Harry D. Bartley

Tektronix, Inc.
P.O. Box 500, M/S 39-645
Beaverton, OR 97077

E-mail: harry.bartley@tek.com

DESIGN REUSE SUBDESIGNS AND MODULES IN A COMPLEX HIERARCHICAL DESIGN ALLEGRO DESIGN ENTRY HDL 15.5

Harry D. Bartley
Tektronix, Inc.
Session Number: 6.9

Introduction

I have been supporting Cadence schematic capture tools for the past 14 years. I have worked in various aspects of CAD/CAE for about 30 years.

The information presented in this paper is based upon current design processes at Tektronix using Cadence tools for schematics and layout. This paper focuses only on the schematic tools.

Cadence has been introducing the Constraint Manager over the past several years. With each release, new features are introduced and old features are changed. The playing field is constantly changing. The design process discussed in this paper was started in release 13 and has been modified for use in releases up to 15.2, and now 15.5.1.

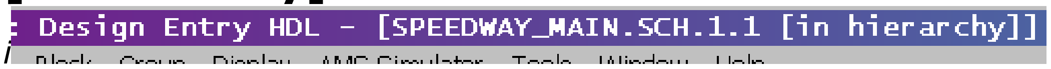
The paper does not focus on the details of the design. The focus is on the steps in the process used to create the design.

To understand the process, you need to understand Schematic States, Schematic Styles, and the interaction of Occurrence Editing and the Constraint Manager.

What are Schematic States?

A schematic may be in one of three different states, or modes, [in hierarchy], [expanded], or [occurrence edit]. The default state of a schematic is [in hierarchy].

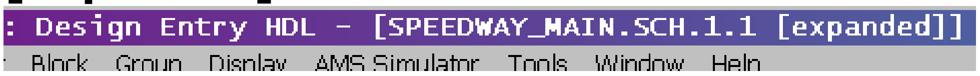
[in hierarchy]



: Design Entry HDL - [SPEEDWAY_MAIN.SCH.1.1 [in hierarchy]]

- Limits searching to the current active display
- You may display and search one page in the design.

[expanded]



: Design Entry HDL - [SPEEDWAY_MAIN.SCH.1.1 [expanded]]

- Compiles information from all sheets in the design

- Temporary files allow searching through all pages of the design.
- Some processes require expansion of the design.
- Constraint Manager and all global search commands require design expansion.

[occurrence edit]

: Design Entry HDL - [SPEEDWAY_MAIN.SCH.1.1 [occurrence edit]]
t Block Group Display AMS Simulator Tools Window Help

- The occurrence state stores information that cannot be stored on the schematic page.

- When a property is attached to a bus, ADDR<15..0>, the property is attached to each member of the bus, and that information is stored in the OPF (Occurrence Property File). ***This has changed in release 15.5!***

- To change the property on one bit of the bus, you may open occurrence editing. The occurrence change will be stored in the OPF. The value in the OPF is the primary value, or winning value if it conflicts with a property value on a schematic page.

- You may view the OPF values with the ATTRIBUTE command when the design is in the expanded mode.

- In a complex hierarchical drawing, when the same block is used multiple times, each occurrence of the drawing will have unique signal names, and unique reference designators.

- In the occurrence edit mode, the unique occurrence values will display when you descend into the schematic pages of a block. You may make additional property modifications in the occurrence edit mode.

What are Schematic Styles?

A schematic may be created using one of three different styles: Flat, Simple Hierarchy, or Complex Hierarchy. The default state of a schematic is Flat. A general understanding of the capabilities and limitations of schematic styles is necessary.

Flat Schematics*, the simplest schematic style***.**

FEATURES OF A FLAT DESIGN

- A flat design uses one name for all of the schematic sheets.
- The only difference in the names of the sheets will be the sheet, or page number.
- All of the page files reside in the same directory.
- Signals are connected by name, no special name syntax or ports are required
- Every symbol instance exists on the schematic pages
- All reference designators and pins may be annotated on the schematic
- Each schematic page is unique.
- Occurrence edit will always affect bus signals.
- If the OPF is deleted, the schematics still contain all of the information required to recreate the information stored in the OPF.
  (If you have not edited properties in the occurrence edit mode.)
- When Constraint Manager is used, it will read constraint properties, of correct syntax, from the schematic.

When constraints are created in the Constraint Manager, the property information is stored in the constraint directory <*drawing*>.dcf. This information is also stored in the OPF. As an OPF property, it may be seen in the ATTRIBUTE form when the schematics are expanded. To annotate the property on the schematic, you must place a placeholder property on the schematic net, and run:

TOOLS > Constraints > Schematic Update

What is hierarchy?

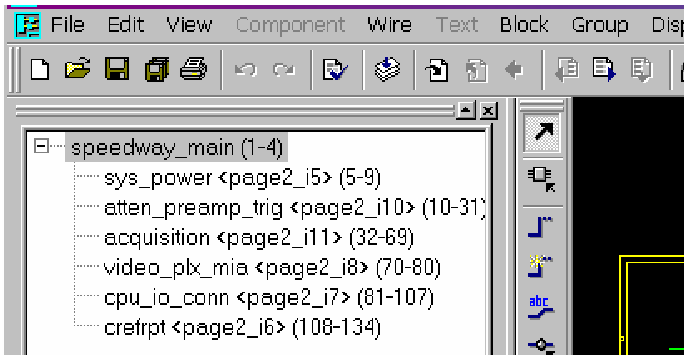
Simple hierarchy and complex hierarchy

FEATURES OF HIERARCHICAL DESIGNS

- A hierarchical design usually has a block diagram at the top level, with one or more schematic sheets under each block.
- A block is similar to a library symbol, but the information related to pin usage is provided by schematics instead of through a chips file. **Block names may not contain spaces**.
- A block and supporting schematic sheet will have the same name, and be located in the same directory.
- Individual blocks will have different names, and the sheets will be saved in separate directories.

Simple Hierarchy

Simple hierarchy is a block symbol and underlying schematics, when the block is used just one time in the design. A simple hierarchical design and a flat schematic design are very similar. Both types of designs have a single instance of each schematic symbol. Backannotation will place reference designators and pin numbers on each symbol.



speedway_main (1-4)
sys_power <page2_i5> (5-9)
atten_preamp_trig <page2_i10> (10-31)
acquisition <page2_i11> (32-69)
video_plx_mia <page2_i8> (70-80)
cpu_io_conn <page2_i7> (81-107)
crefrpt <page2_i6> (108-134)

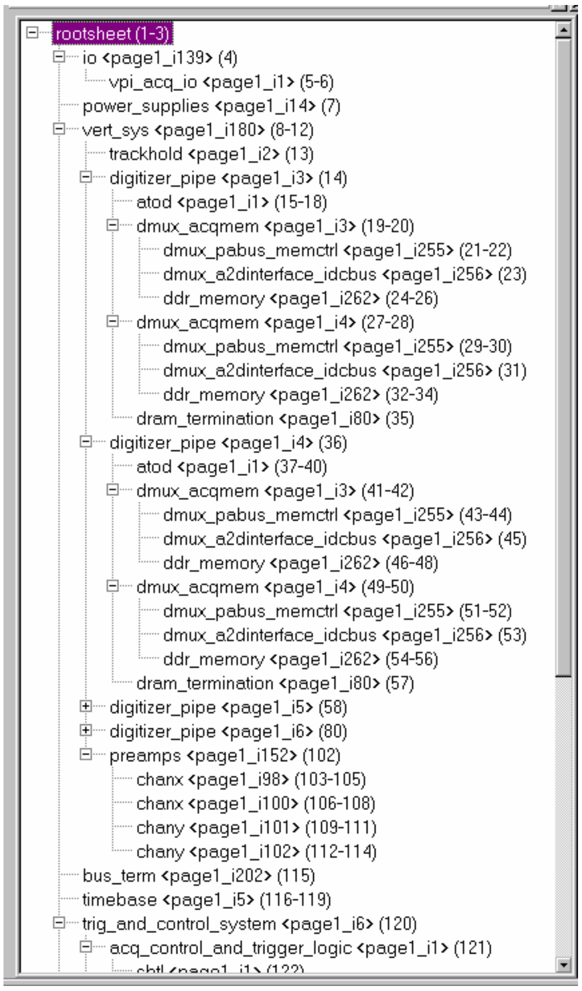
- A simple hierarchy uses each block only once
- Every symbol is represented on the schematics

Complex Hierarchy, *the most difficult style of schematics*

A complex hierarchical design uses one or more blocks more than one time. Whenever the same block is reused within a design, the design is a complex hierarchy. The same set of schematic pages is referenced each time the block is used. When the design is packaged, and the first instance of a block is encountered, the schematics under the block are packaged. Reference designators are assigned in sequence. When the packager comes across another instance of the same block, the schematics are again packaged and reference designators are assigned in sequence. Now, each symbol on the block level schematic has two different reference designators. The reference designators are stored in the Occurrence Property File. One set of reference designators for each block occurrence.

To view or plot each occurrence of the reference designators, the design must be placed in the Occurrence Edit Mode. There will be no similarity of reference designators, to easily identify the same part, used in different instances of the same block.

To maintain some similarity in the reference designator assignments, the block may be packaged as a subdesign. This creates a reference designator "root" that is made unique by appending a suffix.



```
rootsheet (1-3)
  io <page1_i139> (4)
    vpi_acq_io <page1_i1> (5-6)
  power_supplies <page1_i14> (7)
  vert_sys <page1_i180> (8-12)
    trackhold <page1_i2> (13)
    digitizer_pipe <page1_i3> (14)
      atod <page1_i1> (15-18)
      dmux_acqmem <page1_i3> (19-20)
        dmux_pabus_memctrl <page1_i255> (21-22)
        dmux_a2dinterface_idcbus <page1_i256> (23)
        ddr_memory <page1_i262> (24-26)
      dmux_acqmem <page1_i4> (27-28)
        dmux_pabus_memctrl <page1_i255> (29-30)
        dmux_a2dinterface_idcbus <page1_i256> (31)
        ddr_memory <page1_i262> (32-34)
      dram_termination <page1_i80> (35)
    digitizer_pipe <page1_i4> (36)
      atod <page1_i1> (37-40)
      dmux_acqmem <page1_i3> (41-42)
        dmux_pabus_memctrl <page1_i255> (43-44)
        dmux_a2dinterface_idcbus <page1_i256> (45)
        ddr_memory <page1_i262> (46-48)
      dmux_acqmem <page1_i4> (49-50)
        dmux_pabus_memctrl <page1_i255> (51-52)
        dmux_a2dinterface_idcbus <page1_i256> (53)
        ddr_memory <page1_i262> (54-56)
      dram_termination <page1_i80> (57)
    digitizer_pipe <page1_i5> (58)
    digitizer_pipe <page1_i6> (80)
    preamps <page1_i152> (102)
      chanx <page1_i98> (103-105)
      chanx <page1_i100> (106-108)
      chany <page1_i101> (109-111)
      chany <page1_i102> (112-114)
  bus_term <page1_i202> (115)
  timebase <page1_i5> (116-119)
  trig_and_control_system <page1_i6> (120)
    acq_control_and_trigger_logic <page1_i1> (121)
      cbtl <page1_i1> (122)
```

- Any time the same block is used more than one time in a schematic design, the design is a complex hierarchy.
- A block used multiple times always calls the same schematic page or pages.
- Each part in the design is not uniquely represented on the schematics.
- Reference designators unique to each instance of the block are stored in the OPF. The schematic can only store one value for each property.
- With multiple use of the same block, all unique property information for each instance of the block is stored in the OPF. unique information for each page instance is viewable in Occurrence Edit Mode.
- Property passing, macros, may be used in Complex Hierarchical designs, but it is very limited, and totally disabled if you use the Constraint Manager.
- Packaging a complex hierarchy is a multi-step process.

General Information for hierarchical designs

- Local signals are recognized only within individual blocks of the hierarchy.
- Interface signal names must match block pin names
- The PINNAMES command creates a symbol with matching signal names.
- Interface signals must be identified by a port symbol, or a \I suffix on the signal name.
    - INPORT
    - OUTPORT
    - IOPORT
    - or FLAG
- Global signals are identified by the HDL_POWER property attached to the power symbols, or by a \G suffix attached to the signal name.
- Global signals are recognized by name throughout the entire design, no port symbols are required.
- The same signal name may not be used with more than one signal scope.

Occurrence Editing

The occurrence editor stores information that is not stored as part of the schematic page. The value of a property edited in occurrence edit mode is stored in the OPF, not on the schematic page. If you view the schematic page in hierarchy mode, you will see the unedited value of the property. If you view the schematic page in the expanded mode, you will not see any difference in the schematic, but if you use the ATTRIBUTE command, you will see the schematic value of the property and the occurrence value of the property. When you view, or plot, the schematic in the occurrence edit mode, you will see the value of the property as changed in the occurrence edit mode.

Currently, some bus property information, and Constraint Manager information is also stored in the OPF as occurrence values. The OPF is master, so the values stored there are the values used by the packager. This has caused confusion, and will be changed.

- Some occurrence properties are automatically created and stored in the OPF.

- In occurrence edit mode, you may create or modify information stored in the OPF.

- In a flat design, occurrence editing may lead to problems with "phantom" property values. The value stored in the OPF takes precedence over the value stored on the schematic page.

- Occurrence editing may be used with complex hierarchical designs to manually create unique property values for different instances of a block that has been used multiple times.
NOTE: If there are several changes, this can be a very repetitive manual process.

- If you modify property values in occurrence edit mode, deletion of the OPF will cause you to lose all of your modifications.

- File contention is not controlled for occurrence editing.

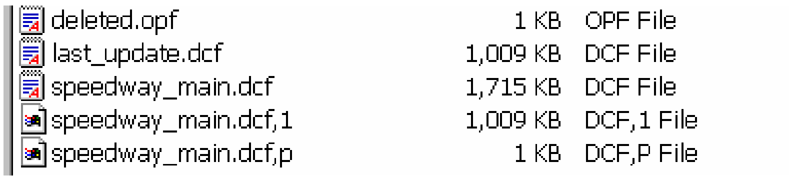
- The OPF is a design file. If a page is copied, the OPF information is not copied to another design.

Constraint Manager

The Constraint Manager is a spreadsheet tool used to create and modify design constraint properties. This replaces the older practice of placing constraint properties directly on wires in the schematic pages. We typically use propagation delay, relative propagation delay, differential pair, net physical type, and net spacing type properties.

The constraint information is stored in a .dcf file located under the constraints directory. The root design contains the master constraints directory, and each subdesign will contain the constraints directory for the subdesign. The Constraint Manager will now read information from lower level constraint directories. When the same subdesign is used multiple times, the same subdesign constraint information is read each time. Suffix numbers are automatically added to keep group names, reference designators used in pin pairs, and differential pair names unique. It is confusing that the suffixes used have no relationship to any suffixes you have assigned.

CONSTRAINTS DIRECTORY



deleted.opf	1 KB	OPF File
last_update.dcf	1,009 KB	DCF File
speedway_main.dcf	1,715 KB	DCF File
speedway_main.dcf,1	1,009 KB	DCF,1 File
speedway_main.dcf,p	1 KB	DCF,P File

- Constraint properties may be created in the Constraint Manager spreadsheet.

- The Constraint Manager will read properties from the schematic pages, if the syntax is correct. The message "failing to push constraints" indicates a problem with the syntax or content of a schematic property. (This is being changed)

- Constraint properties are stored in the constraints directory, project.dcf file, and in the OPF. (This is being changed)

- Constraint properties may be seen in the ATTRIBUTE form when the schematic has been expanded.

- Constraints may be annotated on the schematic.
  A placeholder property must be placed on the schematic net.
  Tools > Constraints > Update Schematic will then annotate the value in the placeholder property.
  Annotation of bus signals is not always possible; there are current limitations to the annotation of the top-level signal name, and to bus bit names. (This is being changed)

- If a net name is changed, the constraints associated with the old net name will be lost.

- Obsolete nets are not automatically flushed you must Audit Obsolete Objects, and identify the obsolete objects to be deleted.

- The schematic Constraint Manager does not recognize power pins identified in the chips.prt file, or the power_group property. Only visible signal pins may be used in pin pairs.

- Packaging changes the logical signal name to a physical signal name. The physical signal names are shorter, and easier to read in the Constraint Manager.

- Deletion of the constraints directory will eliminate all constraints except those that exist as properties on schematic nets.

- File contention is not controlled for the Constraint Manager.

- The constraint information is design information, and not copied with individual pages of the design.

Enough background information. How was this used to create a complex hierarchical design?

The example design for this paper is a four channel design. Each channel is broken into smaller segments. The top level of the design, rootsheet, contains a block diagram. In this block diagram, several blocks are used once each, while the vert_sys block is used four times. Vert_sys contains four digitizer blocks, and the digitizer contains two demux blocks. Each of the reused blocks is packaged as a subdesign. Each subdesign is used to create a layout module.

```
ROOTSHEET                        (LEVEL #4) USED 4 VS
      VERT_SYS                   (LEVEL #3) USED 4 DP
            DIGITIZER_PIPE       (LEVEL #2) USED 2 DA
                  DMUX_ACQMEM(LEVEL #1)
```

After the planning stage, the lowest level blocks were created first. The detail of the design is added from the bottom up to the top. Demux subdesign and demux modules were created. Then the demux subdesign was used in the creation of the digitizer. The demux modules were used in the creation of the digitizer module. The digitizer subdesign was used in the creation of the vert_sys subdesign. The digitizer module was used in the creation of the vert_sys module. The top level of the design used the vert_sys blocks, representing the vert_sys subdesign that contains all of the lower level subdesigns.

Each subdesign was constrained as it was created. The Constraint Manager was used to control trace length, matched traces, and trace width. Some traces were matched in groups, and others were matched in differential pairs. One of the layout problems is locking the modules so they are not individually modified after they have been placed.

Constraints added at the subdesign level are now read by upper levels of the design. This makes it much easier to keep group names, differential pair names, and reference designators used in pin pairs unique when a subdesign is used multiple times. The upper level Constraint Manager adds suffix numbers as the constraint information is reused with another instance of the same block. It is confusing that the suffix numbers used do not match any subdesign_suffix values you have placed in your design.

The assignment of reference designator suffixes in your schematic is controlled by the subdesign_suffix property, and the ability to specify a separator value. We have size limits on reference designators. The Engineer planning this design assigned numbers and letters to different levels of the design so the reference designator indicated where the component exists in the design. Following shows how the design .cpm file was modified to create each subdesign. Saving the modified .cpm files makes it easier to modify and package the design as changes are made.

Building the hierarchy from the bottom level to the top

NOTE: When making changes and packaging you must begin packaging from the lowest level modified.

At the **first level**, build a schematic and block for demux_acqmem.

Setup the .cpm file and the packager options to generate a subdesign.

```
START_GLOBAL
design_name 'dmux_acqmem'
END_GLOBAL

START_PKGRXL
FORCE_SUBDESIGN
GEN_SUBDESIGN 'dmux_acqmem'
END_PKGRXL
```

Save a copy of the modified .cpm as 1_dmux_acqmem.cpm. This is the lowest level of the design. The modified .cpm file may be used to open, edit, and package this level of the design.

Packaging option `GEN_SUBDESIGN 'dmux_acqmem'` creates
`pxl_DMUX_ACQMEM.state file when packaged.`
`(reference designator example C464)`
--
At the **second level**, build a schematic and block for digitizer_pipe. The schematic contains two dmux_acqmem blocks. Setup with an

underscore "_" as separator and specify subdesign_suffix=0 and subdesign_suffix=1 on the digitizer blocks.

```
START_GLOBAL
design_name 'digitizer_pipe'
END_GLOBAL
```

Setup the .cpm file and the packager options to generate a subdesign for digitizer_pipe, and to force the use of the dmux_acqmem subdesign state file. Save the modified .cpm as 2_digitizer_pipe.cpm.

```
START_PKGRXL
FORCE_SUBDESIGN 'dmux_acqmem'
GEN_SUBDESIGN 'digitizer_pipe'
END_PKGRXL
```

This .cpm Creates pxl_DIGITIZER_PIPE.state when packaged.
(reference designator examples C464_0 and C464_1)
--
At the **third level**, build a schematic and block for vert_sys. The schematic contains four digitizer_pipe blocks. Setup with no underscore "_" separator and specify subdesign_suffix=A and subdesign_suffix=B, C, and D on the digitizer blocks.
Setup the .cpm file and the packager options to generate a subdesign for vert_sys, and to force the use of the digitizer_pipe subdesign state file. Save the modified .cpm as 3_vert_sys.cpm.

```
START_GLOBAL
design_name 'vert_sys'
END_GLOBAL

START_PKGRXL
sd_suffix_separator ''
FORCE_SUBDESIGN 'digitizer_pipe'
GEN_SUBDESIGN 'vert_sys'
END_PKGRXL
```

This modified .cpm creates pxl_VERT_SYS.state when packaged.
(reference designator examples
C464_0A, C464_1A, C464_0B, C464_1B, C464_0C, C464_1C, and C464_0D, C464_1D)

--
At the **fourth or top level** in this design, build a schematic rootsheet. The schematic contains four vert_sys blocks. Setup with no underscore "_" separator, and specify subdesign_suffix=1, 2, 3 and 4. Setup the .cpm file and the packager options to force the use of the vert_sys subdesign state file. Save the modified .cpm as 4_rootsheet.cpm

```
START_GLOBAL
design_name 'rootsheet'
END_GLOBAL

START_PKGRXL
sd_suffix_separator ''
FORCE_SUBDESIGN 'vert_sys'
GEN_SUBDESIGN
END_PKGRXL
```

Creates pxl_VERT_SYS.state when packaged
(reference designator examples
C464_0A1, C464_1A1, C464_0B1, C464_1B1, C464_0C1, C464_1C1, and C464_0D1, C464_1D1 (channel 1)
C464_0A2, C464_1A2, C464_0B2, C464_1B2, C464_0C2, C464_1C2, and C464_0D2, C464_1D2 (channel 2)
C464_0A3, C464_1A3, C464_0B3, C464_1B3, C464_0C3, C464_1C3, and C464_0D3, C464_1D3 (channel 3)
C464_0A4, C464_1A4, C464_0B4, C464_1B4, C464_0C4, C464_1C4, and C464_0D4, C464_1D4 (channel 4)

For information on module creation and use, refer to the presentation by George Patrick.

This is not comprehensive, but should provide you with an overview of the process we have used to create Complex hierarchical designs.