

# High Level Synthesis Design Space Exploration

Benjamin Carrion Schafer<sup>1</sup>, Kazutoshi Wakabayashi

DAC 2010 User Track

 **U can change.**

NEC Corporation  
1753, Shimonumabe, Nakahara-ku  
Kawasaki, Japan  
[schaferb@bq.jp.nec.com](mailto:schaferb@bq.jp.nec.com)<sup>1</sup>

# Outline

- CyberWorkBench (ESL design Platform)
- Design Space Exploration: Goals and Tradeoffs
- Exploration Options
  - Local Attributes
  - Global Options
  - Functional Unit Count
- Image Processing IP example
- Results and Discussion
- Conclusions

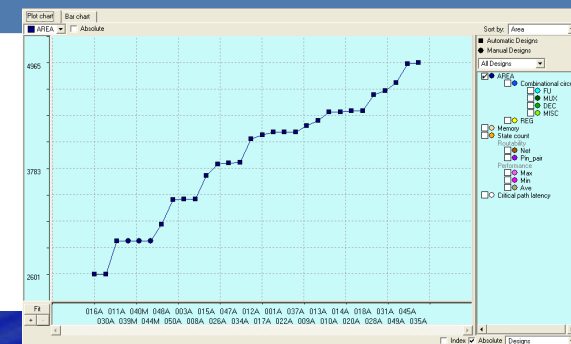
## Motivation

# Initial Design (C, C++, Matlab)

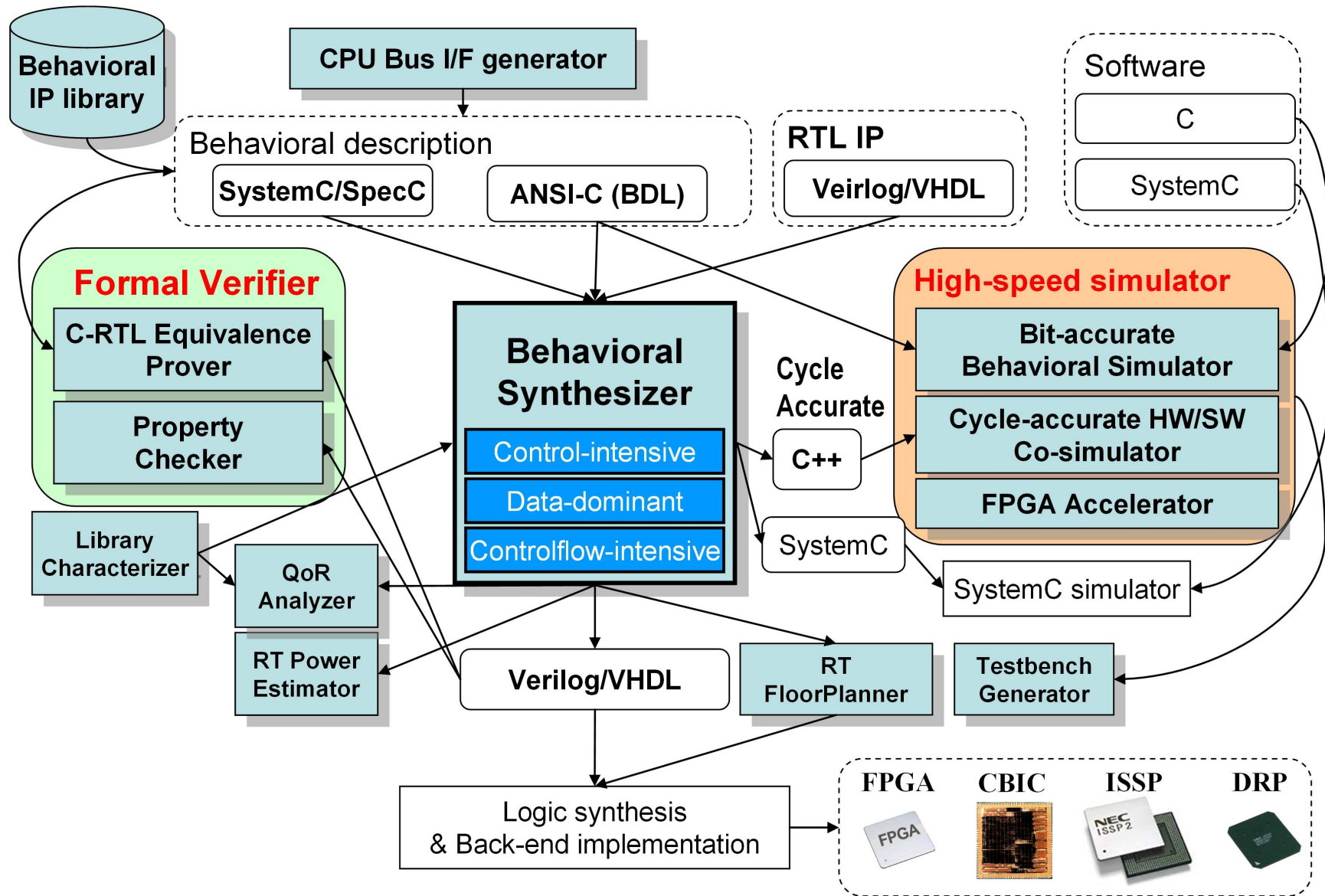


- # C for Hardware (SystemC, BDL)
- Fixed point data
  - Loop unrolling
  - Function inlining, goto
  - Arrays (logic, memory, regs)

- # C for Hardware (SystemC, BDL)
- Fixed point data
  - Loop unrolling
  - Function inlining, goto
  - Arrays (logic, memory, regs)



# ESL Synthesis Tool Overview (CyberWorkBench )



# Motivational Example

- Design computes the average of 8 numbers
- Depending on how the array, loops and function are synthesized different area vs. latency designs are created

```
process main(){  
  in ter(0:8) in0 ;  
  out ter(0:8) out0 ;  
  var(0:8) sum;  
  var (0:8) fifo[8] /* Cyber array=reg */{ 0, 0, 0, 0, 0, 0, 0, 0 } ;  
  var(0:4) i ;
```

Explore 1

```
/* Cyber unroll_times=0 */  
for( i = 7 ; i > 0 ; i-- )  
  fifo[i] = fifo[i-1] ;
```

Explore 2

```
  fifo[0] = in0;
```

```
/* Cyber unroll_times=0 */  
for(i=1;i<8;i++){  
  /* Cyber func=inline */  
  sum = addition(sum, fifo[i]);  
}
```

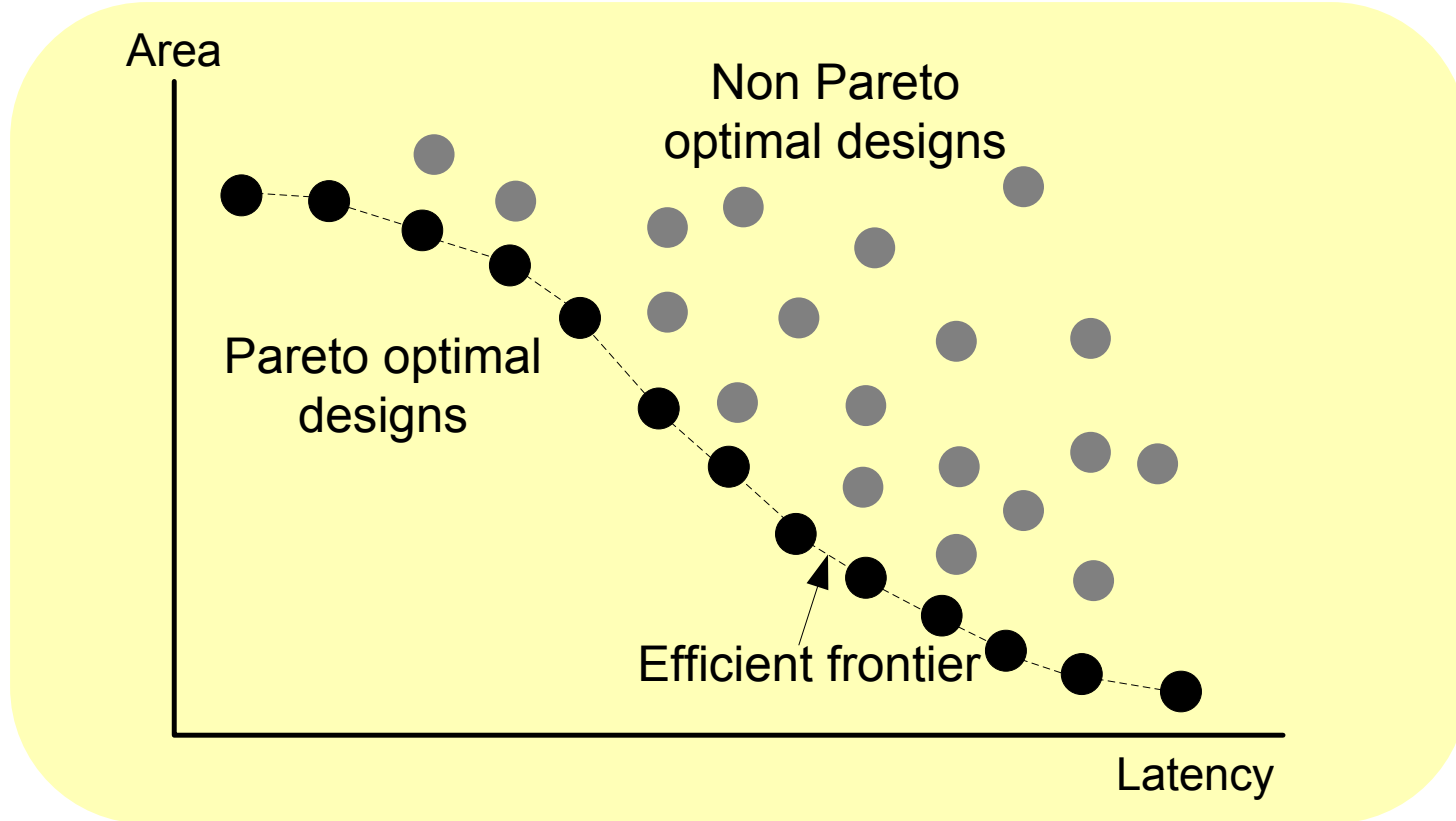
```
  out0 = sum / 8;  
}
```

Explore 3

array	loop1	loop2	function	Area	Latency
ram	0	0	goto	1362	24
ram	all	all	inline	2684	19
reg	0	all	inline	2915	6
reg	0	0	inline	3544	5
reg	all	all	inline	4352	1

# Pareto Optimal Designs (Efficient Frontier)

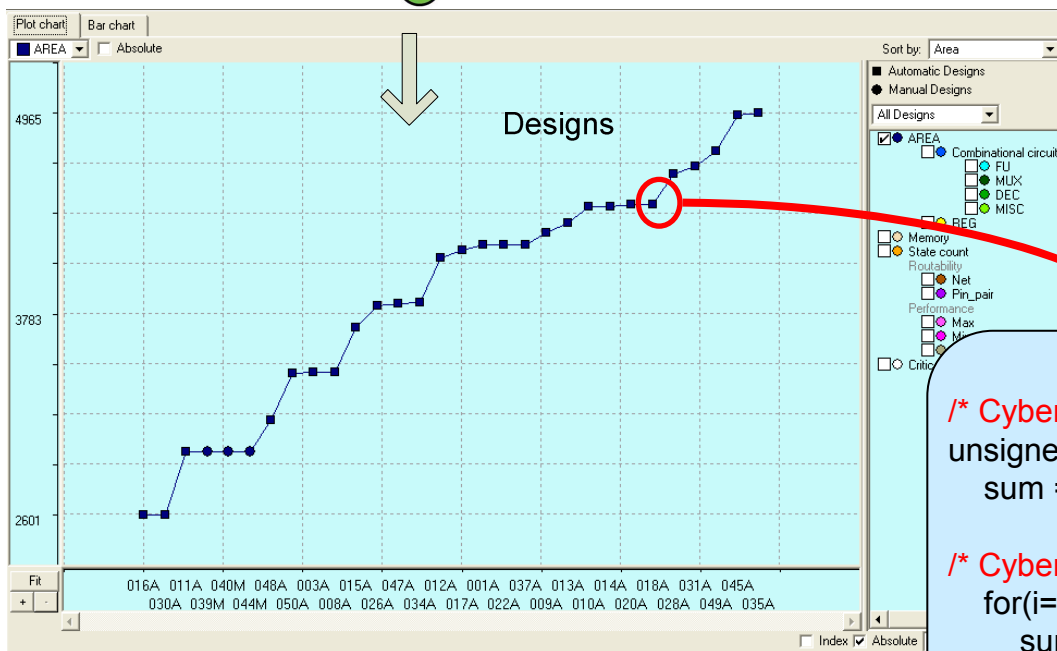
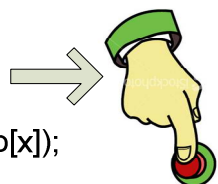
- CyberWorkBench has 300+ optimization options and applies different synthesis heuristics based on the C code and synthesis options
- Ideal exploration generates only Pareto optimal designs ( most efficient designs)



# Exploration Overview

## C code:

```
unsigned var(63..0) add{  
    sum = fifo[0];  
  
    for(i=1;i<8;i++){  
        sum = gen_sum(l,fifo[x]);  
    }  
}
```



1

Synthesis Options

2

Attributes

3

Functional  
Resource constraint

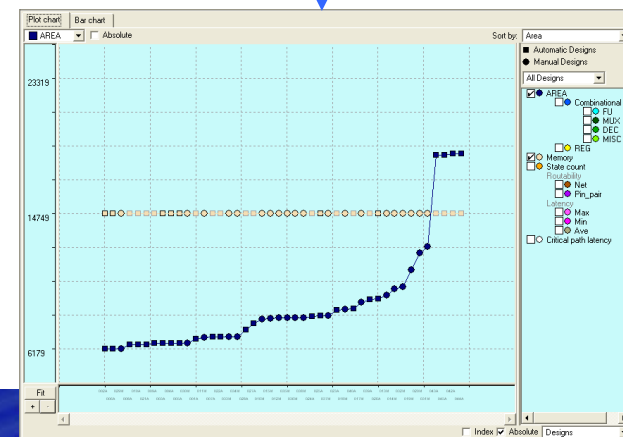
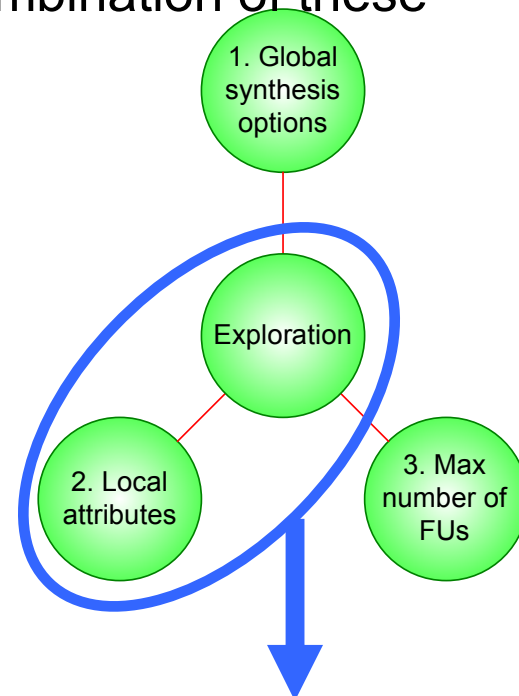
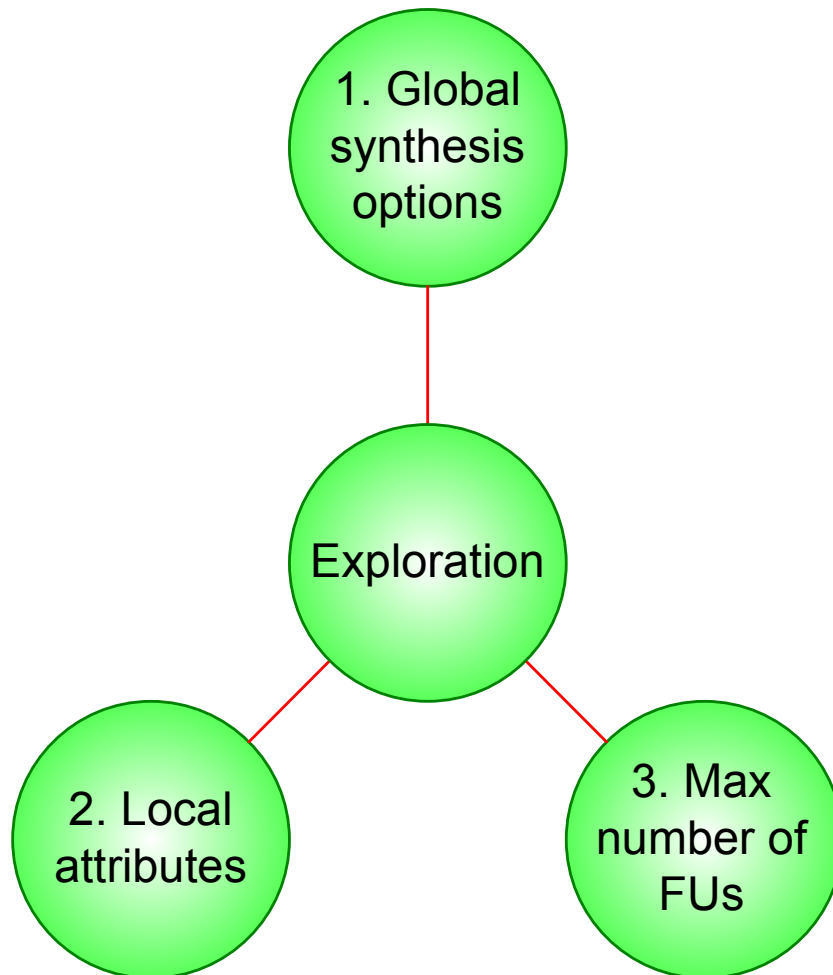
```
/* Cyber function = inline */  
unsigned var(63..0) add{  
    sum = fifo[0];  
  
/* Cyber unroll_times = all */  
    for(i=1 ; i<8 ; i++){  
        sum = gen_sum(l, fifo[x]);  
    }  
}
```

➔ Completely Automatic “push button exploration”

➔ Automatic RTL generation for each design on the curve

# Exploration Options

- Explorer can search the global synthesis options, local attributes, number of Functional units (FUs) or any combination of these



# Local Attributes

- Specified directly at the source code using pragmas e.g. :

```
/* Cyber unroll = all *  
for(x=0; x<10; x++){
```

```
/* Cyber func=inline*/  
Int mult_func(int a, int b){
```

- Explorable attributes
  - Loops
  - Functions
  - arrays

	Attribute
Loop	Unroll=0
	Unroll=x
	Unroll=all
Functions	Folding=x
	Func=inline
	Func=goto
	Func=operator
Arrays	Array=logic
	Array=expand
	Array=reg
	Array=memory

# Global Synthesis Options

- CWB Synthesis has over 100 synthesis options
- Each option affects all operators at the same time

-fi: All functions  
inlined

-am: map all  
arrays to memory

- e.g.

- `%bdltran -fi -SS -am foo.IFF -lfc foo.FCNT...`

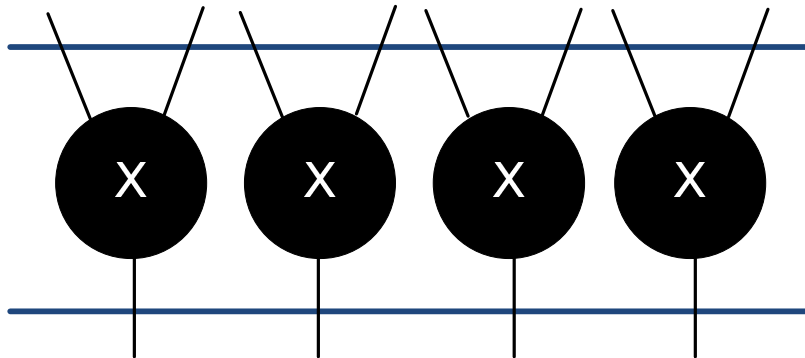
-SS: branch  
parallelization

➔ Global synthesis options apply to the entire design

# Functional Units Exploration

- Functional Constraint File (FCNT) file contains the max number of FUs allowed

```
/* Cyber unroll=all */  
for( i = 7 ; i > 0 ; i-- )  
    fifo[i] = fifo[i-1]*a;
```



foo.FCNT

```
@FCNT{  
    NAME    mul4 _08u  
    LIMIT   4  
}
```

→ The maximum number of FUs allowed will drastically impact the synthesis results

# Design Space Exploration Flow

1 Call cwbexplorer with following inputs:

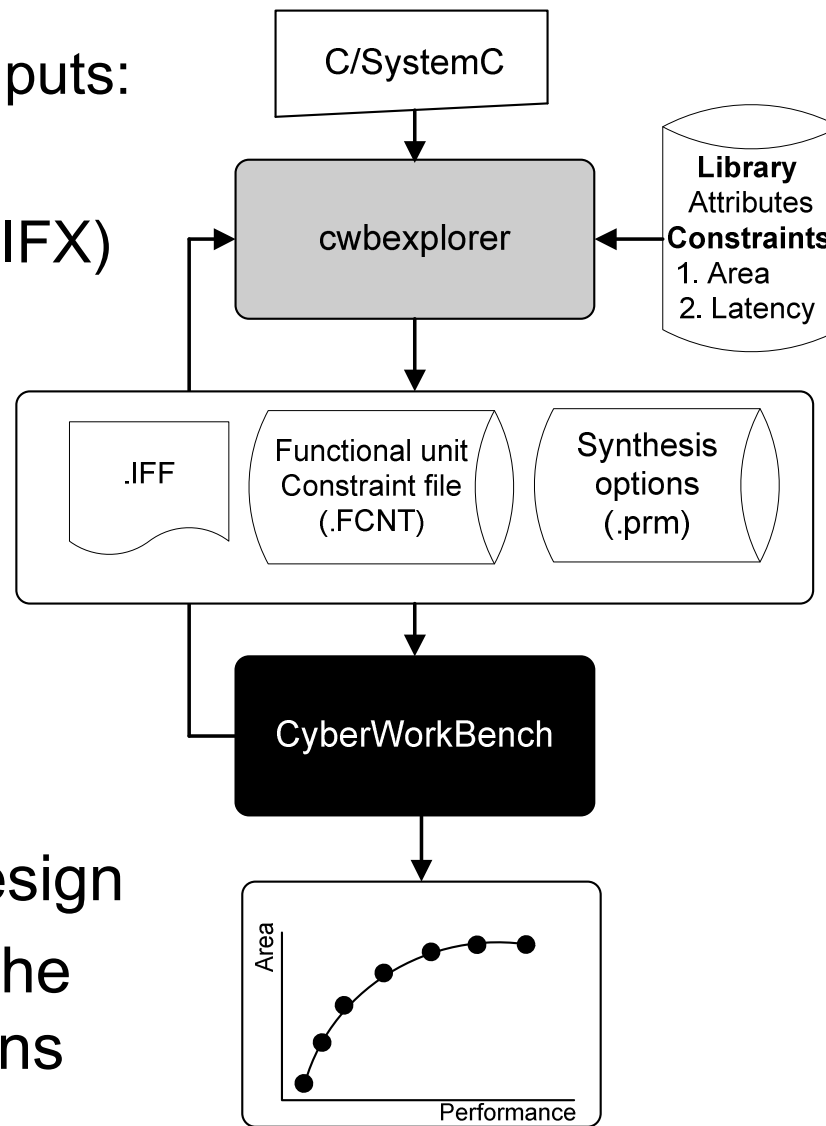
- C/SC file
- Exploration configuration file (IFX)
- Attribute library
- Constraint file

2 Cwbexplorer generates:

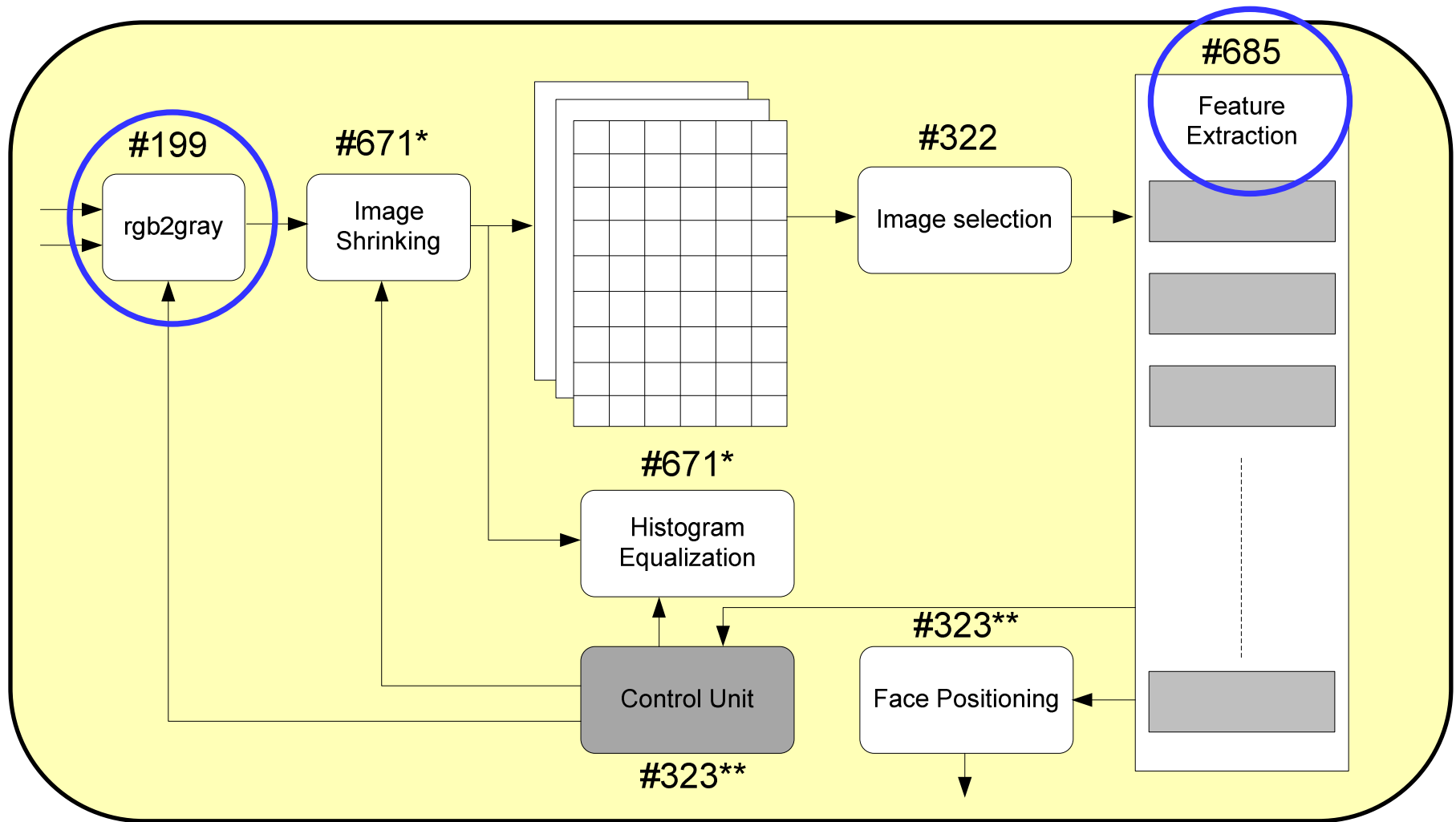
- Global synthesis options
- Functional Constraint File
- New C file with new set of local attributes

3 Call CWB to synthesize the design

4 Read results back to analyze the impact of the exploration options



# Example: Face Detection IP



# = lines of C code

➔ Fully described and synthesized in C

## Design Characteristics

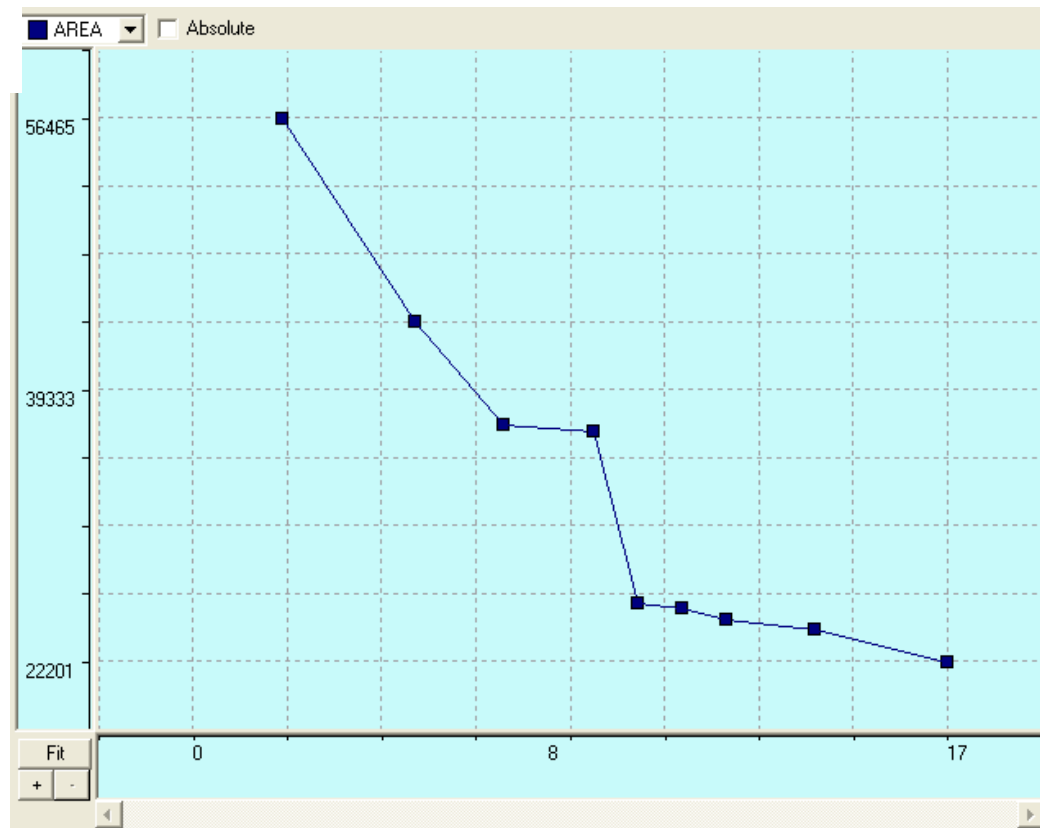
No	Unit	# of C	# of RTL	Development Effort (man/month)
1	rgb2grey	199	1,065	0.1
2	Image Shrinking & Histogram Equalization	671	15,954	0.75
3	Integral Image Selection	322	3,226	0.4
4	Control Unit	323	8,336	0.25
5	Feature extraction	685	37,454	1
	<b>TOTAL</b>	<b>2,200</b>	<b>66,035</b>	<b>2.5</b>

→ All units Freq=200Mhz  
→ 8 frames/second

## Exploration Results: rgb2grey

Name	# lines C	# designs explored	# Optimal designs found	Runtime [s]
rgd2grey	199	82	9	1,518 (~0.4h)

Area

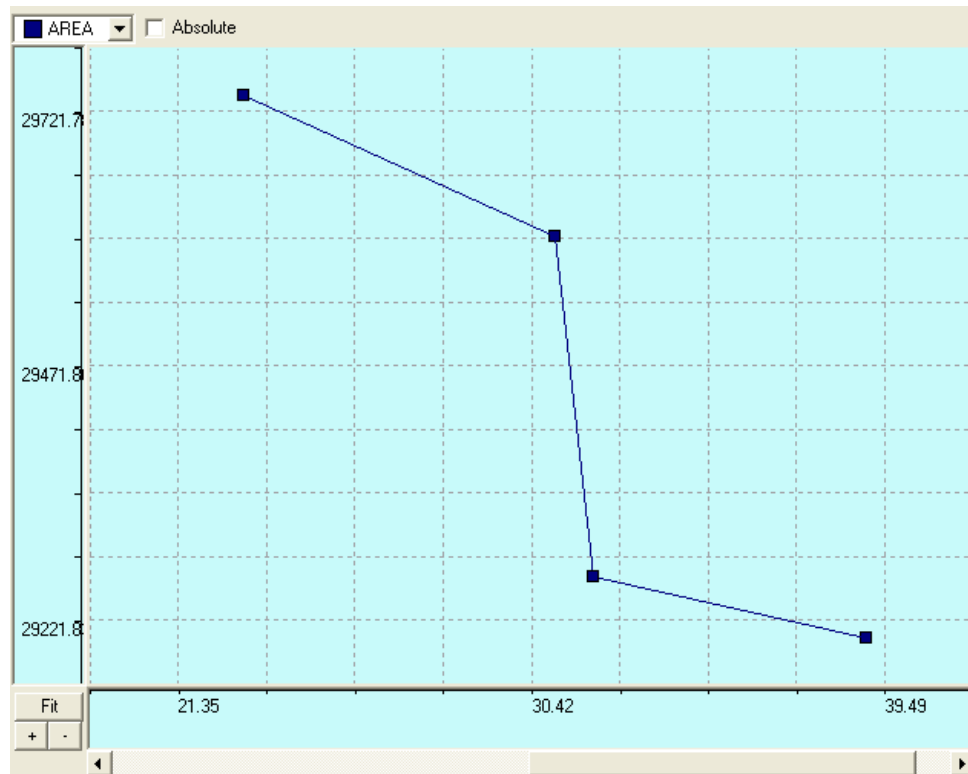


Latency

# Exploration Results: Feature Extraction

Name	# lines C	# designs explored	# Optimal designs found	Runtime [s]
Feature extraction	685	685	4	41,184 (~11h)

Area



Latency

## Limitations

- The design space is extremely large, therefore runtime can be unacceptable  
→ **Solution:** Reduce design space by specifying directly at the source code which attributes to explore for each operation
- Performance measure currently based on state count of the control unit  
→ **Solution:** Need to perform a cycle accurate simulation for each design explored → but increases runtime

## Conclusions

- Automatic exploration tool to bridge the gap between pure algorithmic descriptions and RTL
- Graphical display of the results for easy analysis
- Example using a face detection IP completely written in C
- Main issues:
  - Runtime
  - How to measure circuit performance more accurately
  - How to guarantee finding all or most of Pareto optimal designs

# Q&A

