EE244: Design Technology for Integrated Circuits and Systems

Outline

Lecture 2.1

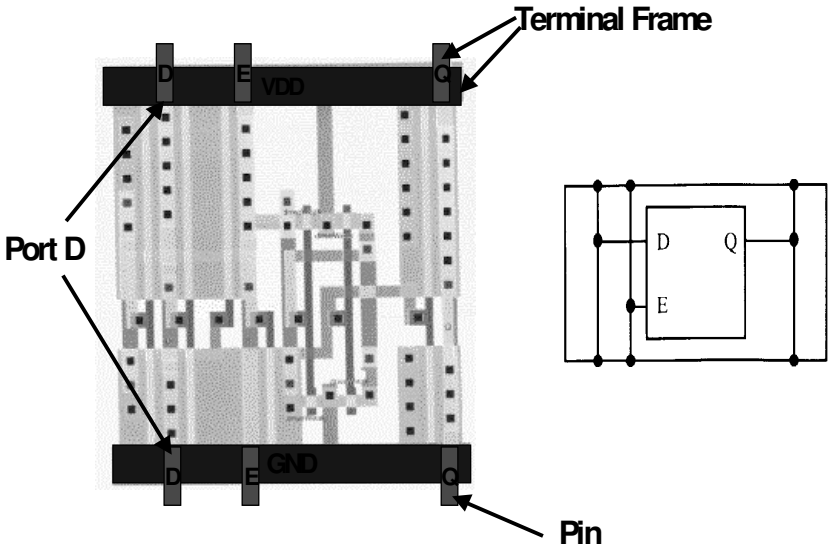
- **Representing the Design: Hierarchy, Abstraction**
- **Netlists**
- **The EDIF Design Interchange format.**

Design Representation

- **Some variation of entity-relationship model most common today.**
- **Data model usually based on existing interchange format or design language (e.g. EDIF, VHDL).**
- **Open Issue: Forced Consistency (conventional database approach) versus Periodic Check (C "Lint" approach).**
- **Integration Environment provides *lingua franca* for design representation. Useful side-effect of standards meetings is a common understanding of terms and data objects.**

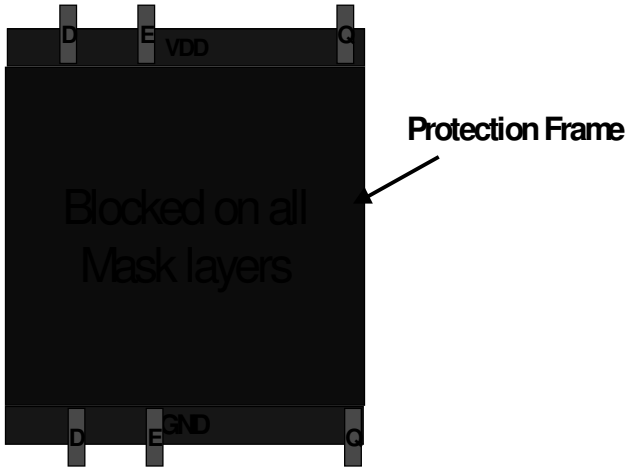
Copyright © 1997, A. Richard Newton

Abstracting Layouts



EE244 Fall 97 1.2.3

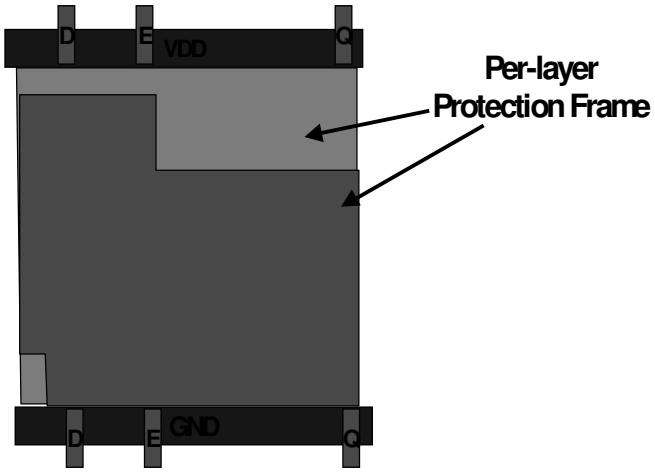
Abstracting Layouts



EE244 Fall 97 1.2.4

Copyright © 1997, A. Richard Newton

Abstracting Layouts



The Netlist View

- List of parts used in a design and their electrical connections:
- net-oriented (e.g. EDIF)
  - partName, partType, portNames (perhaps implicit)
  - netName, ports (or pins) it connects

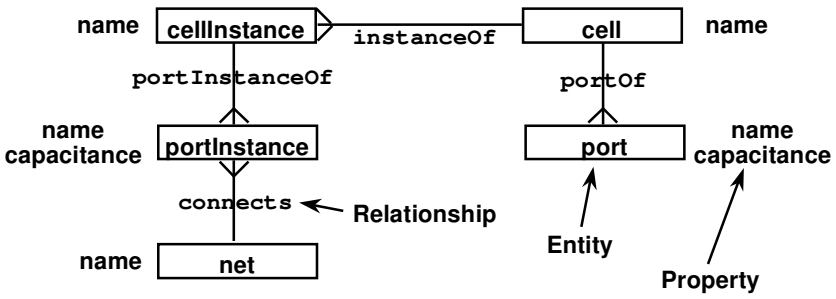
```
(instance Dlatch latch1)
(instance Dlatch latch2)
(net (joined (portReference latch1 D) (portReference latch2 Q)))
```

- or part-oriented (e.g. Spice)
  - partName, nets connected to, partType

```
M1 1 2 3 11 NMOS
M2 4 5 2 11 PMOS
```


Copyright © 1997, A. Richard Newton

Entity-Relationship Approach to Data Representation



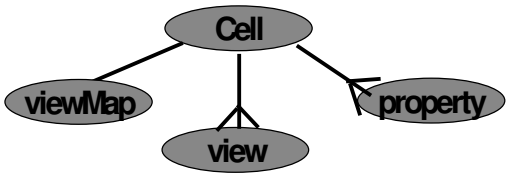
What's in a Name?

- Just about Everything!
- Efficient name resolution - resolving references to design objects - is one of the most important, "undecided" research problems.
- Strongly related to multiprocessor distributed cache consistency problem, distributed file system problem, general distributed data management problem.
- Ultimate issue is efficient pruning of "global search." (replication of read-only data, use of "hints," management of domains and dynamic data migration are all important.)

Copyright © 1997, A. Richard Newton

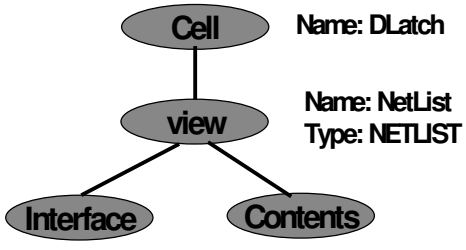
Introduction to EDIF

```
(cell cellNameDef   a named cell definition
     < viewMap > |  relates ports & instances of views
   {  view       |  different views of the cell
     property }     properties of the cell
)
```



Introduction to EDIF: Cells and Views

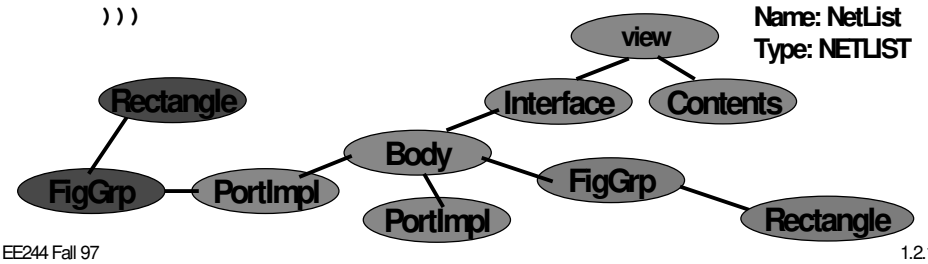
```
(cell DLatch
  (view NETLIST NetList
    (interface
      abstraction of the cell
    )
    (contents
      implementation of the cell
    )))
```



Copyright © 1997, A. Richard Newton

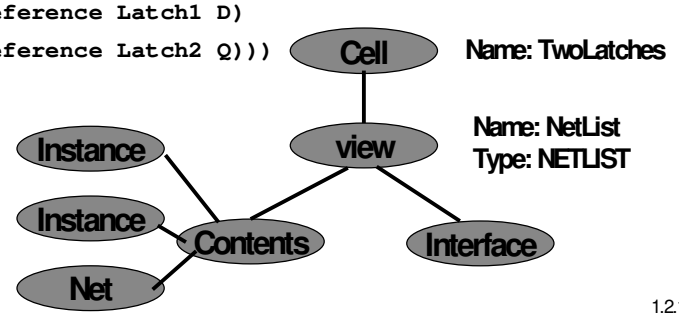
Introduction to EDIF: Abstraction

```
(interface
  (body PROTECTION_FRAME
    (portImplementation D (figureGroup …)
    (portImplementation E (figureGroup …)
    (portImplementation Q (figureGroup …)
    (figureGroup GREEN (rectangle (point x y) ..))
    )
    (contents
      lots of figure groups containing rectangles
    )))
```



Introduction to EDIF: Contents

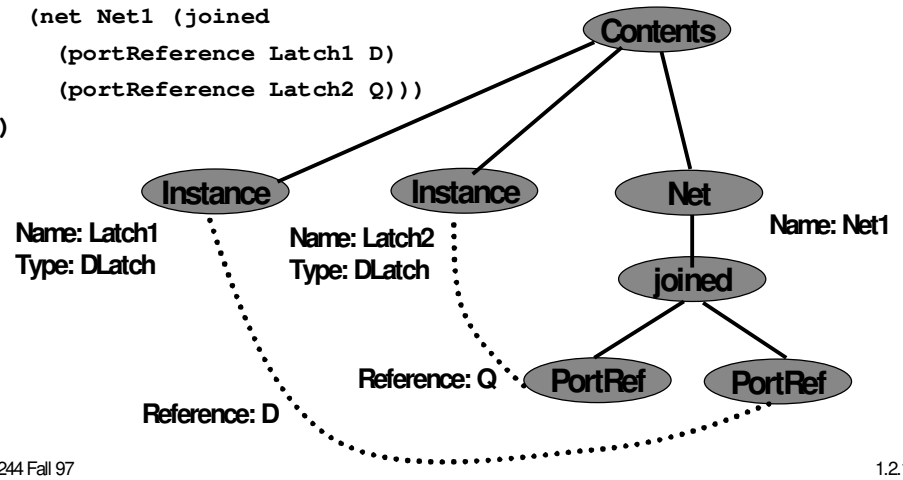
```
(cell TwoLatches
  (view NETLIST NetList
    (interface abstraction of the cell )
    (contents
      (instance DLatch NetList Latch1)
      (instance DLatch NetList Latch2)
       (net Net1 (joined
        (portReference Latch1 D)
        (portReference Latch2 Q)))
))))
```



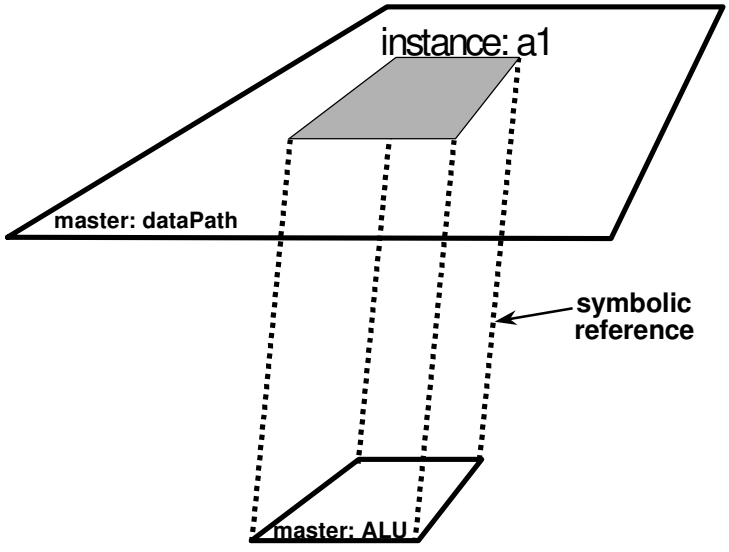
Copyright © 1997, A. Richard Newton

Introduction to EDIF: Netlists

```
(contents
  (instance DLatch NetList Latch1)
  (instance DLatch NetList Latch2)
  (net Net1 (joined
    (portReference Latch1 D)
    (portReference Latch2 Q)))
)
```

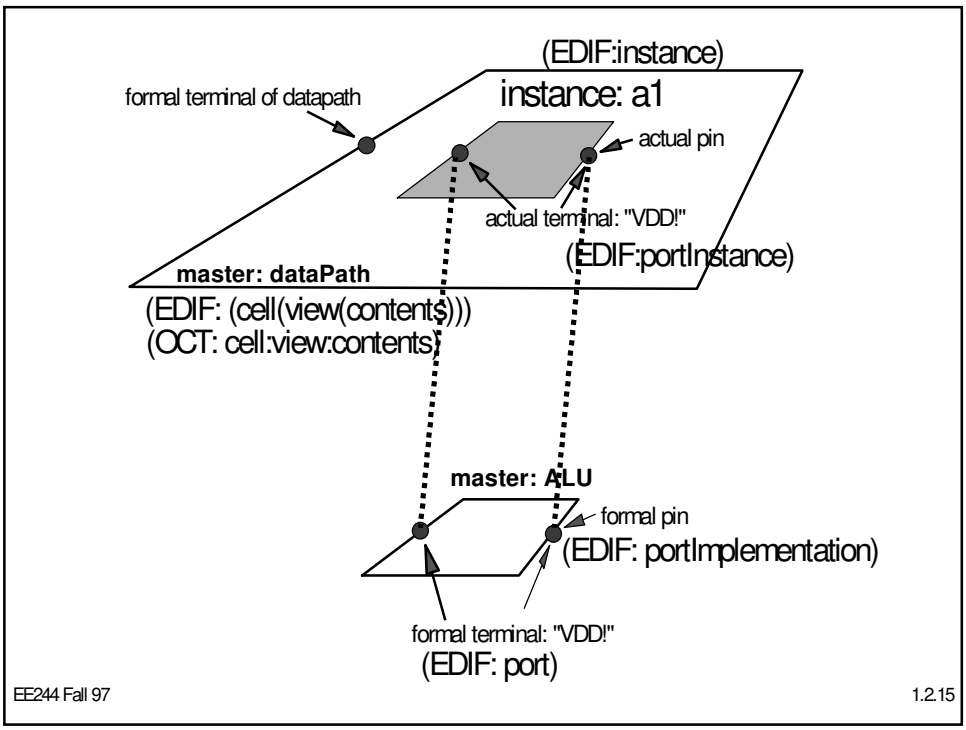


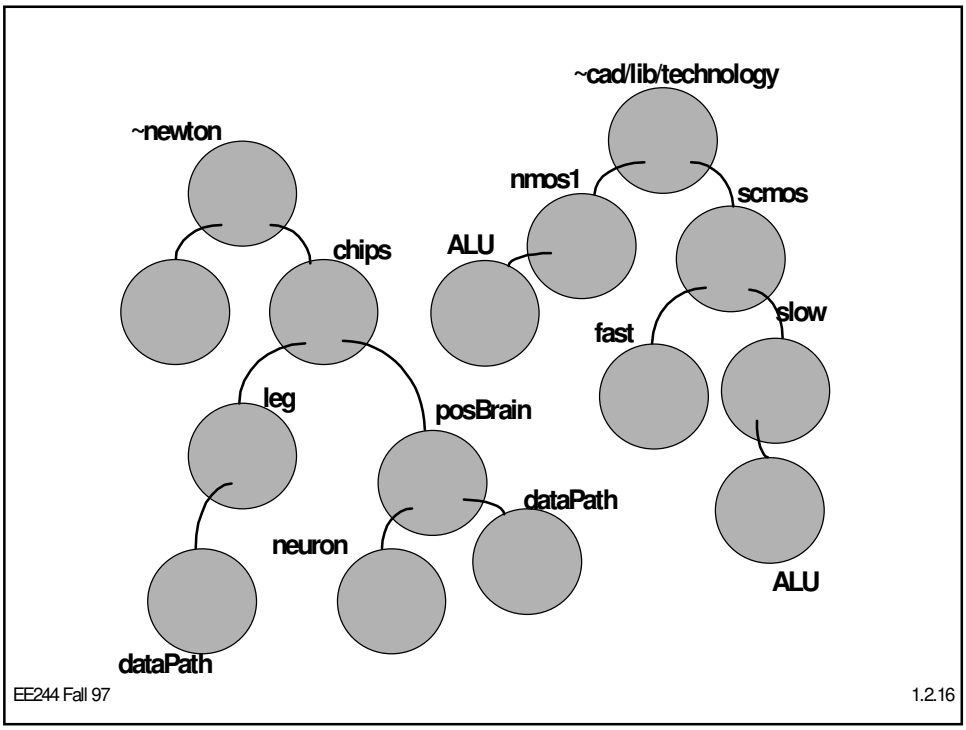
EE244 Fall 97 1.2.13



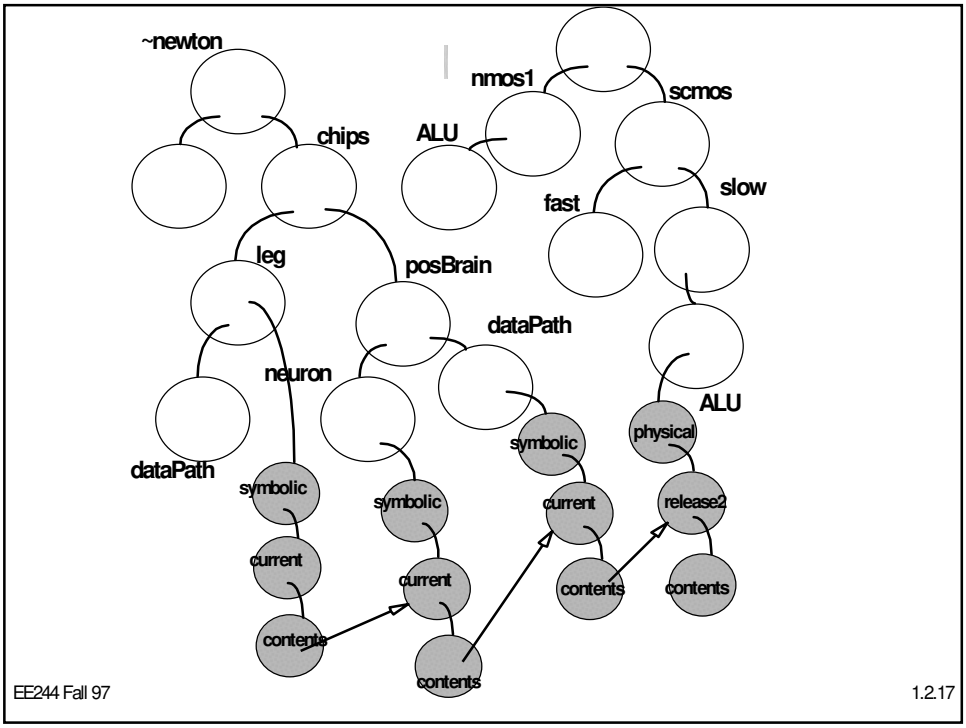
EE244 Fall 97 1.2.14

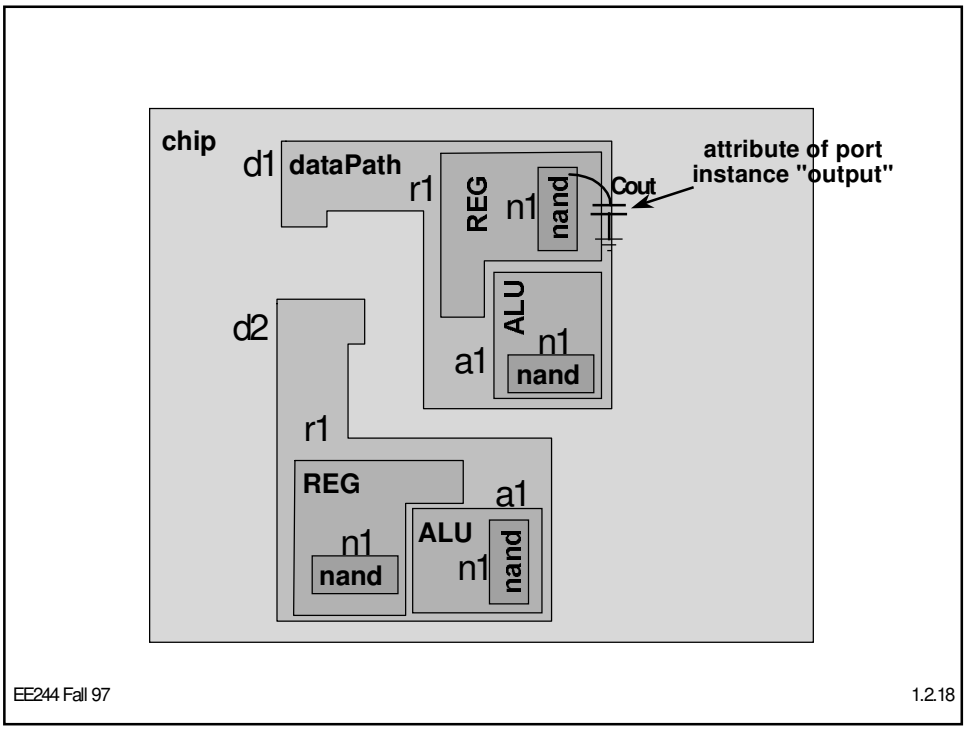
Copyright © 1997, A. Richard Newton



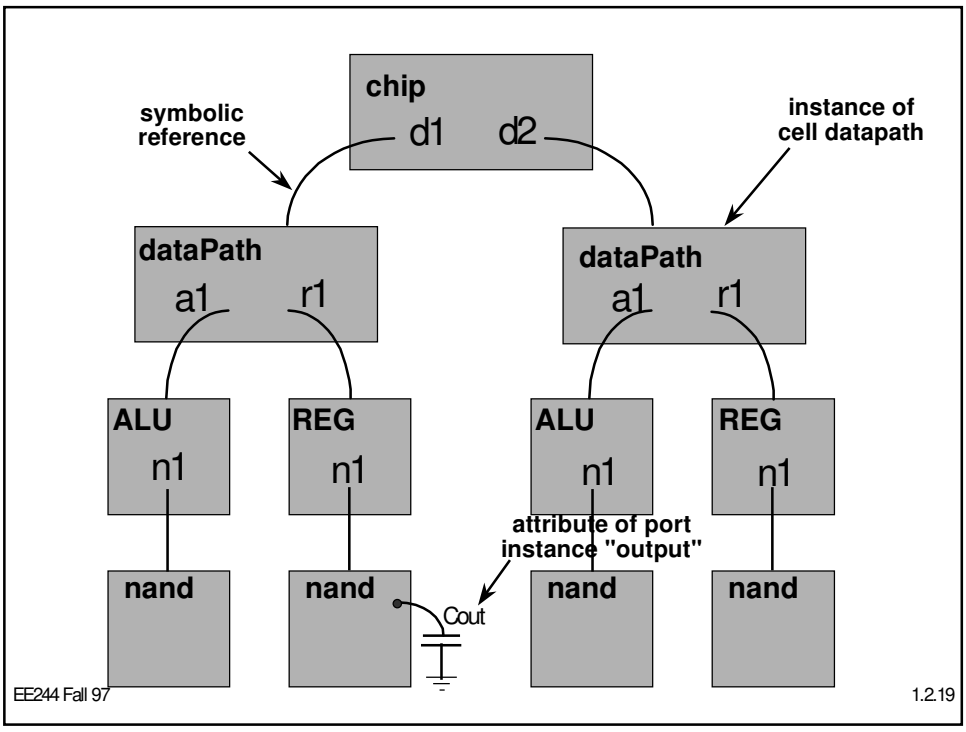


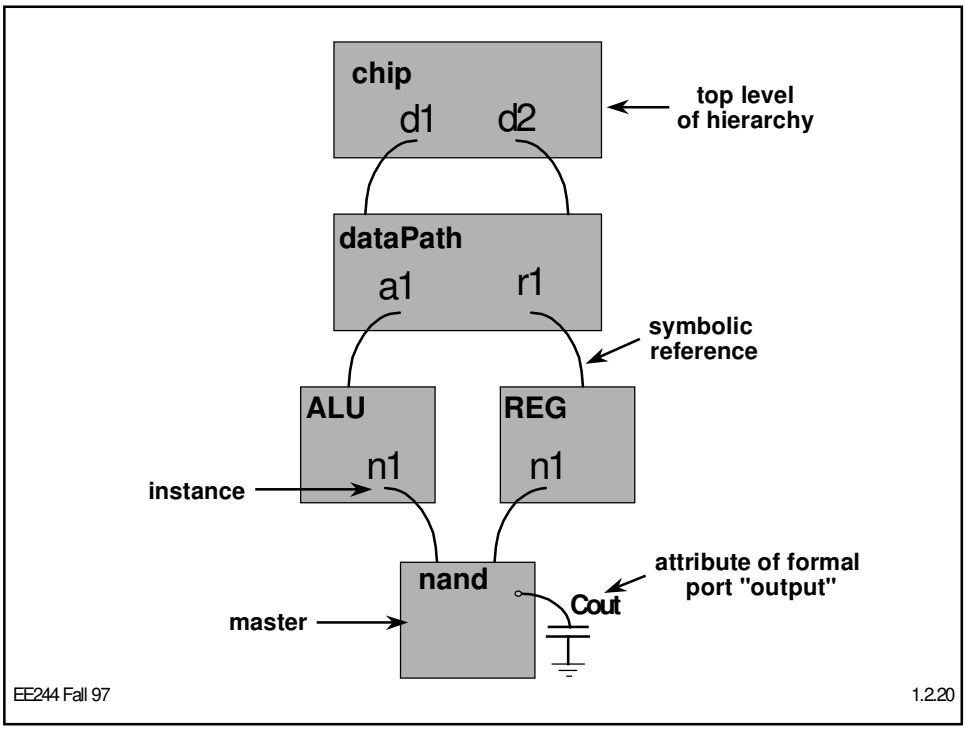
Copyright © 1997, A. Richard Newton





Copyright © 1997, A. Richard Newton

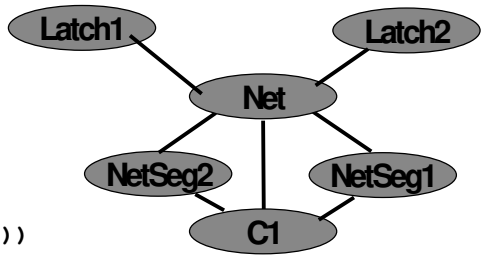




Copyright © 1997, A. Richard Newton

Introduction to EDIF: Nested Nets

```
(instance DLatch Symbolic Latch1)
(instance DLatch Symbolic Latch2)
(net Net1 (joined
   (portReference Latch1 D)
   (portReference Latch2 Q))
   (instance M1M2 Symbolic c1)
   (net netseg1 (joined
      (portReference Latch1 D)
      (portReference C1 viaPort))
   (net netseg1 (joined
      (portReference Latch2 Q)
      (portReference C1 viaPort)))
```



Managing Access to Data

Example: The OCT Approach

Copyright © 1997, A. Richard Newton

OctObject Structure

```
struct octObject {
    octObjectType type;
    octId objectId;
    union {
        struct octFacet facet;
        struct octInstance instance;
        struct octProp prop;
        struct octTerm term;
        struct octNet net;
        struct octBox box;
        struct octPolygon polygon;
        struct octCircle circle;
        struct octPath path;
        struct octLabel label;
        struct octBag bag;
        struct octLayer layer;
        struct octPoint point;
        struct octEdge Edge;
        struct octChangeList changeList;
        struct octChangeRecord changeRecord;
    } contents;
};
```


The OctFacet Object

```
struct octFacet {          /* facet object */
  char *cell;                  /* cellName */
  char *view;                  /* viewName */
  char *facet;          /* "interface" or "contents" */
  char *version;        /* OCT_CURRENT_FACET */
  char *mode;                  /* "r", "w" or "a" */
};
```


Copyright © 1997, A. Richard Newton

The octPoint and octBox Objects

```
struct octPoint {                    /* oct Point */
  octCoord x;                /* x coordinate (32-bit int )*/
  octCoord y;                /* y coordinate (32-bit int) */
};

struct octBox {                              /* oct Box */
  struct octPoint lowerLeft;
  struct octPoint upperRight;
};
```


Operations on Facets

```
void octBegin()
void octEnd()

octStatus octOpenFacet(octObject *facet)
octStatus octCloseFacet(octObject *facet)

octStatus octOpenMaster(octObject *instance, *facet)
octStatus octOpenRelative(octObject *rfacet, *facet, int location)
octStatus octFlushFacet(octObject *facet)
octStatus octWriteFacet(octObject *new, *old)
octStatus octCopyFacet(octObject *new, *old)
octStatus octFreeFacet(octObject *facet)
octStatus octGetFacetInfo(octObject *facet, struct octFacetInfo
  *info)
octFullName(octObject *facet, char **name)
```


Copyright © 1997, A. Richard Newton

Operations on Data Items

octStatus octCreate(octObject *cnt, *obj)
octStatus octDelete(octObject *obj)
octStatus octModify(octObject *obj)

octStatus octAttach(octObject *cnt, *obj)
octStatus octDetach(octObject *cnt, *obj)

octStatus octAttachOnce(octObject *cnt, *obj)
octStatus octIsAttached(octObject *cnt, *obj)

octStatus octPutPoints(octObject *obj, int32 num, octPoint *pnts)
octStatus octGetPoints(octObject *obj, int32 *num, octPoint *pnts)

Retrieving Data Items

octStatus octInitGenContents**(**
octObject *cnt, octObjectMask mask, octGenerator *gen)
octStatus octInitGenContainers**(**
octObject *obj, octObjectMask mask, octGenerator *gen)
octStatus octGenerate**(octGenerator *gen, octObject *obj)**

Values for mask:

OCT_FACET_MASK
OCT_NET_MASK
OCT_PROP_MASK
OCT_POLYGON_MASK
OCT_CIRCLE_MASK
OCT_LABEL_MASK
OCT_POINT_MASK
OCT_FORMAL_MASK
OCT_CHANGE_RECORD_MASK
OCT_TERM_MASK
OCT_INSTANCE_MASK
OCT_BAG_MASK
OCT_BOX_MASK
OCT_PATH_MASK
OCT_LAYER_MASK
OCT_EDGE_MASK
OCT_CHANGE_LIST_MASK

Copyright © 1997, A. Richard Newton

Use of Generators

```
/* proper way to generate */
    while (octGenerate(&gen, &obj) == OCT_OK) {
        /* do something */
    }

/* XXX wrong way to generate */
    while (octGenerate(&gen, &obj) !=
  OCT_GEN_DONE) {
        /* do something */
    }
```

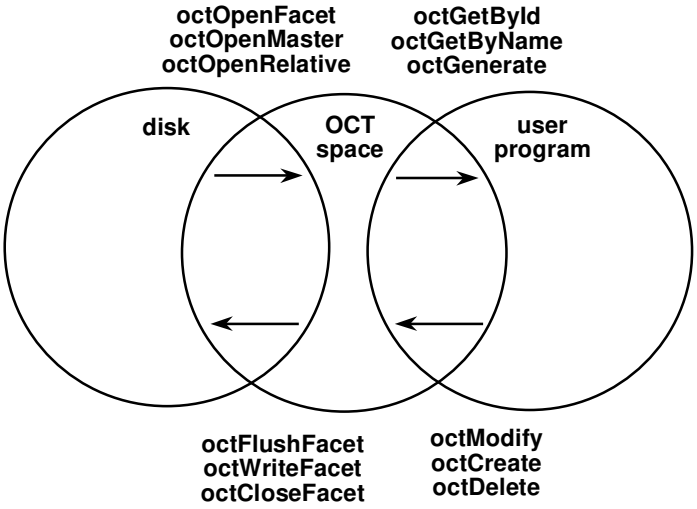

Generator Examples

```
 octInitGenContents(&facet, OCT_NET_MASK, &gen);
        while (octGenerate(&gen, &net) == OCT_OK) {
            /* do something */
        }
octInitGenContents(&facet, OCT_NET_MASK, &gen);
        while (octGenerate(&gen, &net) == OCT_OK) {
            octDelete(&net);
        }

/* XXX will loop infinitely */
newnet.type = OCT_NET;
newnet.contents.net.name = "new net";
octInitGenContents(&facet, OCT_NET_MASK, &gen);
        while (octGenerate(&gen, &net) == OCT_OK) {
            octCreate(&facet, &newnet);
        }
```


Copyright © 1997, A. Richard Newton

OCT Operations and the Environment



EE244 Fall 97

OCT Program Example

```
/*
 * program to generate over all geometries in the facet
 */
#include "copyright.h"
#include "port.h"
#include "oct.h"

main(argc, argv)
int argc;
char **argv;
{
    /* declare the oct objects to be used */
    octObject facet;      /* facet to be opened          */
    octObject layer;      /* layer containing the geometry */
    octObject geo;        /* geometry on the layer       */

    /* declare the oct generators to be used */
    octGenerator lgen;    /* generator for the layers    */
    octGenerator ggen;    /* generator for the geometries */

    /* initialize oct - allocate tables, notify design managers, etc. */
    octBegin();
```

EE244 Fall 97

Copyright © 1997, A. Richard Newton

OCT Program Example

```
/*
 * open the facet
 */
facet.type = OCT_FACET;
facet.contents.facet.cell = argv[1];
facet.contents.facet.view = argv[2];
facet.contents.facet.facet = "contents";
facet.contents.facet.version =
 OCT_CURRENT_VERSION;
facet.contents.facet.mode = "r";

if (octOpenFacet(&facet) < OCT_OK) {
   octError("opening facet to be generated");
   exit(-1);
}
```


OCT Program Example

```
/*
 * generate over all layers
 */
   (void) octInitGenContents(&facet, OCT_LAYER_MASK, &lgen);
   while (octGenerate(&igen, &layer) == OCT_OK) {

      /*
       *  generate over all geometries on the layer
       */
      (void) octInitGenContents(&layer, OCT_GEO_MASK, &ggen);
      while (octGenerate(&ggen, &geo) == OCT_OK) {

         /*
         * process the geometry
         */
      }
   }
 /* close down oct - release memory, notify design managers, etc. */
   octEnd();
   exit(0);
}
```


Copyright © 1997, A. Richard Newton

OCT Example Program

```
/*
 * generate over all layers
 */

(void) octInitGenContents(&facet, OCT_LAYER_MASK, &lgen);
while (octGenerate(&igen, &layer) == OCT_OK) {

   /*
    *  generate over all geometries on the layer
    */

   (void) octInitGenContents(&layer, OCT_GEO_MASK, &ggen);
   while (octGenerate(&ggen, &geo) == OCT_OK) {

      /*
       * process the geometry
       */
   }
}
```

EE244 Fall 97

Copyright © 1997, A. Richard Newton