Datapath Width Optimization for Customizable Processors

Hiroto Yasuura
System LSI Research Center
Kyushu Univ., Japan

Assumptions

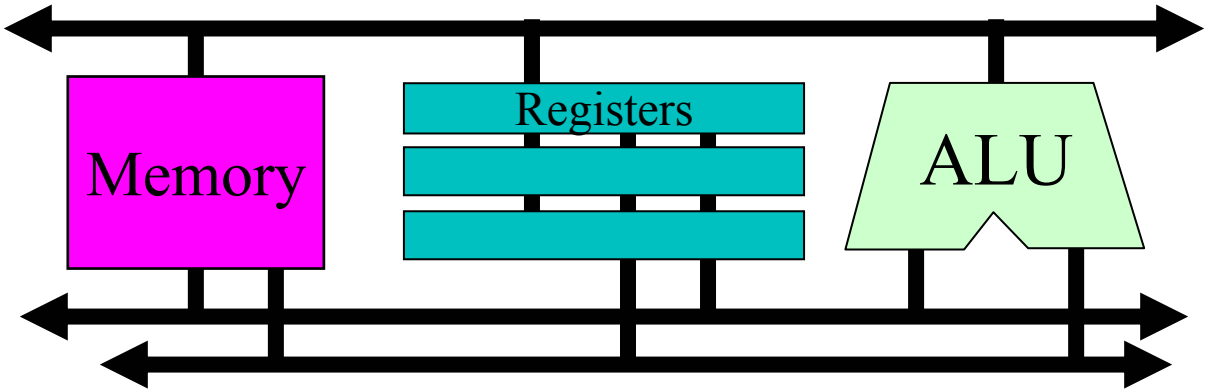
- SOCs for
  - Battery Operated Consumer Products
  - Cost < $50
  - Power Consumption < 1W
  - # of Products < 1M
- First, functionality of an SOC is designed and then optimization of cost and energy is done under the constraints on performance.

Datapath Width Optimization for Customizable Processors

- **Problem Definition**
- Basic Techniques
  - » Effective Bit Analysis (Variable Size Analysis)
  - » Customizable Processor
  - » Programming Language and Compiler
- Optimization Flow
- Memory Architecture
- Quality Driven Design

Datapath Width

- Width of Data Buses
- Length of Data Registers
- Bit Width of Operation Units
- Word Length of Memories



We don't use all bits on the Datapath

(Ex. MPEG-2 Video Decoder)

ize of Program :
275 lines (written in C)
he number of *int*:406

Bit Width	# of variables
1 bits	50
2 bits	17
3 bits	11
4 bits	11
5 bits	10
11 bits	6
12 bits	17
13 bits	0
14 bits	46
15 bits	2
16 bits	39

Bit Width	# of variables
17 bits	2
18 bits	3
19 bits	0
20 bits	6
21 bits	0
27 bits	4
28 bits	3
29 bits	3
30 bits	7
31 bits	0
32 bits	5

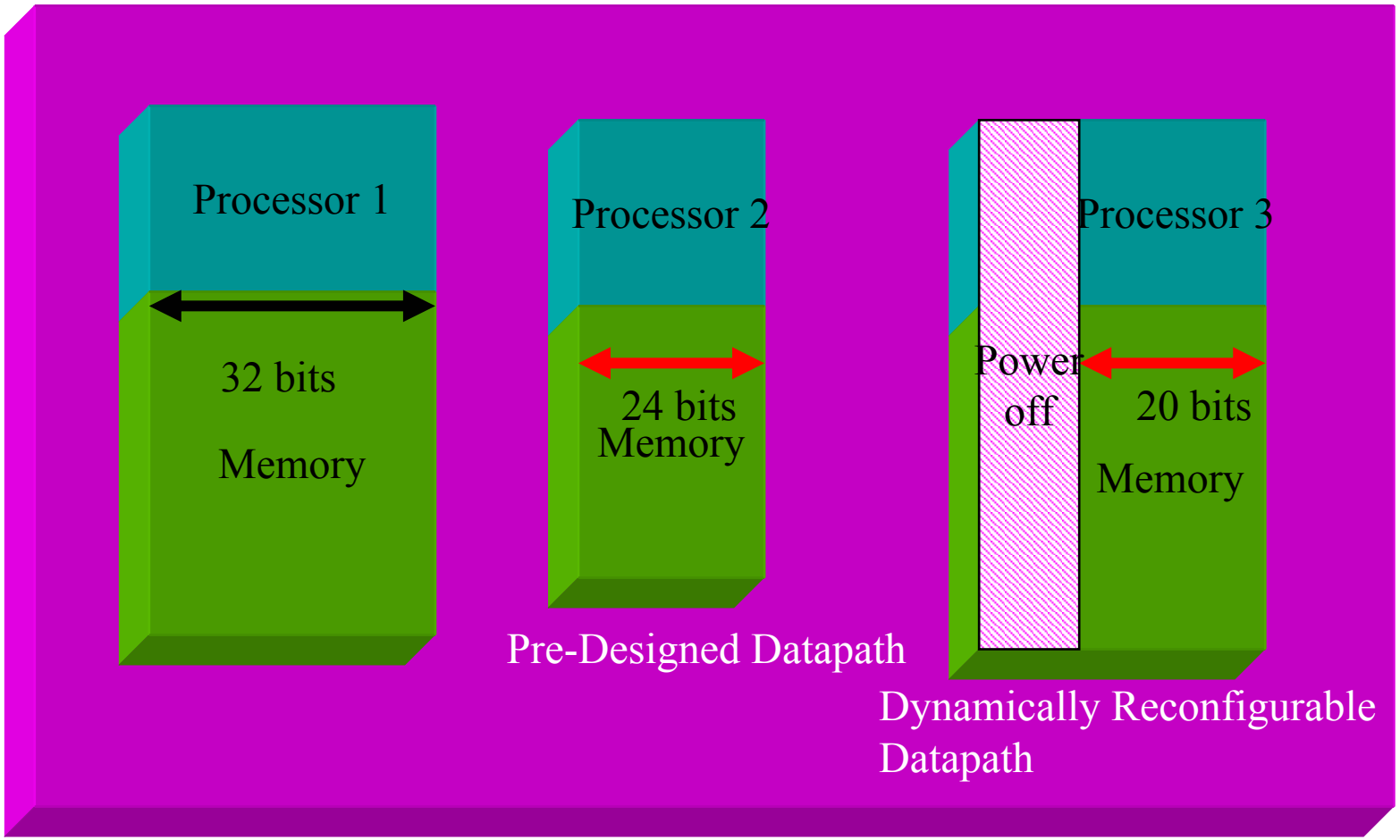
Bit Width	Arrays
1 bits	9 * 4
3 bits	10 * 1 20 * 1
4 bits	4 * 1
9 bits	9 * 1

$$\frac{\text{Actually Used Bits in Variables}}{406 \times 32 \text{ bits}} * 100 = \underline{35\%}$$

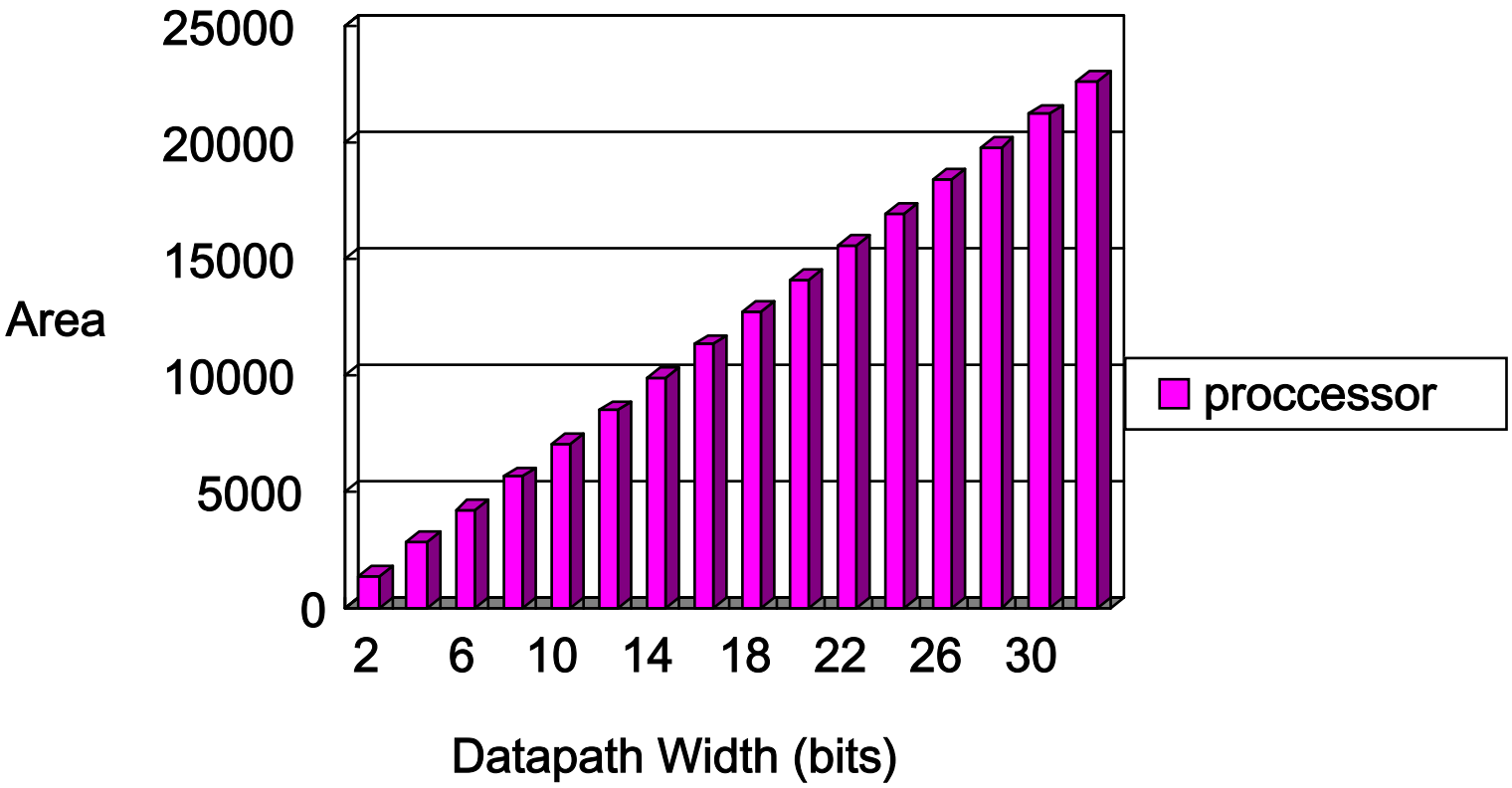
Datapath Width Optimization

- For a given set of application programs, find datapath width which minimizes overheads of area, energy, and performance.
- Assumptions:
  - » Keep functionality and accuracy of computation.
  - » Redesign of processors and memories
    - – We use customizable soft-core processors in which datapath width can be redesigned.
    - – Bit width of data buses, registers, operation units, and memory words are customizable.

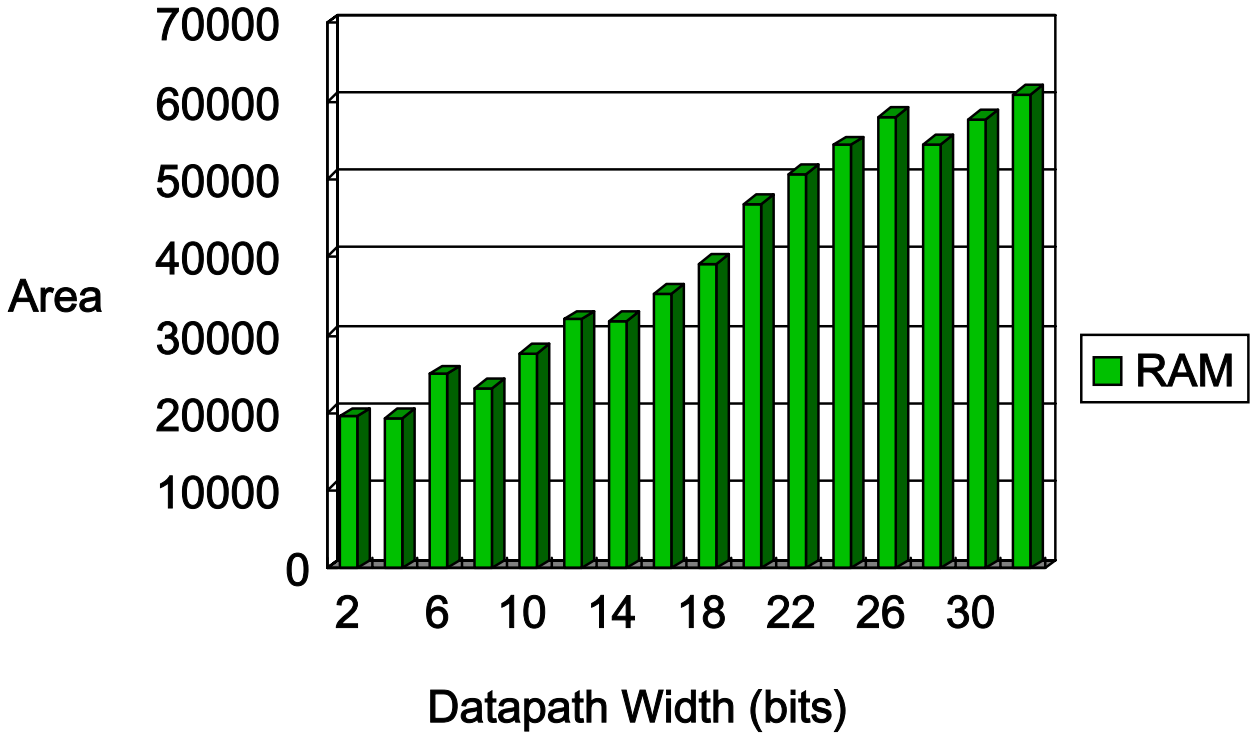
Application to MPSOC Design



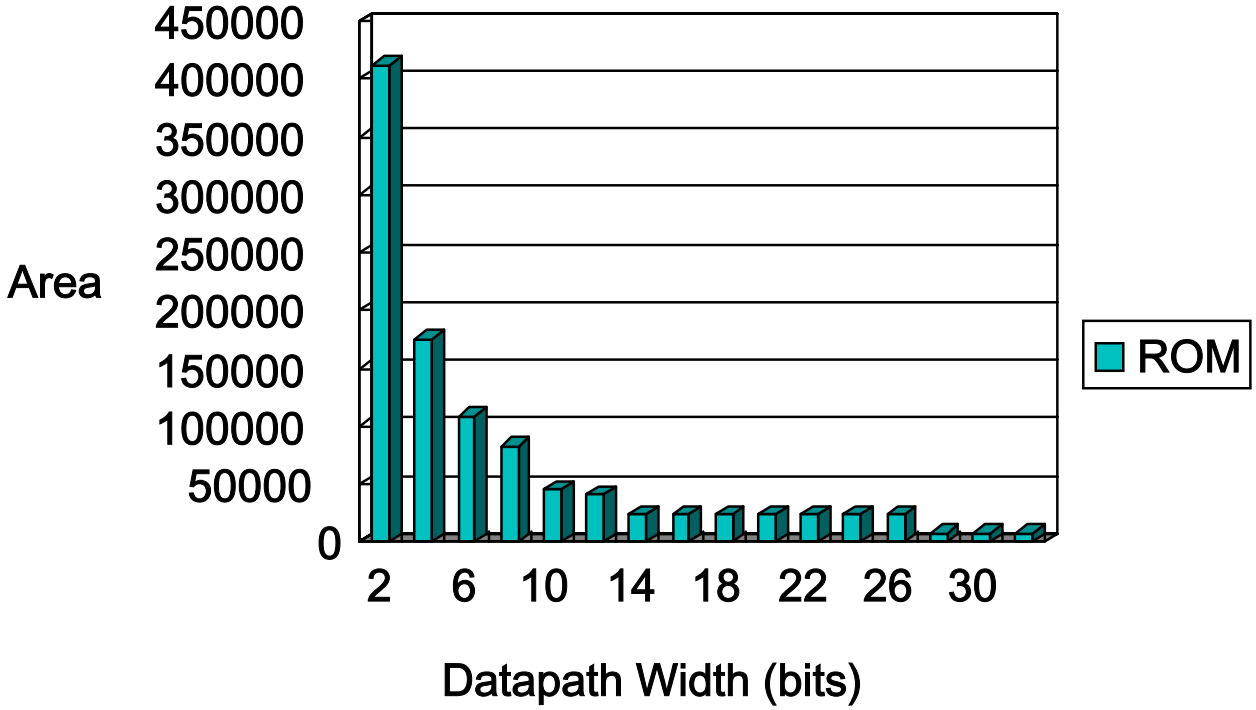
Processor Area and Datapath Width



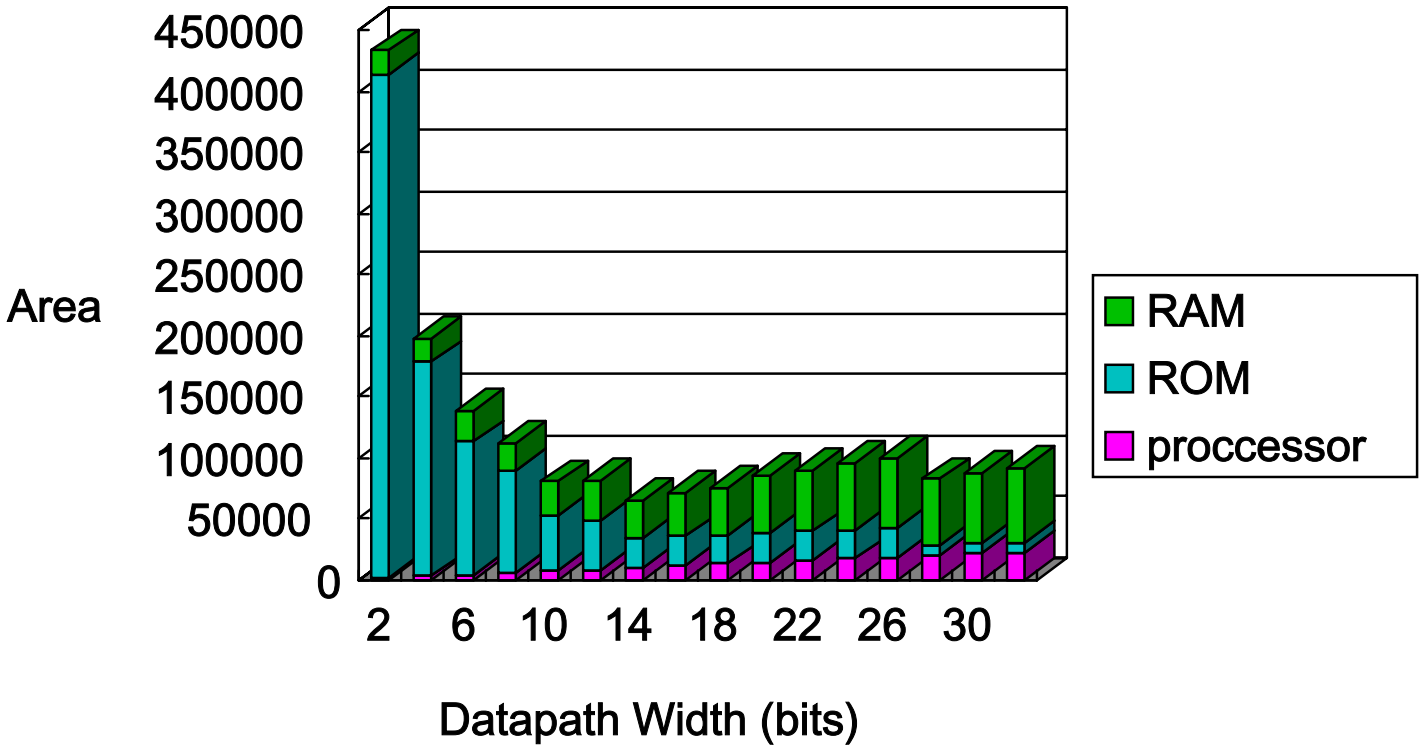
Data RAM Area and Datapath Width



Program ROM Area and Datapath Width



Total System Area and Data Path Width



Issues on Datapath Width Optimization

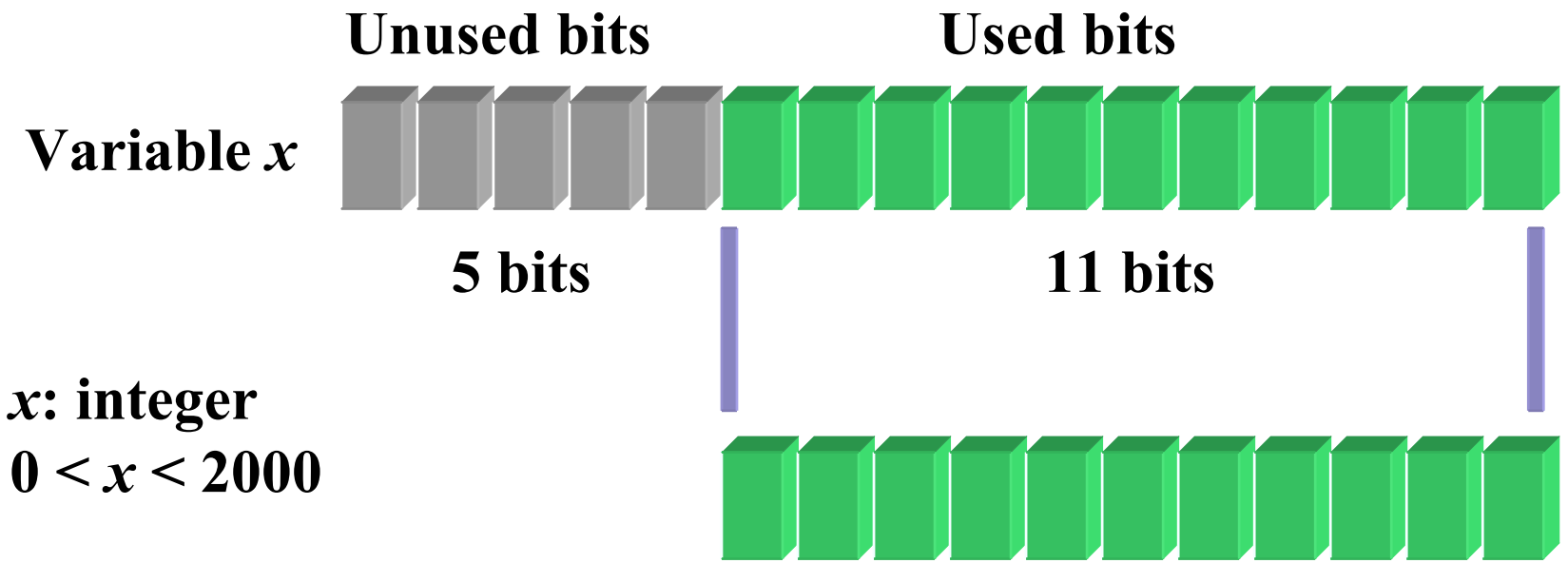
- Analysis of Programs
  - » Effective Bit Analysis
- Soft-Core Processor
  - » Customizable Datapath
  - » Programming Language and Compiler
- Design Flow
- Optimization of Memory Architecture

Datapath Width Optimization for Customizable Processors

- Problem Definition
- **Basic Techniques**
  - » **Effective Bit Analysis (Variable Size Analysis)**
  - » Customizable Processor
  - » Programming Language and Compiler
- Optimization Flow
- Memory Architecture
- Quality Driven Design

Effective Bit Width of a Variable

The number of bits actually used for a variable in computation.



The effective bit width of *x* is 11.

Static Analysis

Using symbolic simulation and formal verification techniques, the effective bit width of each variable can be calculated from the range of input and /or output variables.

- Assumptions
  - Programs have no recursion.
  - The range of each input and/ or output data is known.
  - Programs are well-structured.

Static Analysis Method

Static analysis method

1: Analyze the range of each input variable

2: Calculate the bit width of input variables from the range of its value with following the equations.

- x : unsigned integer

$$e(x) = \lceil \log_2 (x_{\max} + 1) \rceil$$

- x : signed integer

$$e(x) = \lceil \log_2 \eta \rceil + 1$$

where

$$\eta = \max(|x_{\max}| + 1, \ |x_{\min}|)$$

e(x) : Effective bit width of x
x_{max} : The largest value of x
x_{min} : The smallest value of x

Static Analysis (1)

```
inc func(int x, int y)
{
  int a, b;

  a = x + y;

  if (a > 0)   b = x * 2;
       else   b = y * 3;

  a = x * func2(b);

  while( a < 10 ) {
    a = a - y;
          ⋮
  }

  return( a );
}
```

The range of input parameters is known.

Static Analysis (2)

```
inc func(int x, int y)
{
  int a, b;

  a = x + y;

  if (a > 0)   b = x
       else   b = y * 3;

  a = x * fun

  while( a <
    a = a - y;
          ⋮

  }

  return( a );
}
```

For an assignment, the range of a variable in the left side is calculated from range of variables, constants and operators in the right side.

Static Analysis (3)

```
inc func(int x, int y)
{
  int a, b;

  a = x + y;

  if (a > 0)   b = x * 2;
       else   b = y * 3;

  a = x * func2(b);

  while ( a
    a = a -

  }

  return( a
}
```

For a conditional statement, the *then* and the *else* parts are analyzed separately.
With merging these obtained ranges, we can obtain the range of variables.

Static Analysis (4)

```
inc func(int x, int y)
{
  int a, b;

  a = x + y;

  if (a > 0)   b = x * 2;
       else   b = y * 3;

  a = x * func2(b);

  while ( a < 10 ) {
    a = a - y;
        ⋮
  }

  return( a );
}
```

We can analyze the function call with the range of its parameters. An assignment statement which has an function call statement is analyzed with the result of the function call analysis.

Static Analysis (5)

```
inc func(int x, int y)
{
  int a, b;

  a = x + y;

  if (a > 0)   b
       else   b

  a = x * func2(3);

  while (a < 10) {
    a = a - y;
        ⋮

  }

  return( a );
}
```

Bounded loop:

We can analyze a range of each variable with expanding the loop into a straight-line program.

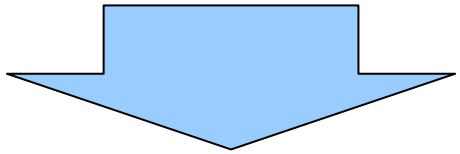
Unbounded loop:

We must analyze this statement with a dynamic analysis.

Dynamic Analysis

Statements which are difficult to be analyzed by static method are

- unbounded loops.
- pointers.



Simulation base approach

- Execute the program with typical input data and monitor the values assigned to each variable.
- Calculate the bit width of each variables from their range.

A Result of Analysis of MPEG-2 Video Recoder

Size of Program : 6275 lines (written in C)

Variable size analysis result :

Bit Width	# of variables
1 bits	50
2 bits	17
3 bits	11
4 bits	11
5 bits	10
6 bits	14
7 bits	16
8 bits	9
9 bits	7
10 bits	3
11 bits	6
12 bits	17
13 bits	0
14 bits	46
15 bits	2
31 bits	0

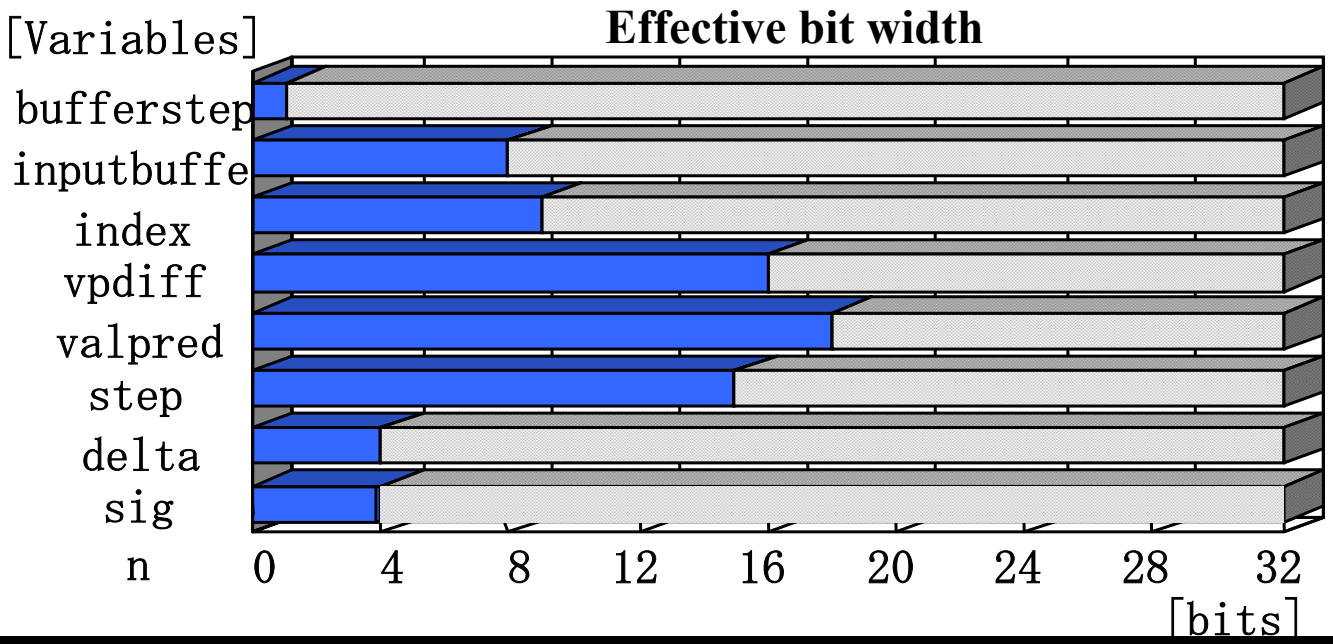
Bit Width	# of variables
17 bits	2
18 bits	3
19 bits	0
20 bits	6
21 bits	0
22 bits	0
23 bits	0
24 bits	13
25 bits	0
26 bits	2
27 bits	4
28 bits	3
29 bits	3
30 bits	7

Bit Width	Arrays
1 bits	9 * 4
3 bits	10 * 1 20 * 1
4 bits	4 * 1
9 bits	9 * 1
11 bits	3 * 1 9 * 1
32 bits	3 * 3

x * y : x is # of elements
y is # of arrays

Application of Effective Bit Analysis

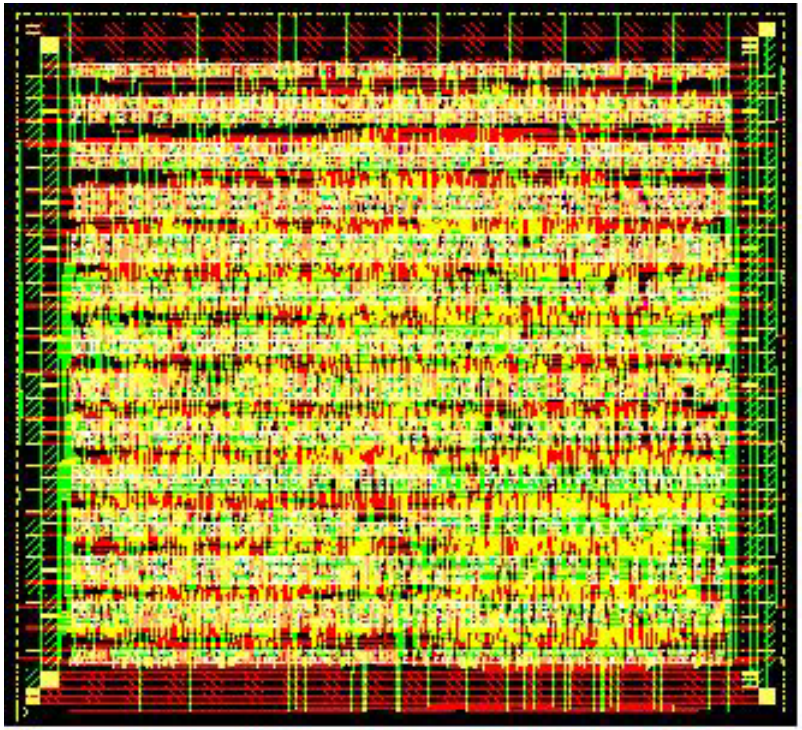
ADPCM Decoder



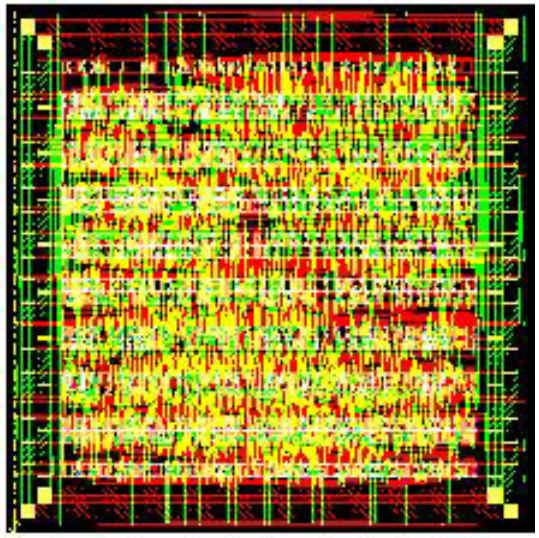
	ADPCM32	ADPCM18	Reduction
Area	1220.8×1196.0 [μ･]	865.2×865.2 [μ･]	49%
# of Cells	1379	669	52%
# of Transistors	13006	5864	55%
Energy Consumption	367 [nJ]	239 [nJ]	35%

Process technology : NEL 0.5μm 2M1P

Implementations of ASICs



ADPCM32



ADPCM18

Datapath Width Optimization for Customizable Processors

- Problem Definition
- Basic Techniques
  - » Effective Bit Analysis
  - » **Customizable Processor**
  - » Programming Language and Compiler
- Optimization Flow
- Memory Architecture
- Quality Driven Design

Customizable Core Processors

- Parameterized core processors
  - » Synthesizable HDL description
  - » Logic/Layout tools
  - » Tensilica, Arc etc.
- Software development environment
  - » Compiler (Retargetable compiler)
  - » Operating systems and debugger
- HW / SW codesign environment
  - » Co-simulation and estimation tools
  - » Optimization methods

Soft-core Processor

- A Customizable core processor
  - » Design parameters
    - – **the datapath width**
    - – the number of general registers
    - – Instruction set
    - – Data/instruction memory space
- Logic and layout synthesis tools
- Programming language and retargetable compiler

A Soft-Core Processor: Bung DLX

- 32-bit DLX RISC Architecture
  - Non pipelined Harvard Architecture
  - 32 general registers
  - 72 instructions
  - **the datapath width 32 bits**
  - the instruction length 32 bits
  - VHDL Description 7,000 lines
  - Synthesized circuit 23,282 gates

Customization of Bung DLX

- Design Modification Table
  - **The datapath width** **(32 bits)**
  - The data memory space (2^{32} words)
  - The instruction length (32 bits)
  - The instruction memory space (2^{32} words)
  - The number of general registers (32)
  - The number of instructions (72)
- Automatic synthesis from the modification table

Datapath Width Optimization for Customizable Processors

- Problem Definition
- Basic Techniques
  - » Effective Bit Analysis
  - » Customizable Processor
  - » **Programming Language and Compiler**
- Optimization Flow
- Memory Architecture
- Quality Driven Design

Valen-C and a Retargetable Compiler

- Valen-C
  - » Programmers can specify the effective bit width for each variable.

    int20 x, y, z
  - » The semantics of the program can be independent from processor architecture.
- Retargetable compiler
  - » Processor Definition + Valen-C Program

    ⇒ Assembly code for the processor

Compilation from Valen-C Code

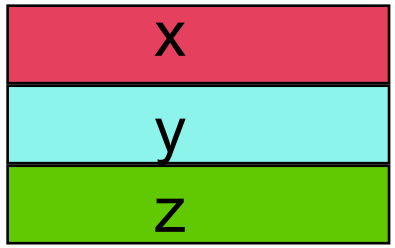
Valen-C code

Int20 x, y, z;

…

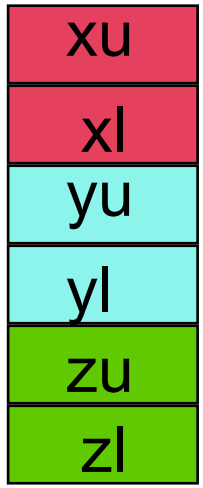
z = x + y;

20-bit Processor



add x y z

10-bit Processor



add xl yl zl
addc xu yu zu

Relation between Datapath Width and Program Size

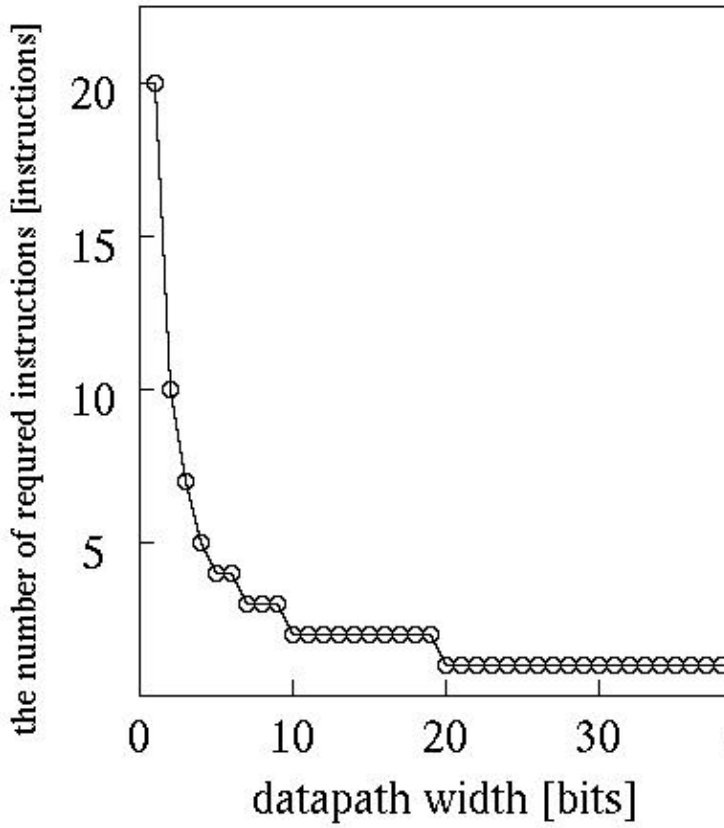
```
nt20  x, y, z;
z = x + y;
```

datapath width = 20 bits

```
add  z, x, y
```

datapath width =10 bits

```
add  z_low, x_low, y_low
addc  z_high, x_high, y_high
```

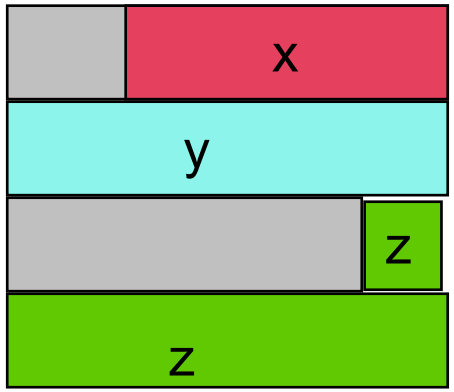


Datapath Width and Data Memory

Valen-C Program

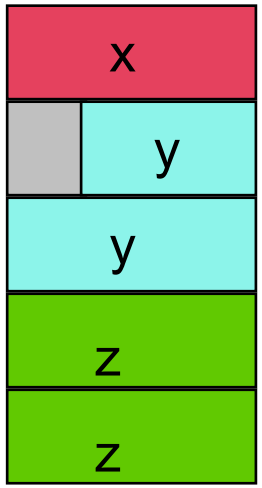
```
int12 x;
int20 y;
int24 z;
```

20-bit processor



unused: 24 bits
total: 80 bits

12-bit processor



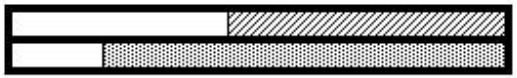
unused: 4 bits
total: 60 bits

unused bits

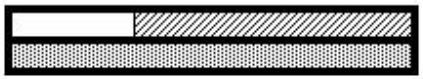
Relation between Datapath Width and Size of Data Memory

```
main()
{
   int18 x;
   int26 y;
   ........
}
```

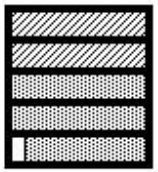
32 bits x 2 words = 64 bits

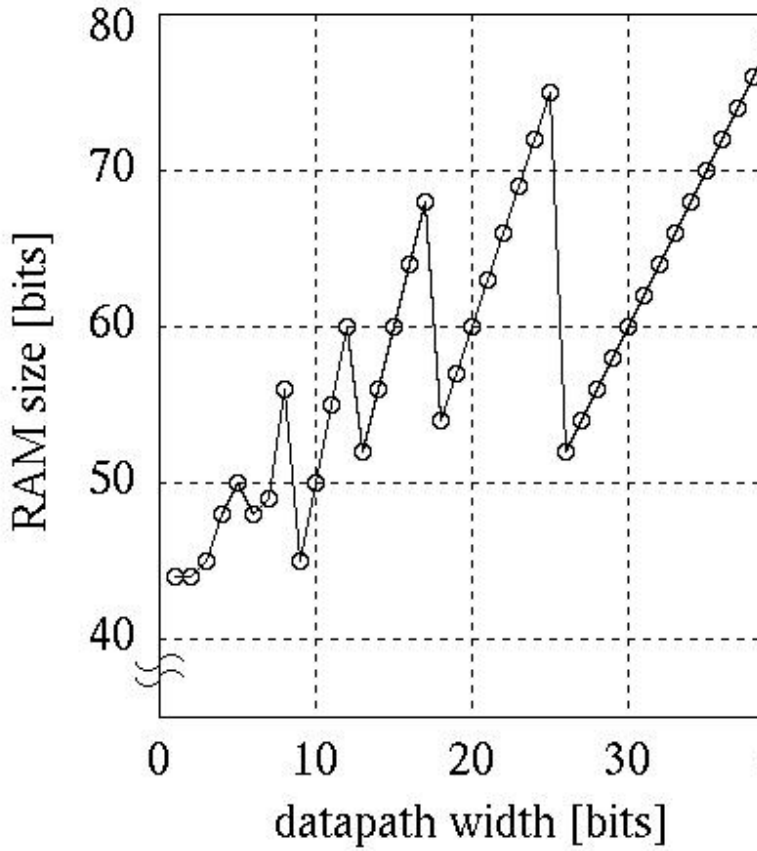


26 bits x 2 words = 52 bits

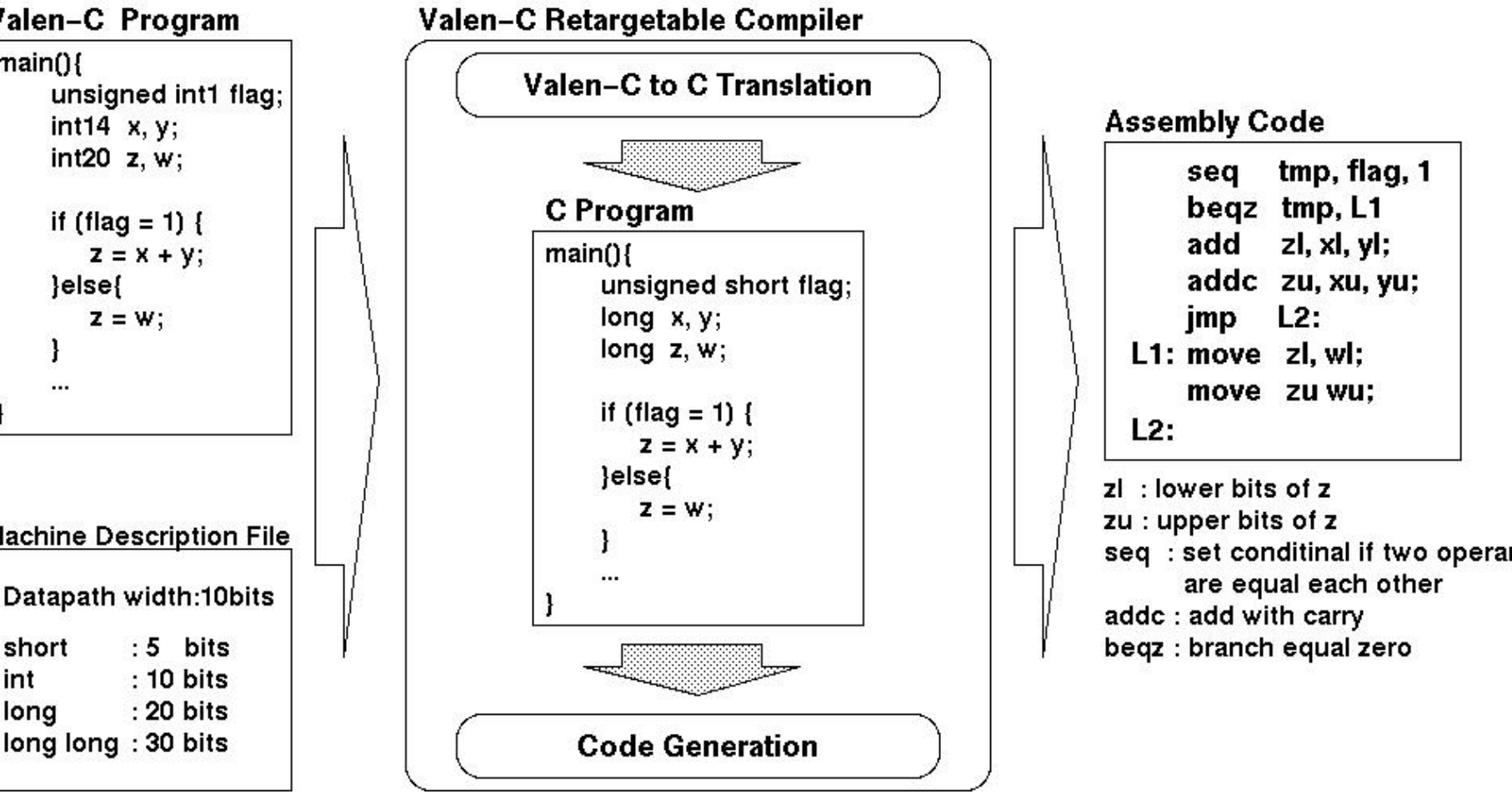


9 bits x 5 words = 45 bits





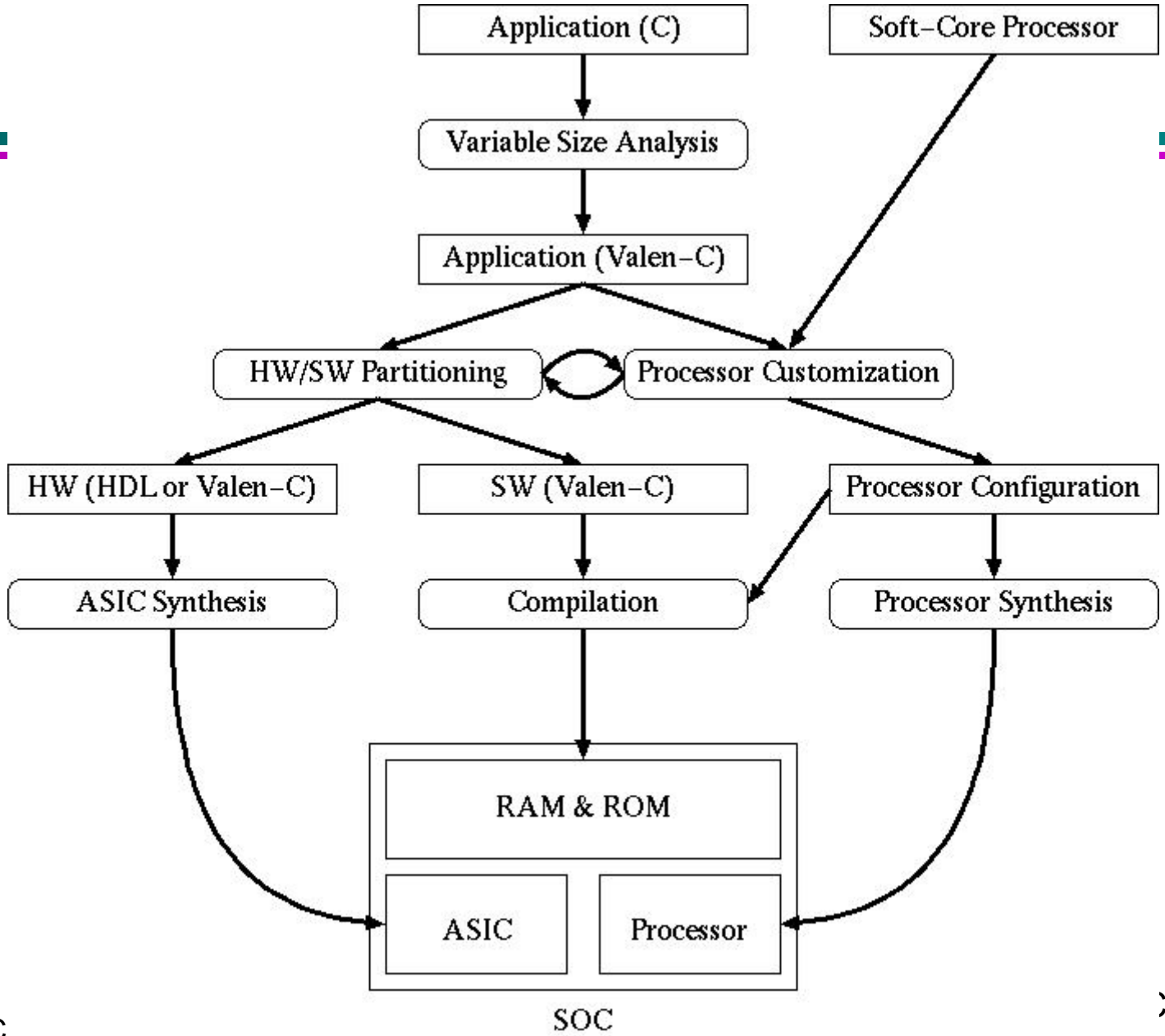
Compilation of a Valen-C Program



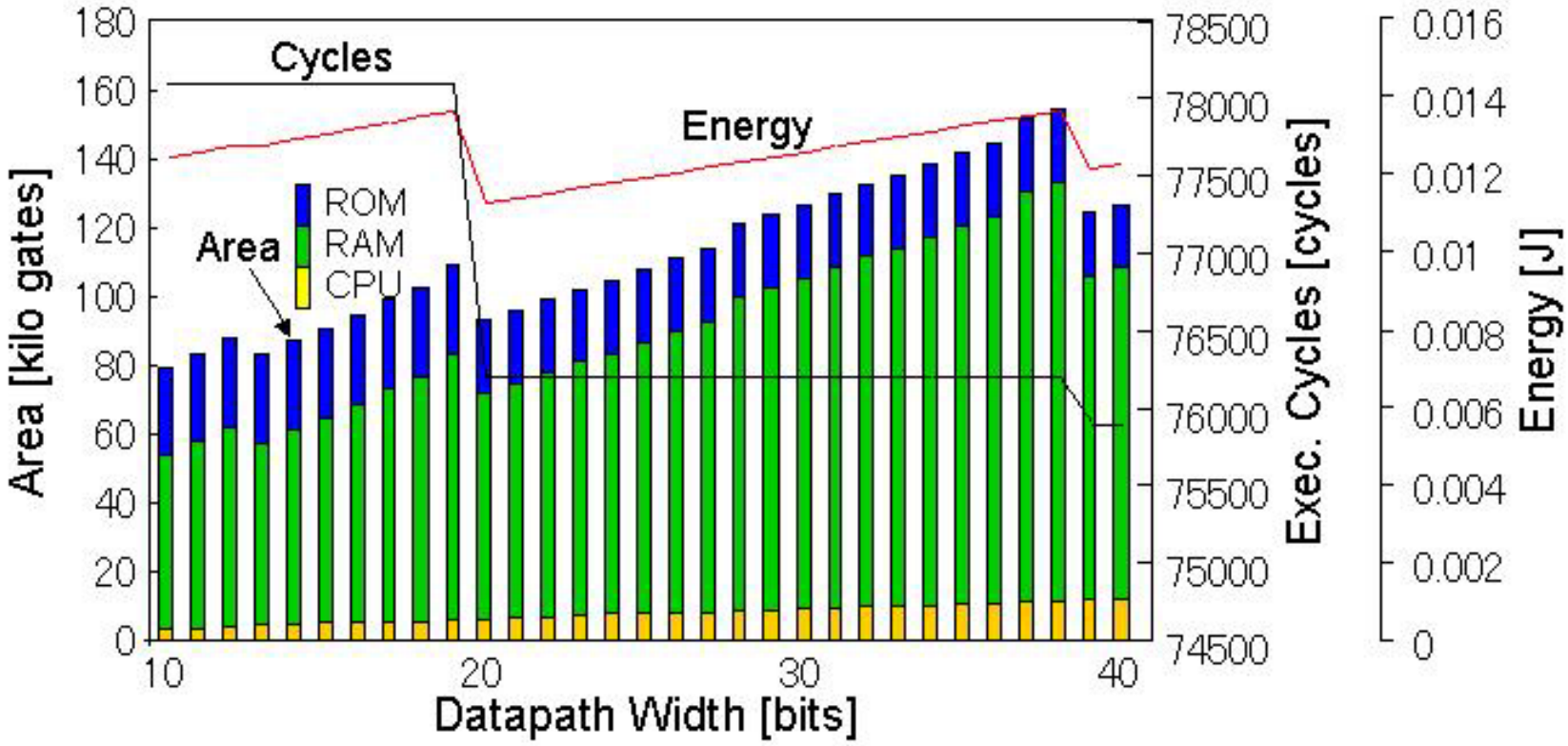
Datapath Width Optimization for Customizable Processors

- Problem Definition
- Basic Techniques
  - » Effective Bit Analysis
  - » Customizable Processor
  - » Programming Language and Compiler
- **Optimization Flow**
- Memory Architecture
- Quality Driven Design

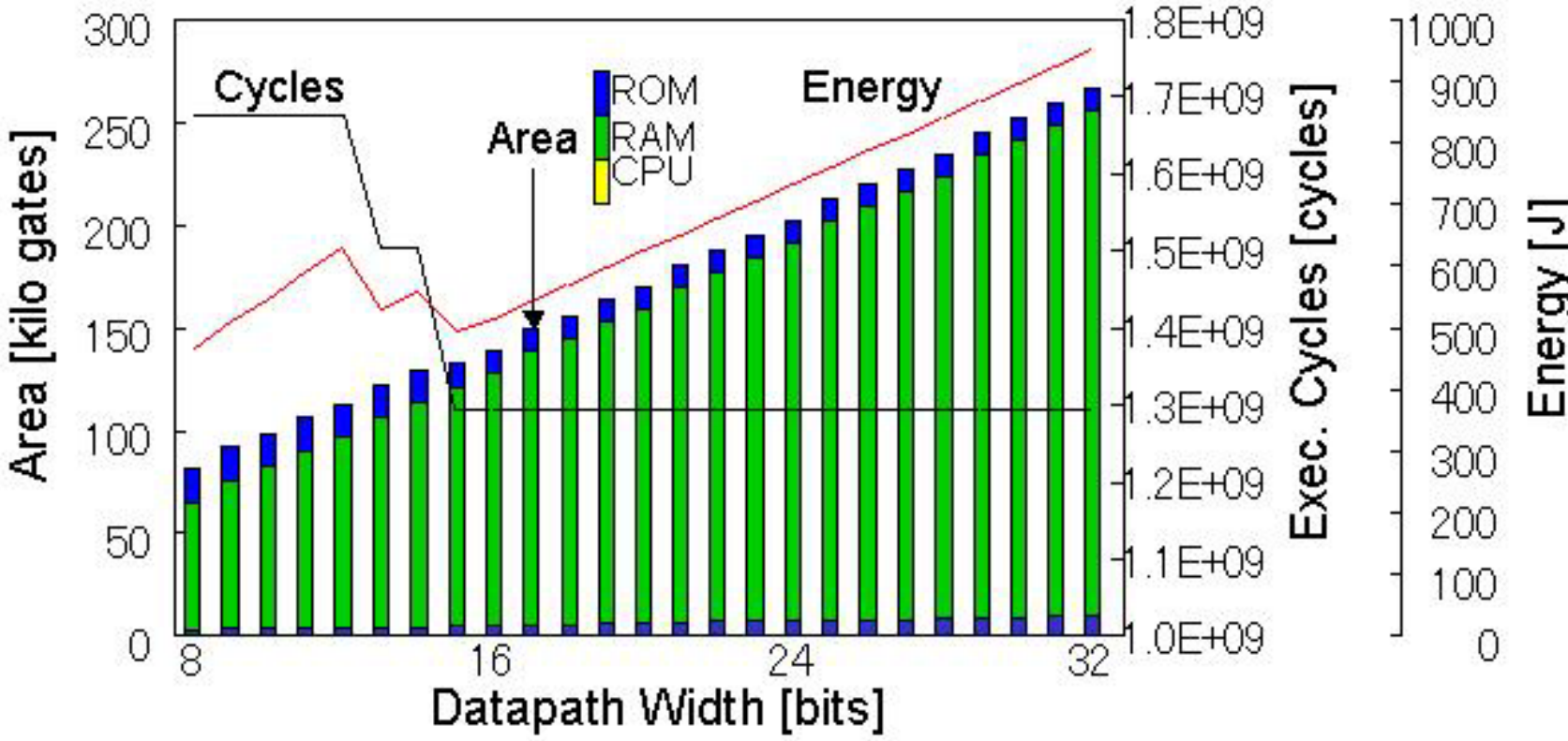
Design Flow


Application (C)
Soft-Core Processor
Variable Size Analysis
Application (Valen-C)
HW/SW Partitioning
Processor Customization
HW (HDL or Valen-C)
SW (Valen-C)
Processor Configuration
ASIC Synthesis
Compilation
Processor Synthesis
RAM & ROM
ASIC
Processor
SOC

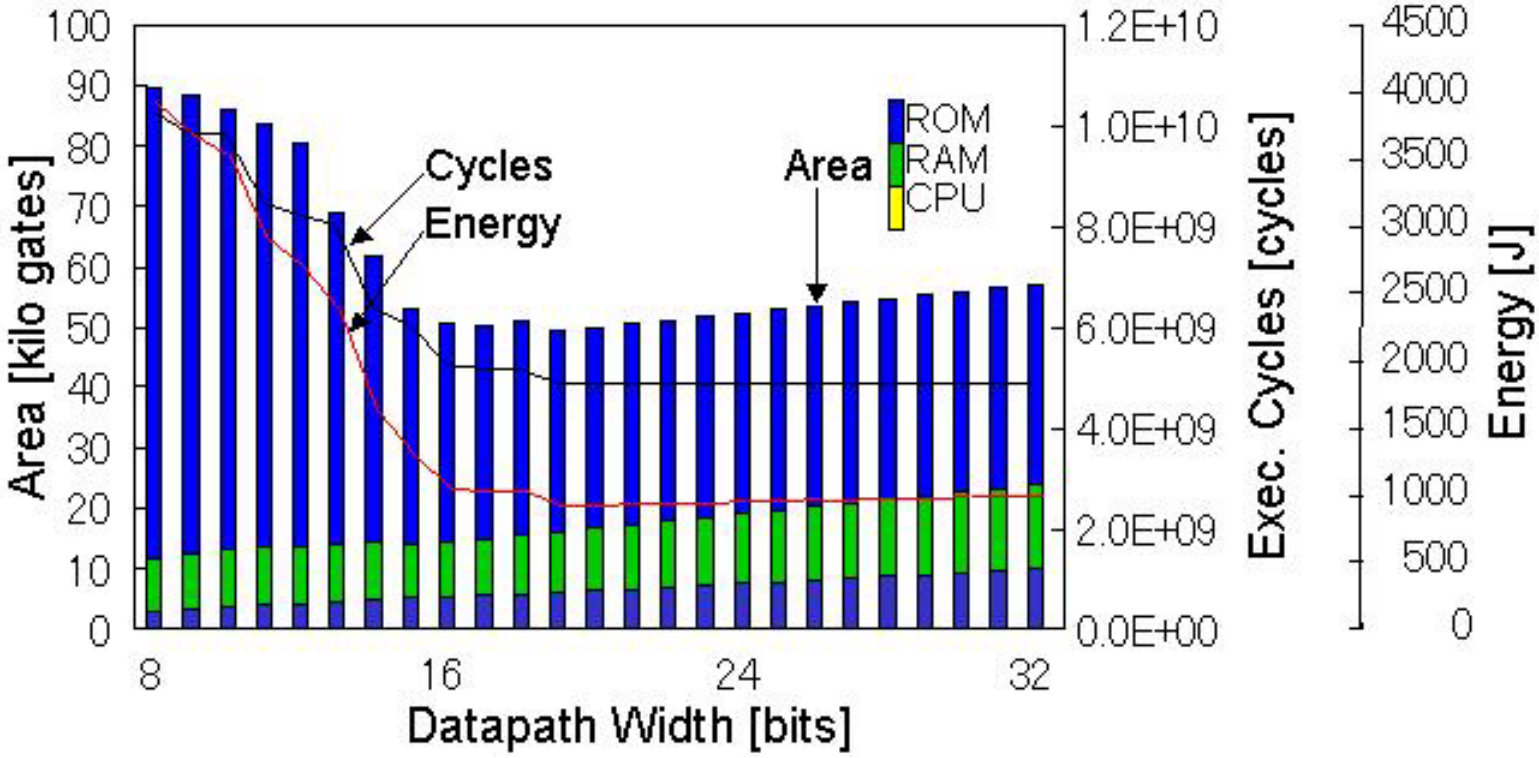
Design Example: Calculator



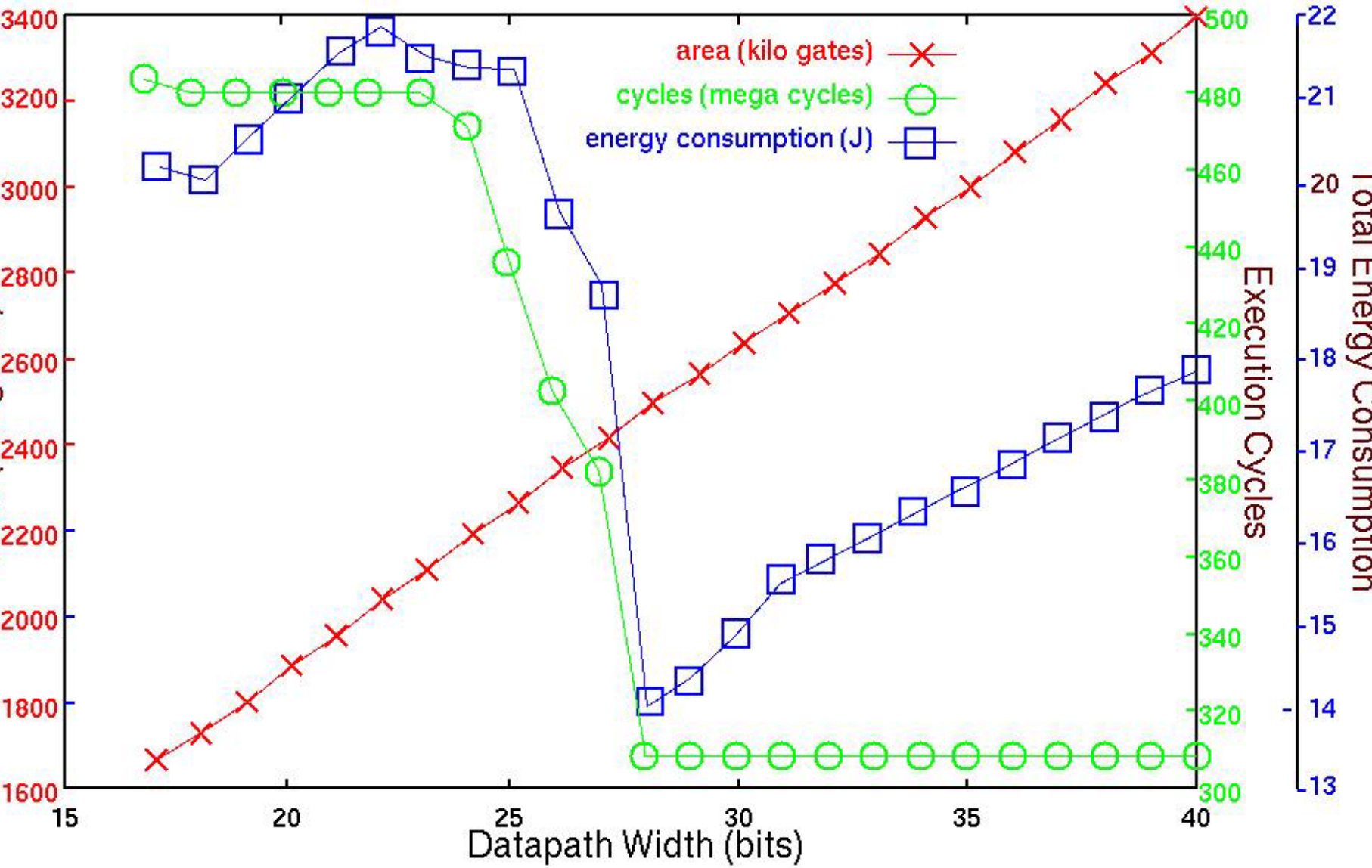
Design Example: Lempel-Zip Encoder/Decode



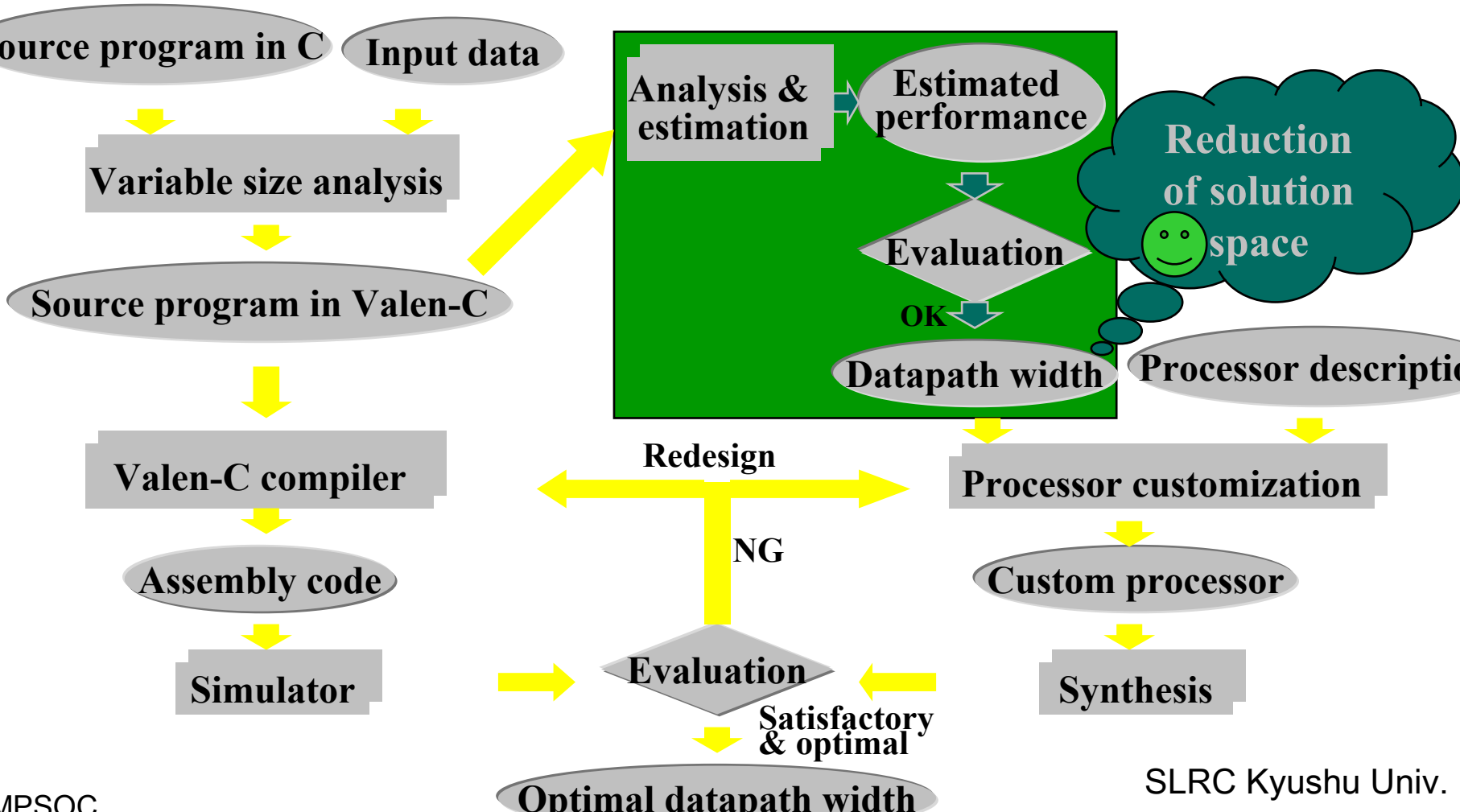
Design Example: ADPCM Decoder



Design Example:MPEG2 Video Decoder



Reduction of Exploration Space



Datapath Width and Program Size

```
int main()
{
   int16   x;
   int26   y;
   int30   z;

   z = x + y;

}
```

Sample program
(Valen-C)

32-bit processor

```
Load     x,R1;
Load     y,R2;
Add      R2,R1;
Store    R1,z;
```

21-bit processor

```
Load     x,R1;
Load     y_low,R2;
Load     y_up,R3;
Add      R1,R2;
Addc     #0,R3;
Store    R2,z_low;
Store    R3,z_up;
```

Estimation of # of Execution Cycles

Analysis result

Operand	Width
x	16
y	26

Operator	Width
+	26
=	30

Result	Width
z	30

Simulation Results for 32-bit

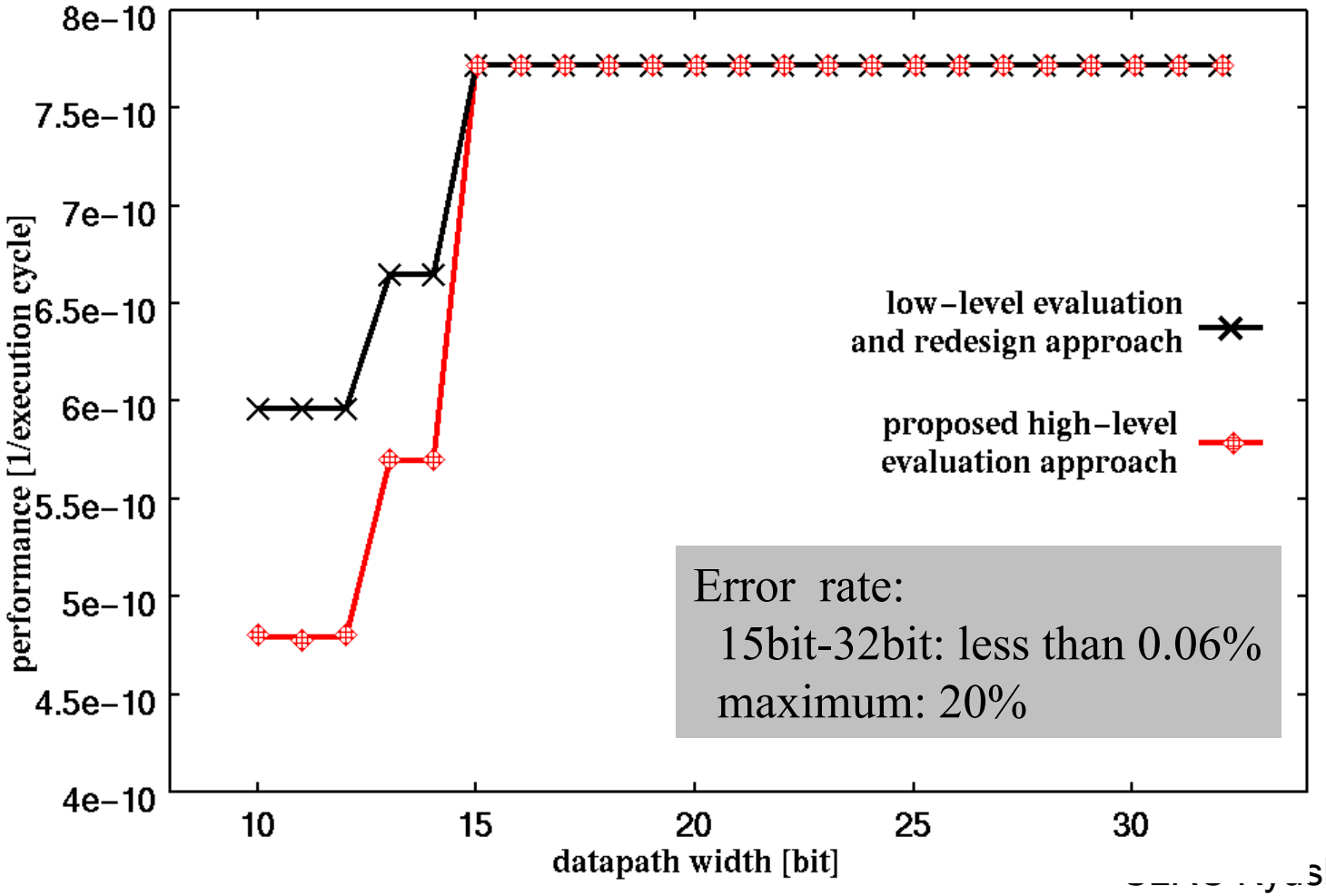
32-bit processor

x :	1 SP Load
y :	1 SP Load
+ :	1 SP Add
= :	1 SP Store
Total :	4 SP instr.

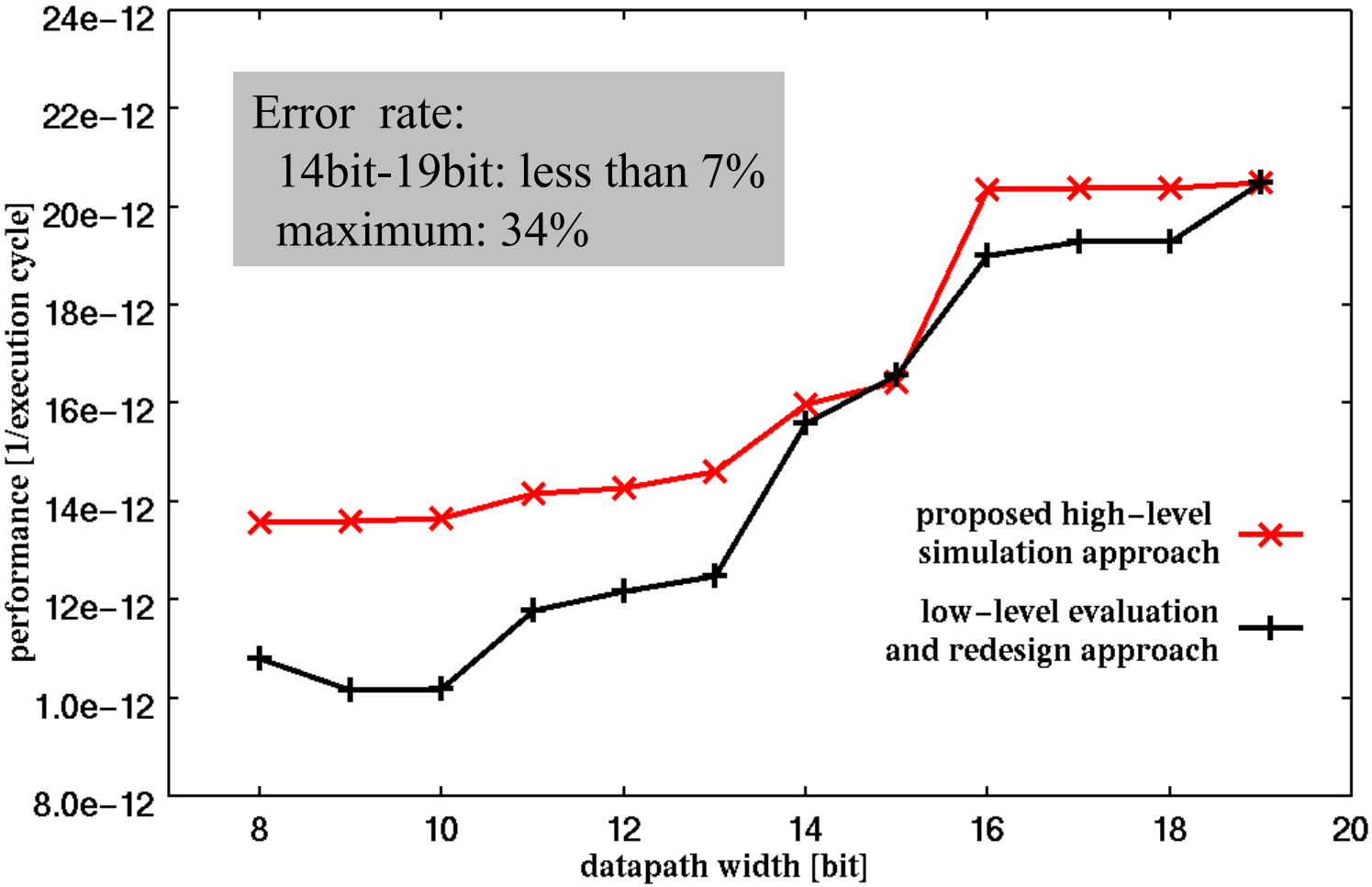
21-bit processor

x :	1 SP Load
y :	2 SP Load
+ :	2 SP Add
= :	2 SP Store
Total:	7 SP instr.

Accuracy Analysis (Lempel-Ziv)



Accuracy Analysis (adpcm)



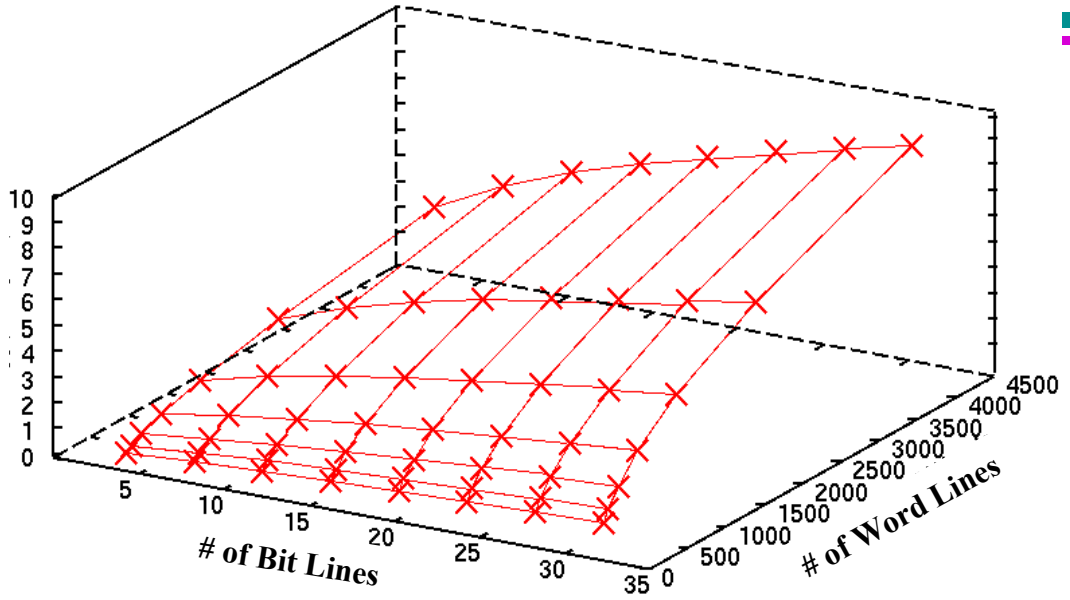
Datapath Width Optimization for Customizable Processors

- Problem Definition
- Basic Techniques
  - » Effective Bit Analysis
  - » Customizable Processor
  - » Programming Language and Compiler
- Optimization Flow
- **Memory Architecture**
- Quality Driven Design

Memory Architecture

- Word length significantly affects area and energy consumption of a system.
- Memory architecture is an important design parameter.
- Area and energy consumption of memories are often dominant in the system.

Relation between Memory Size and Energy



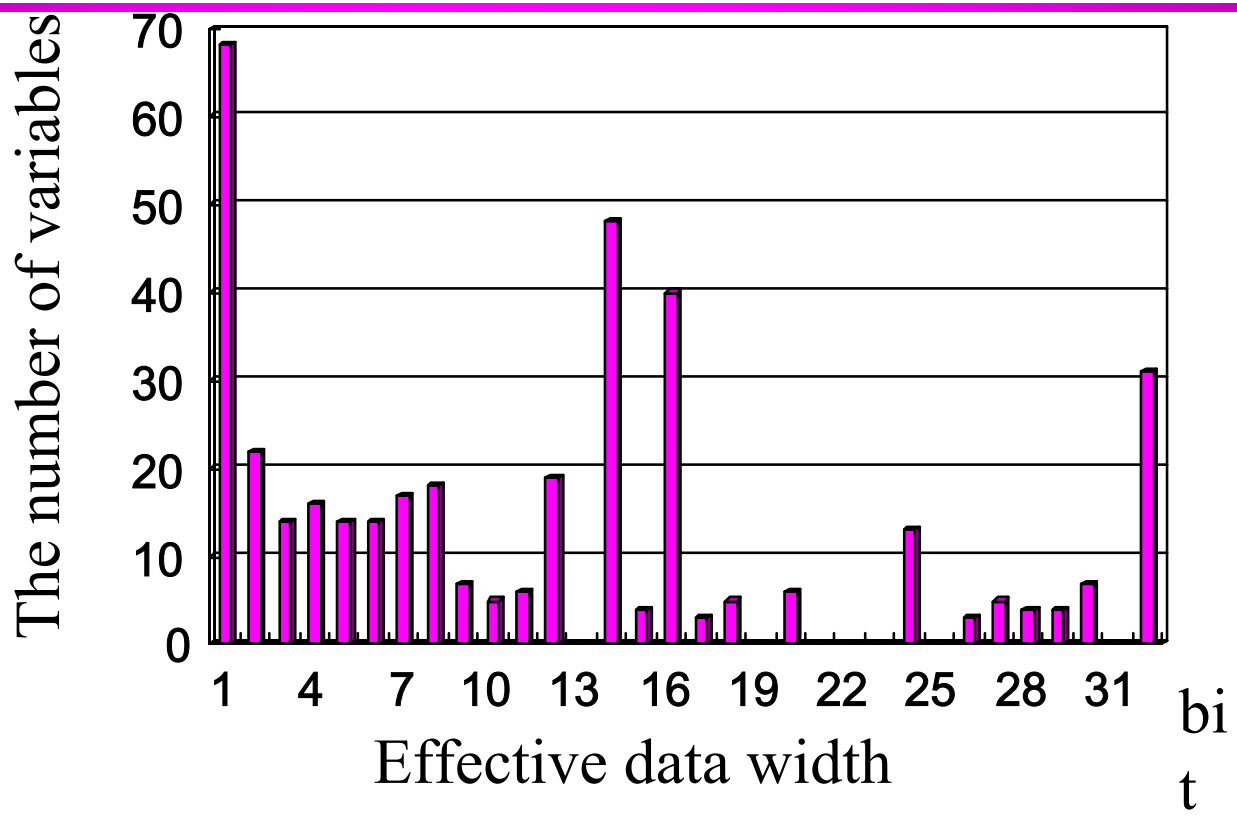
- Energy consumption of a single read access

$$e_r = 24.9 \times \sqrt{(\#\text{of bitline}) \times (\#\text{of wordline})} + 56[pJ/cycle]$$

- Energy consumption of a single write access

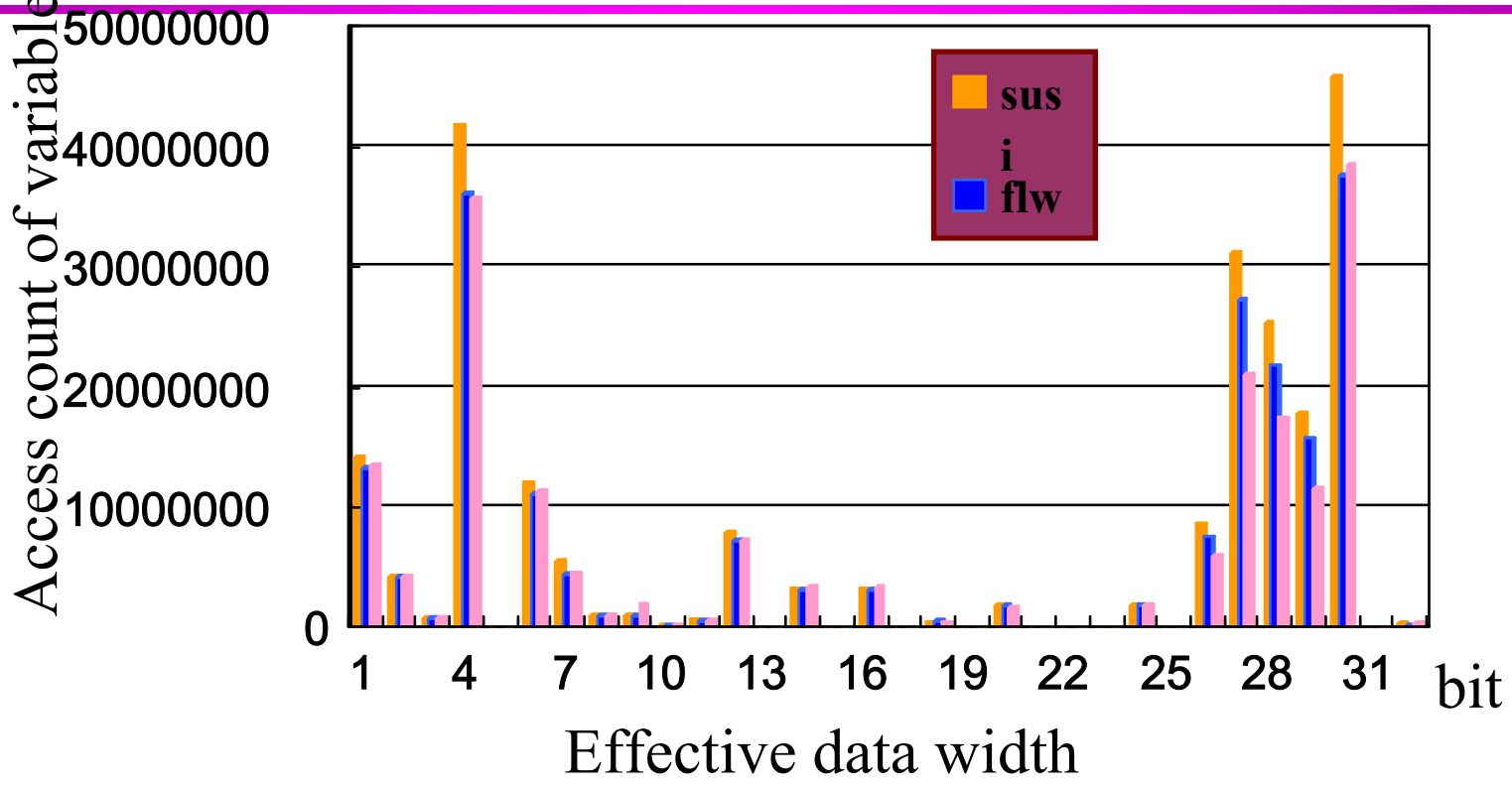
$$e_w = 197 \times \sqrt{(\#\text{of bitline}) \times (\#\text{of wordline})} + 369[pJ/cycle]$$

Distribution of Effective Bit Width



MPEG-2 video decoder

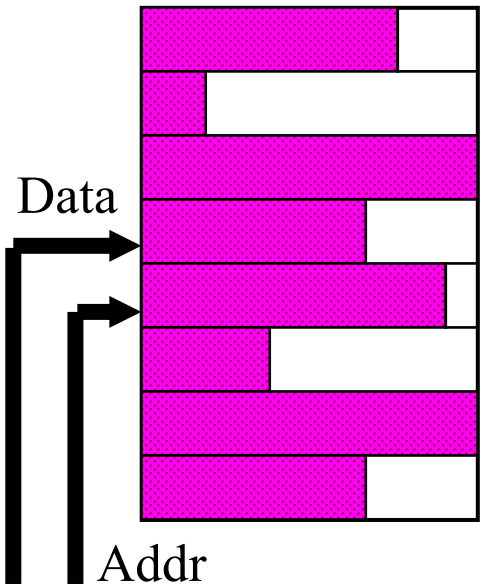
Distribution of Variable Accesses



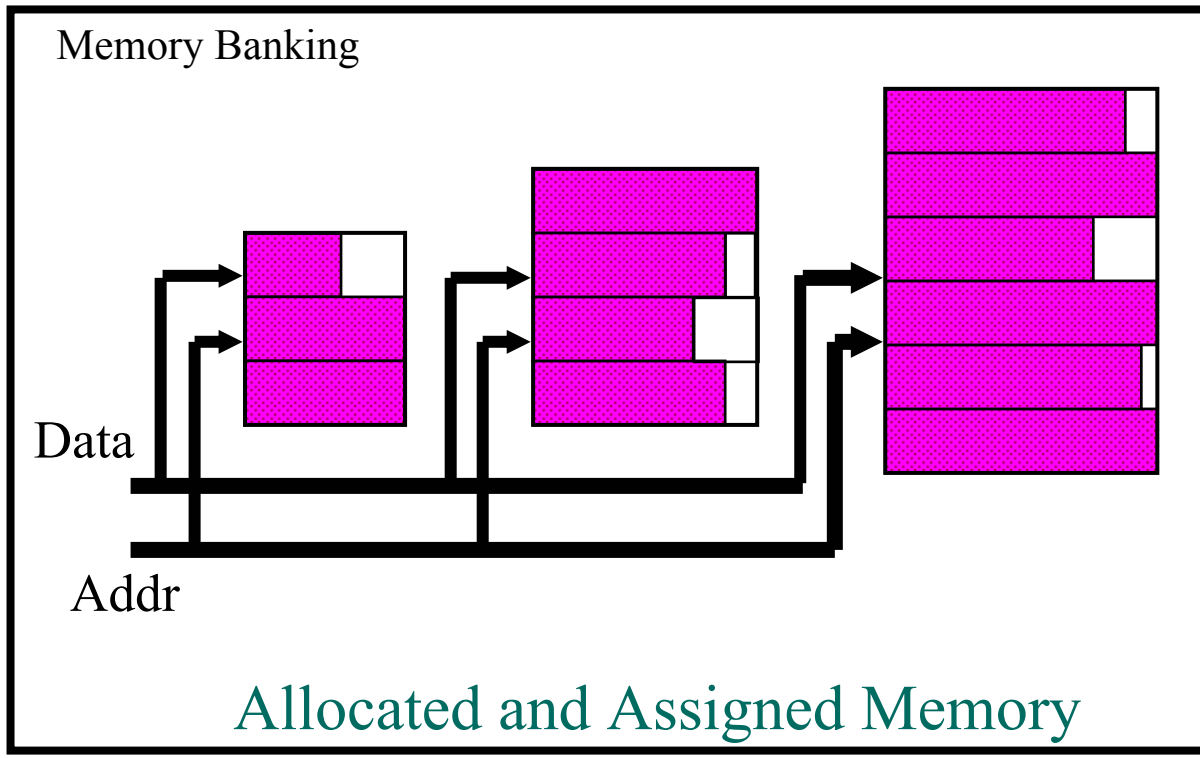
MPEG-2 video decoder

Memory Banking

- **Allocate variables with higher access ratio into a small memory.**

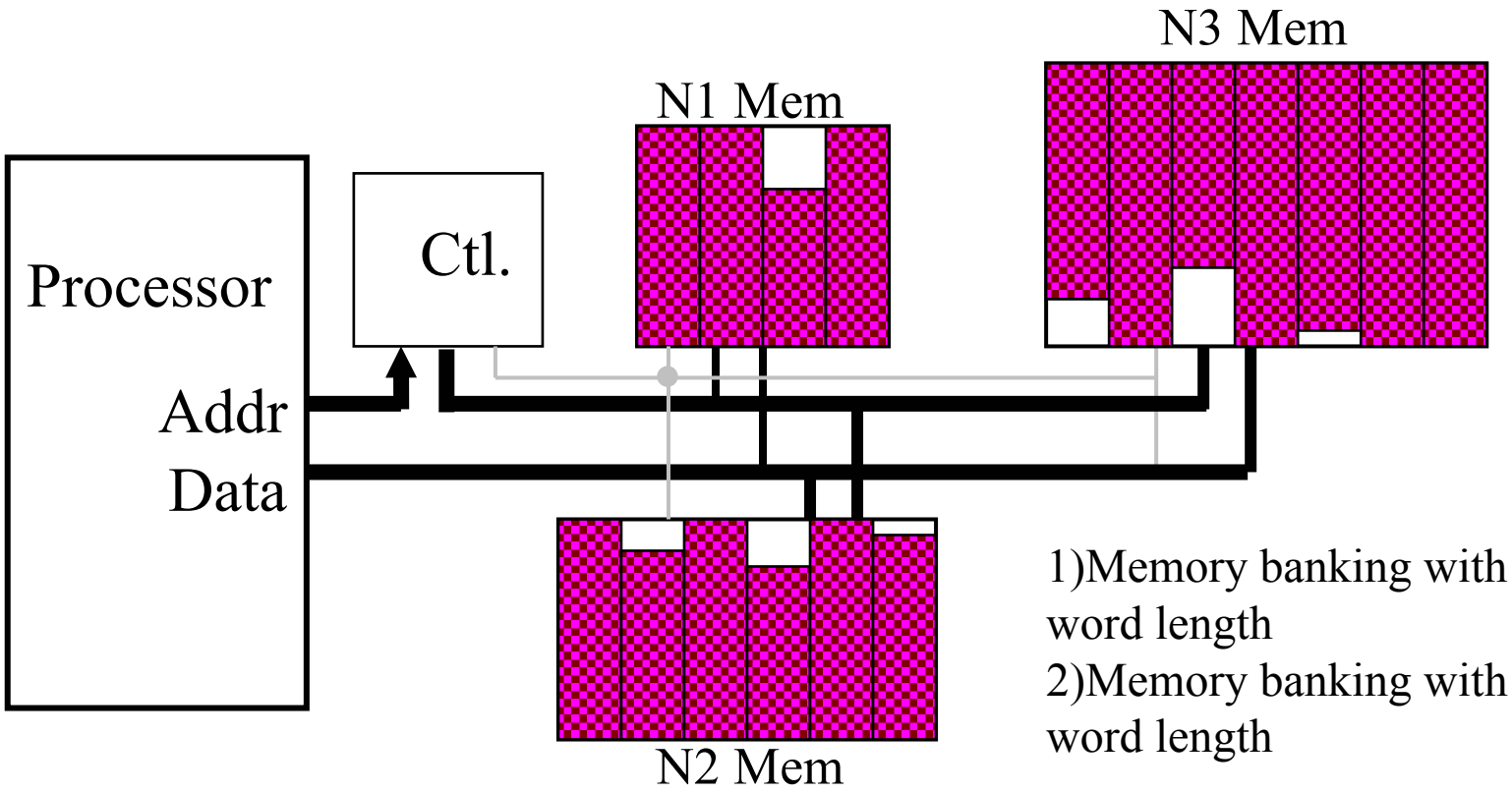


Monolithic Memory



Allocated and Assigned Memory

Experiments



Assumption:We can use arbitrary size of memories.

Experimental Results

Applications	Energy (J)	Memory banking (Uniform Length)			Optimized Memory Banking		
		Configuration	TE (J)	Sav.	Configuration	TE (J)	Sav.
Calculator	1.27 mJ	85 rows 154rows 533rows	0.87 mJ	-31.5%	85rows X 8b 154rows X 32b 533rows X 32b	0.76 mJ	-40.2%
Lempel-Ziv	1.37	830rows 3rows 1663rows	0.89	-35.0%	830rows X 13b 3rows X 15b 1663rows X 15b	0.69	-49.6%
ADPCM	1.63	20rows 16rows 86rows	1.10	-32.5%	20rows X 10b 16rows X 14b 86rows X 19b	0.80	-50.9%
MPEG2AAC	1.05	30rows 2374rows 4804rows	0.39	-62.8%	30rows X 20b 2374rows X 32b 4804rows X 32b	0.37	-64.8%
MPEG2Video	145.1 kJ	26559rows 26557rows 28127rows	120.1 kJ	-17.2%	26559rows X 8b 26557rows X 30b 28127rows X 32b	105.2 kJ	-27.5%

Datapath Width Optimization for Customizable Processors

- Problem Definition
- Basic Techniques
  - » Effective Bit Analysis
  - » Customizable Processor
  - » Programming Language and Compiler
- Optimization Flow
- Memory Architecture
- **Quality Driven Design**

Multi-media Data and Output Devices

- Large amount of high-quality multi-media data stored by a standardized format (MPEG, JPEG, MP3, etc.)
- Variety of output devices as human interfaces
  - » Mobile devices, low cost devices, high-quality devices, and ultra high-quality devices
- Share a decoding algorithm but compute energy effectively.
  - » Computation with required quality
  - » Quality or accuracy is another design parameter

Quality Driven Design

Digital Contents

High resolution decoder

High quality

Super high-resolution Display

Consumer decoder

Low Cost

Home TV

Mobile device decoder

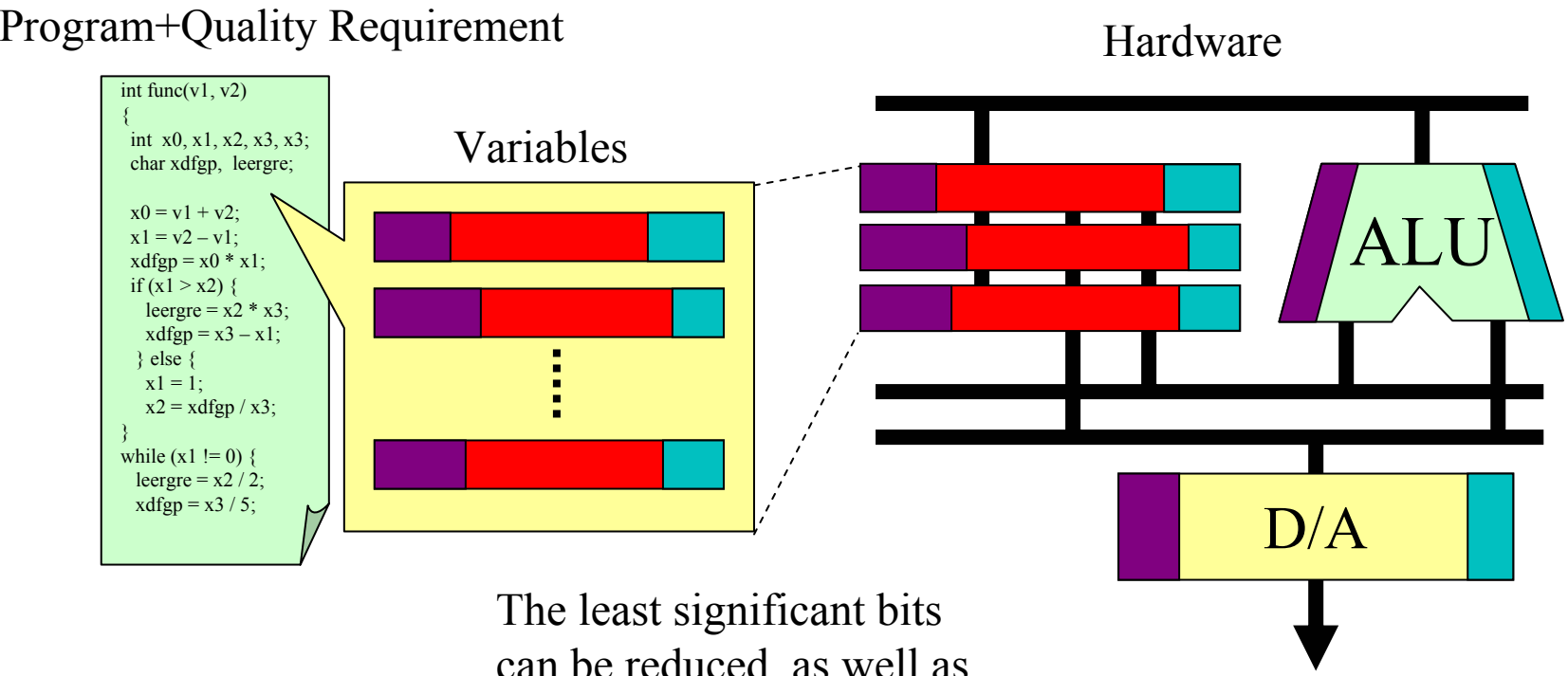
Low Energy
Low Cost

Mobile phone

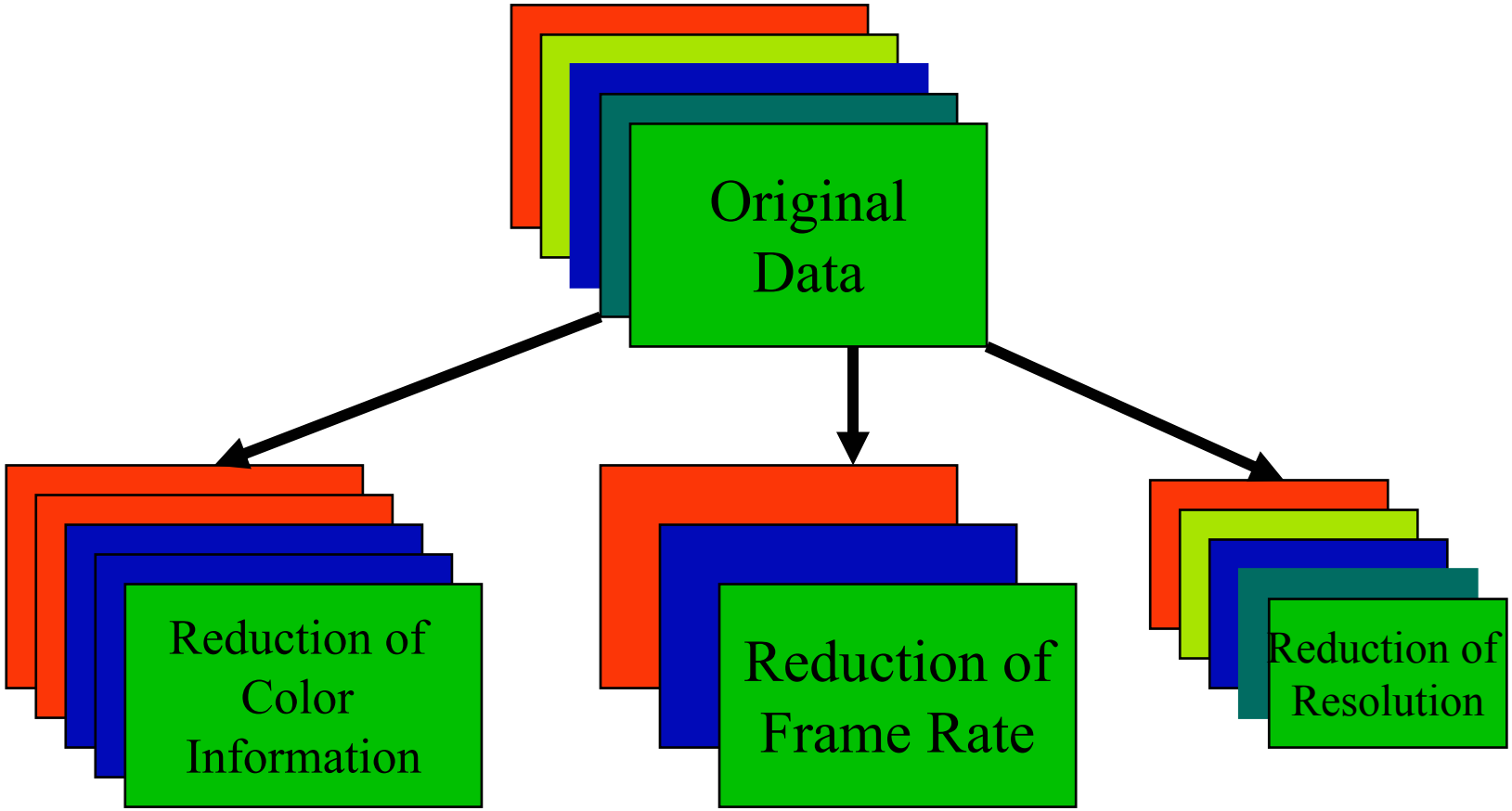
Same algorithm but different accuracy of computation

Reduction of Accuracy

Prepare the least width of datapath for the requested accuracy of the computation



Examples for Image Decoding



Reduction of Color Information in IDCT Algorithm

bit

12bit

3bit

4bit

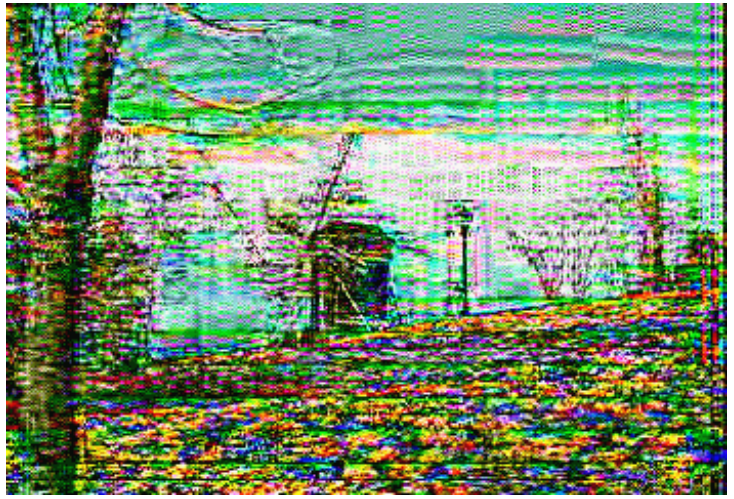
Reduction of Color Information in IDCT Algorithm

16bit

12bit

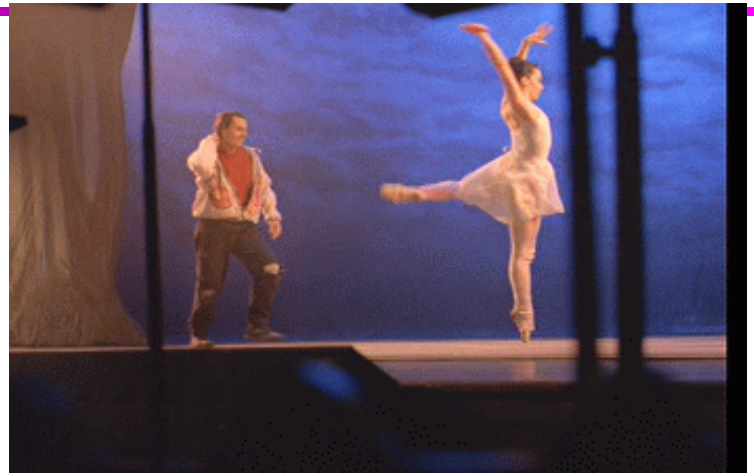
8bit

4bit

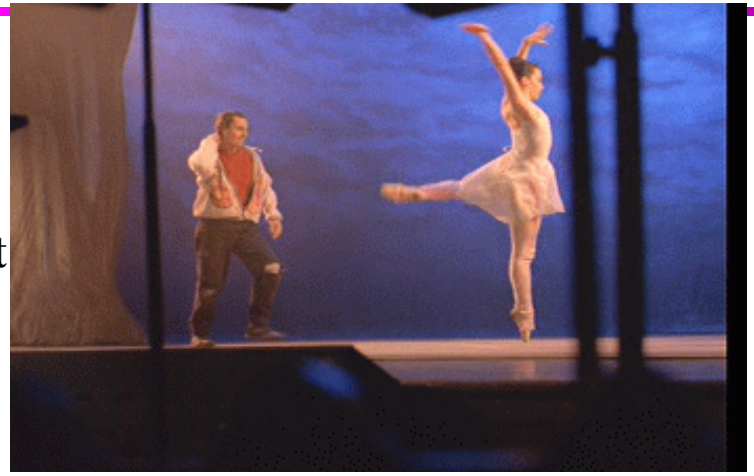


Reduction of Color Information in IDCT Algorithm

16bit

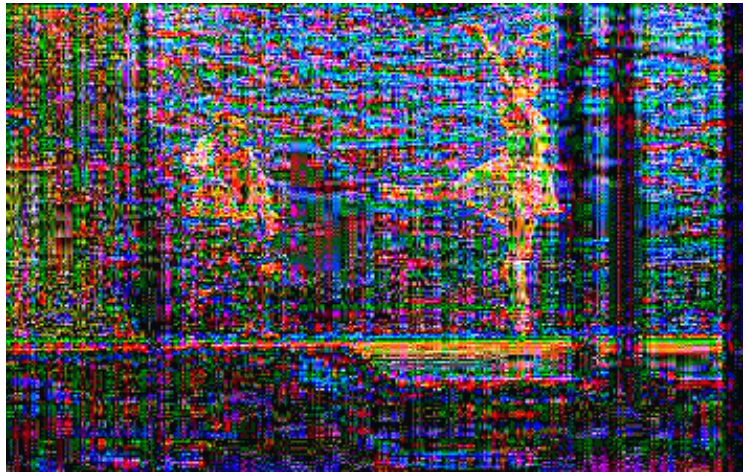


12bit



8bit

4bit



Technologies for Quality Driven Design

- Definition of the quality
  - » Images and Audio Data
  - » Measures and Measurement Tools
- Relation between accuracy and quality
  - » Sensitivity Analysis
- Automatic program transformation
  - » From original algorithm, define required accuracy of computation and transform the original program

Conclusion

- Each application requests different datapath width.
- Find and prepare the optimum datapath width for a given set of applications.
  - » Optimization without loss of quality
  - » Optimization with quality loss (Quality Driven Design)
    - – Variation of output devices and requirement of quality
- Trade-off between area and performance
- Reduction of unused circuits and meaningless switching
  - » Reduction of energy consumption by dynamic power consumption and leakage currents.
- Techniques are available for both HW and SW design.

References

[1] Y. Cao and H. Yasuura ``A System-level Energy Minimization Using Datapath Optimization", International Symposium on Low Power Electronics and Design, August 2001.

[2] B. Shackleford, et al, ``Memory-CPU Size Optimization for Embedded system Designs," Proc. of 34th Design Automation Conference (34th DAC), June 1997.

[3] T. Ishihara and H. Yasuura, ``Programmable Power Management Architecture for Power Reduction," IEICE Trans. on Electronics, vol. E81-C no. 9, pp.1473-1480, September 1998.

[4] H. Yamashita, H. Yasuura, F. N. Eko, and Yun Cao, ``Variable Size Analysis and Validation of Computation Quality", Proc. of Workshop on High-Level Design Validation and Test, HLDVT00, Nov. 2000.

[5] M. Stephenson, J. Babb, and S. Amarasinghe, ``Bitwidth Analysis with Application to Silicon Compilation, "Conf. Programming Language Design and Implementation, June 2000.

References (cont.)

[6] M.-A. Cantin and Y. Savaria,"An Automatic Word Length Determination Method", Proc. of The IEEE International Symposium on Circuit and Systems, V53-V56, May. 2001.

[7] S. Mahlke, R. Ravindran, M. Schlansker, R. Schreiber, and T. Sherwood, ``Bitwidth Cognizant Architecture Synthesis of Custom Hardware Accelerators," IEEE Trans. CAD, vol. 20, no. 11, pp. 1355--1371, Nov. 2001.

[8] H. Yasuura, H. Tomiyama, A. Inoue and F. N. Eko, ``Embedded System Design Using Soft-core Processor and Valen-C", IIS J. Info. Sci. Eng., vol. 14, pp.587-603, Sept. 1998.

[9] F. N. Eko, et.al., ``Soft-Core Processor Architecture for Embedded System Design," IEICE Trans. Electronics, vol. E81-C, no. 9, 1416-1423, Sep. 1998.

[10] A. Inoue, et al. ``Language and Compiler for Optimizing Datapath Widths of Embedded Systems," IEICE Trans. Fundamentals, vol. E81--A, no. 12, pp. 2595--2604, Dec. 1998.

[11] C.N. Taylor, S. Dey, and D. Panigrahi, ``Energy/Latency/Image Quality Tradeoffs in Enabling Mobile Multimedia Communication", Proc. of Software Radio: Technologies and Services, Enrico Del Re, Springer Verlag Ltd., January 2001.

[12] Y. Cao and H. Yasuura, ``Video Quality Modeling for Quality-driven Design", the 10th Workshop on System and System Integration of Mixed Technologies (SASIMI 2001), Oct. 2001.