

A horizontal band with a background of colorful, abstract, scribbled lines in shades of red, yellow, green, and blue, overlaid on a grey, textured background.

# **Exporting RedHawk ECO to Milkyway Database**

**Steven Chen**

# Contents

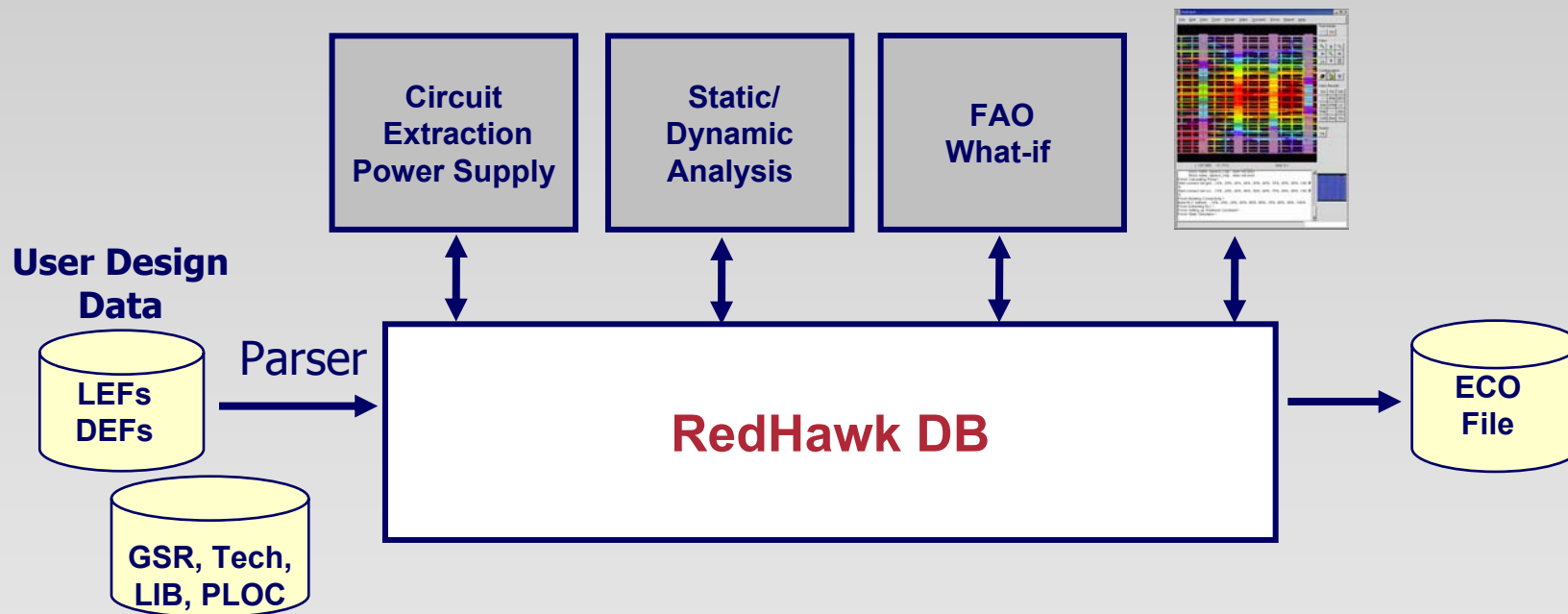
- **RedHawk Environment**
- **RedHawk using Milkyway C-API**
  - **MW2RH**
    - Importing a Milkyway design to RedHawk DB
  - **RH2MW**
    - RedHawk performs Fix and Optimization (FAO)/what-if analysis
    - Exporting RedHawk design changes (ECO) back to Milkyway
- **RH2MW Implementation**
- **Future Work**
- **Discussion**

The title 'RedHawk Environment' is centered in a bold, red, sans-serif font. The background of the slide is a light gray with a complex, abstract pattern of overlapping lines and shapes in various colors, including yellow, green, and red, creating a sense of depth and movement.

# **RedHawk Environment**

# RedHawk Power Integrity/Signal Integrity (PI/SI) Environment

Silicon Integrity



# RedHawk Data Objects


- **User design data**
  - Logic objects: std cells, macro cells, instances, ports, nets...
  - Physical objects: wires, vias, via arrays, pin geometry
  - Hierarchical v.s. Flat design
- **Circuit for simulation and power analysis**
  - R, L, C branches
  - Circuit nodes
  - Current, voltage, sinks etc.

A horizontal band with a background of colorful, abstract, scribbled lines in shades of red, orange, yellow, and green, overlaid on a grey, textured background.

# **RedHawk using Milkyway C-API**

# Milkyway C-API Advantages

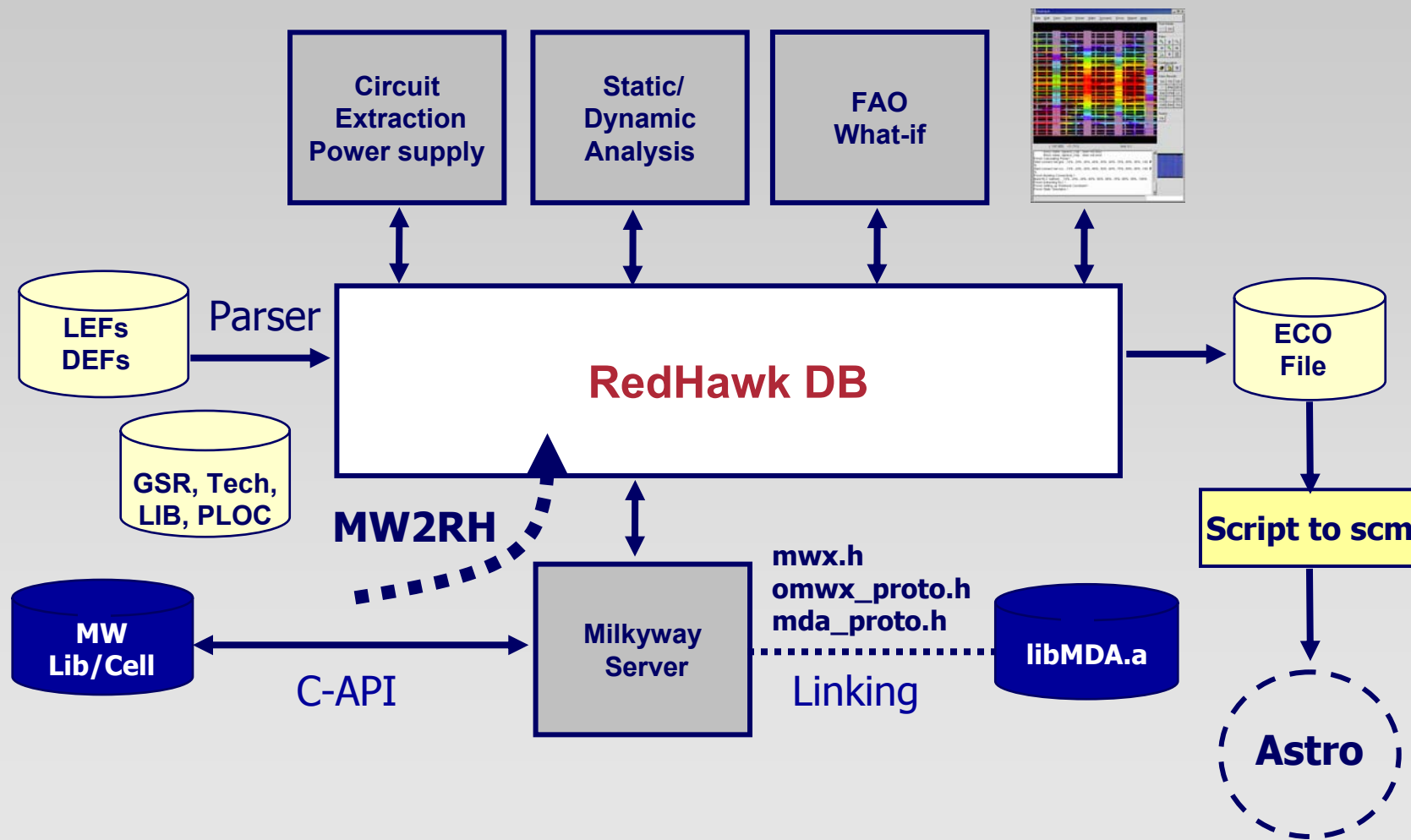
- No parser version problem
- No parser run time space overhead
- No ambiguous syntax v.s. Connectivity problem, e.g., ( \* out ) in 2 nets
- Limited number of necessary input lib directories (MW DB has embedded hash )
- Direct file access on design objects
- Extraction of exact connectivity
- Processed polygon pin geometry
- Exact via array configurations

A horizontal band across the middle of the slide contains a background graphic of overlapping, colorful lines in shades of red, orange, yellow, and green, resembling a signal or data visualization. The text 'MW2RH' is overlaid on this graphic.

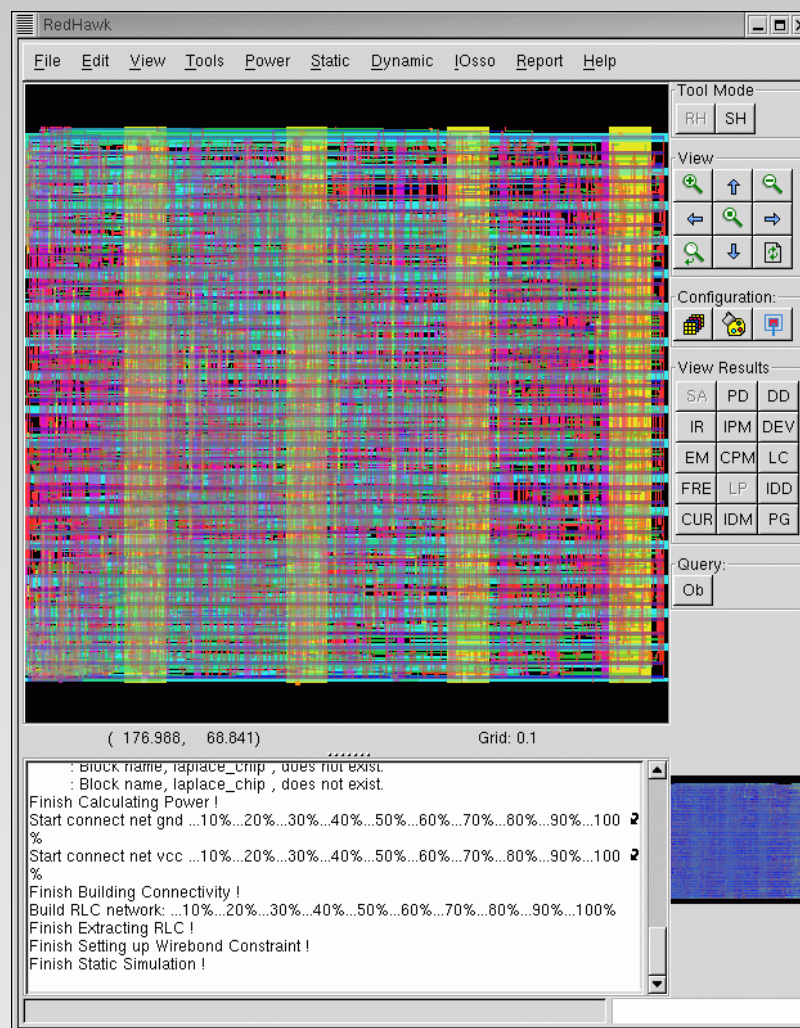
# **MW2RH**

**Milkyway DB to RedHawk DB Translation**

# RedHawk using Milkyway C-API MW2RH



# Importing MW Library and Cell



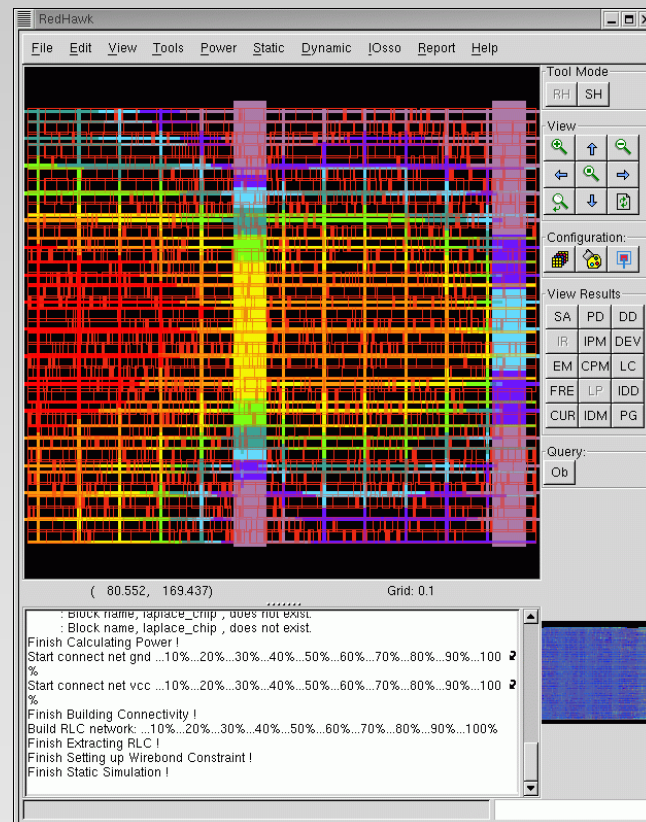
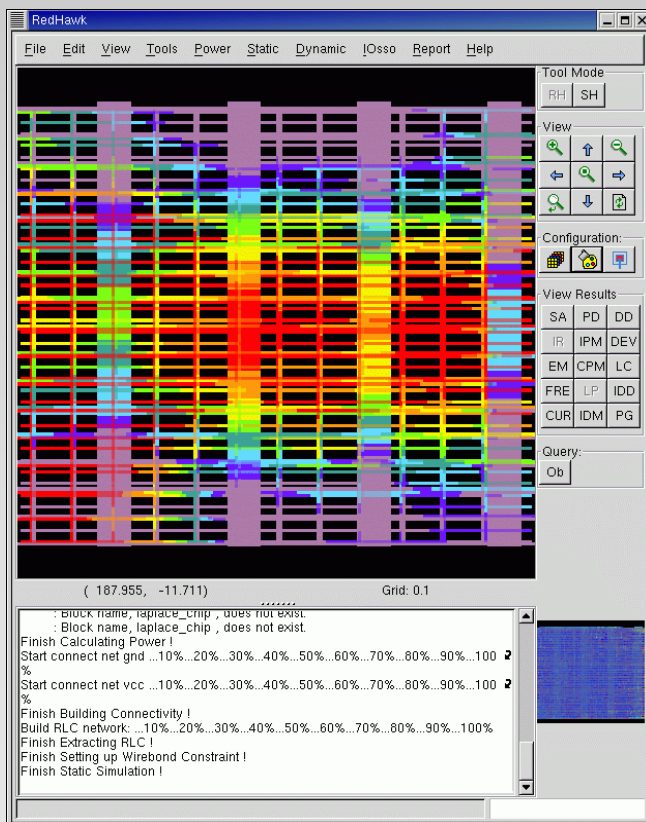
# Importing Another MW Top Design


Silicon Integrity



# RedHawk Analysis Flow

## IR-Drop / Dynamic Voltage Drop Display

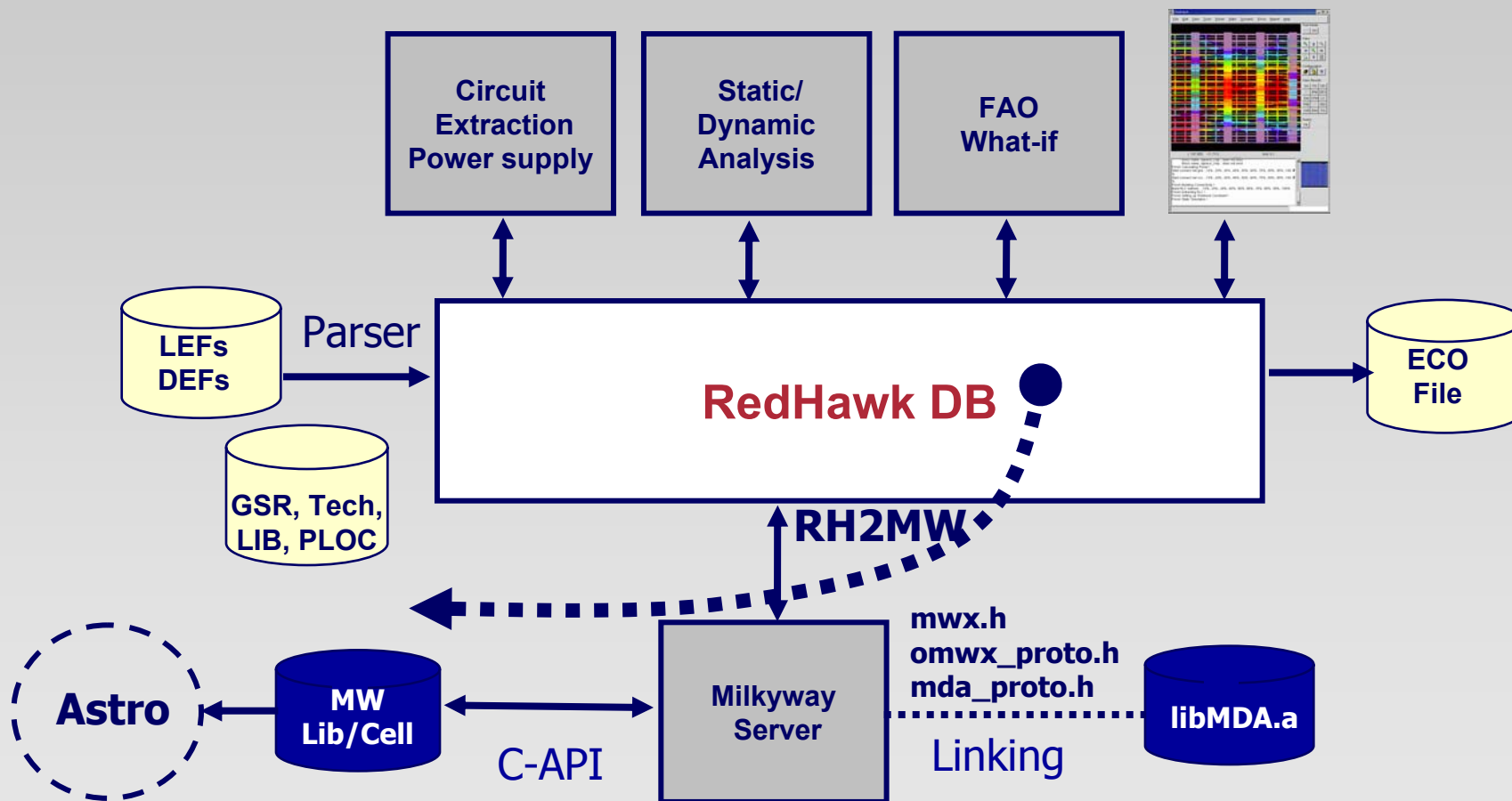


A horizontal band with a grey background and a pattern of diagonal lines. Overlaid on this are several colorful, jagged lines in red, orange, yellow, and green, resembling a signal waveform or data plot.

# **RH2MW**

**RedHawk ECO Changes to Milkyway DB Translation**

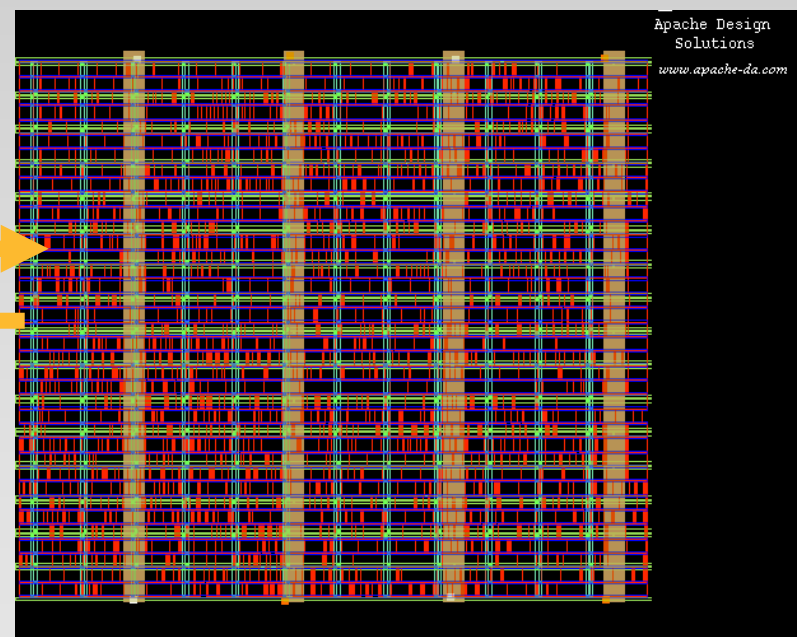
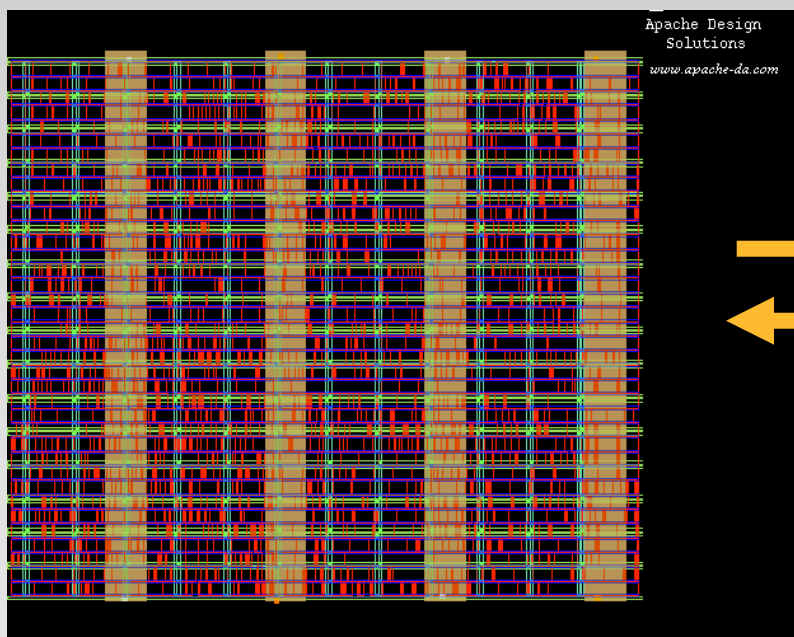
# RedHawk using Milkyway C-API RH2MW



# Power Grid Sizing What-If Analysis

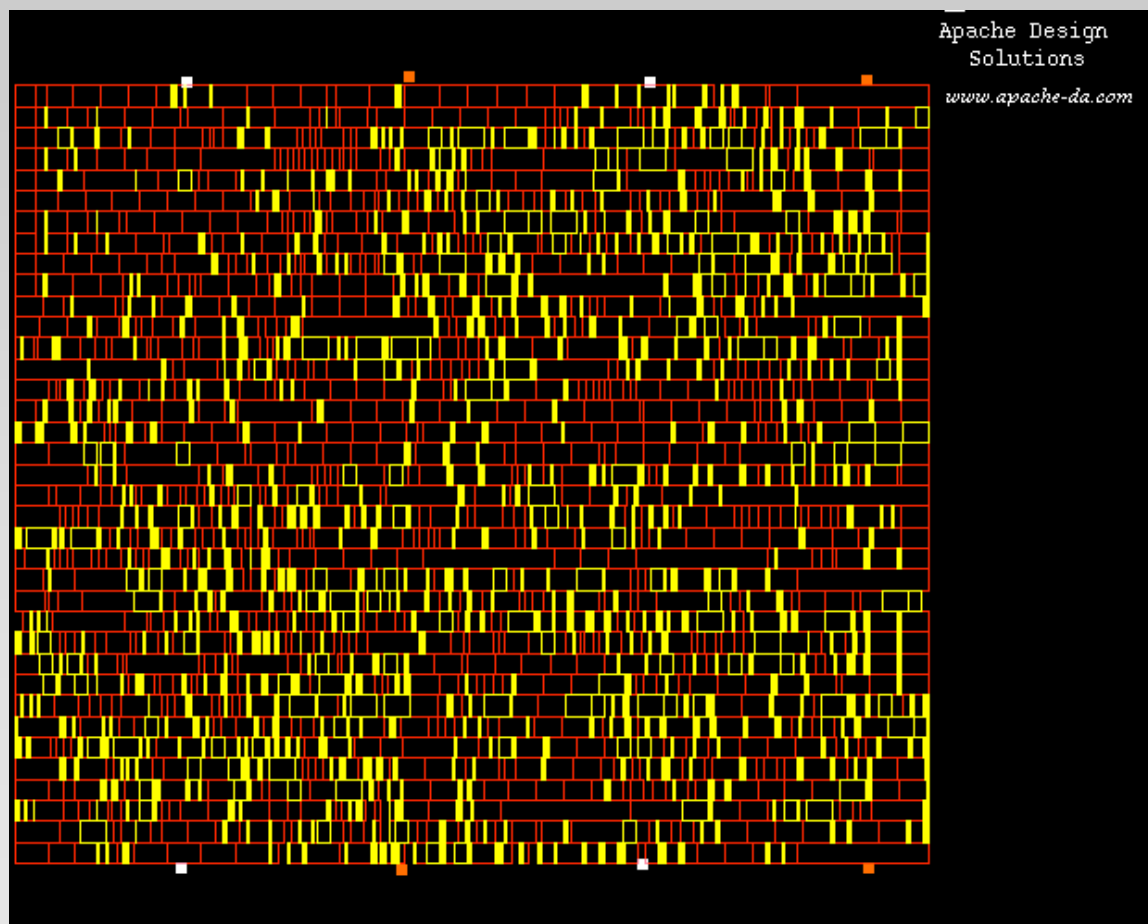
- Sizing Up

- Size Down



# Add Decoupling Capacitors Power Gating Cells

Silicon Integrity



# RedHawk ECO output

```

#Redhawk_Eco_20
DESIGN gpio
UNIT 2000
DELETE wire eco_wire1 gnd MET6 58620 0 82620 326040
DELETE wire eco_wire2 vcc MET6 248700 0 272700 326040
ADD wire eco_wire3 gnd MET6 254700 0 266700 326040
ADD wire eco_wire4 vcc MET6 64620 0 76620 326040
DELETE via eco_via1 gnd Via_10_MET5_VIA5_MET6_26_2_460_460 70620
1320
DELETE via eco_via2 vcc Via_10_MET5_VIA5_MET6_26_2_460_460 70620
21780
ADD via eco_via35 gnd MWVia_5_MET5_VIA5_MET6 -1 260700 1320
ADD via eco_via36 vcc MWVia_5_MET5_VIA5_MET6 -1 260700
161700
DELETE inst xofiller!filler_!filler_32x!1
DELETE inst xofiller!filler_!filler_32x!2
ADD decap filler_1x_SH0 filler_1x N 35640 2640 660 8580
ADD decap filler_1x_SH1 filler_1x S 36300 2640 660 8580
ADD decap filler_1x_SH2 filler_8x N 39600 2640 660 8580
ADD decap filler_1x_SH3 filler_16x S 46860 2640 660 8580

```

...



**RH2MW  
Implementation**

# RedHawk ECO Objects and Actions

- **RH Objects to be changed**
  - **Wires**
    - Vertical/horizontal wires
  - **Vias**
    - Single contact
    - Contact Arrays
  - **Cell Instances**
    - Decoupling cap Cells
    - Power Gating Cells
- **Change actions**
  - **Addition**
  - **Deletion**

# Flows

- **Global**
  - MWXLibId\_t libId ;
  - MWXCellId\_t topCellId ;
  
- **Open write session**
  - MWXDb\_Open\_Lib (libName, &libId)
  - MWXDb\_Open\_Cell (topCellName, 'w' , &topCellId)
  
- **Action session**
  - Add/Delete\_Wires/Vias/CellInstances (objId)
  
- **Close and save change session**
  - MWXDb\_Close\_Cell\_And\_Save (topCellId)
  - MWXDb\_Close\_lib ()

# DELETE Wire

- Id the wire to be deleted in MW

```
int n = 0 ;
```

```
MWXObjId_t *objIds =NULL ;
```

```
MWXDb_Get_Objects_ByBBox ( topCellId,  
MWX_VERTICAL(HORIZONTAL)WIRE,  
x1, y1, x2, y2, &n, &objIds )
```

- wireObjId > wireMasterIndex > wireMasterObjId > width/layer

For each objId in objIds[n] do the following

```
MWXDb_Get_Vertical(Horizontal)Wire_wireMasterIdx (  
topCellId, ... , &Index)
```

```
MWXDb_GetWireMasterId ( topCellId, layer, index, &wireMasterId)
```

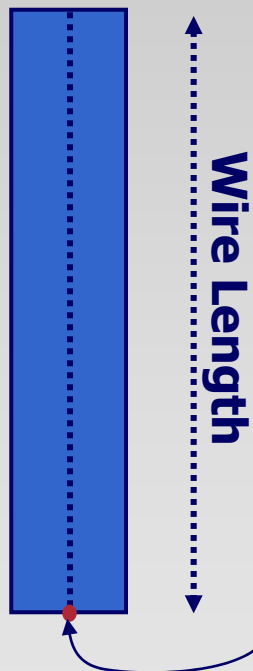
```
MWDb_Get_WireMaster_width(topCellId, wireMasterId, &width)
```

# DELETE Wire

- **Disconnect from net**
  - **MWXDb\_Disconnect ( topCellId, net, objId)**
  
- **Delete the object**
  - **MWXDb\_Delete\_Object ( topCellId, objId )**

# MW Wire

- Vertical Wire



- Horizontal Wire



Center Lower Left starting point

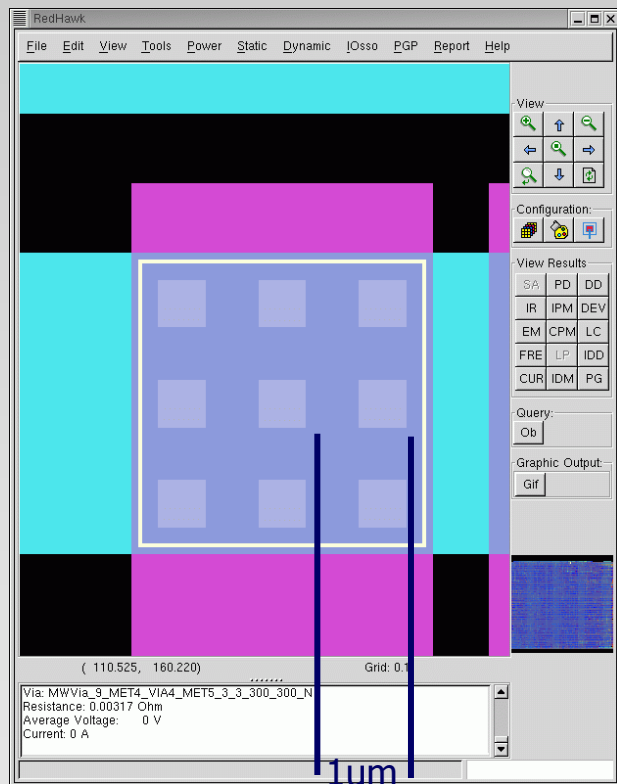
## ADD Wire

- Find existing wire masters of layer/width
  - Preprocess or build beforehand (during MW2RH)
  
- For new layer/width
  - `uchar index = 0 ;`
  - `MWXObjId_t masterId = 0 ;`
  - `MWXDb_Create_Wire_Master ( topCell, layer, width ... &index)`
  - `MWXDb_Get_WireMasterId ( topCell, ... , index, &masterId)`

# ADD Wire

- **Create the wire**
  - **int32 wireLength = ... ;**
  - **MWXPoint\_t centerLowerLeftPt = ... ;**
  - **MWXDb\_Create\_Vertical(Horizontal)\_Wire ( topCell, masterId, wireLength, centerLowerLeftPt, &wireId)**
  
- **Connect to net**
  - **MWXDb\_Connect ( topCellId, net, wireId)**

# Contact Code in MW



- Instance-based
- no via model as in LEF
- xPitch = yPitch = 1 um
- xDuplicate = yDuplicate = 3

Via Model Name:

MWVia\_9\_MET4\_VIA4\_MET5\_3\_3\_1000\_1000\_N

MWXDb\_Get\_ContactArray\_CodeIdx()

# Via Array C-APIs

## MW2RH

```
MWXDb_Get_ContactArray_xDuplicate()
MWXDb_Get_ContactArray_yDuplicate()
MWXDb_Get_ContactArray_xPitch()
MWXDb_Get_ContactArray_yPitch()
```

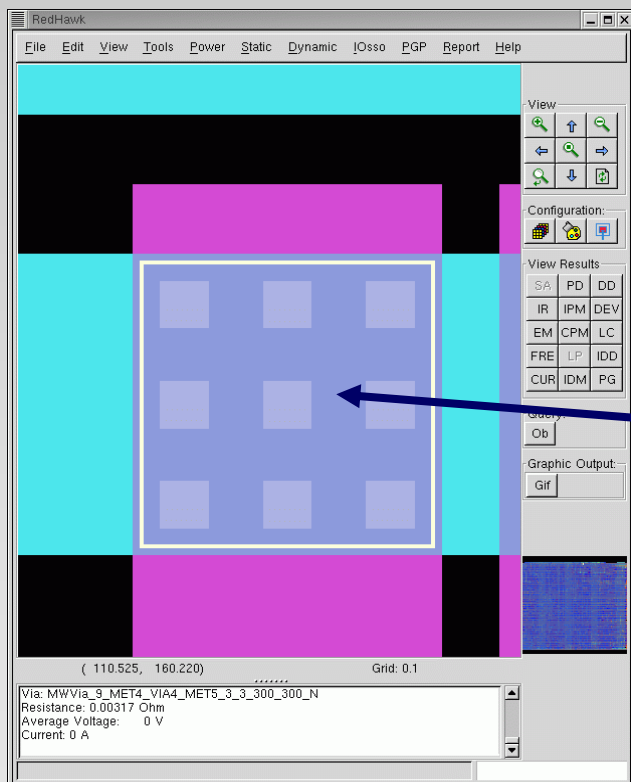
### Get the center location

```
MWXDb_Get_ContactArray_position()
```

```
MWXDb_Get_ContactArray_transformation()
MWXDb_Get_ContactArray_bbox()
```

## RH2MW

```
myGetCutConfig ( defViaModel* vp,
                 int32 &xPitch, int32 &yPitch,
                 uint16 &xDup, uint16 &yDup )
```



# DELETE via, via array

- Find associated **contact code index** for top/cut/bot layers
  - Preprocess or build beforehand (during MW2RH)
  
- Id the **contact(Array)** to be deleted in MW w/ **via(array)** bounding box
  - **int n = 0 ;**
  - **MWXObjId\_t \*objIds =NULL ;**
  - **MWXDb\_Get\_Objects\_ByBBox ( topCellId,**
  - **MWX\_CONTACT(ARRAY) , x1, y1, x2, y2, &n, &objIds )**

# DELETE via, via array

- **viaObjId > contactCodeIndex > contactObjId > cut/top/bottom layers**
  - For each objId in objIds[n]
  - int32 code = 0 ;
  - MWXDb\_Get\_Contact\_contactCodeIdx ( topCell, objId, &code)
  - MWXDb\_Get\_TechContactLayer ( code, &cutLayer, &botLayer, &topLayer )

filter out by center position, cut layer (from contact code)
  
- **Disconnect from net**
  - MWXDb\_Disconnect ( topCellId, net, objId ) ;
  
- **Delete the object**
  - MWXDb\_Delete ( topCellId, objId ) ;

# ADD via, via array

- Find associated contact code index for top/cut/bot layers
  
- Create via(array)
  - `MWXPoint_t ctrPt = ... ;`
  - `MWXObjId_t viaObjId = 0 ;`
  - `MWXDb_Create_Contact(Array)_WithRoutType ( topCellId, &ctrPt, contactCodeIndex, ... , &viaObjId ) ;`
  
- Connect via to net
  - `MWXDb_Connect ( topCellId, net, viaObjId ) ;`

# DELETE inst

- **Get the inst**
  - **MWXObjId\_t instIdMW ;**  
**MWXDb\_Get\_CellInstance\_ByName (topCellId, instName, &instIdMW)**
  
- **if exist**
  - **Disconnect net from pin instances for each pin instances in MW**
  - **MWXObjId\_t netIdMW ;**  
**MWXDb\_Get\_Net\_ByName (topCellId, netName, &netIdMW)**  
**MWXDb\_Disconnect\_PortInstance\_ByName ( topCellId, netIdMW, instName, pinInstName )**
  - **Delete the MW object**  
**MWXDb\_Delete\_Object ( topCellId, instIdMW )**

# ADD inst

- Using

## MWXDb\_Create\_CellInst (

topCellId (current opened MW top cell MWXCellId\_t)

libName (current opened MW lib name char\* )

cellName (cell master for the new inst char\* )

rotation ( "0" , "90" , "180" , "270" )

mirror ( "no" , "X" , "Y" )

origin (MW origin MWXPoint\_t\* )

newObjId ( returned new objId MWXObjId\_t\* )

)

# ADD inst

- **Orientation**

- **LEF/DEF**

- N, FN, S, FS, E, FE, W, FW

- **MW rotation/mirror**

- R90, R180, R270, mirror-X, and mirror-Y

- **Origin**

- **LEF/DEF**

- Always lower left corner of cell boundary

- **MW**

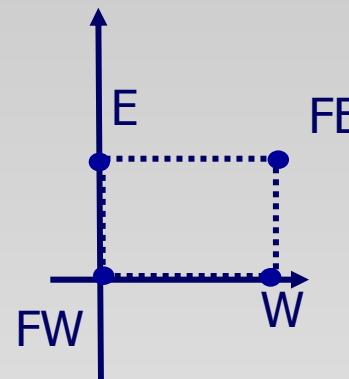
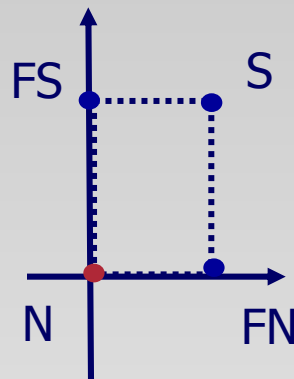
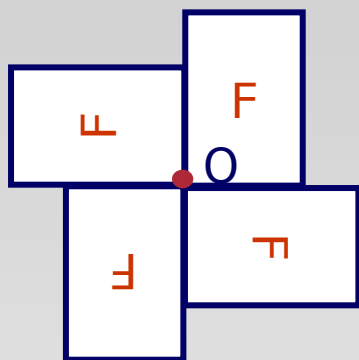
- The corner BEFORE R90, R180, R270, mirror-X and mirror-Y translation

# ADD inst

- **Orientation string translation** *.pdf pp. 3-101*
  - **N : “0” “no” MWXcRotate0**
  - **W : “90” “no” MWXcRotate90**
  - **S : “180” “no” MWXcRotate180**
  - **E : “270” “no” MWXcRotate270**
  - **FN : “0” “Y” MWXcRotate0MirrorY**
  - **FW: “90” “X” MWXcRotate90MirrorX**
  - **FS : “0” “X” MWXcRotate0MirrorX**
  - **FE : “90” “Y” MWXcRotate90MirrorY**

# ADD inst

## ■ Origin Translation to MW



$O_w (x + \text{height} , y)$   
 $O_s (x + \text{width} , y + \text{height} )$   
 $O_E (x , y + \text{width})$

$O_{FN} (x + \text{width} , y)$   
 $O_{FS} (x , y + \text{height} )$   
 $O_{FE} (x + \text{width}, y + \text{height} )$   
 $O_{FW} (x , y)$

## ADD inst

- **Connectivity in MW**
  - Hook up net for each pin instance in MW
  - **MWXDb\_Connect\_PortInstance\_ByName ( topCellId, netIdMW, instName, pinInstName)**
  
- **Skip connection to power/ground nets**
  - Wildcard ( \* **VDD** ) in DEF
  - High fan-out nets not recorded in MW
  - Return **MWXFail** if not

# RH2MW Flow Optimization

- **MW Objects needed for create new objects**
  - **Wire masters**
    - Of different directions, widths, layers
  - **Via code indices**
    - Of different via/top/bottom layer combination
- **Setup cache beforehand to memorize**
  - MW2RH to store in cache
  - RH2MW to reference without search MW again

# Implementation Details

- **Complexity**
  - **MW2RH**
    - 3700+ lines C++ code
  - **RH2MW**
    - 1100+ lines C++ code
  
- **Current status on RH2MW**
  - QA/Release test
  
- **Formal release**
  - RedHawk release 6.2

# Major Milestones


- **MW2RH**
  - **7/25/2003** Started to implement
  - **9/10/2003** Finished prototyping
  - **1/2004** Formal released in RH 4.1
  - SNPS provided more platform supports
  - **10/2006** Most current Astro users adopted the flow (TI, Marvell, Samsung etc.)
  
- **RH2MW**
  - **9/10/2006** Started to implement
  - **10/10/2006** Finishing coding and testing

## Future Work

- **\*.so file dynamic linking to cut down binary size of libMDA.a comparing to RH**
- **Access GDSII data from Milkyway DB to get accurate IP block info**
- **Attach circuit objects into Milkyway DB to save RLC extraction time**

# Suggestions

- Open **MWXDb\_Get\_CellRow\_bbox()**
  - Needed for placement applications
  
- ECO marking handling functions
  - Debugging eco changes on Astro side
  
- Provide a “**MWXDb\_omwxEnd()**” to release Milkyway license w/o occupying in application after **MWXDb\_omwxInit()**
  
- Figures in Pdf documentaions focusing on Creation of the API flow
  - Cell masters
  - Instances
  - Via arrays

The background of the slide is a light gray with a subtle, abstract pattern of diagonal lines. Overlaid on the left side is a series of colorful, wavy lines in shades of red, orange, yellow, and green, resembling a signal or data visualization.

# **Discussion**

## **Q & A**

# Milkyway C-API (for dummies)

- **Study MDA\_Example 1**
  - Follow example1 to write your first application
  - Dump object either to your GUI or MW AMonitor to understand
  
- **Components to know**
  - FlexIm: license checker
  - Header files: mwx.h, omws\_proto.h, mda\_proto.h
  - Library : libMDA.a (platform-based)
  - Makefile from MDA\_Example1
  - PDF file for documentation search
  
- **Other templates and application examples**
  - **mw2oa utility from Si2**