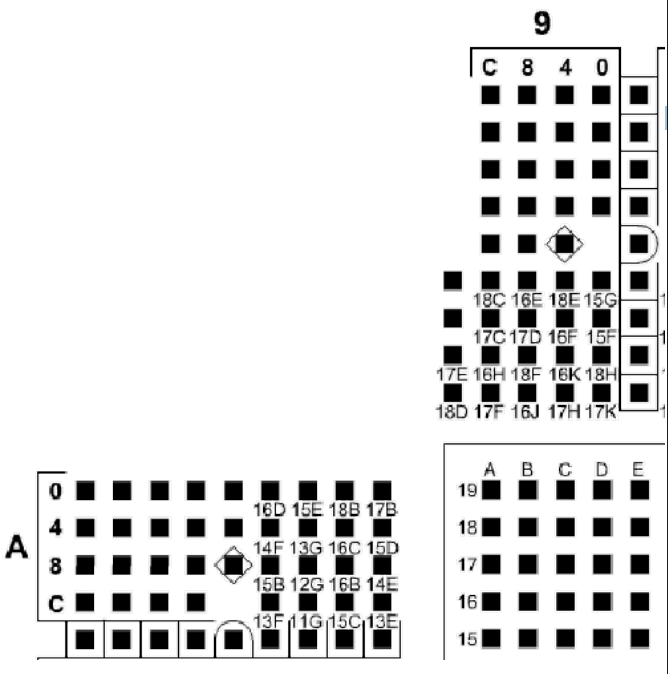
Wiring the DUT Card

- Essentially two choices:
  - Solder wires on a PGA DUT card
    - Remember that VDD and GND are not connected to tester channels
    - Probably only want to do this once for the whole class
    - Which means standardizing VDD and GND!
  - Use a "Quick-Connect" card
    - Uses 3M Scotch-Connect to wire (using wire-wrap wire) from the tester channels to the chip socket
    - Can also use quick-connect for VDD and GND

Quick-Connect DUT Card



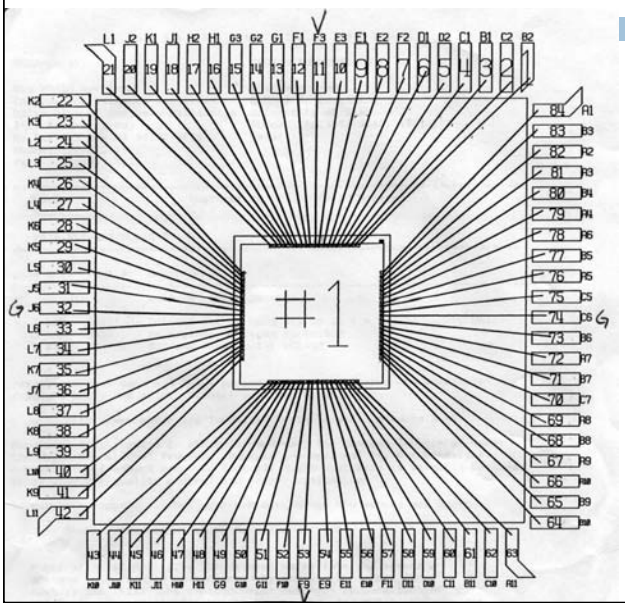
Quick-Connect DUT Card

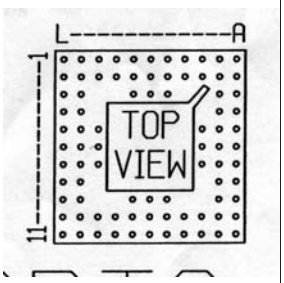


Knowing What to Wire

- A "Bonding Diagram" is a picture that shows how your chip was bonded to the chip frame
- It also shows how the chip frame is connected to the chip pins

Bonding/Chip Diagram





Map Your Pins to Channels

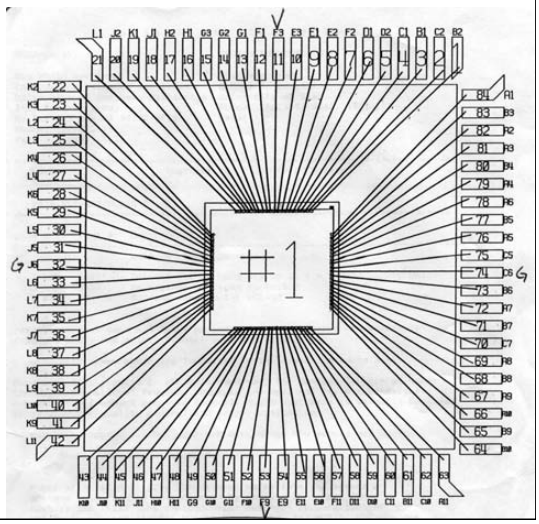
- Pick tester sector.channel assignments for each of your pins
  - Signals that need the same voltage characteristics should be grouped in the same sector
    - Each sector gets common voltage ranges
    - More on this later…
  - Signals that need the same timing should be grouped in the same quadrant
    - Sectors 0-3, 4-7, 8-b, c-f are the four quadrants
    - More on this later…
  - Wire things up!
    - Remember to keep a list of what you've wired!

Class DUT Card

- Pre-wired for class chips
  - 84 pin PGA with specific VDD and GND placements in the pad ring
  - On class web page http://www.eng.utah.edu/~cs6712
  - Two different DUT cards
    - DUTmap.txt: for the LV514 tester
    - DUTmap-new.txt: for the LV512 tester

DUTmap.txt

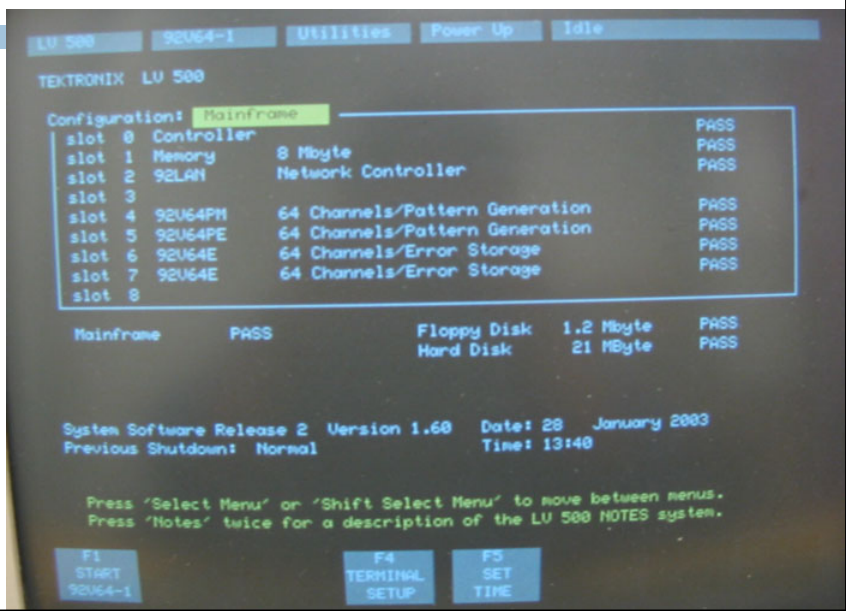
PAD	TESTER PGA	SIGNAL sec.chn	NAME (no spaces)
1	B02	6.C	
2	C02	7.7	
3	B01	6.B	
4	C01	7.6	
5	D02	7.D	
6	D01	7.C	
7	F02	GND	GND
8	E02	8.1	
9	E01	8.0	
10	E03	8.7	
12	F01	8.6	
13	G01	8.A	etc.…



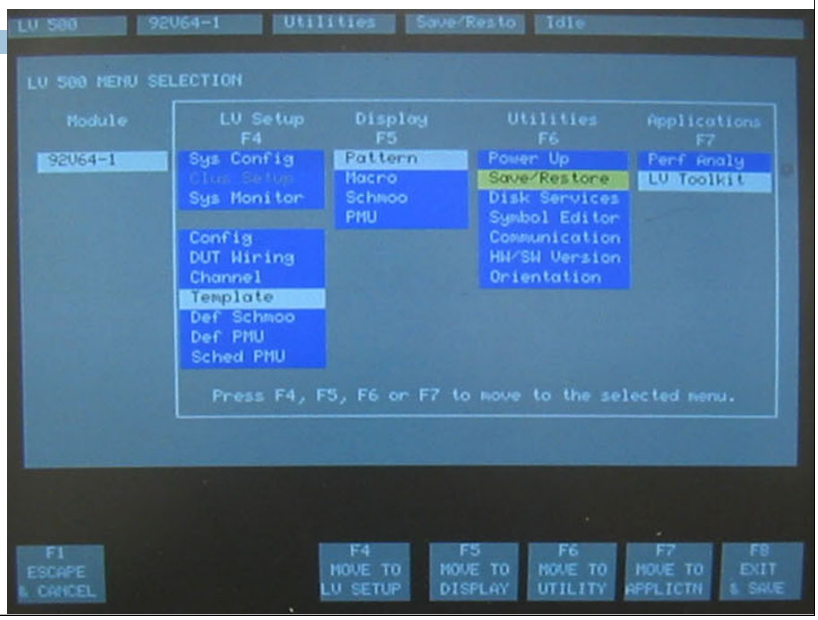
Finished DUT Card

- Now you have part 1 – a wired DUT card that connects your chip to the tester
- On to part 2 – configuring the tester…

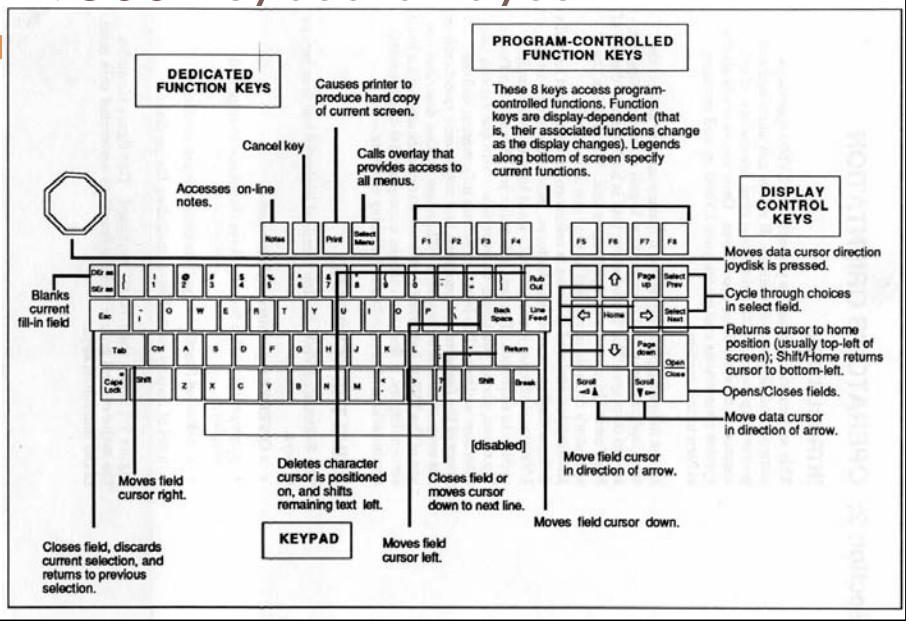
LV512 Boot Menu



LV500 Main Menu



LV500 Keyboard Layout



Important Menu Choices

- Config Menu
  - Defines voltages for VDD, VTT, GND
  - Defines voltages for two force/compare sets
- DUT Wiring menu
  - Defines how your signals are assigned to tester sector.channels
- Channel menu
  - Defines how your signals are collected into groups (i.e. buses)
    - ALL signals must be a part of some group
    - Groups are assigned to specific timing templates (clock phases)

Important Menu Choices

- Template Menu
  - Defines timing of tests
    - How long is a "test cycle?"
    - When to force data within that cycle?
    - When to compare data within that cycle?
- Schmoo Menu
  - Defines which variables to vary, and by how much
- Pattern Menu
  - Defines data vectors (force and compare) for each tester cycle

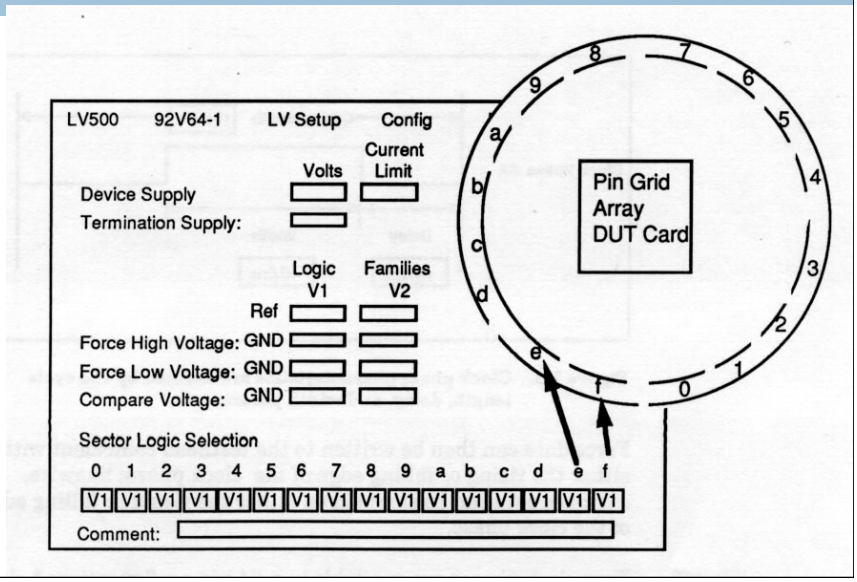
Basic Procedure…

- Tell tester which chip signals are connected to which channels (DUT wiring menu)
- Combine signals into groups (Channel menu)
- Define timing for each group (Template menu)
  - Only four "clock phases" per quadrant
  - A "template" assigns clock phases to groups, and timing of clock phases…
- Define patterns (Pattern menu)
  - Each pattern starts with a template
  - Includes force, compare, and mask data for each test cycle

Config Menu

- Defines the electronics for this test
  - VDD, VTT, GND, current limit, etc.
- You can also define two different "force" and "compare" voltage sets for data channels
  - Each sector uses one of these two sets

Config Menu (diagram)



Config Menu (LV512)

DUT Wiring Menu

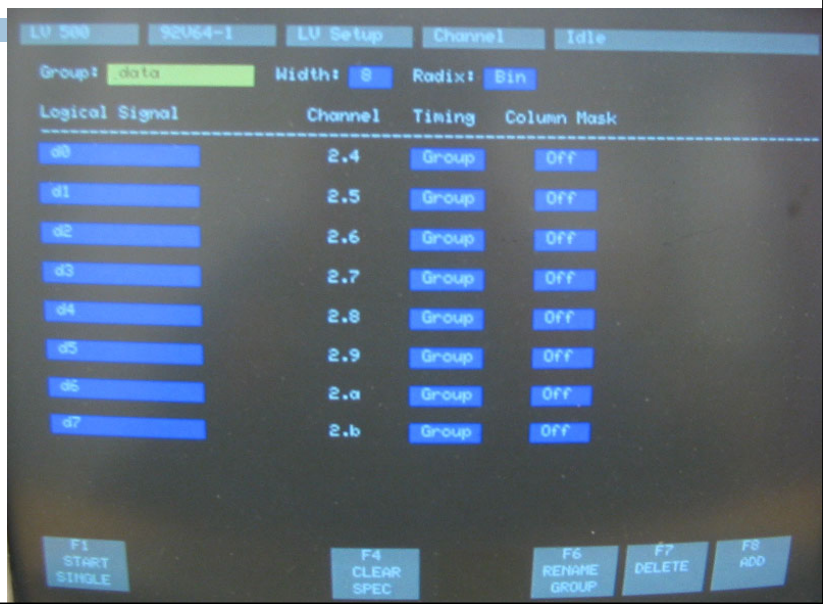
- Defines how your signals are assigned to tester sector.channels
  - List signal names
  - Define which tester channels they connect to
  - Optionally define which actual chip (DUT) pins they are connected to
    - This is just a comment for documentation

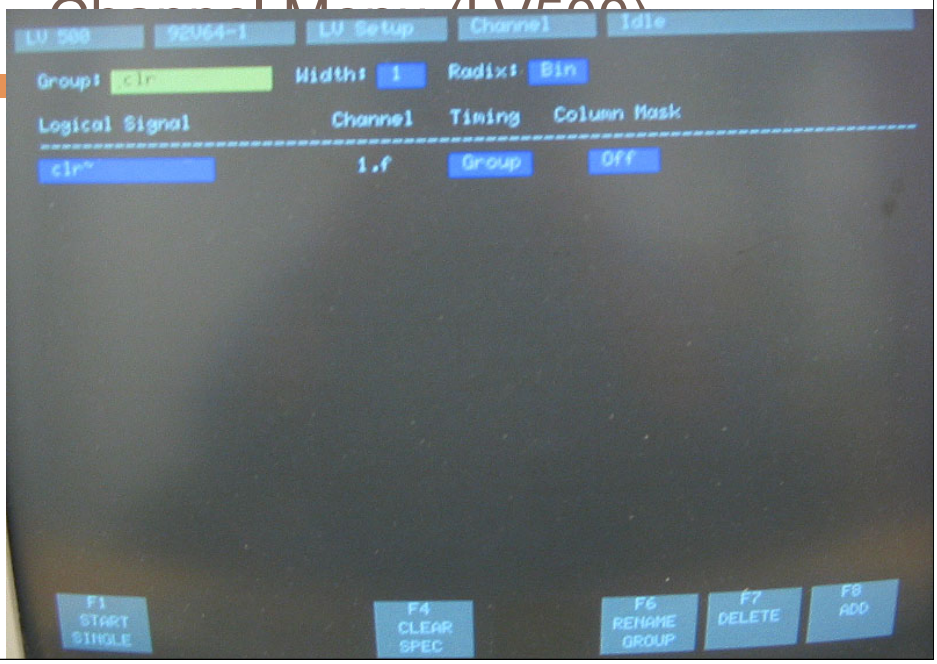
DUT Wiring Menu

Channel Menu

- Defines how your signals are collected into groups
- EVERY signal must be a part of some group (even single signals)
  - Groups can make data entry and evaluation easier
  - Can define how group data is printed
    - Dec, Hex, Oct, Bin
  - Can specify timing once for the whole group
  - In general, inputs vs. outputs is a good group…
    - Or control vs. data, etc.

Channel Menu (LV500)



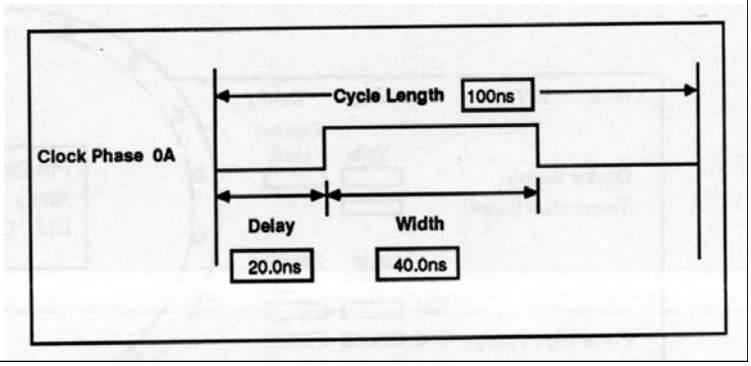


Templates

- Templates
  - Defines timing of tests
    - When to force data?
    - When to compare data
    - When to ignore data?
  - Set up using a "clock phase"
    - Bad name – really a timing waveform
    - Defines when things happen in each tester cycle
  - You can define up to four clock phases per quadrant

Clock Phases

- Cycle Length: 20ns – 496ns
- Delay is delay to Leading Edge
  - Can be 0ns
- Width is delay from Leading to Trailing edge



DUT Card Quadrants

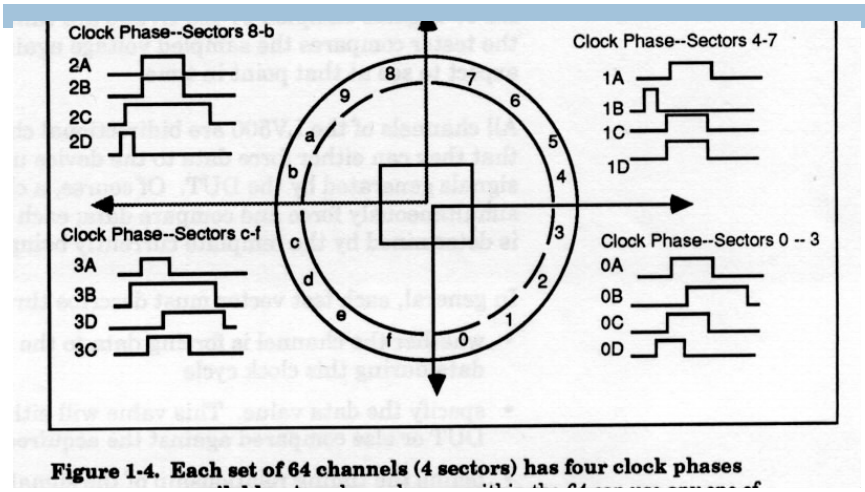
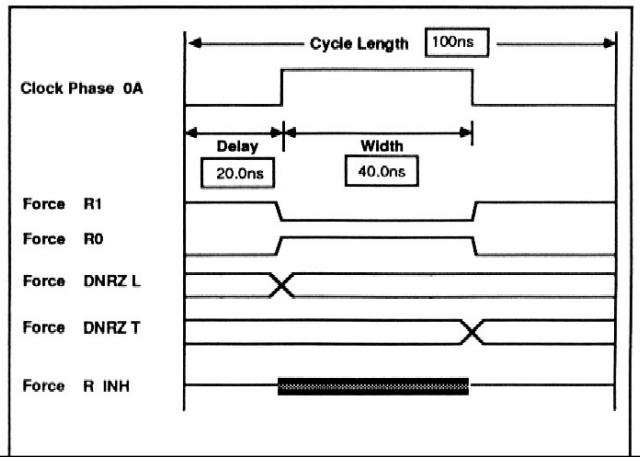


Figure 1-4. Each set of 64 channels (4 sectors) has four clock phases

Each quadrant has up to four timing waveforms you can use to control signal timing (called "Clock Phases" in LV500-speak)

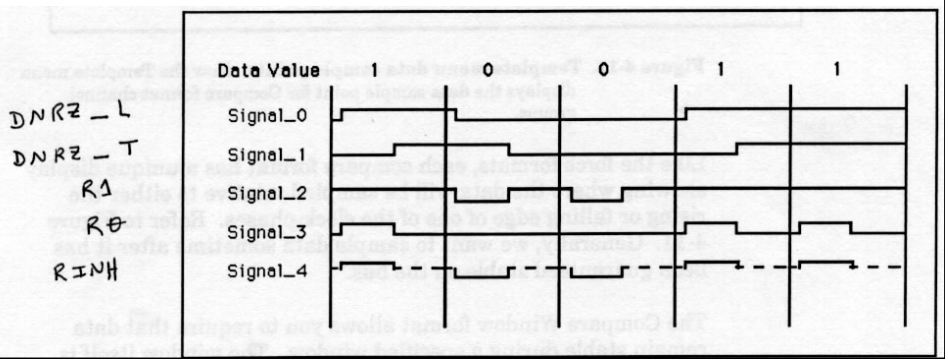
Force Formats

- Within a clock phase, you can define when values are "forced" to your chip in relation to the edges



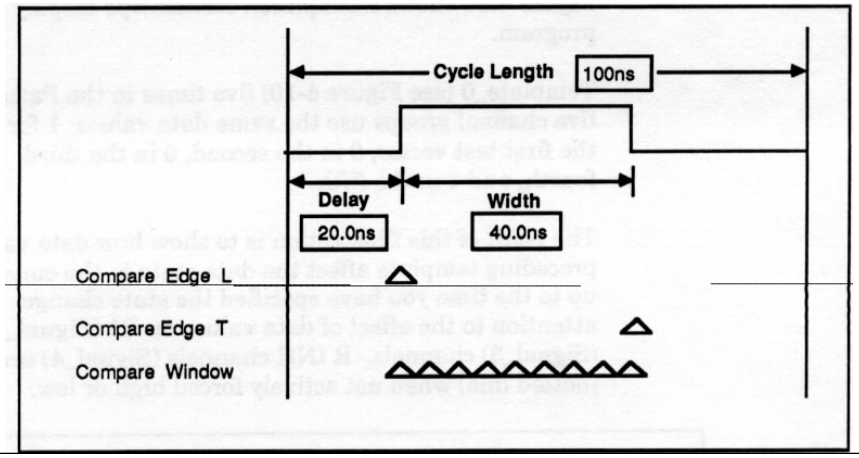
Force Formats Example

- This is an example of a pattern driven on five consecutive tester cycles with each of the different force formats

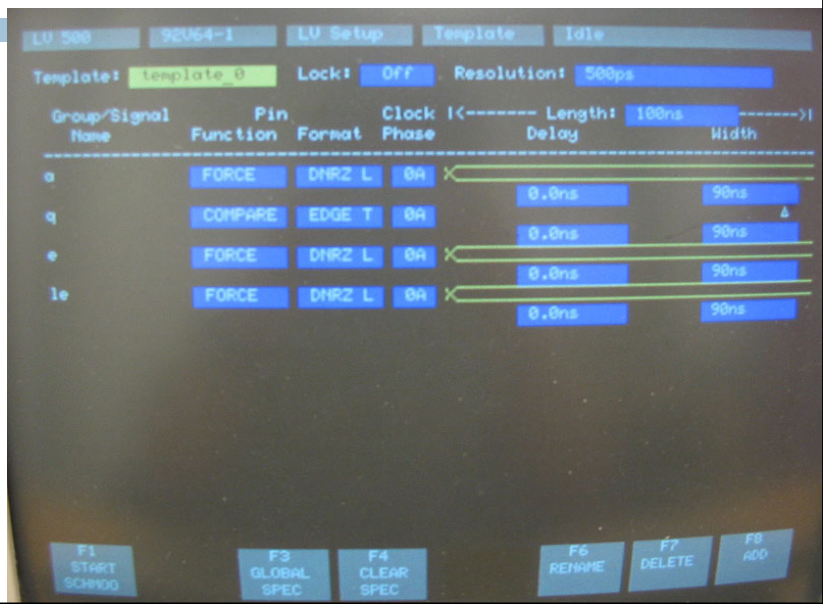


Compare Formats

- You can also define when you Compare outputs in relation to the clock phase edges



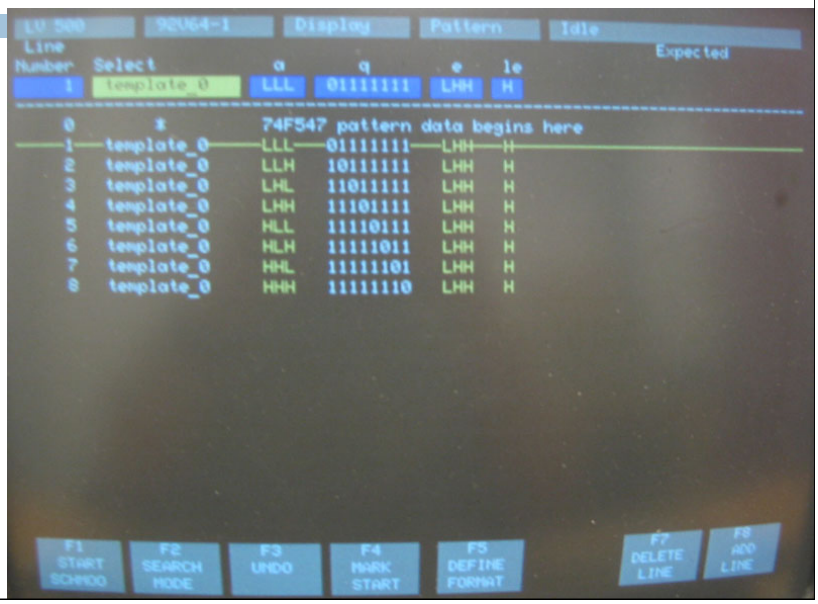
Template Menu



Pattern Menu

- Defines data vectors for each tester cycle
  - Data for each signal is defined in the data vector
  - Some of those signals are "Force", some are "Compare" and some are "Mask"
    - These are set in the templates
    - Assign a template to each vector
    - On each tester cycle, the next vector, with that vector's template, is applied to the DUT and compared

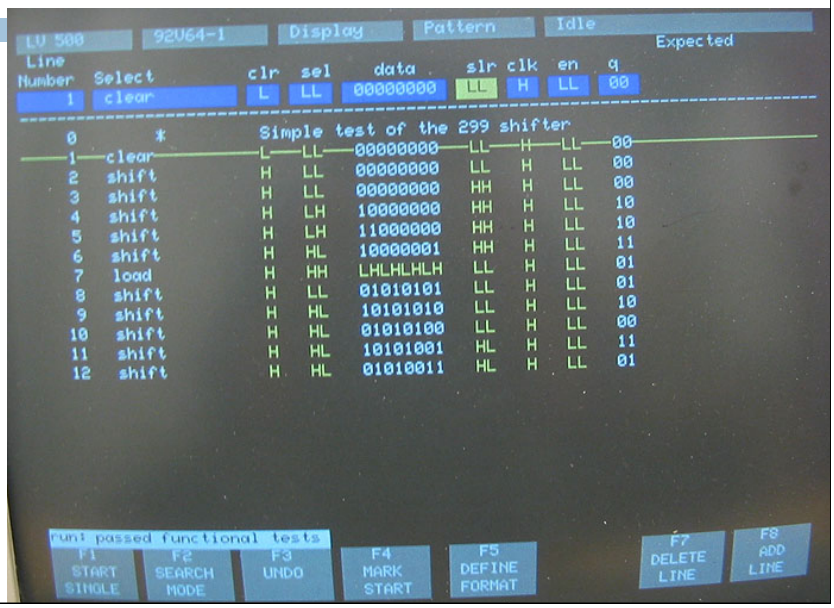
Pattern Menu



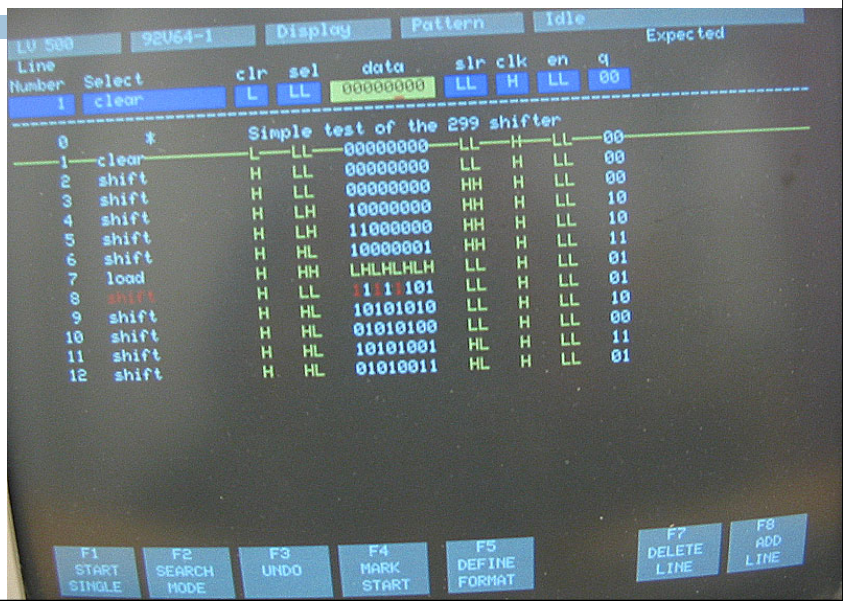
Pattern Display

- The Pattern screen is where you see the results of your test
  - Before the test you can see all the vectors (and their templates) that you will be using
  - After running the test you see the same display with any errors highlighted in red
    - Red means that the output of the DUT didn't match the expected output vector
  - You run the test with F1-Start (the F1 function key)

Successful Test



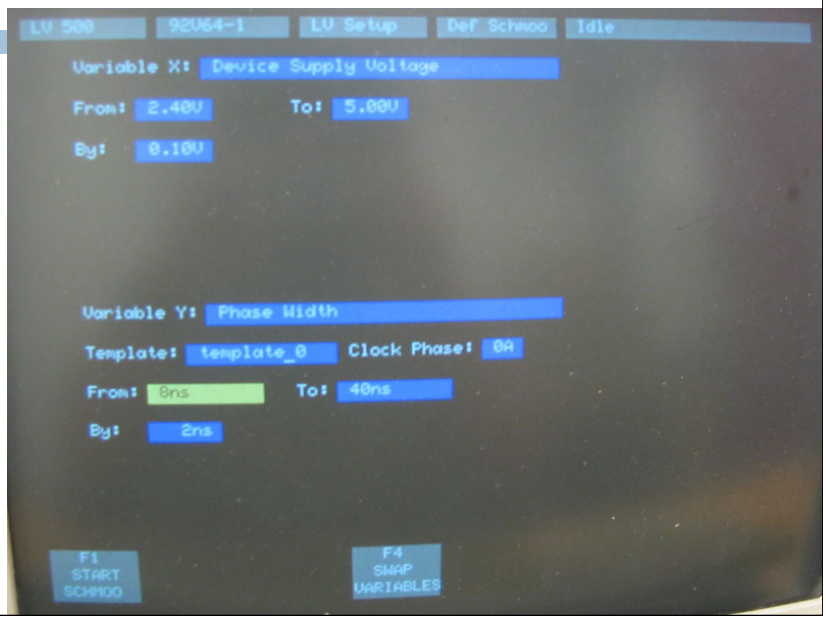
Failed Test



Schmoo Menu

- After you have your basic test working, you can run a Schmoo test
  - Repeat the test while changing 1 or 2 variables
    - Variables can be things like VDD voltage, delay time, cycle time, compare voltage, etc.
  - Generates a graph showing where the part worked or didn't work

Schmoo Menu



Schmoo Result

Logistics

- The LV500 is old and cranky…
  - Basic rule – if you're not *SURE* about what you're doing, ask me first!!!!
    - Replacement parts are very hard (impossible?) to find.
  - Leave terminal ON
    - Turn down brightness when you leave,
    - Check brightness when you come into the lab
  - Do NOT turn the LV500 off without good cause!
    - We'll leave the LV512 up and running for tutorials, and then switch to the LV514 when chips come back…

Logistics continued

- Be very gentle with the DUT cards
  - They connect to the machine through elastomer connectors
    - These are basically rubber-like connectors wrapped with wire
    - They are very fragile, and a little worse for wear
    - We have no replacements…
- Schedule some time with me to run tests!
  - Once you've got some LV500 time under you belt you can go it alone…

Tester Setup Simplified

- All this stuff can be defined in a .msa file
  - Module Setup, Ascii
  - Each section of the .msa file corresponds roughly to a tester menu
  - You can (fairly easily) write your own .msa file
    - Templates and examples on class web page http://www.eng.utah.edu/~cs6712

MSA File

```
/*****************************************************************/
/* config section */
/*****************************************************************/

resolution = 500ps;

dev_supply_voltage     =       5.00v;
dev_supply_current     =       1.00a;

term_supply_voltage    =       3.00v;

force_high_family_u    =       2.40v;
force_low_family_u     =       0.50v;
compare_family_u       =       1.40v;

force_high_family_v    =       4.50v;
force_low_family_v     =       0.50v;
compare_family_v       =       2.50v;

/* The "v" parameters are appropriate for CMOS */
sector_logic_family    =       {
        v, v, v, v, v, v, v, v, v, v, v, v
};
```

MSA File

```
/* address pins for the 74F547  */
group "a" {
        radix = bin;
        force_fmt = dnrz_1;
        compare_fmt = edge_t;
        phase = 0a;
        signal "a2" {
                dut = "17";
                sector = 0h2;
                channel = 0he;
        }
        signal "a1" {
                dut = "7";
                sector = 0h2;
                channel = 0hd;
        }
        signal "a0" {
                dut = "6";
                sector = 0h2;
                channel = 0hc;
        }
}
```

```
/* Active-low output pins       */
group "q" {
        …
     }

/* chip-enable pins             */
group "e" {
        …
     }

/* Active-low Latch Enable      */
group "le" {
        …
     }
```

MSA File

```
/* for this example, you want to force the address, enable, */
/* and latch-enable, and see which output is asserted (low) */
template "template_0" {
       cycle = 100ns;
       phase 0a {delay = 0ns; width = 90ns;}
       group "a" {
              function = force;
       }
       group "q" {
              function = compare;
       }
       group "e" {
              function = force;
       }
       group "le" {
              function = force;
       }
}
```

MSA File

```
/* This section must (I think) be the last section in the    */
/* .msa file.                                                */
/* Pin groups are used in the order in which they are defined */
/* Here the E (enable) pins are e1~, e2, and e3              */
/* LE (latch enable) is active low                           */
Pattern
       *  "74F547 pattern data begins here";
       *  "templ     A     Q     E   LE";
       "template_0" 000 01111111 011 1;
       "template_0" 001 10111111 011 1;
       "template_0" 010 11011111 011 1;
       "template_0" 011 11101111 011 1;
       "template_0" 100 11110111 011 1;
       "template_0" 101 11111011 011 1;
       "template_0" 110 11111101 011 1;
       "template_0" 111 11111110 011 1;
```

Tester Setup with msa Files

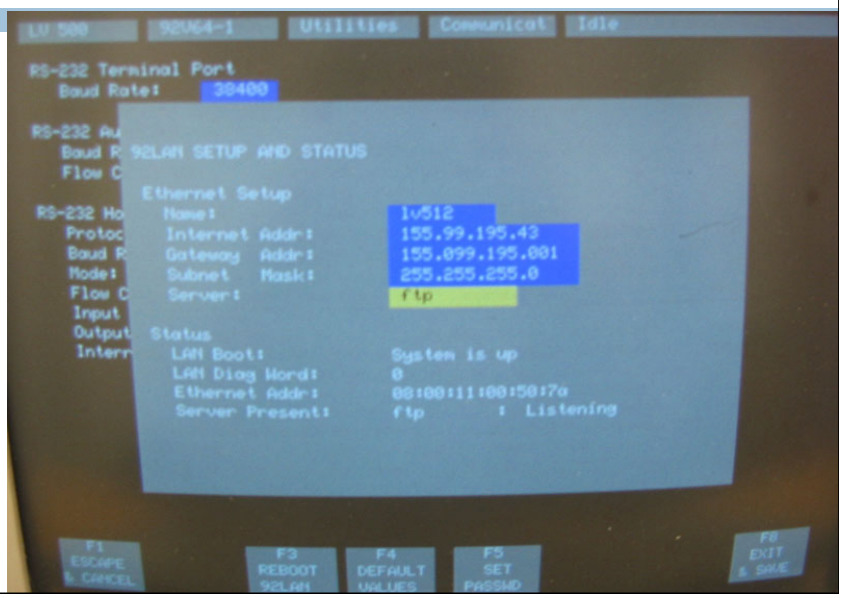
- You can ftp to the lv500 and upload the .msa file which defines your test
  - Should work from CS or CADE
  - lv512.cs.utah.edu, lv514.cs.utah.edu
  - ftp lv514.cs.utah.edu
  - No username/password required…
  - Put your .msa file into the Simulation directory on the LV500
  - Convert to tester setup using the LV Toolkit menu

Tester Setup with msa Files

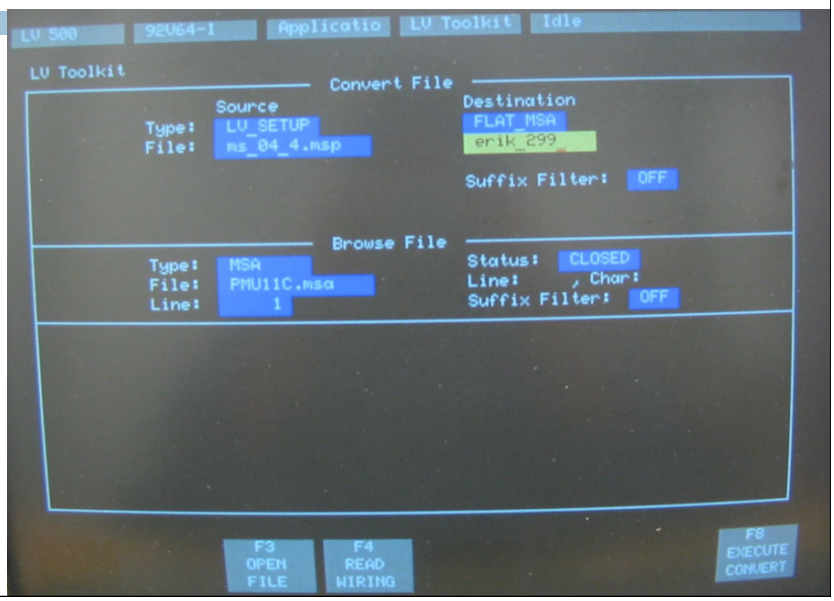
- You can ftp to the lv500 and upload the .msa file which defines your test
  - Should work from CS or CADE
  - lv512.cs.utah.edu, lv514.cs.utah.edu
  - ftp lv514.cs.utah.edu
  - No username/password required…
  - Put your .msa file into the Simulation directory on the LV500
  - Convert to tester setup using the LV Toolkit menu

This is not working at the moment!

LV512 LAN Screen



LV Toolkit Menu (LV512)



LV Toolkit Issues

- Note that the conversion process goes to an ms_04_4.msp file (or something close to that)
  - You are not allowed to change this name!
  - If you want to save this setup under a different name you need to convert to the standard name, and then save the setup to a new name using the Disk Services menu.
- Once the .msa is converted, you can look at the setup using all the previous menus

Running Tests

- The .msa conversion is a great first step
  - But, after that's running you may want to change things or try new things
    - Like Schmoo, or changing parameters
  - You can change the data using the menus shown earlier
  - You can also save the changed tests into new .msa files
  - And you can retrieve those new .msa files using FTP if you like

Overview

- On every tester cycle the LV500:
  - Applies a set of signals to the DUT
    - The data to "Force" is defined in the Pattern
    - Which signals are "Forced" on this cycle is defined in the template
    - When the data are applied is defined relative to the "clock phase"
    - The names of the signals and which tester channels they are on are defined in the DUT wiring menu
  - At the right time (defined in the template) the tester captures and compares the data from the DUT
    - Compares against the data in the Pattern

Procedure

1. Get your bonding diagram and map where your signals are on your chip
2. Decide how those pins will map to tester channels (DUTmap.txt)
3. Decide on timing templates for all signals
4. Generate test vectors that include pin names, templates, and data vectors for every cycle
5. Put it all in a .msa file

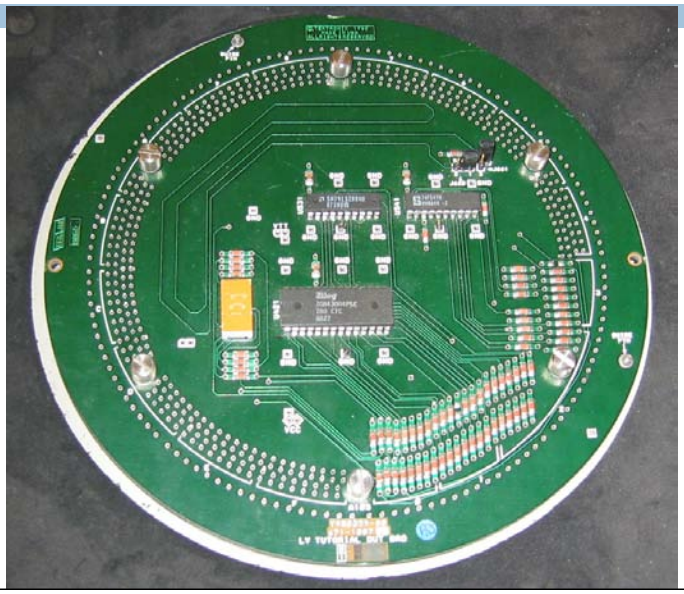
Procedure 2

6. Upload the .msa file to the LV500
7. Convert the .msa file to a tester setup file
8. Check all menus to make sure things are how you want them
    1. Config
    2. DUT wiring
    3. Channel
    4. Template
    5. Pattern

Procedure 3

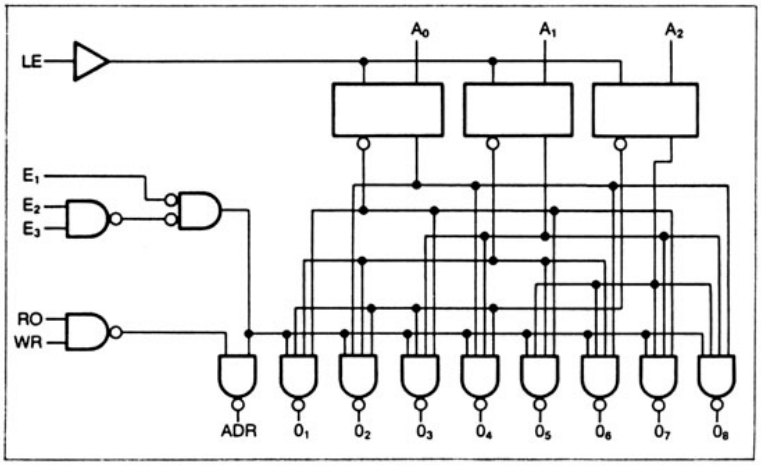
9. Fix or modify test parameters
10. Run your test
11. Look at the results
    1. Celebrate!
    2. Or diagnose and debug…
    3. Or decide to schmoo to get more info…

Tutorial DUT Card



Tutorial 1: 74LS547

- 3 to 8 decoder



74LS547

Table 4-3
PIN MAP FOR 74F547 DECODER

Signal	Sector/Channel	DUT Pin =
a2	2.e	17
a1	2.d	7
a0	2.c	6
q0~	3.6	12
q1~	3.7	2
q2~	3.8	1
q3~	3.9	19
q4~	3.a	18
q5~	3.b	8
q6~	3.c	9
q7~	3.d	11
le	3.2	16
e1~	2.f	15
e2	3.0	14
e3	3.1	13

(~ means negative-true logic)

Table 4-2
DECODER STATUS

Inputs			Latch	Status/Decoder Outputs
e1~	e2~	e3~	le	
L	L	L	L	Transparent Address inputs decoded
L	L	H	H	Storing Latched address decoded
L	H	L	H	Transparent q = HIGH
L	H	H	H	Storing
H	L	L	H	Transparent
H	L	H	H	Storing
H	H	L	H	Transparent
H	H	H	H	Storing

(~ means negative-true logic)

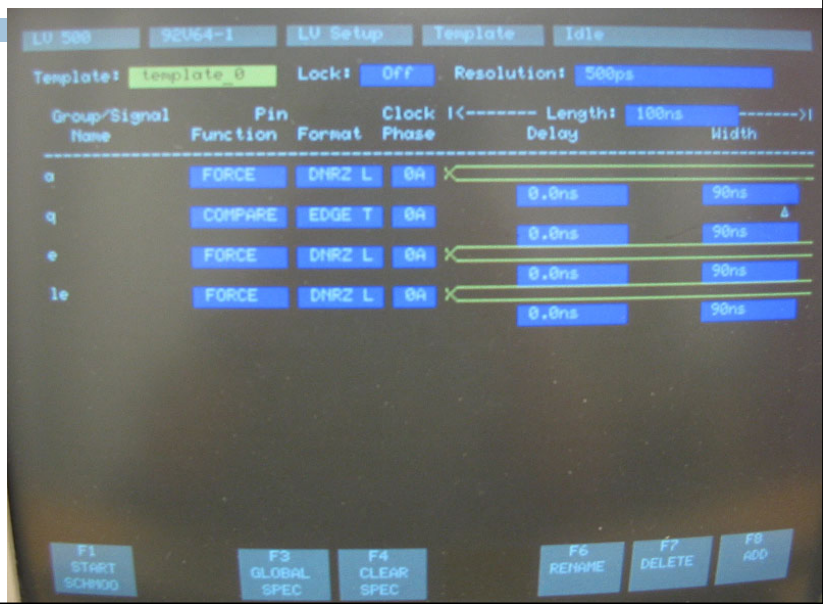
Table 4-1
TRUTH TABLE FOR DECODER

Inputs			Outputs							
a2	a1	a0	q0~	q1~	q2~	q3~	q4~	q5~	q6~	q7~
L	L	L	0	1	1	1	1	1	1	1
L	L	H	1	0	1	1	1	1	1	1
L	H	L	1	1	0	1	1	1	1	1
L	H	H	1	1	1	0	1	1	1	1
H	L	L	1	1	1	1	0	1	1	1
H	L	H	1	1	1	1	1	0	1	1
H	H	L	1	1	1	1	1	1	0	1
H	H	H	1	1	1	1	1	1	1	0

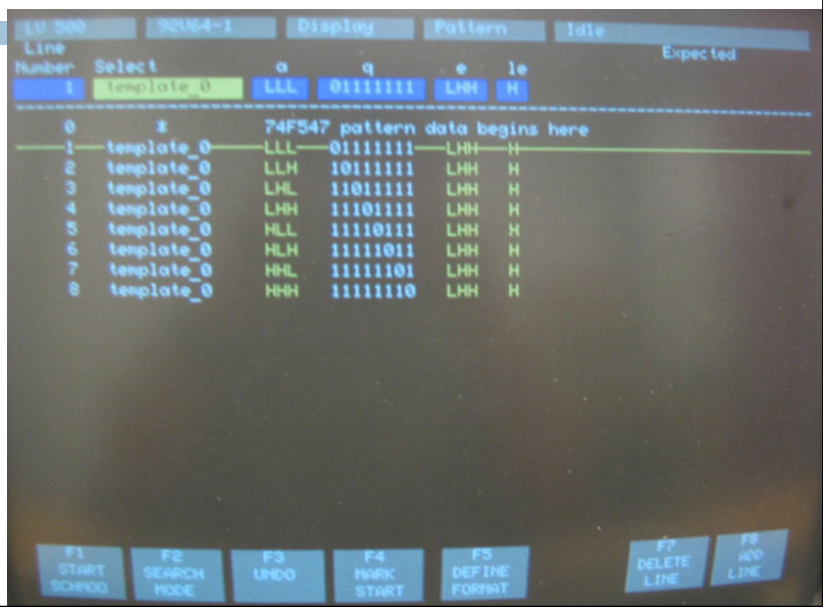
(~ means negative-true logic)

547 DUT Wiring

547 Template



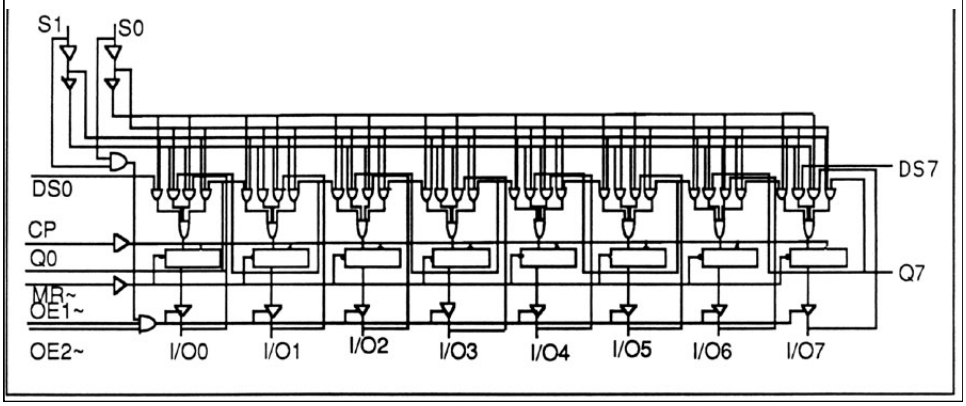
547 Pattern



547 Schmoo

Tutorial 2: 74LS299

- Shift Register, shift L or R, parallel load and output
  - Bidirectional data bus



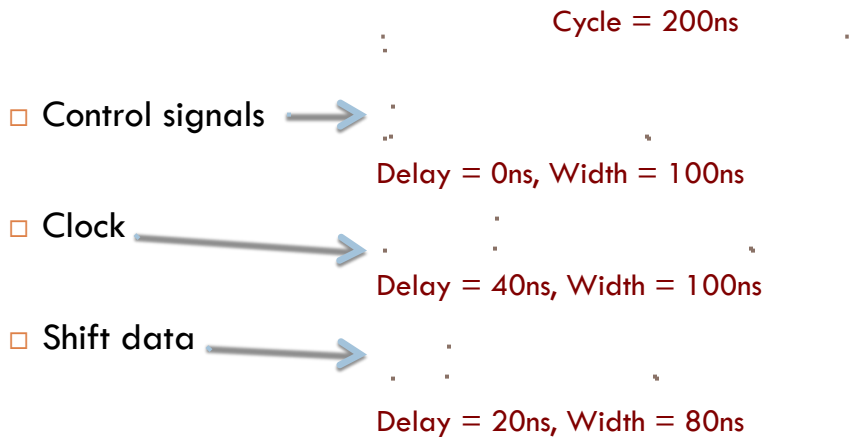
74LS299 Timing

- Control should be set up ahead of the clock
- Data should be sampled after the rising edge of the clock
- Data should be driven after the control is set up
  - Avoid drive fights on bidirectional path

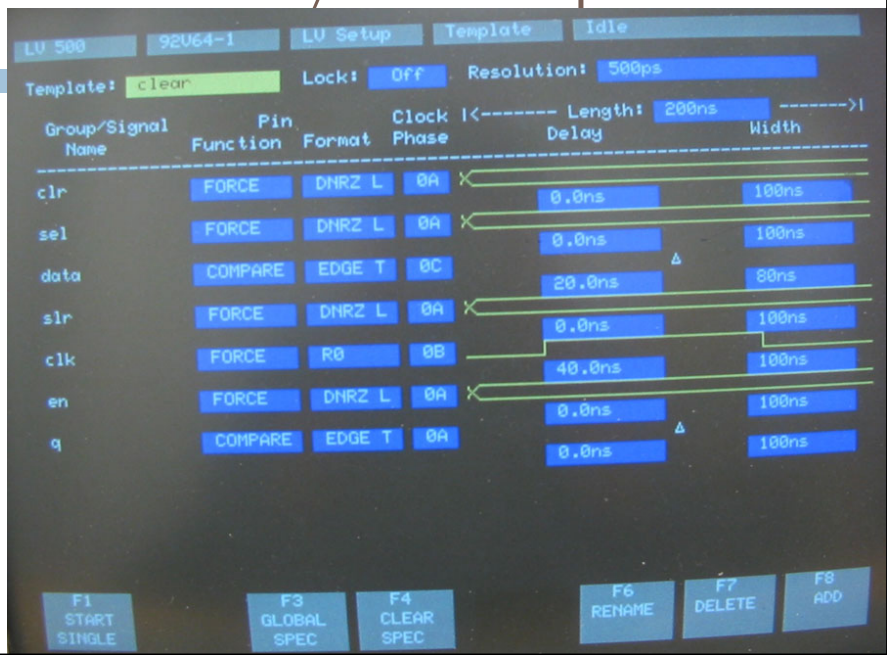
Table 6-1
FUNCTION TABLE FOR 74LS299 SHIFT REGISTER

Inputs				Response
clr~		s1	s0	clk
L	X	X	X	Asynch Reset;d[7:0]=Low
H	H	H	0->1	Parallel Load
H	L	H	0->1	Shift Right
H	H	L	0->1	Shift Left
H	L	L	X	Hold

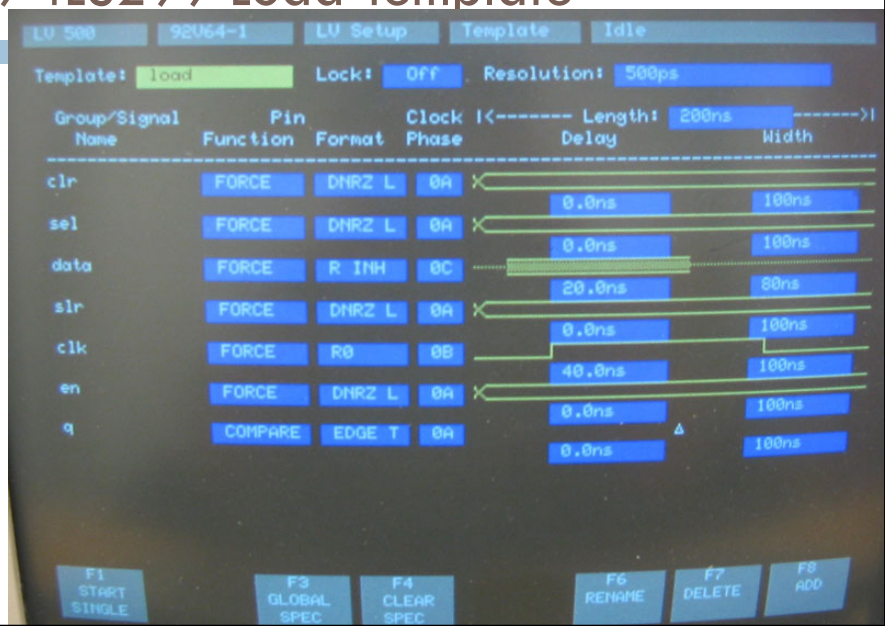
74LS299 Timing



74LS299 Shift/Clear Template



74LS299 Load Template



74LS299 Pattern