

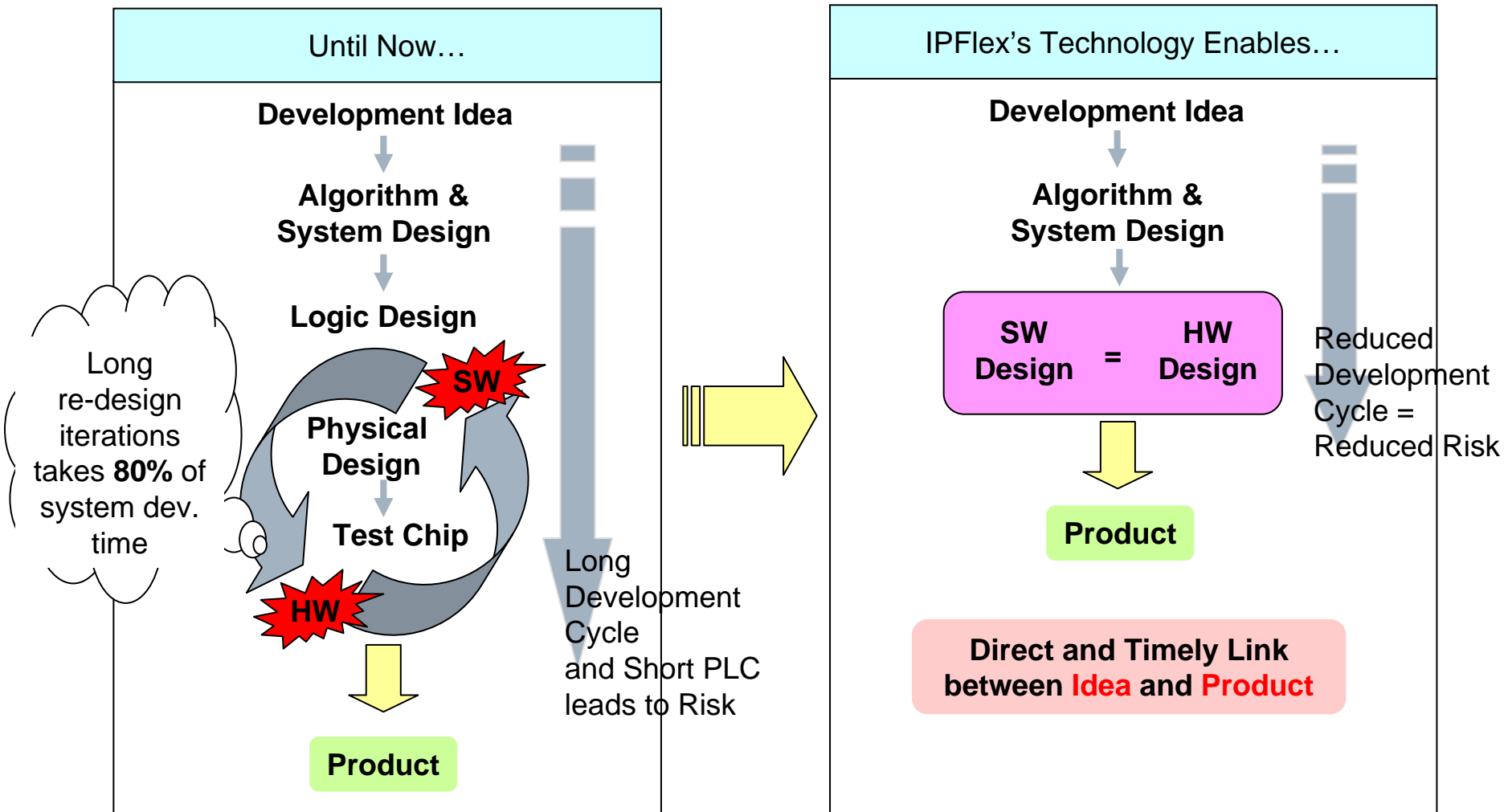
# **C-Based Hardware Design Platform for Dynamically Reconfigurable Processor**

**September 22<sup>nd</sup>, 2005**

**IPFlex Inc.**

- **Merits of C-Based hardware design**
- **Hardware enabling C-Based hardware design**
- **DAPDNA-FW II Design Tool**
- **Programming Example and Performance**

# Short development iteration while achieving high performance



**Dynamically  
Reconfigurable  
Hardware**

Hardware functions reconfigured in  
a single clock

**Software to Silicon™**

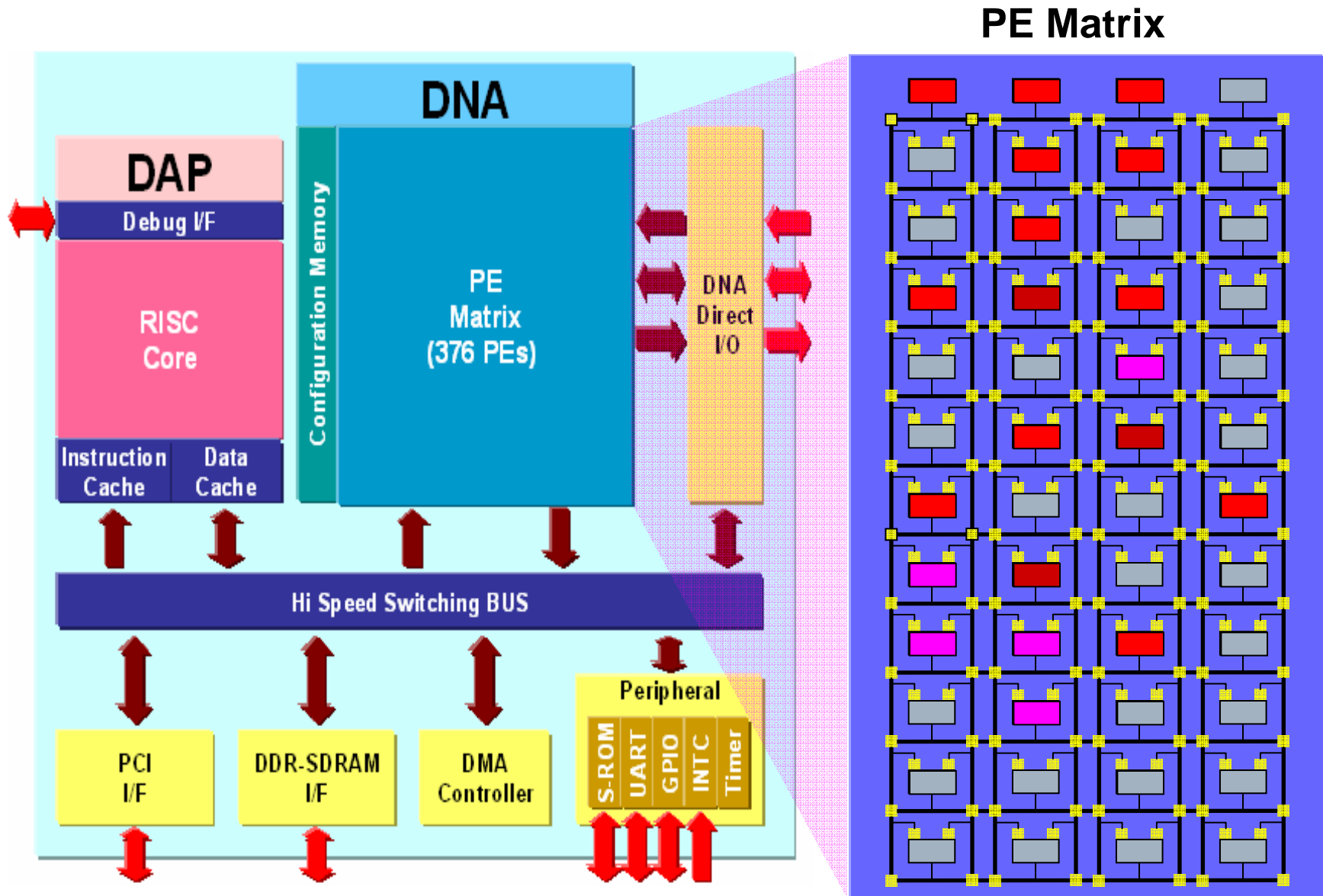
Hardware design with C-like  
language

- Merits of C-Based hardware design
- Hardware enabling C-Based hardware design
- DAPDNA-FW II Design Tool
- Programming Example and Performance

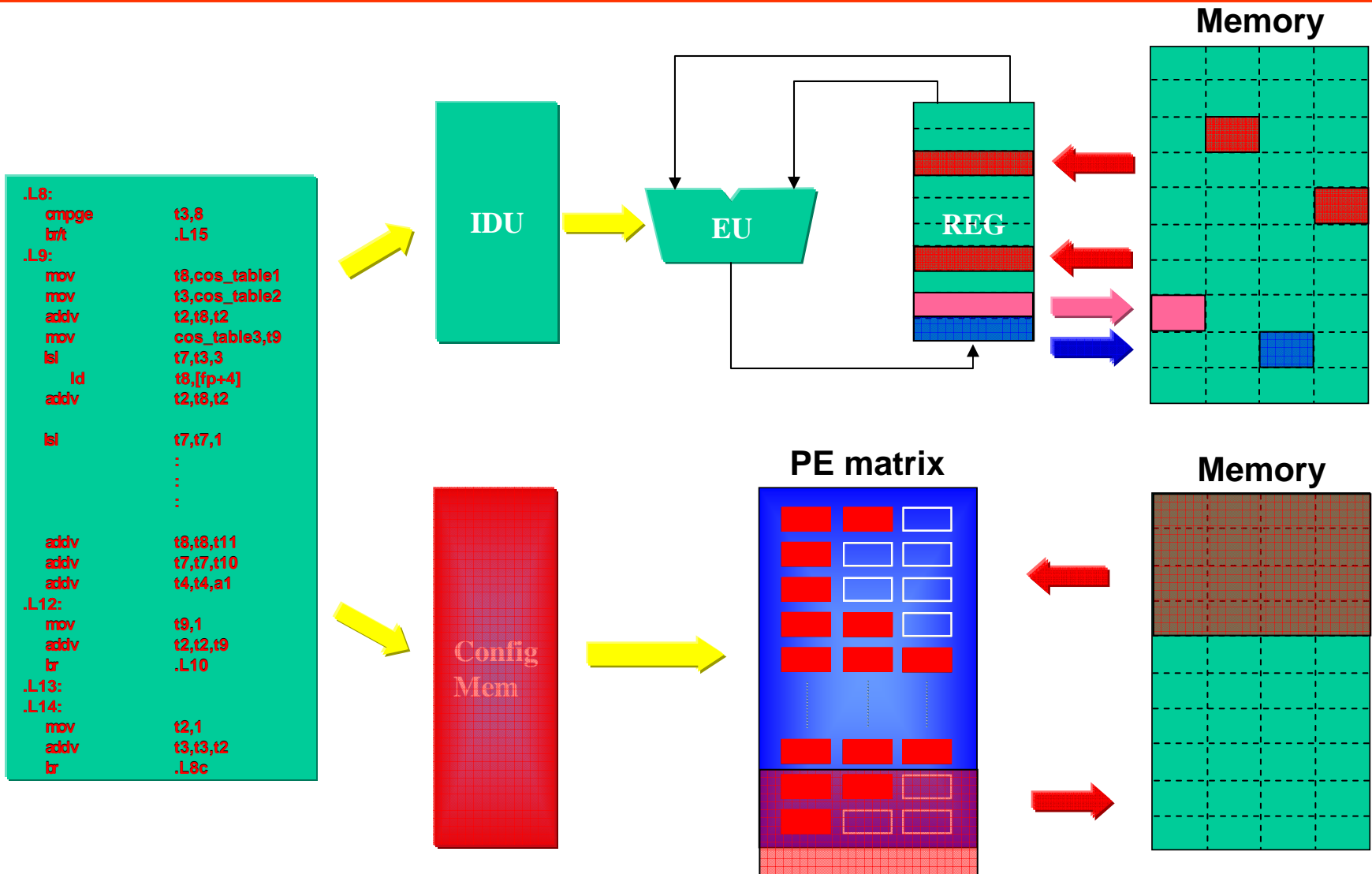


- 166MHz
- 376 32 bit processing elements
- 512KB Internal RAM
- Direct I/O
- DDR SDRAM
- Processor chip, design tool, evaluation boards available

# IPFLEX<sup>®</sup> RISC core and dynamically reconfigurable data flow machine



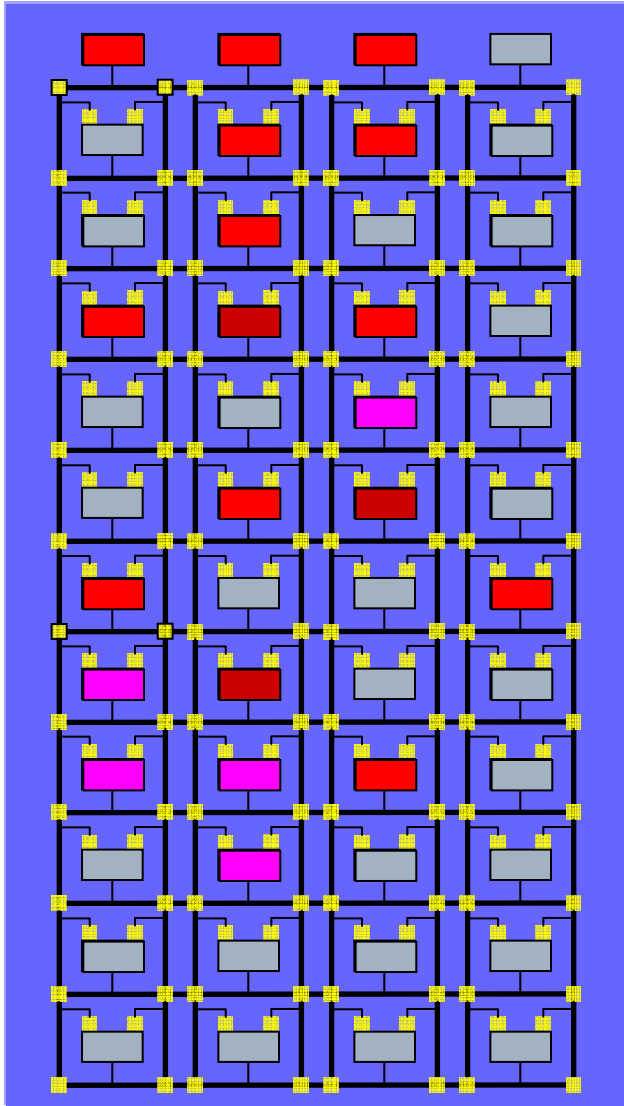
# DAP RISC and DNA Parallel Data Processing





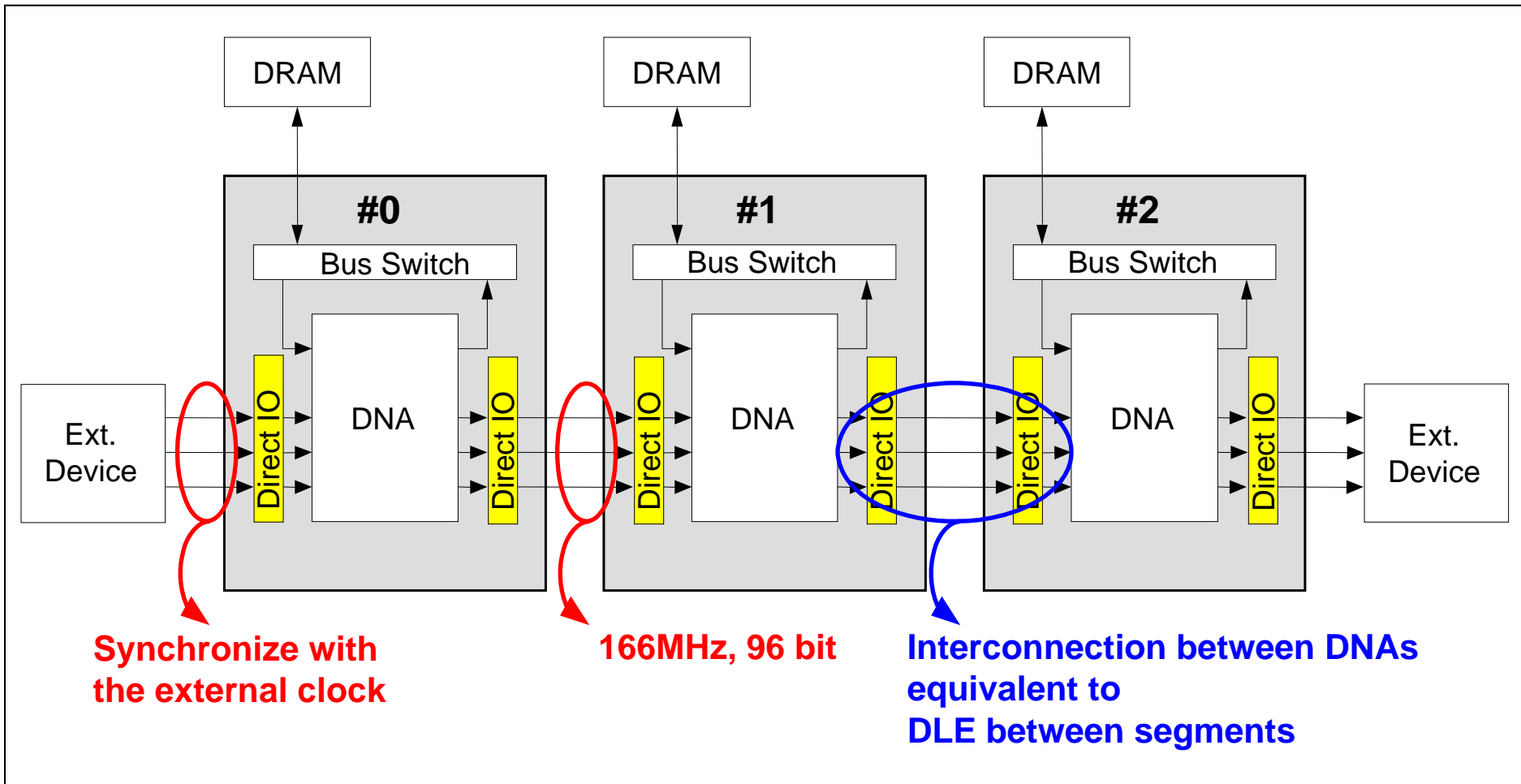
# Heterogeneous PE Matrix

## PE Matrix



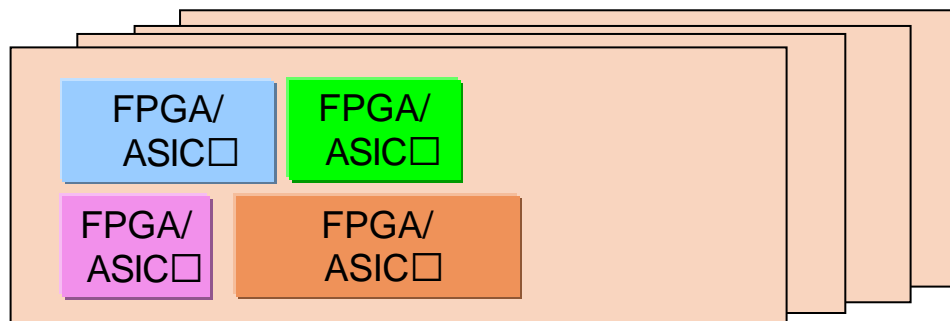
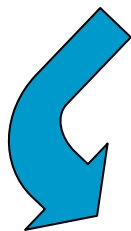
PE	#	Functionality
<b>EXE EXM</b>	<b>112 56</b>	<ul style="list-style-type: none"> <li>• 32 bit 2 inputs / 1 output execution elements</li> <li>• 16bitx16bit-&gt;32bit multipliers</li> </ul>
<b>DLE</b>	<b>136</b>	<ul style="list-style-type: none"> <li>• 32 bit 2 inputs / 1 output delay element</li> </ul>
<b>RAM</b>	<b>32</b>	<ul style="list-style-type: none"> <li>• DNA Internal memory element</li> <li>• 16KBx32 =512KB</li> </ul>
<b>C16E C32E</b>	<b>24</b>	<ul style="list-style-type: none"> <li>• Address generators</li> </ul>
<b>LDx STx</b>	<b>16</b>	<ul style="list-style-type: none"> <li>• I/O buffers</li> </ul>
<b>Total</b>	<b>376</b>	

## 16 DNAs at 16Gbps flow-thru (Total 32 Gbps I/O)



# DAPDNA architecture: solution at any system size

System Example



4 Pipelined Processes

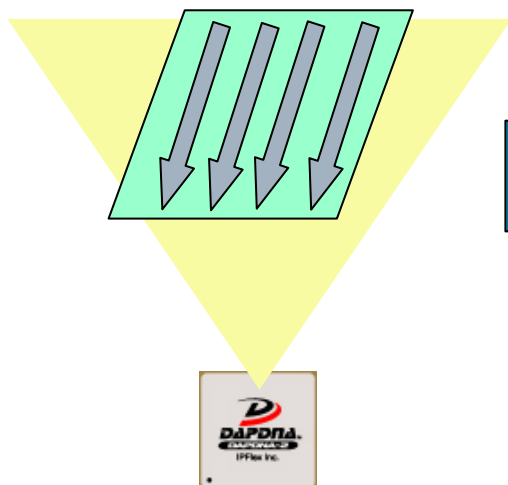
Dynamic Reconfiguration

Scalability

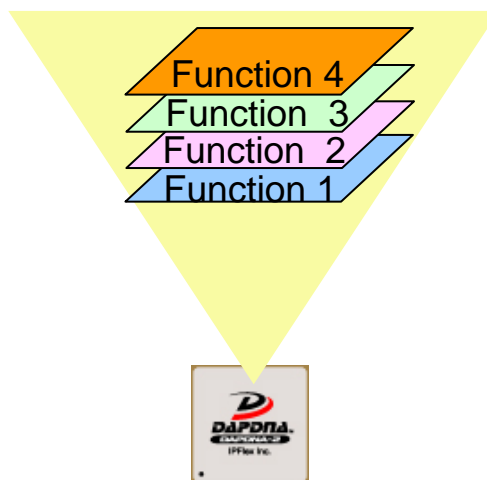
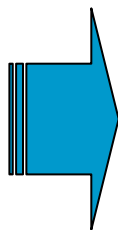
Small scale

Mid/Large size

Ultra large



One DNA Plane



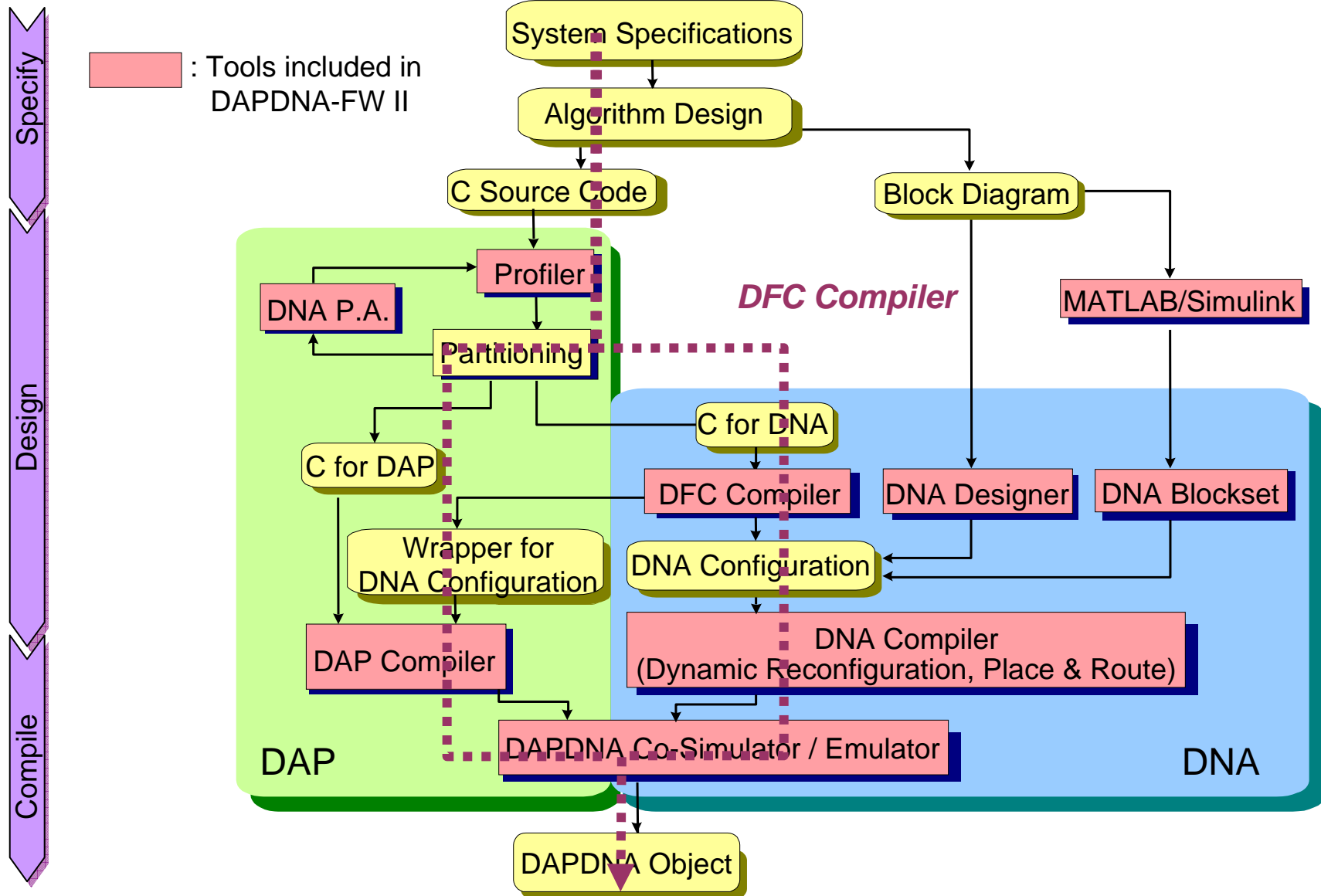
One DAPDNA Processor



Up to 16 DAPDNA Processors

- **Merits of C-Based hardware design**
- **Hardware enabling C-Based hardware design**
- **DAPDNA-FW II Design Tool**
- **Programming Example and Performance**

# Data Flow C compiler: A part of Software to Silicon design tool



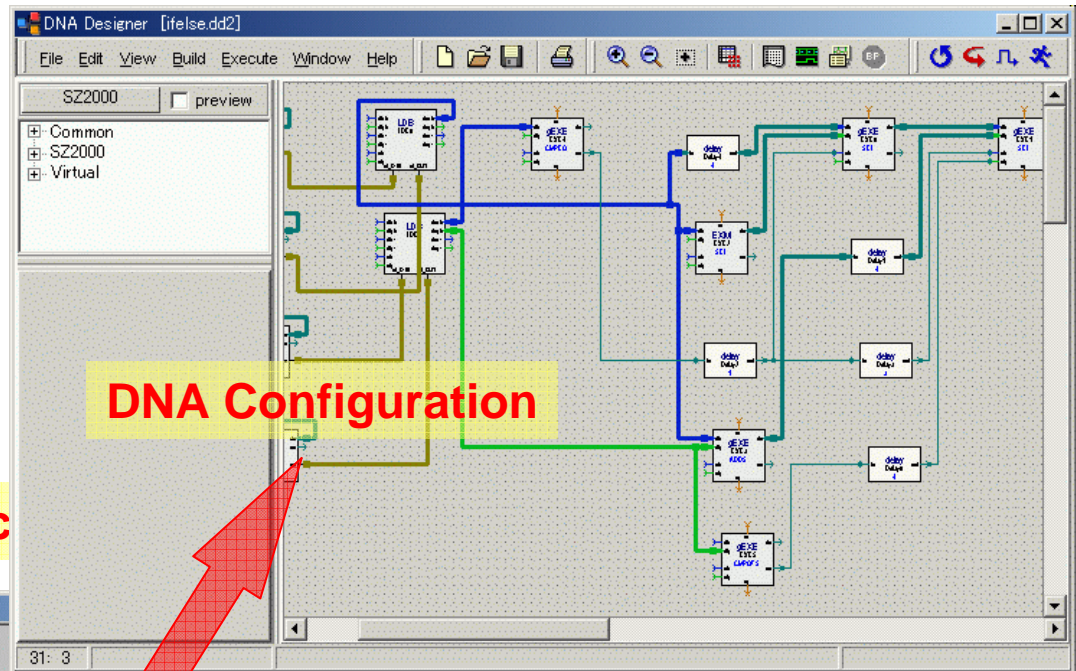
# DFC Compiler: HW configurations from C language

```

ifelse.c - EmEditor
ファイル(F) 編集(E) 検索(S) 表示(V) ツール(T)
ウィンドウ(W) ヘルプ(H)

void ifelse(int a[], int b[], int r[])
{
    int i;
    int data;
    for(i=0; i<5; i++) {
        data = a[i];
        if( b[i] == 0 ) {
            data = a[i] * 50;
        } else if( b[i] > 1 ) {
            data = a[i] + b[i];
        }
        r[i] = data;
    }
}
    
```

**C-based Data Flow Desc**



**DNA Configuration**

```

~/work/eval/dfc/ifelse
sugawara@POSITRON2 ~/work/eval/dfc/ifelse
$ dfc.exe -device t -wt ../fir32/template.c ifelse.c
Warning: non-generic element UQID=3(EXE0) treated as generic based on parameters
Warning: non-generic element UQID=4(EXE1) treated as generic based on parameters
Warning: non-generic element UQID=10(EXE3) treated as generic based on parameter
Warning: non-generic element UQID=15(EXE4) treated as generic based on parameter
Warning: non-generic element UQID=16(EXE5) treated as generic based on parameter
sugawara@POSITRON2 ~/work/eval/dfc/ifelse
$
    
```

**DFC Compiler**

```

DNA Control code for DAP
static void int_handler(void) {
    io_write(DNCINTCLR, 0xFF);
    finished = 1;
}

void ifelse( uint32 a[], uint32 b[], uint32 r[] ) {
    int i;
    int debugData;
    unsigned long tmpData;

    printf("Starting test...\n");

    io_write(DNCCNTPLSW, 0x00000002); // init
    io_write(DNCINTCLR, -1); // clear all interrupt stuff
}
    
```

**DNA Control code for DAP**

# IP FLEX® HW configurations written in DFC as functions

## Application (DAP C code)

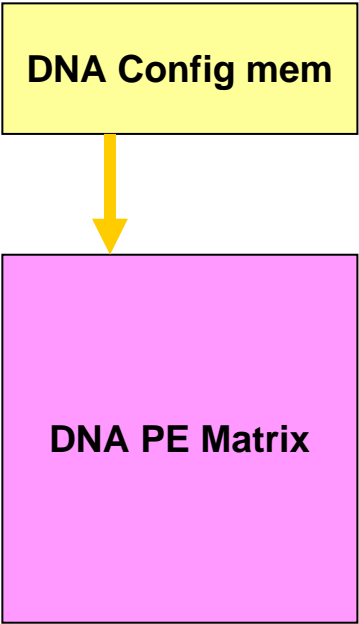
```
void top_level()
{
...
fir(d,h,r);
...
}
```

Call

### DNA Control Code

```
void fir (int d[ ], int h[ ], int r[ ])
{
dna_cfgram_load3(BANK3, cfg_fir);
dna_config(0,BANK3);
dna_run();
...
dna_stop();
...
}
```

### DNA Hardware



Load config data

Bank Switch

DNA Run

DNA Stop

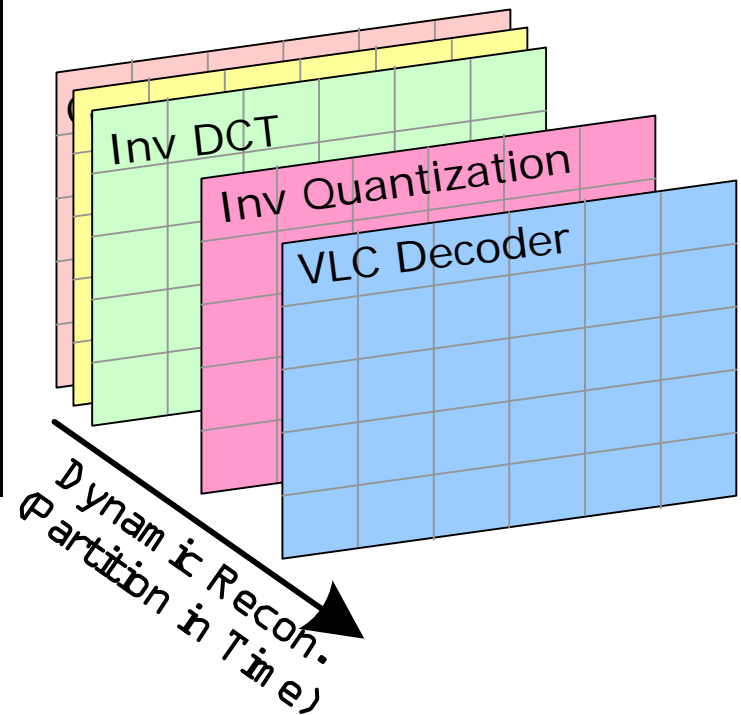
## Application (DAP C code)

```

...
while (1) {
    ...
    VLC_Decoder(Vector, Quant_dat, Input);
    Inv_Quantize(dct_dat, quant_dat);
    Inv_DCT(mc_dat, dct_dat);
    Motion_Comp(mcu_dat, mc_dat);
    Color_Conv(mcu_dat, mcu_dat);
    ...
}

```

## DNA Hardware

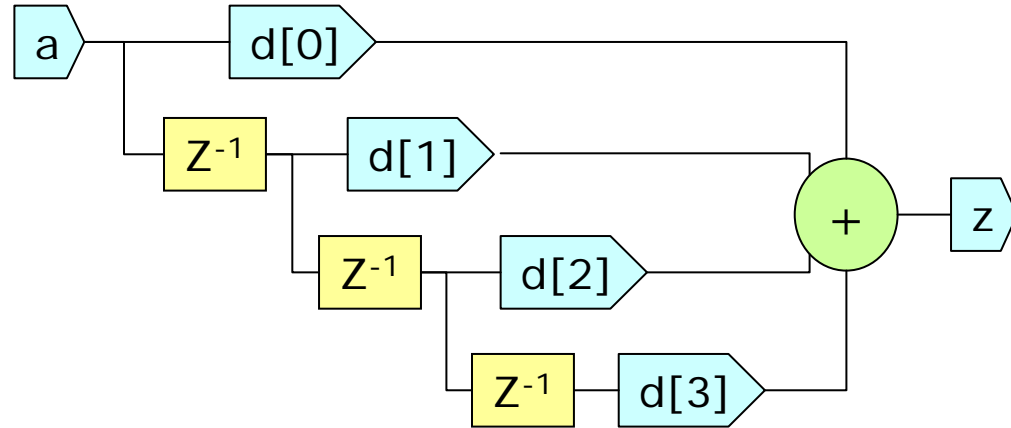




# IP FLEX® Extension on ANSI C for data flow systems

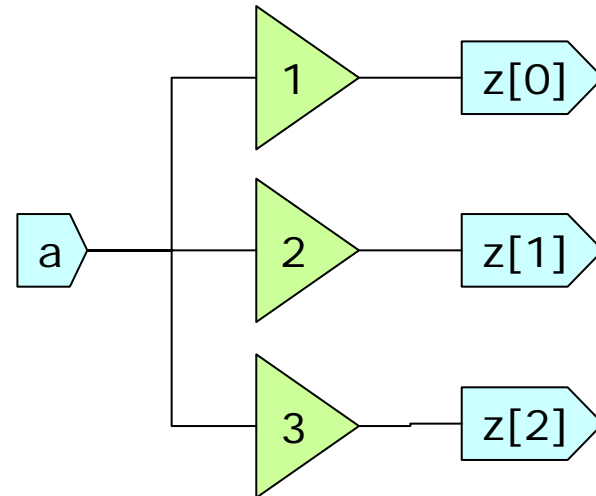
## ● Delayline: delayed data signals

```
delayline d;  
d = a;  
z = d[0] + d[1]  
    + d[2] + d[3];
```



## ● SEQ: Static code replication

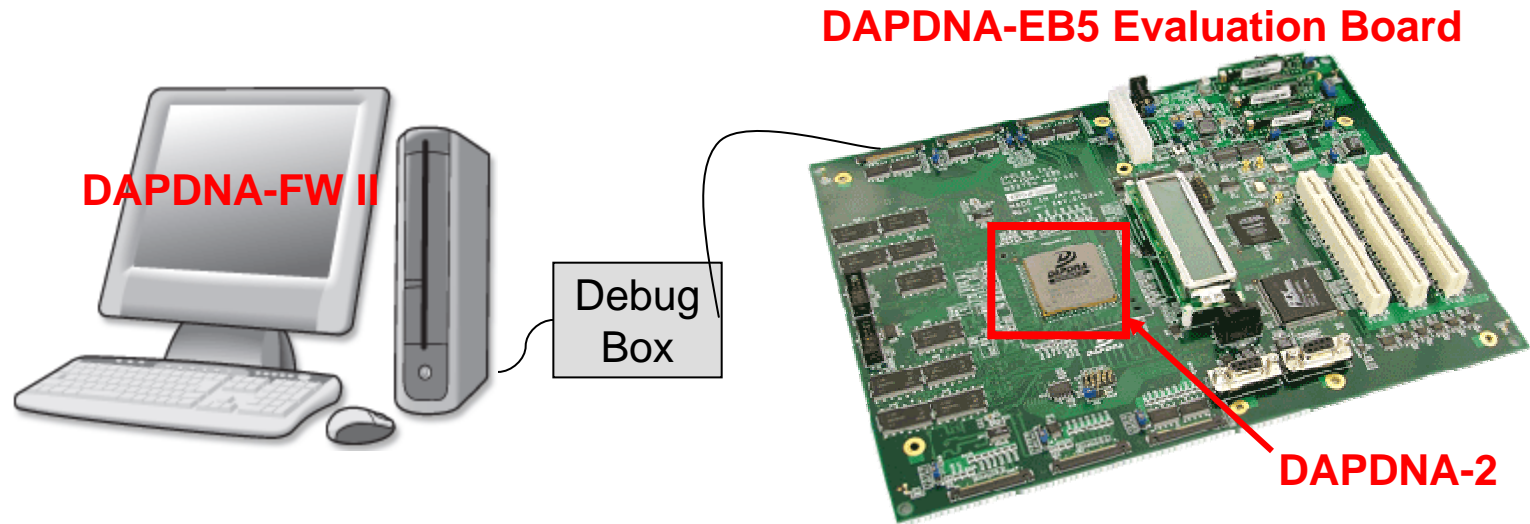
```
int h[] = {1,2,3};  
int z[];  
seq (i=0; i<3; i++) {  
    z[i] = a * h[i];  
}
```



- **Short development period**
  - Rapid prototyping
  - Co-Simulation between Processor (DAP RISC) and Data Flow Machine (DNA PE Matrix)
  - Deterministic placement / guaranteed clock speed @ 166Mhz
- **Flexibility to change HW according to the application in demand**
  - Dynamic reconfiguration in single clock
- **Performance close to custom device**
  - Deep pipelining and massive parallelization
  - 376 processing elements

- **Merits of C-Based hardware design**
- **Hardware enabling C-Based hardware design**
- **DAPDNA-FW II Design Tool**
- **Programming Example and Performance**

# Using dynamic reconfigurability to change image filters

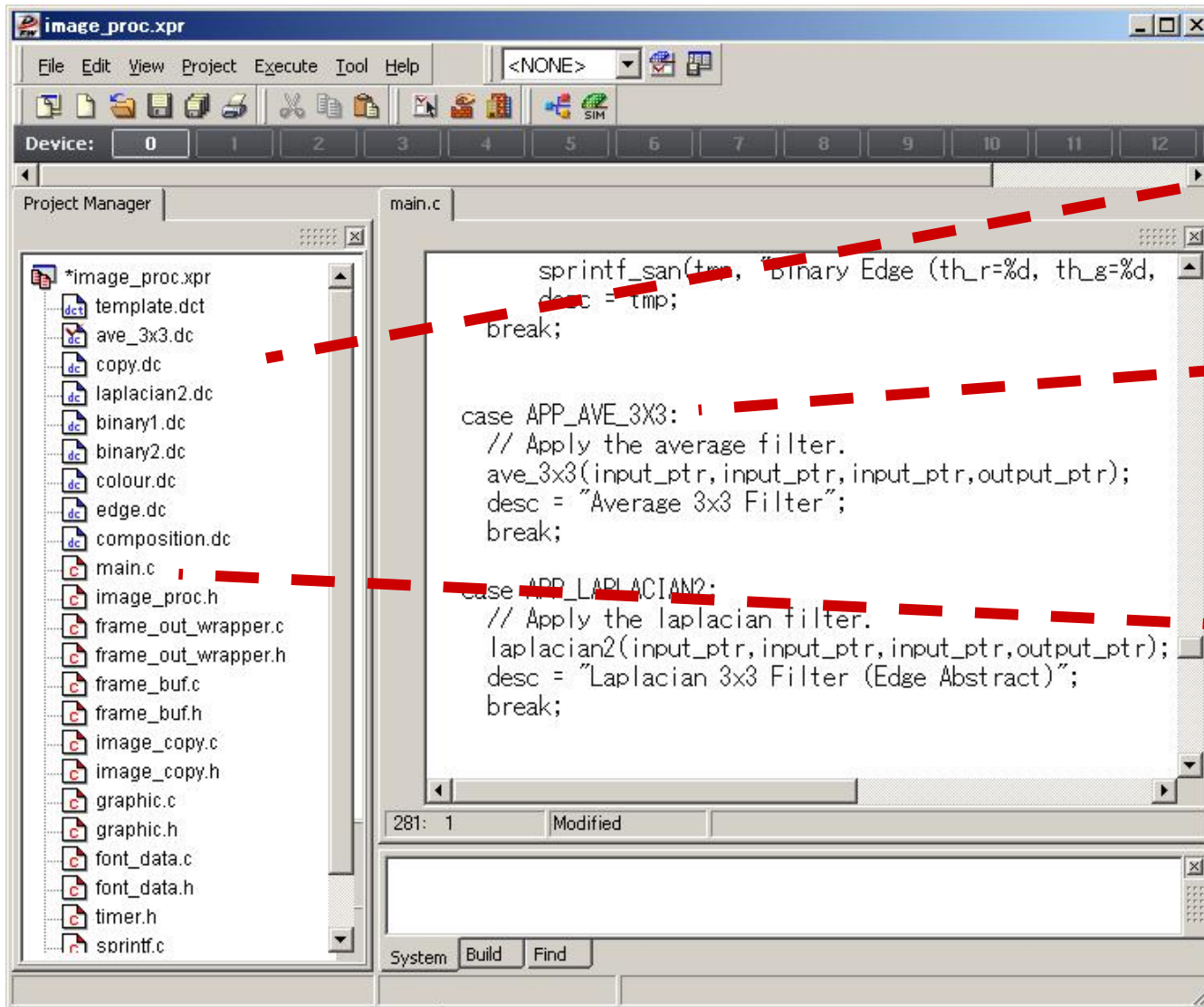


Various image filters written in DFC and compiled

- Laplacian
- Binary
- Average
- Etc.

Filters run on DAPDNA-2 using dynamic reconfigurability of hardware

# DAP calls DNA configurations with simple function calls

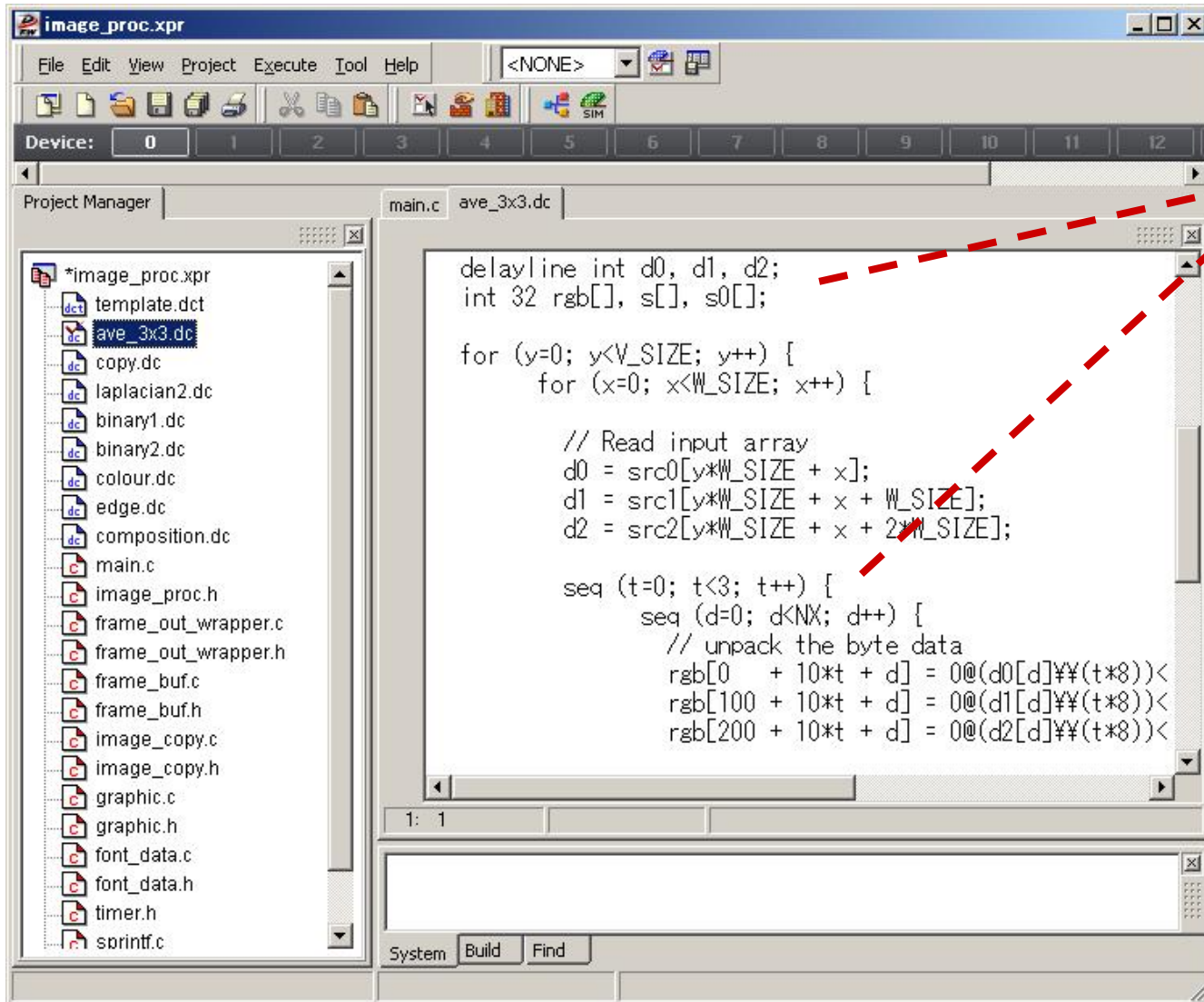


List of filters written in Data Flow c

An Average3x3 filter being called in main.c as a function

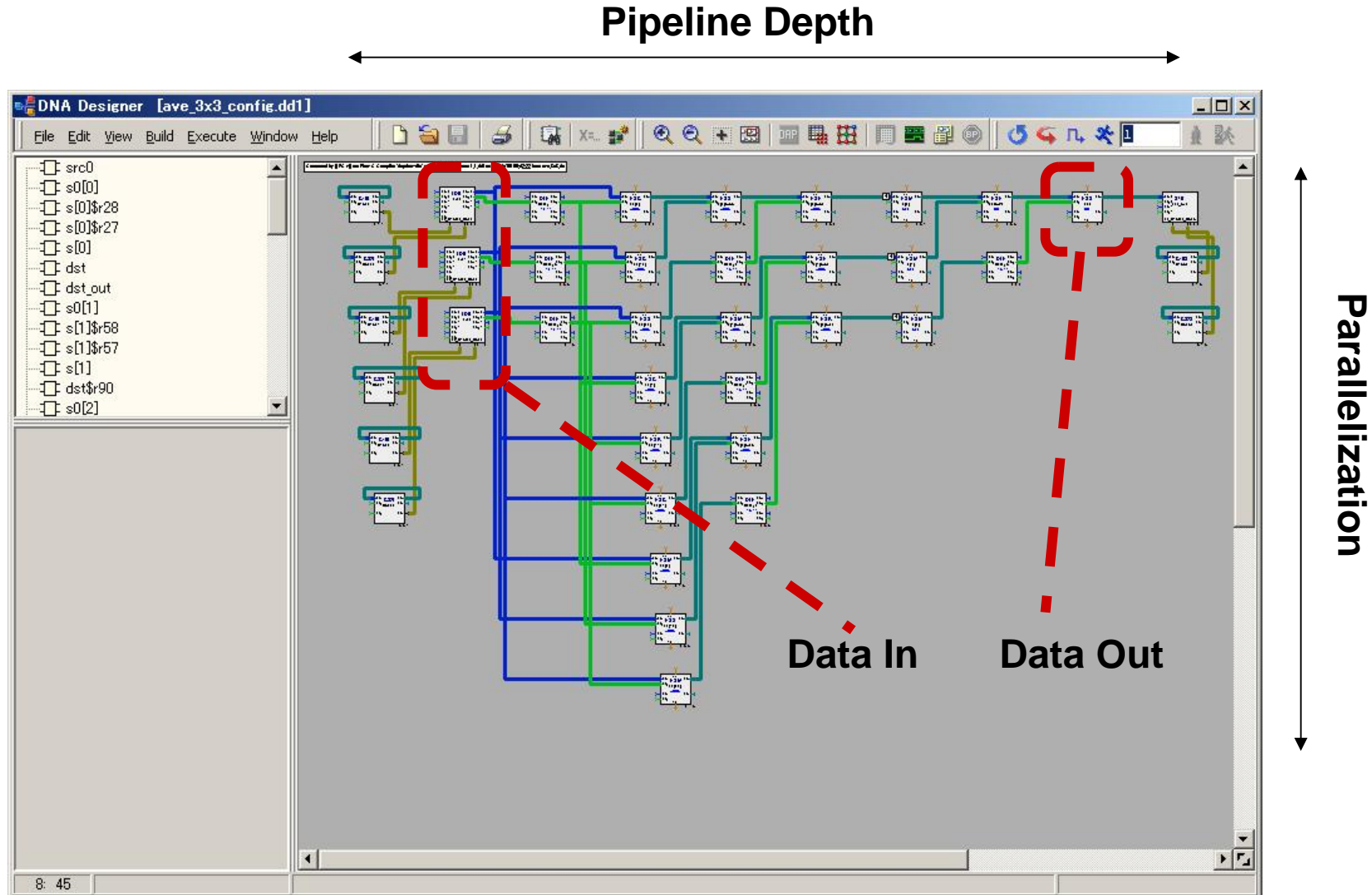
Main.c for DAP RISC

# DFC-described filters compiled into HW with a single click of build button...

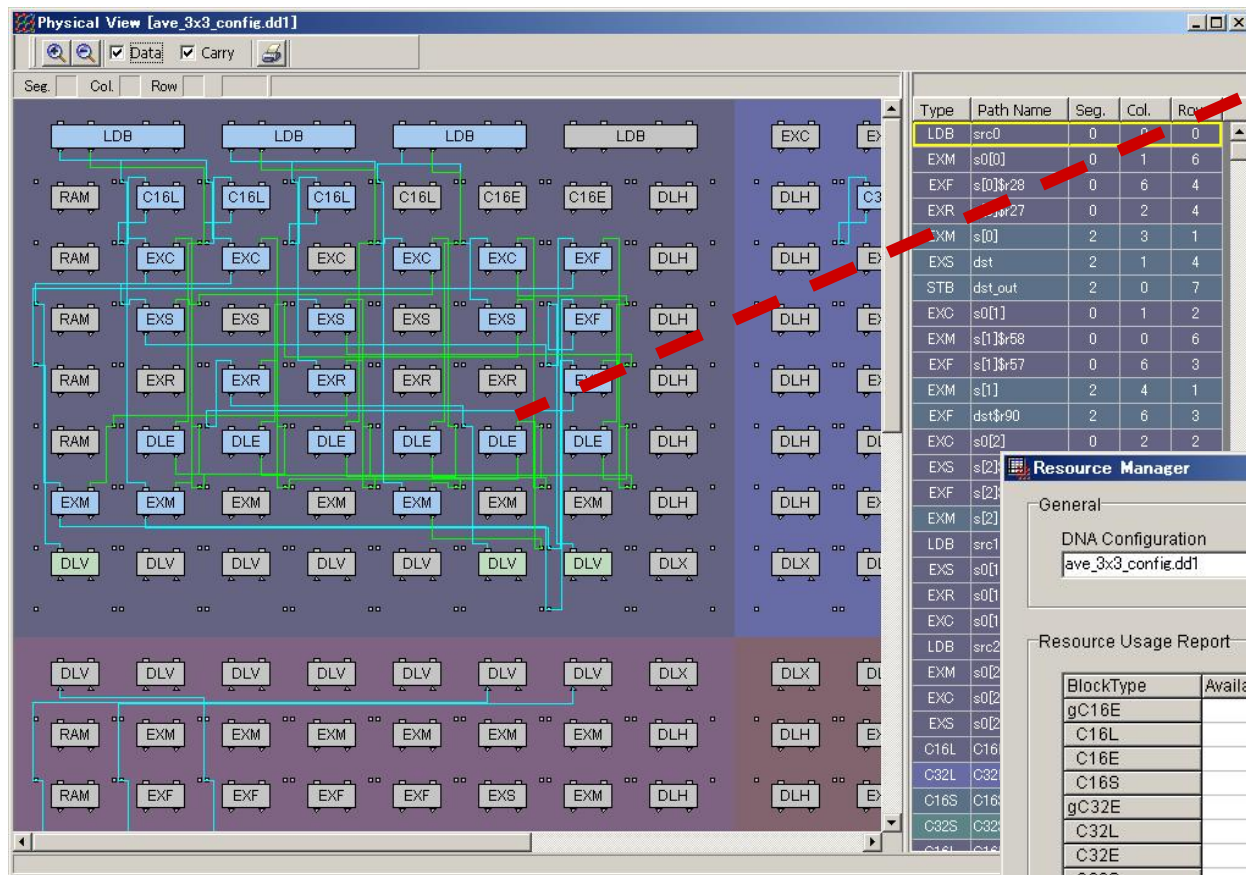


Average3x3 filter described in DFC uses delayline and seq commands to take advantage of parallelization

# ...resulting in a logical representation of the filter with 32-bit processing elements...



# ...as well as physical mapping on DNA



Place & Route result

Resource manager displays usage and power consumption

**Resource Manager**

General

DNA Configuration: ave\_3x3\_config.dd1 Update

---

Resource Usage Report

BlockType	Available	Used	Remaining
gC16E	12	4	8
C16L	4	3	1
C16E	4	0	4
C16S	4	1	3
gC32E	12	4	8
C32L	4	3	1
C32E	4	0	4
C32S	4	1	3
RAM	32	0	32
<b>Total</b>	<b>376</b>	<b>45</b>	<b>331</b>

Power Consumption: 526 [mW]

Error Check    Save Report





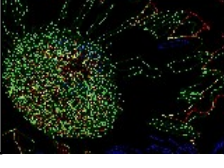
Automatic Update    Close



# DFC compiler result for image filters

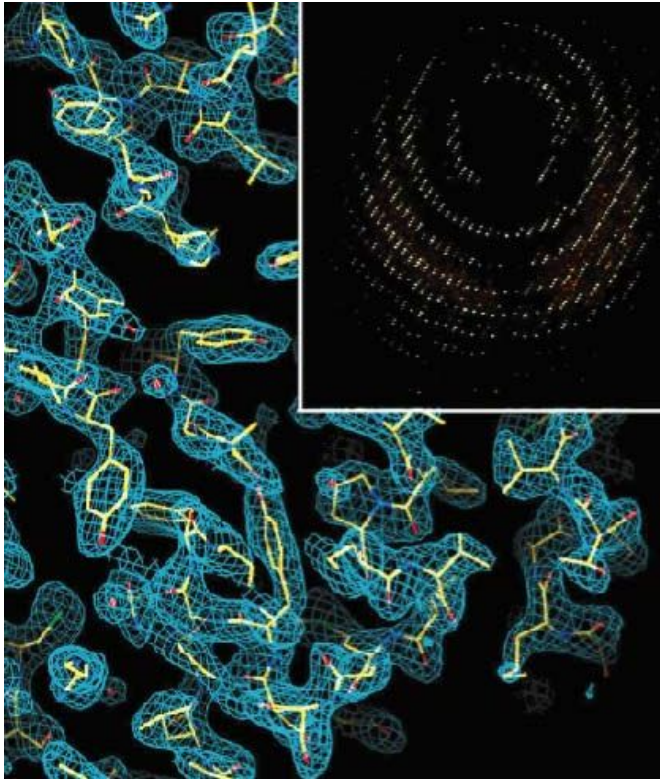
Image Filters		DAPDNA-2 (166 Mhz)						
		EXE	EXM	RAM	DLE	Parallel	Performance	
							M Pixel/sec	Fps(800x600)
3x3 Average	Usage %	12.5	10.7	0	5.1	3	<b>292</b>	203
Binary	Usage %	5.4	3.6	0	1.5	3	<b>498</b>	346
Binary Image Edge Det.	Usage %	5.4	3.6	6.3	1.5	1	<b>292</b>	203

# Optimized result with DNA Designer (Block Diagram) tool

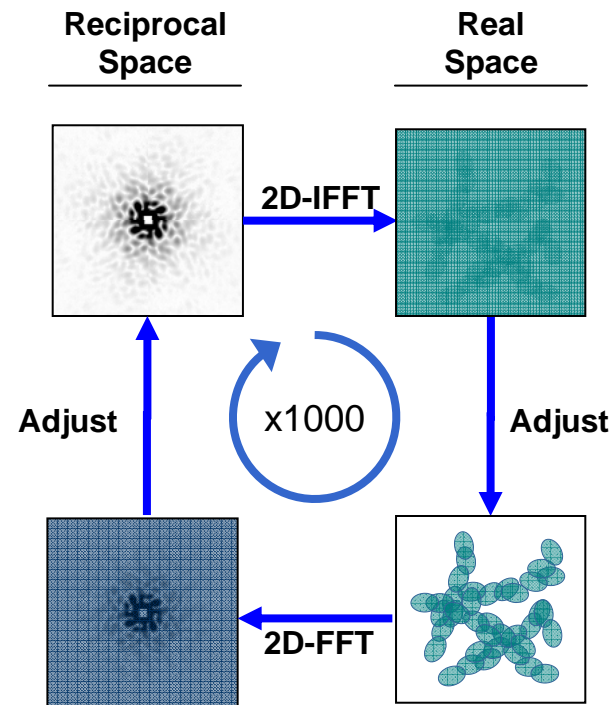
Image Filters	Images	Performance (M Pixels / s)		Performance Multiple
		Pentium 4* (3.06Ghz)	DAPDNA-2 (166Mhz)	
3x3 Average		15.5	<b>664.0</b>	<b>43</b>
3x3 Laplacian (Edge Enhancement)		15.8	<b>664.0</b>	<b>42</b>
3x3 Laplacian (Edge Detection)		15.8	<b>664.0</b>	<b>42</b>
Binary		225.0	<b>839.0</b>	<b>4</b>
Binary Image Edge Detection		38.6	<b>664.0</b>	<b>17</b>

\* Pentium4@3.06GHz, Red Hat Linux 8.0, gcc 3.2, SIMD instruction not used, 24bit color, 800x600 pixels, 1000 Repetitions, Average performance (pixel/sec)

- Extrapolating molecular structure from X-ray diffraction image



- Extrapolation Algorithm (HIO)

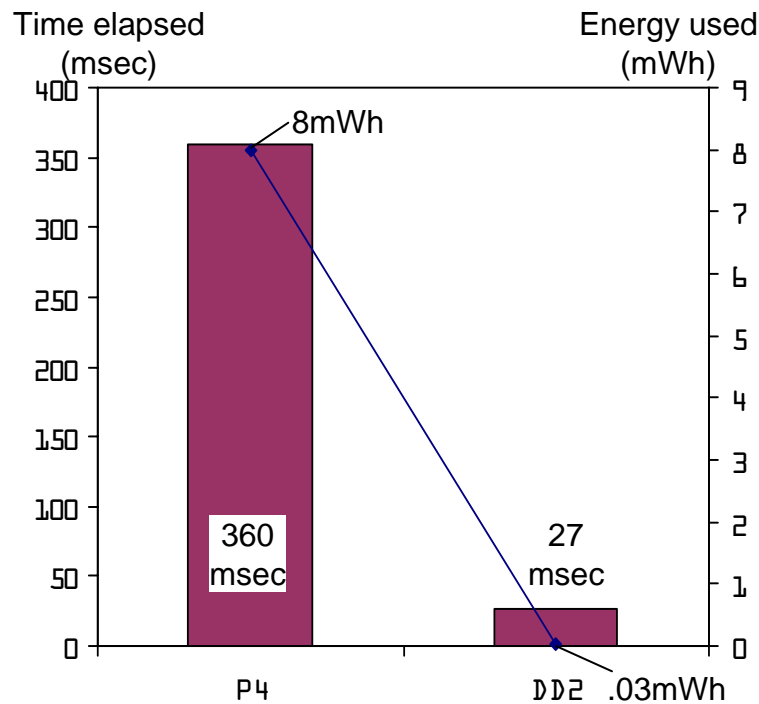


- Requires 1024 X 1024 32-bit FFTs

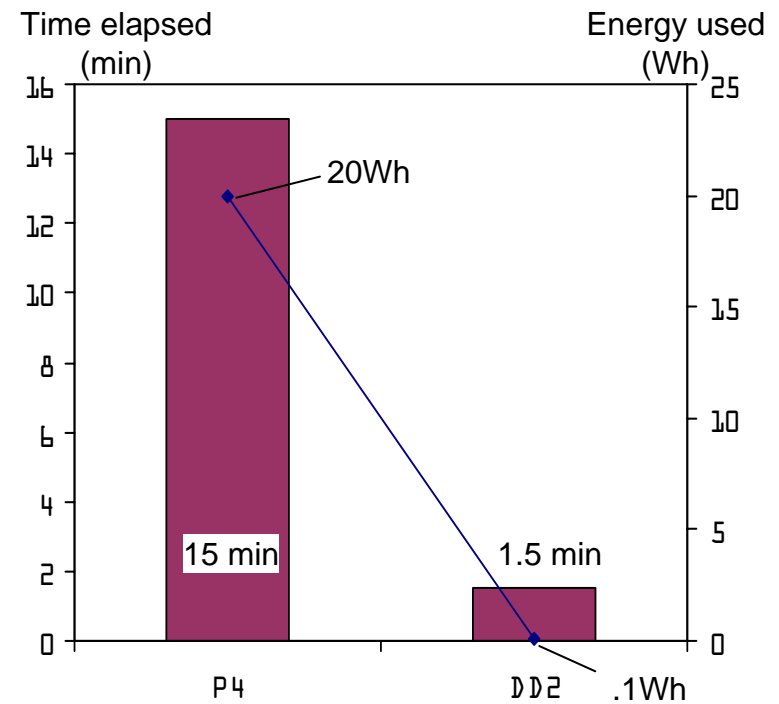


# DAPDNA-2 is 10X faster than P4 with 1/200 energy

### 2D-FFT (1024x1024)



### 1000 iterations (HIO)



Time

13 : 1

10 : 1

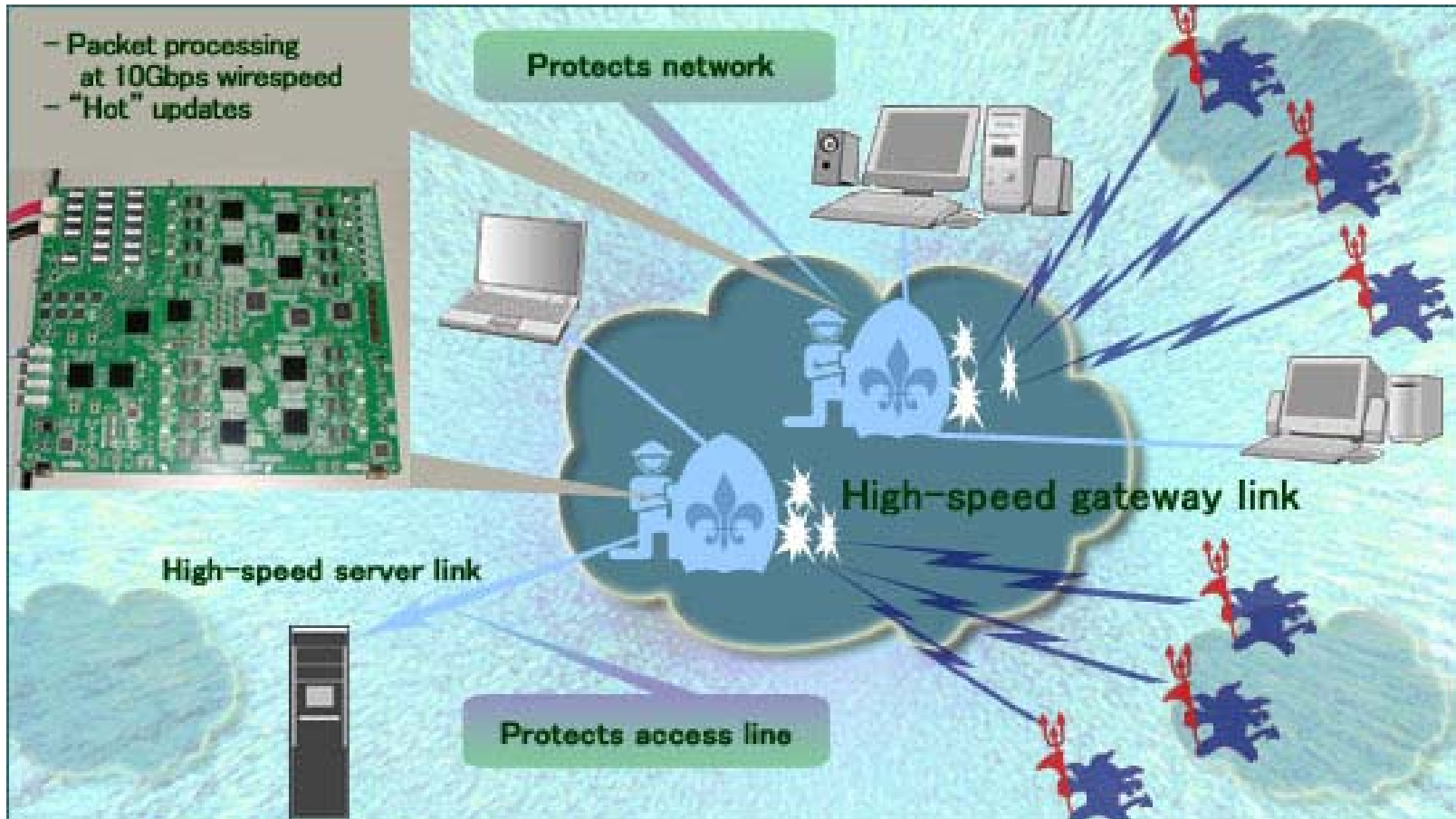
Energy

267 : 1

200 : 1

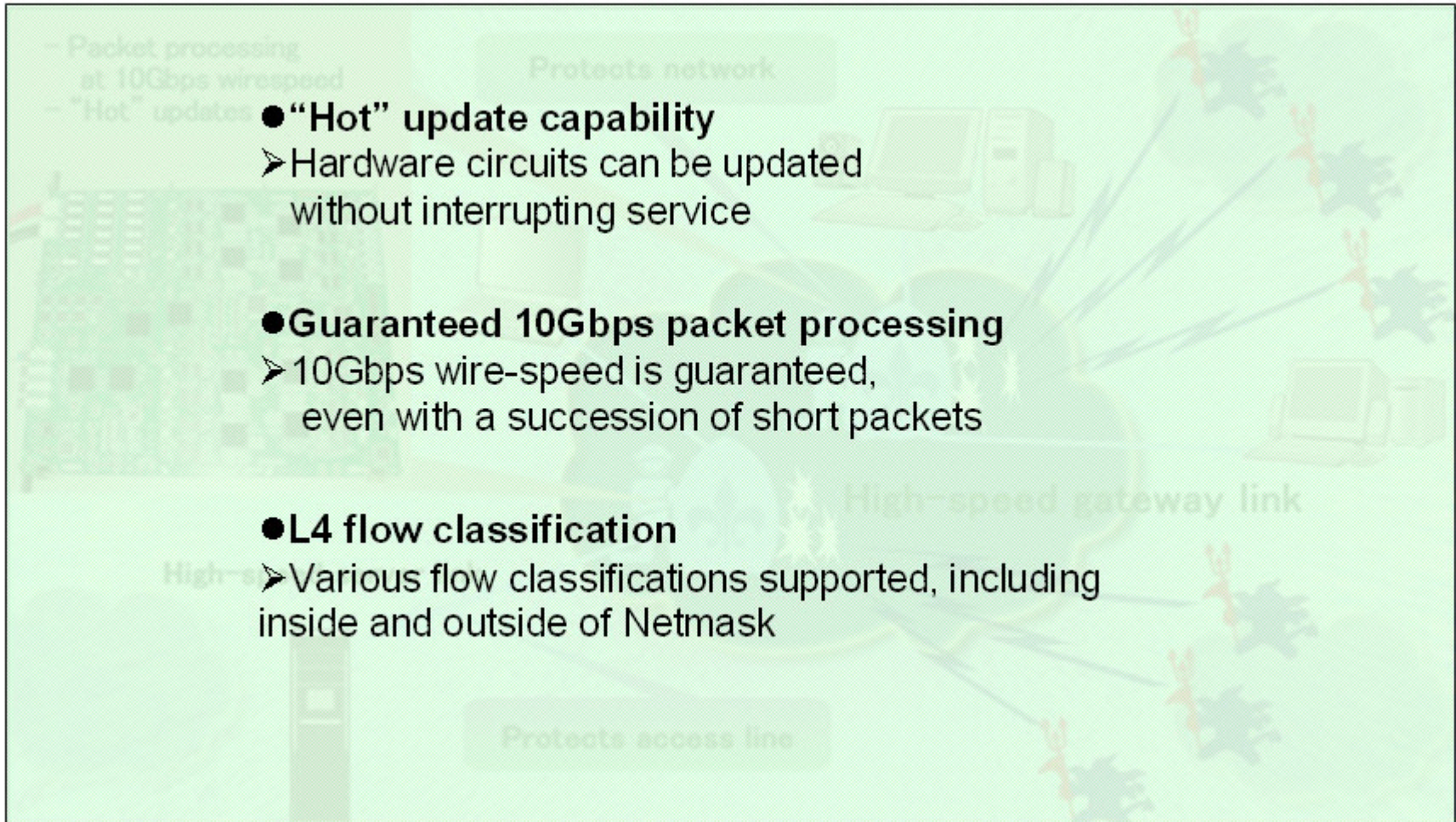
\* Assuming 80W for P4 and 4W for DAPDNA-2  
Source: IPFlex experimentation

# 10Gbps active firewall by NTT



10Gb/s Firewall System using Reconfigurable Processors  
Reconfigurable system technical committee

# 10Gbps active firewall by NTT

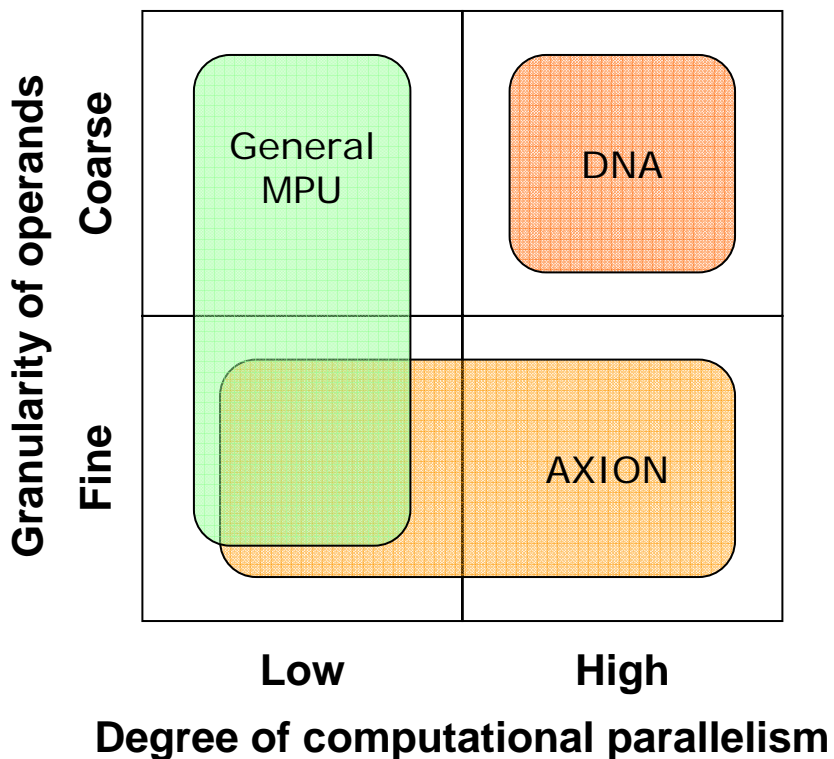


- Packet processing at 10Gbps wire-speed
- "Hot" updates

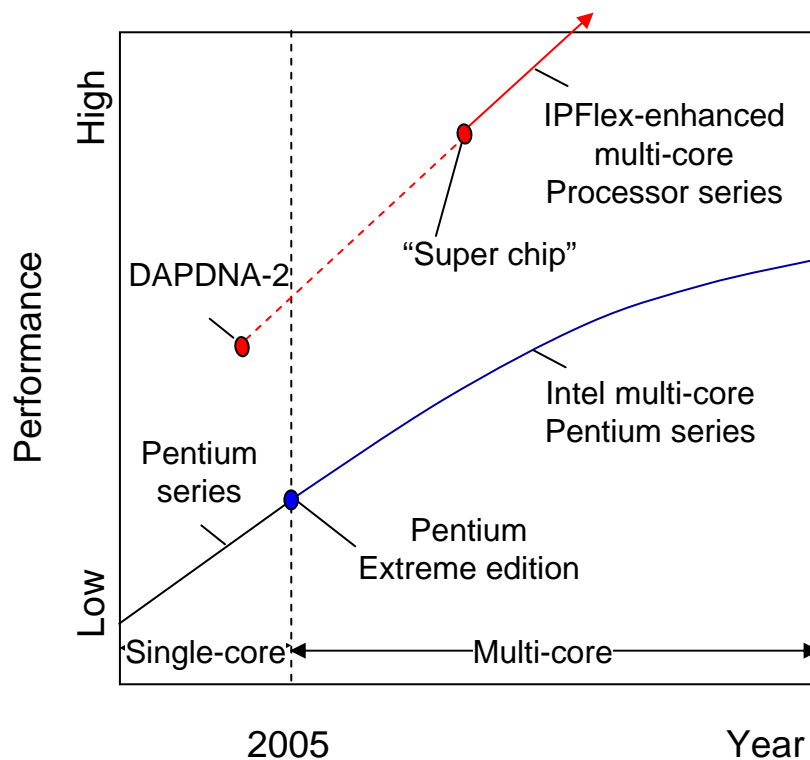
- **"Hot" update capability**
  - Hardware circuits can be updated without interrupting service
- **Guaranteed 10Gbps packet processing**
  - 10Gbps wire-speed is guaranteed, even with a succession of short packets
- **L4 flow classification**
  - Various flow classifications supported, including inside and outside of Netmask

10Gb/s Firewall System using Reconfigurable Processors  
Reconfigurable system technical committee

General MPU like Pentium and DNA+AXION complement well

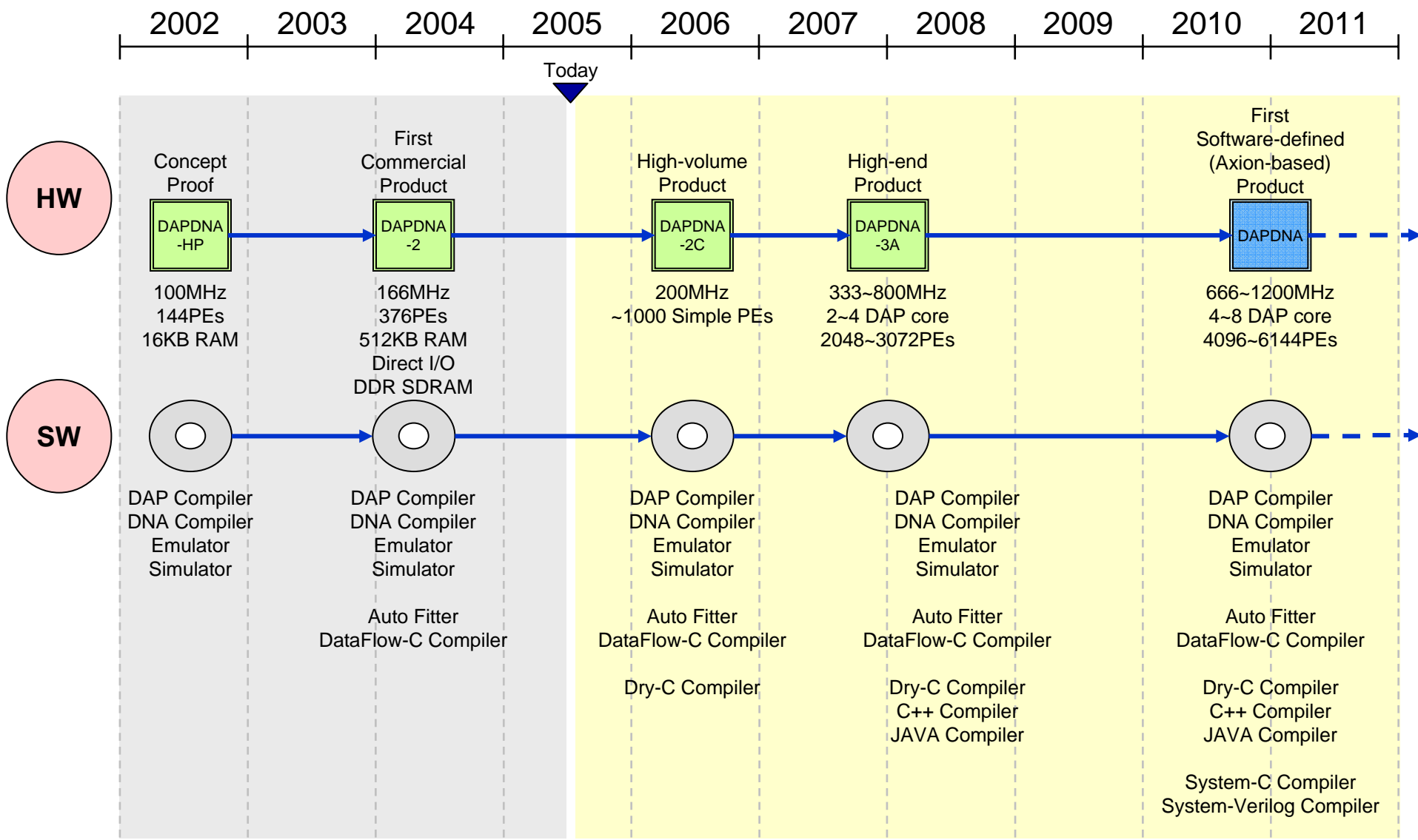


Realizes extreme performance





# IPFlex's technology / product roadmap







# **Dynamically Reconfigurable Processor and C-Based Hardware Design Platform**

**by**

**IPFlex Inc.**

**[www.ipflex.com](http://www.ipflex.com)**

**[info@ipflex.com](mailto:info@ipflex.com)**