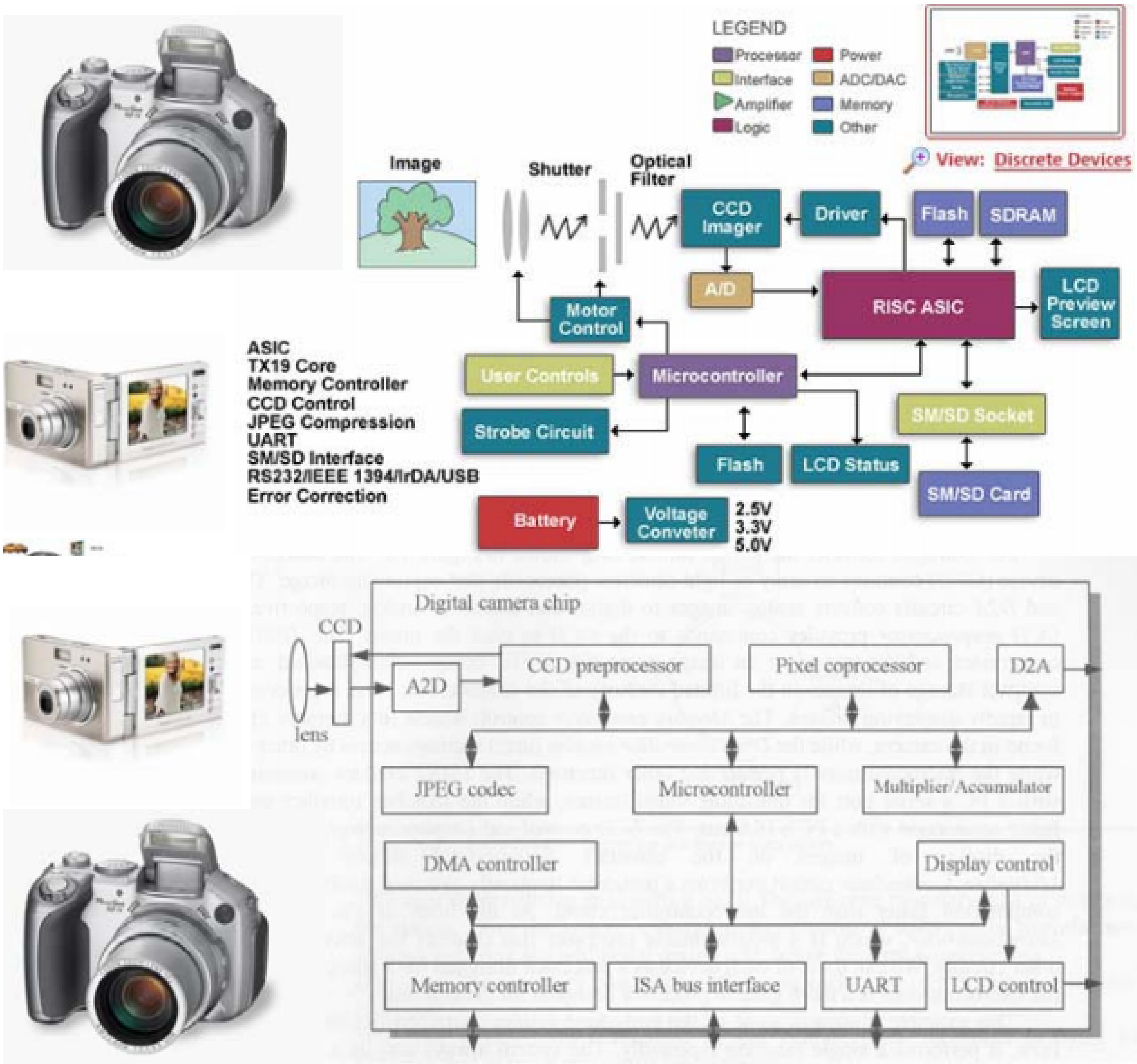
Embedded System
Lecture Note 3. Introductory Example /DSC (Digital Still Camera)

tags: `Embedded`

> — Chenging Fri, Apr 30, 2021

Introduction to Digital Camera



A. Background Information

a. Digital Still Camera (DSC)

1. typical consumer embedded system product
2. Captures and stores pictures in digital format

b. Main operations (simplified)

1. Converts image obtained from CCD device to digital form

   (1) Autofocus (A/F)
   (2) Auto-Exposure (A/E)
   (3) Zero-bias adjustment

2. Image processing

(1) White balancing
(2) Color reproduction

3. Compression (store the image in JPG format)

   (1) FDCT (Forward Discrete Cosine Transformation)
   (2) Quantization
   (3) Huffman encoding

4. Image archiving (storing)

c. Auxiliary operations (simplified)

1. Pre-viewing images
2. Storing images
3. Transferring images

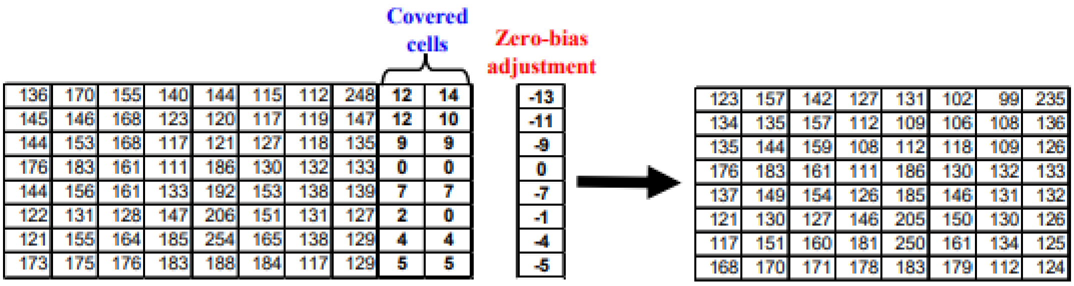
d. Structure of digital camera

B. Introduction of Main operations

1. Capture of Image – CCD (Charged Coupled Device)

- Special sensor that captures an image
- Light-sensitive silicon solid-state device composed of many cells (usually arranged in a two-dimension array)

- Primary stages in image generation:
  - Step 1. charge generation
  - Step 2. collection and storage of the liberated charge
  - Step 3. charge transfer
  - Step 4. charge measurement
- Zero-Bias Adjustment



136	170	155	140	144	115	112	248	12	14
145	146	168	123	120	117	119	147	12	10
144	153	168	117	121	127	118	135	9	9
176	183	161	111	186	130	132	133	0	0
144	156	161	133	192	153	138	139	7	7
122	131	128	147	206	151	131	127	2	0
121	155	164	185	254	165	138	129	4	4
173	175	176	183	188	184	117	129	5	5

-13
-11
-9
0
-7
-1
-4
-5

123	157	142	127	131	102	99	235
134	135	157	112	109	106	108	136
135	144	159	108	112	118	109	126
176	183	161	111	186	130	132	133
137	149	154	126	185	146	131	132
121	130	127	146	205	150	130	126
117	151	160	181	250	161	134	125
168	170	171	178	183	179	112	124

Before zero-bias adjustment

After zero-bias adjustment

- Image Processing – Compression
- JPEG (Joint Photographic Experts Group)
- A standard format for digital images in a compressed form
- Image data divided into blocks of 8 x 8 pixels
- 3 steps performed on each block
  - Step 1. DCT (Discrete Cosine Transform: FDCT/IDCT)
  - Step 2. Quantization
  - Step 3. Huffman encoding (Variable length encoding)

2. Image Processing - Compression

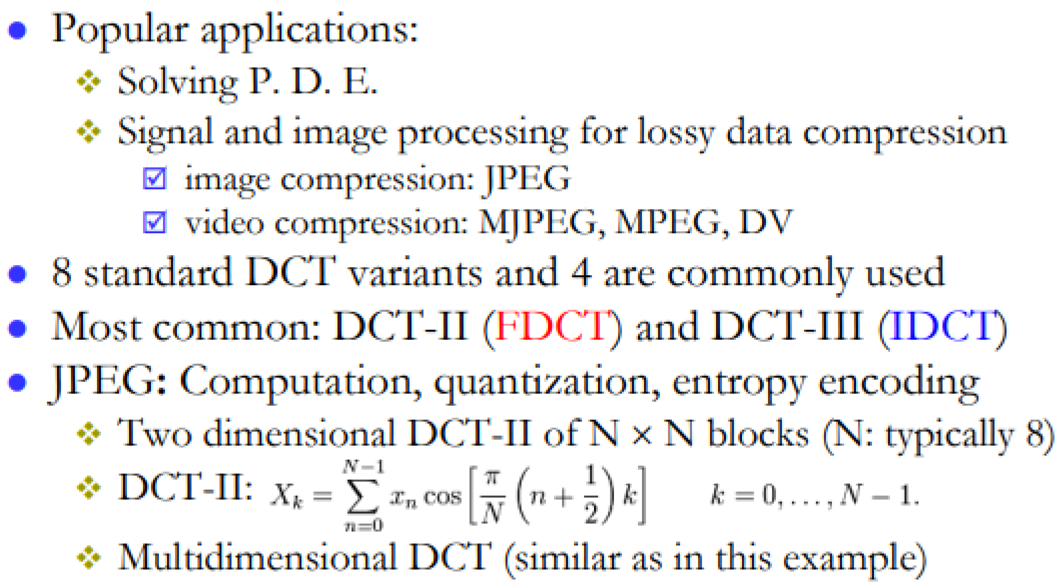
- Save space for storage and time for transmission
- Lossless or lossy (non-lossless)
- JPEG (Joint Photographic Experts Group)
  - A standard format for digital images in a compressed form
  - Image data divided into blocks of 8 x 8 pixels
  - 3 steps performed on each block
    - DCT (Discrete Cosine Transform: FDCT/IDCT)
    - Quantization
    - Huffman encoding (Variable length encoding)

3. Image Processing – DCT

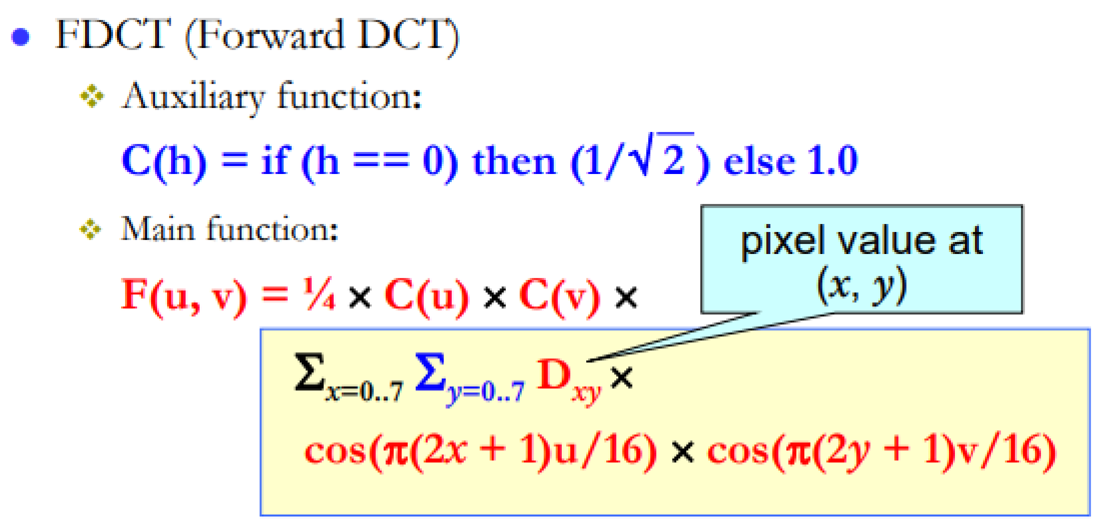
- Discrete Cosine Transform
- Transforms original 8 x 8 block into the domain of cosine-frequency
  - Upper-left corner values represent more of the essence
  - Lower-right corner values represent finer details

- Can reduce precision and retain reasonable image quality
- DCT (Discrete Cosine Transformation)
  - FDCT (Forward DCT): Encoding
  - IDCT (Inverse DCT): Decoding [not needed here]
- Computation intensive

4. Image Processing – more on DCT



5. Image Processing – DCT Formula



6. Image Processing – Implementation of DCT

```c
static double IMG_A[8][8], IMG_B[8][8];
void FDCT(void){
    int u, v, x, y;
    double r, sum;
    for (u = 0; u < 8; u++){
        for (v = 0; v < 8; v++){
            cu = u ? 1.0 : (1.0/sqrt(2.0));
            cv = v ? 1.0 : (1.0/sqrt(2.0));
            for (x = 0; x < 8; x++){
                for (y = 0; y < 8; y++){
                    r = r + IMG_A[x][y] * cos(3.1416*(2*y+1)*v/16.0);
                }
                sum = sum + r*cos(3.1416*(2*x+1)*u/16.0);
            }
        }
        IMG_B[u][v] = 0.25*cu*cv*sum;
    }
}
```

Implementing FDCT:

- Using table look-up
- Using fixed-point arithmetic
  - Floating-point values are multiplied by 32,768 and rounded to nearest integer
  - 32,768 chosen in order to store each value in 2 bytes of memory
- Loop unrolling

```c
static const short COS_TABLE[8][8] = {
  { 32768, 32138, 30273, 27245, 23170, 18204, 12539,  6392 },
  { 32768, 27245, 12539, -6392,-23170,-32138,-30273,-18204 },
  { 32768, 18204,-12539,-32138,-23170,  6392, 30273, 27245 },
  { 32768,  6392,-30273,-18204, 23170, 27245,-12539,-32138 },
  { 32768, -6392,-30273, 18204, 23170,-27245,-12539, 32138 },
  { 32768,-18204,-12539, 32138,-23170, -6392, 30273,-27245 },
  { 32768,-27245, 12539,  6392,-23170, 32138,-30273, 18204 },
  { 32768,-32138, 30273,-27245, 23170,-18204, 12539, -6392 }
};
```

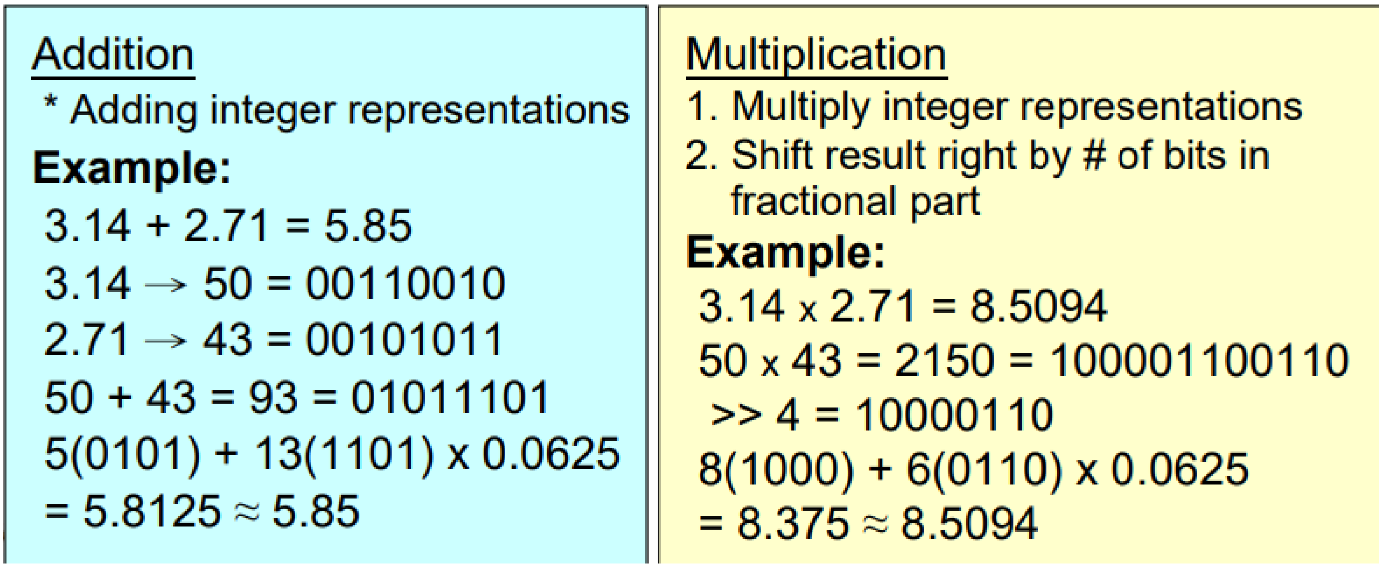
```c
static short ONE_OVER_SQRT_TWO = 23170;
static double COS(int xy, int uv) {
  return COS_TABLE[xy][uv] / 32768.0;
}
static double C(int h) {
  return h ? 1.0 : ONE_OVER_SQRT_TWO / 32768.0;
}
```

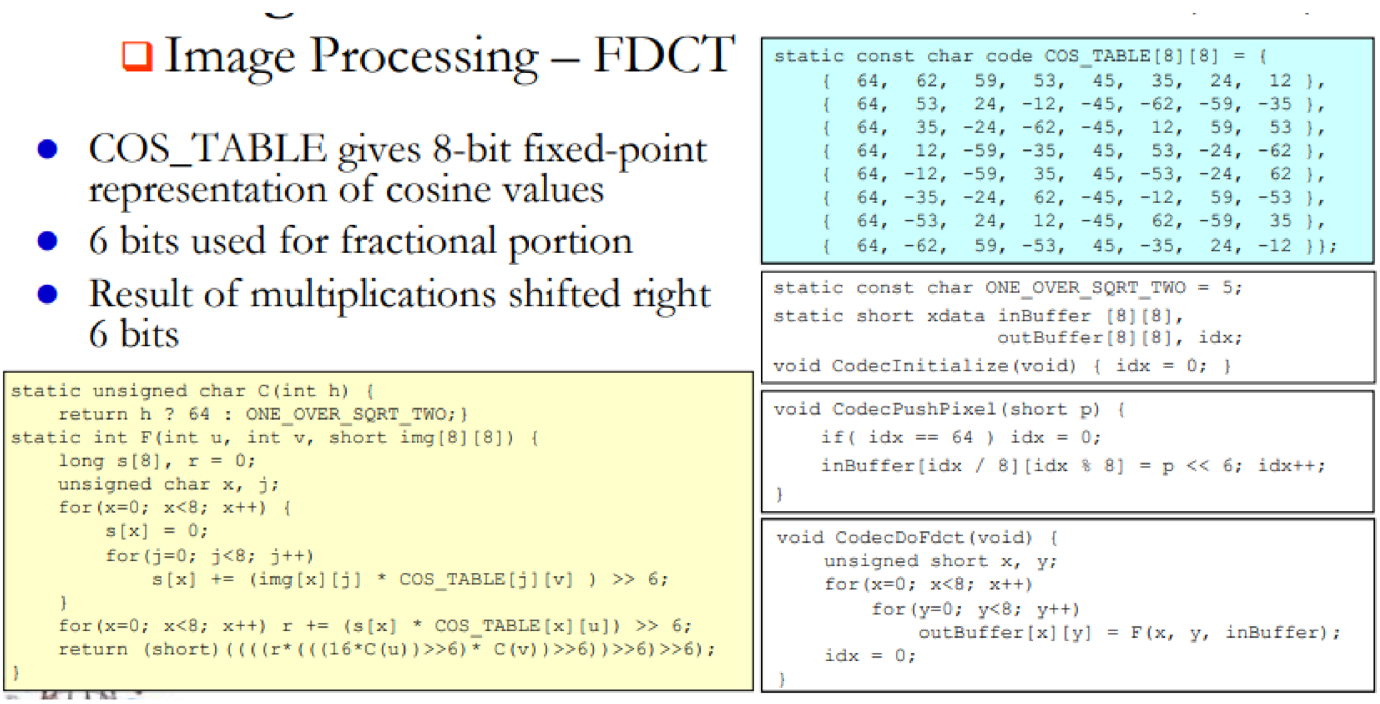
```c
static int FDCT(int u, int v, short img[8][8]) {
   double s[8], r = 0; int x;
   for(x=0; x<8; x++) {
    s[x] = img[x][0] * COS(0, v) + img[x][1] * COS(1, v) +
           img[x][2] * COS(2, v) + img[x][3] * COS(3, v) +
           img[x][4] * COS(4, v) + img[x][5] * COS(5, v) +
           img[x][6] * COS(6, v) + img[x][7] * COS(7, v);
   }
   for(x=0; x<8; x++) r += s[x] * COS(x, u);
   return (short)(r * .25 * C(u) * C(v));
}
```

- Translating a real value to a fixed-point representation
  - Multiply real value by 2 ^ (# of bits used for fractional part)
  - Round to nearest integer
- Example : 3.14 as 8-bit integer with 4 bits for fraction
  - 2^4 = 16
  - 3.14 * 16 = 50.24 = 50 = 00110010
  - 16(2^4) possibrel values for fraction, each represents 1/16
  - The last 4 bits (0010) = 2 => 2*0.0625(1/16) = 0.125

- 3 (0011) + 0.125 = 3.125 與3.14算相近
  more bits for fraction would increase accuracy

注意乘法會較加法不精準





7. Image Processing – Quantization

- Achieve high compression ratio by reducing image quality
- Reduce bit precision of encoded data:
  - Fewer bits needed for encoding
  - One way is to divide all values by a factor of 2 (right shift)

- Reverse the process for dequantization

1150	39	-43	-10	26	-83	11	41
-81	-3	115	-73	-6	-2	22	-5
14	-11	1	-42	26	-3	17	-38
2	-61	-13	-12	36	-23	-18	5
44	13	37	-4	10	-21	7	-8
36	-11	-9	-4	20	-28	-21	14
-19	-7	21	-6	3	3	12	-21
-5	-13	-11	-17	-4	-1	7	-4

After being decoded using DCT

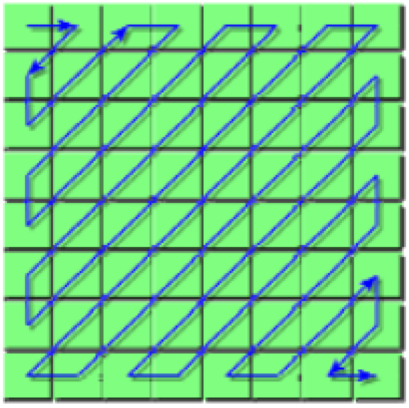
Divide each cell's value by 8 →

144	5	-5	-1	3	-10	1	5
-10	0	14	-9	-1	0	3	-1
2	-1	0	-5	3	0	2	-5
0	-8	-2	-2	5	-3	-2	1
6	2	5	-1	1	-3	1	-1
5	-1	-1	-1	3	-4	-3	2
-2	-1	3	-1	0	0	2	-3
-1	-2	-1	-2	-1	0	1	-1

After quantization

8. Image Processing – Hoffman Encoding

- Variable length encoding
    - More frequently occurring pixels assigned short binary code
    - Longer binary codes left for less frequently occurring pixels
- Serialize 8 x 8 block of pixels and values are converted into single list using zigzag pattern
- Each pixel in the serial list is converted to Huffman encoded value
- Huffman encoding is reversible (No code is a prefix of another code)

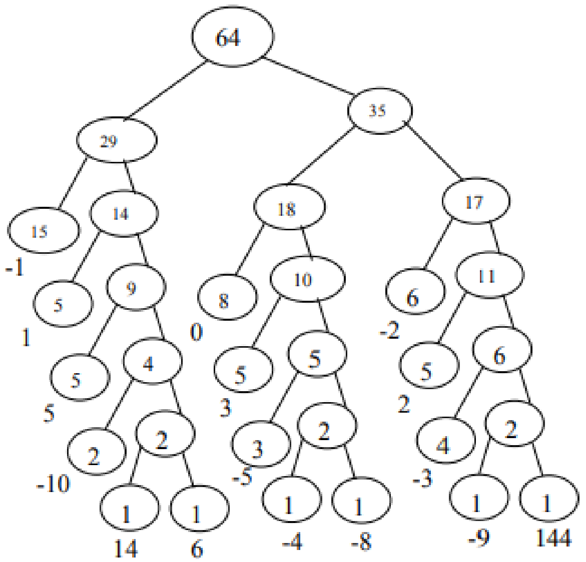


- Build Huffman tree from bottom up
    - leaf for pixel
    - internal node of min sum
- Traverse tree from root to obtain leaf's pixel value
- Append
    - 0 : for left
    - 1 : for right traversal.

Pixel frequencies

-1	15x
0	8x
-2	6x
1	5x
2	5x
3	5x
5	5x
-3	4x
-5	3x
-10	2x
144	1x
-9	1x
-8	1x
-4	1x
6	1x
14	1x

Huffman tree



Huffman codes

-1	00
0	100
-2	110
1	010
2	1110
3	1010
5	0110
-3	11110
-5	10110
-10	01110
144	111111
-9	111110
-8	101111
-4	101110
6	011111
14	011110

C. The List of Requirements

Product Requirement List							
Doc ID		Product	Low-end Digital Still Camera			Date	/ /
Functional Requirements				Non-Functional Requirements			
F01	Taking photo & processing			N01	Processing fast enough	~ 1 sec. ?	
F02	Processing and storing photos in digital format	JPEG?		N02	Cost: IC ≤ $25, Tot ≤ $100	USD	
				N03	IC gate count <= 200K+		
F03	Storing at least 50 photos			N04	Reasonable size with IC		
F04	Image resolution: Low			N05	Reasonable weight suitable for hand-held		
F05	Loading the photo images to a PC computer			N06	Operating w/o cooling fan	< 50°C ?	
				N07	Low power consumption for long battery life		
				N08	Time-to-Market < 6 Mon.		
Reviewer		/ /		Approved by		/ /	

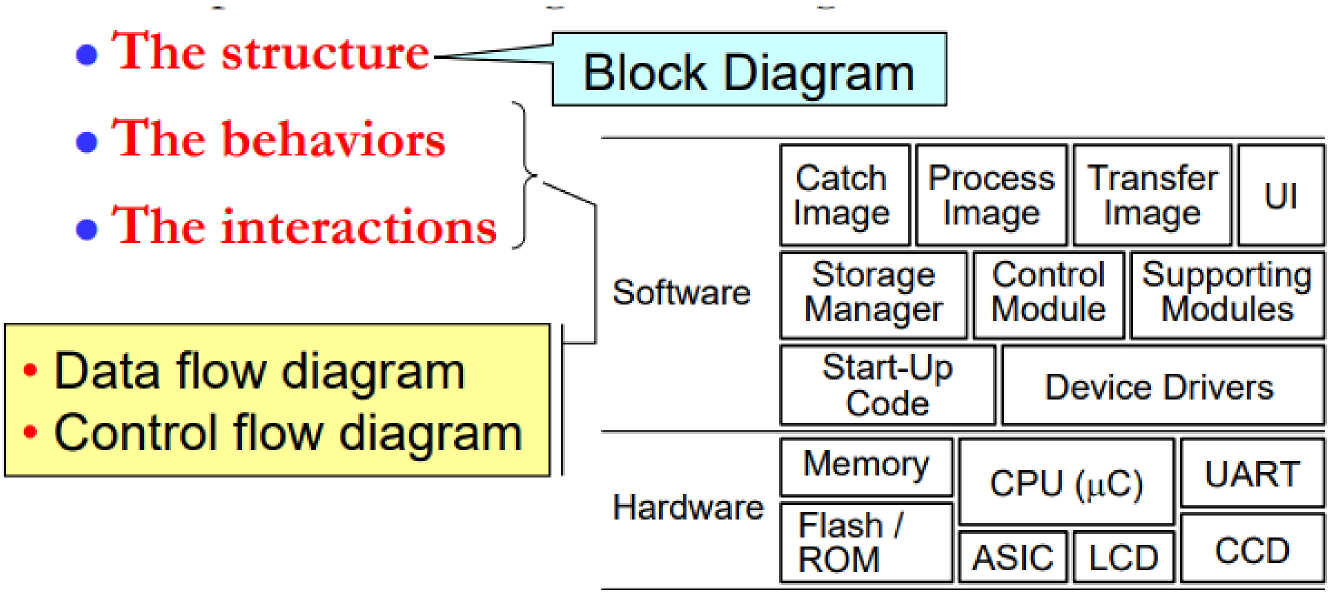
D. The List of Specifications

Product Specifications				
Doc ID		Product: Low-end Digital Still Camera		Date: / /
Item	Function	Specification/Description		Note
S01	Taking photos	1. Digital image format: jpg of resolution 320 × 160 2. Performance: image processing time about 1 sec.		
S02	Storage	1. Capacity: at least 50 photo images 2. Writing photo image for storing 3. Reading stored photo image for transferring		
S03	Connection	1. Connect to PC computers via RS-232C interface 2. Data Rate: auto-set and at least 9600 bits/sec.		UART
S04	User Interface	No (Auto transferring or operating at computer side)		Switches
S05	Miscellaneous	1. Can work at 0 ~ 50°C and no cooling fan is allowed 2. Low power consumption for long battery life 3. Weight (w/o battery) <= 500g and reasonable size		
Reviewer		/ /	Approved by	/ /

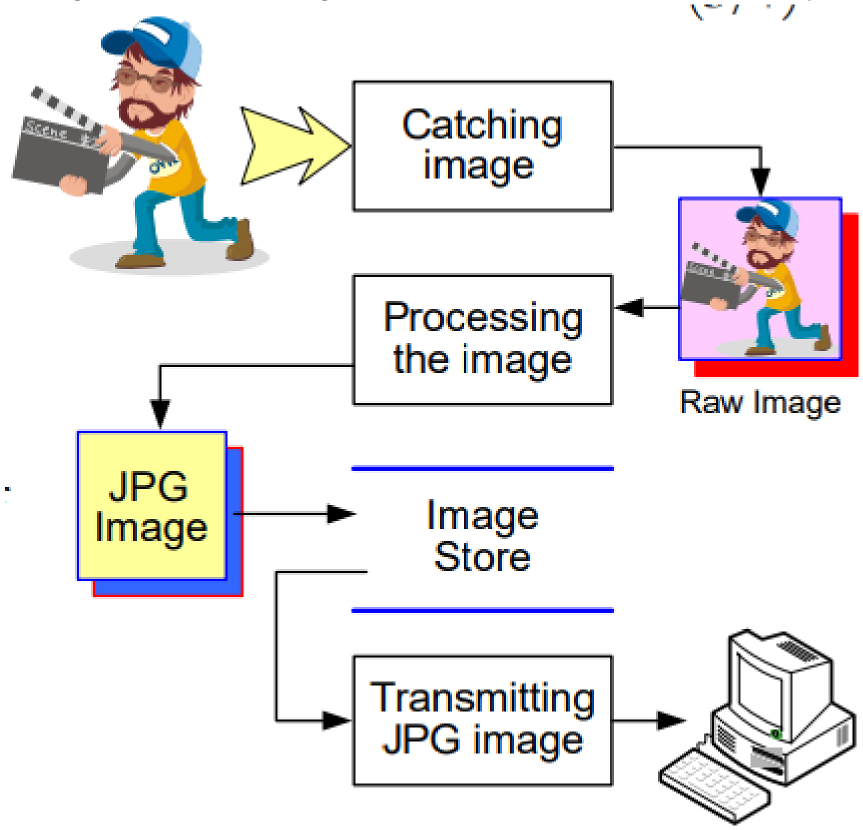
E. The Design

- Description of system architecture
  - Hardware components
  - Software structure
- Mapping functionality to the architecture
  - One function on one processor
  - One function on many processors

- Many functions on one processor
- One person’s specification may be the implementation of another person!

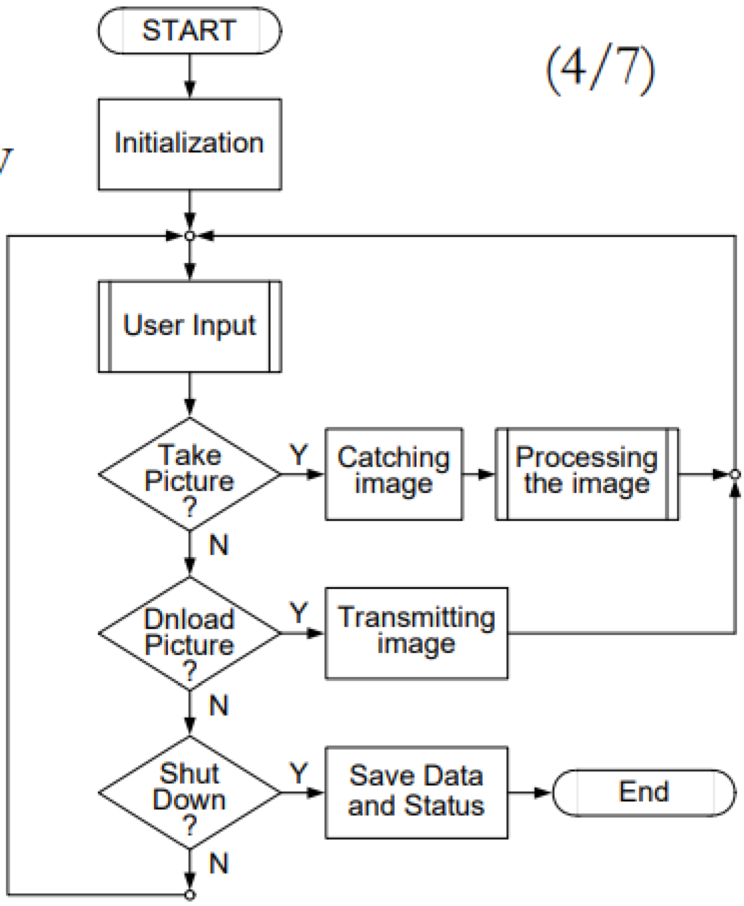


Using a data flow diagram to describe the functional specifications

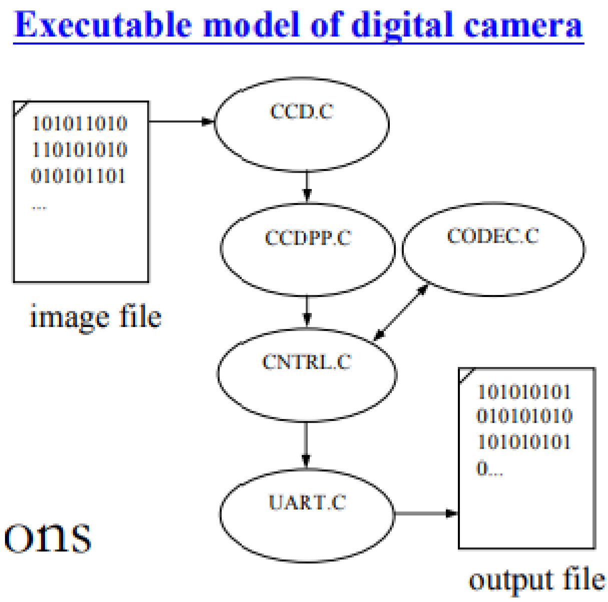


(4/7)

W



A high-level executable model of specifications

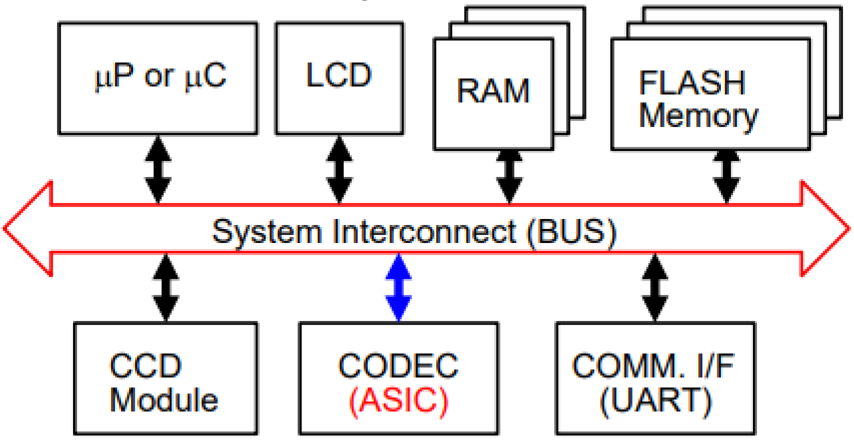


An example of the unified H/W and S/W approach

Function	Operations	Design	Mapping
Capture an image	1. CCD operation 2. Zero-bias adjustment	1. CCD module 2. CCD driver	H/W+S/W
Convert an image into JPG	1. FDCT 2. Quantization 3. Huffman encoding	Encoder	H/W or S/W
Store an image	1. Copy JPG image 2. Manage directory	SaveImage	S/W
Transfer an image	1. UART connection 2. Transmission 3. Manage directory	1. UART module 2. UART driver 3. ReadImage 4. SendImage	H/W+S/W

- System architecture
  - CCD module
  - Processor: General-purpose or special-purpose or both
  - Memory: RAM and Flash Memory
  - Bus
  - I/O Devices
  - Application-specific hardware components (ASIC)
  - Software

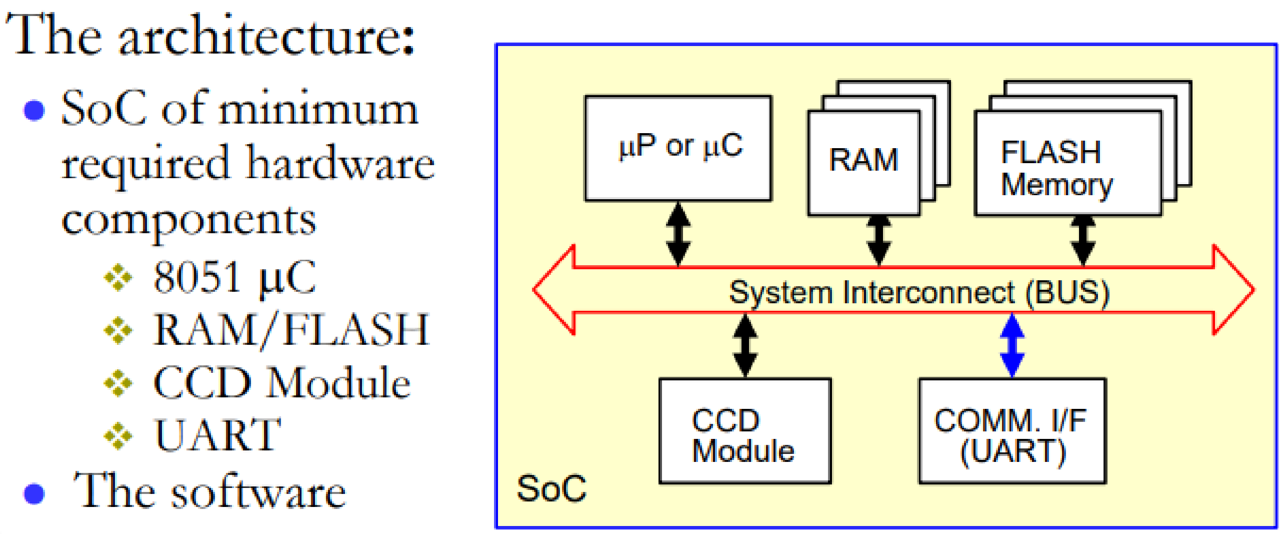
- Basic architecture of a digital camera

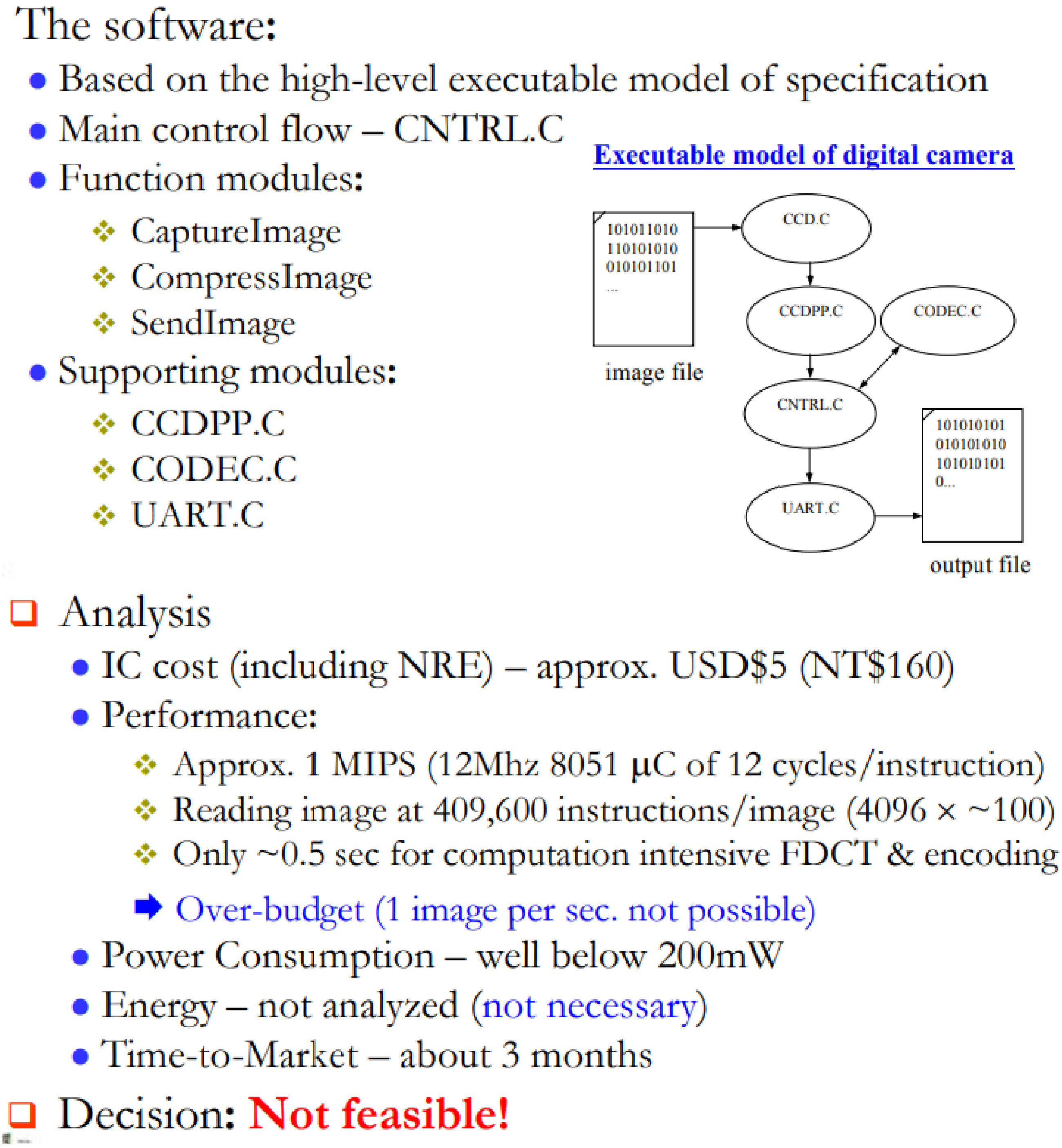


Different Implementation

Implementation 1 – Microcontroller (micro C) alone:

- Hardware – min. required hardware with 12MHz 8051 □C
- Software – all function modules running on the 8051

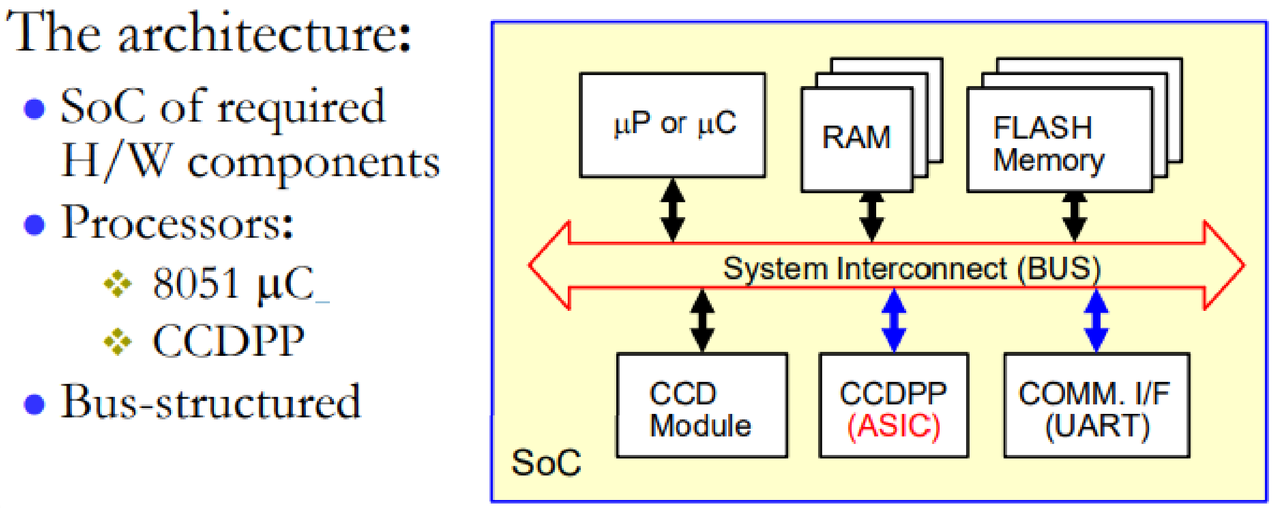


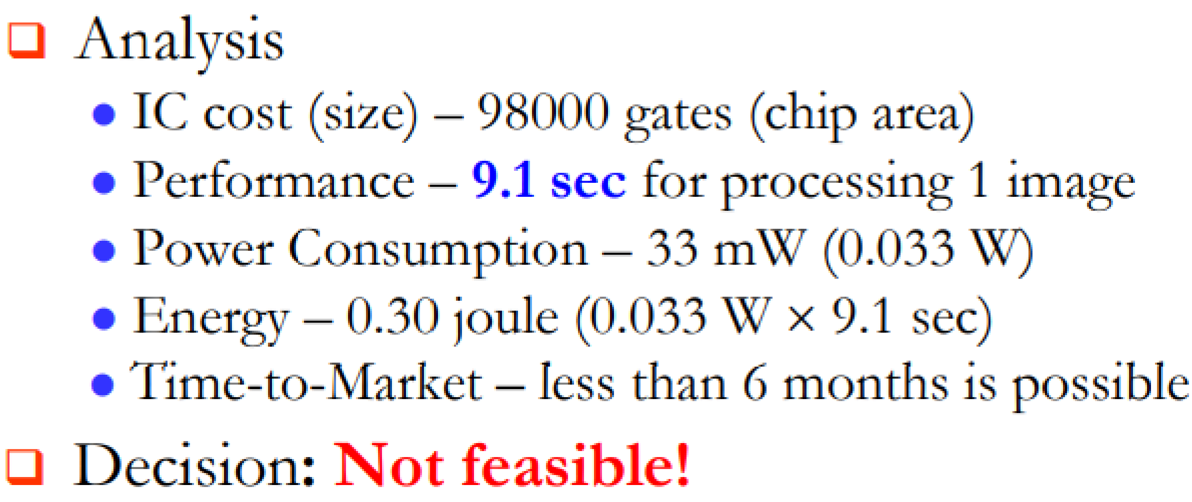


Implementation 2 – micro C + H/W Function Unit:

- Hardware – SOC of 12MHz 8051 micro C and CCDPP

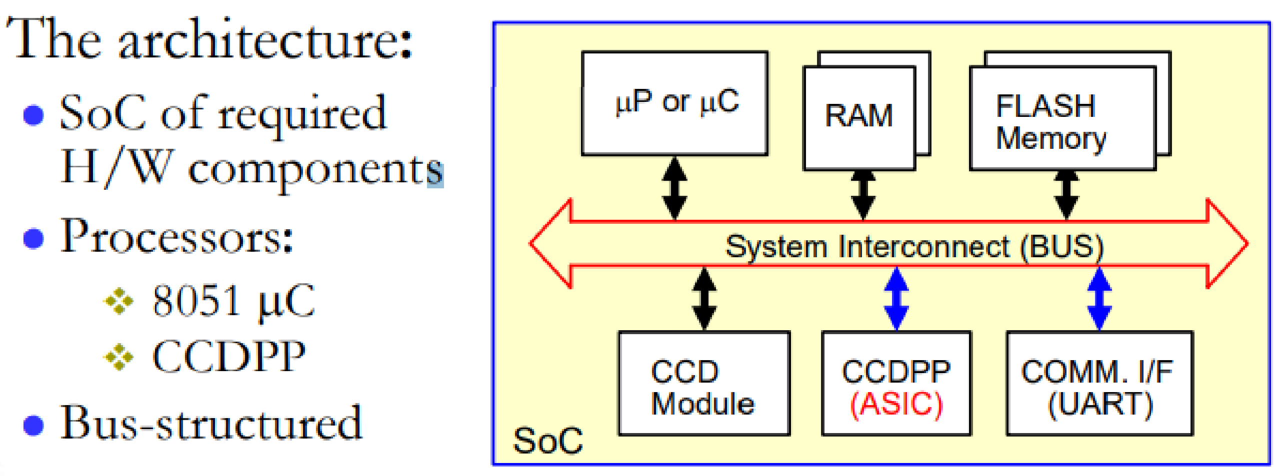
- Software – function modules and control module of CCDPP





Implementation 3 – micro C + H/W Function Unit:

- Hardware – SOC of 12MHz 8051 micro C and CCDPP
- Software – function modules with fixed-point FDCT and



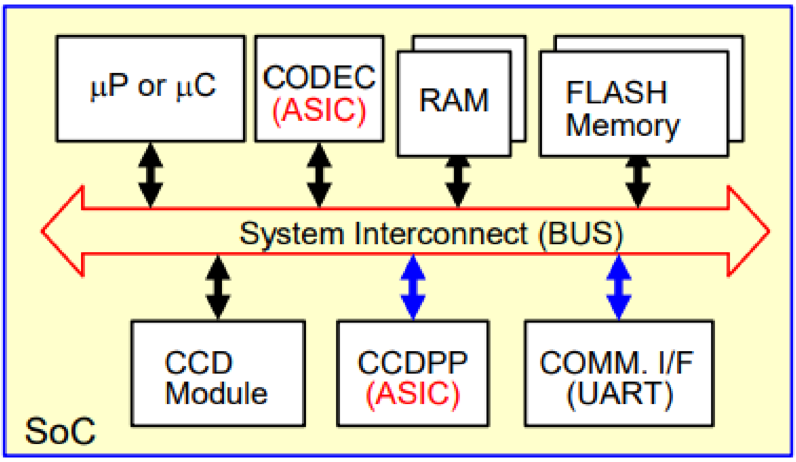
- Analysis
  - IC cost (size) – 90000 gates (chip area)
  - Performance – 1.5 sec for processing 1 image
  - Power Consumption – 33 mW (0.033 W)
  - Energy – 0.050 joule (0.033 W × 1.5 sec)
  - Time-to-Market – less than 6 months is possible
- Decision: A possible solution

Implementation 4 – □C + H/W Function Units:

- Hardware – SOC of 12MHz 8051 micro C, CCDPP and CODEC
- Software – function modules with fixed-point FDCT and control modules of CCDPP and CODEC

The architecture:

- SoC of required H/W components
- Processors:
  - 8051 µC
  - CCDPP
  - CODEC
- Bus-structured



- Analysis
  - IC cost (size) – 128000 gates (chip area)
  - Performance – 0.099 sec for processing 1 image
  - Power Consumption – 40 mW (0.040 W)
  - Energy – 0.0040 joule (0.040 W × 0.099 sec)
  - Time-to-Market – 6 months might not be possible (i.e. > 6 Mon. or 6^+ Mon.)
- Decision: A possible solution

Comparison and Discussion

🗁 Comparison and Discussion:

Implementation	1	2	3	4
Performance	>> 1 sec.	9.1 sec.	1.5 sec.	0.099 sec.
Size (no. gates)	N/A	98000	90000	128000
Power (mW)	<< 200	33	33	40
Energy (joule)	N/A	0.30	0.050	0.0040
Time-to-Market	~ 3 Mon.	< 6 Mon.	< 6 Mon.	? (6+ Mon.)

- Which is better?
- Main lesson: **Trade-off** between **H/W** and **S/W**

業界通常選3，性能不要太好