Deduction and Logic Implementation of the Fractal Scan Algorithm

Zhangjin Chen, Feng Ran, Zheming Jin
Microelectronic R&D center, Shanghai University
Shanghai, China

and

Meihua Xu
School of Mechatronical Engineering and Automation, Shanghai University
Shanghai, China

ABSTRACT

This paper presents the deduction and logic implementation of the fractal scan algorithm based on the mathematical model of the optimal scan architecture. Through the exploration of the sub-space code sequences and bit code sequences for different gray levels, we deduce the general formulae for the sequences and propose the corresponding logic implementations. Moreover, the parameterized IP core for various gray levels, embedded in flat panel display scan controller, can efficiently increase the scan utilization and imaging quality. This research provides a new engineering way to solve a pressing problem of high resolution flat panel display technology.

Keywords: Flat Panel Display, Logic Implementation, Fractal Scan Algorithm, Code Sequence, IP Core

1. INTRODUCTION

The Flat Panel Display (FPD) that has been rapidly replacing bulky CRT based traditional displays plays an important role in exploring the progress of modern IT devices [1]. As the high resolution flat panel display technology for large display size has been one of the key research fields, one of the bottlenecks which impede the technology development from current small display-size flat panel displays to high resolution large display-size ones are low grayscale scan utilization and efficiency [3, 4]. Xu et al. [2, 3] proposed an innovative fractal scan structure which significantly increases scan efficiency. In their paper, they studied different scan schemes and constructed a mathematical model through the analysis of the fractal characteristics of space-time mapping topology. The theoretic model proved an optimal scan structure with maximum scan utilization.

Based on the aforementioned mathematical model, this paper presents the logic implementation of the fractal scan algorithm as a fractal scan intellectual property (IP) core for various gray levels. Compared with traditional scan methods, simulation and real application show 100% scan utilization and a significant decrease of system clock frequency (a multiple of 29 for 256 gray levels) [2, 3]. Thus, the paper provides a new light to solving one of the pressing problems of high resolution FPD technology.

The remainder of this paper is organized into five sections. Section two introduces fractal scan topology. Section three presents the detailed deduction of the algorithm. Section four describes logic implementation of the algorithm. Section five describes design and simulation of the fractal scan IP core. Section six contains some concluding remarks.

2. OVERVIEW OF THE FRACTAL SCAN ARCHITECTURE

Fractal Space-time Mapping Topology

A key to the research of grayscale image scan algorithm is to convert the grayscale storage matrix into pixel lightening time. Hence, a space-time mapping topology for grayscale scan can be constructed at first. The grayscale matrix in space is then mapped to a grayscale scan space-time plane. Based on this mapping architecture, the optimal scan structure is explored. Since the paper is mainly focused on the logic implementation of the optimal scan algorithm, interested readers can refer to the literature [3, 4, 5] for more details on different scan structures and methods.

Consider a scan line and rotate it by 90 degrees backward. Then the line and the time axis form a time-space plane, in which the horizontal axis represents the pixels and the vertical axis the grayscale scan time. For n-bit grayscale (n bits in the binary representation of the gray level), the entire time-space plane is partitioned vertically into \( M_n \) smaller planes (sub-spaces), each being of the same width. In each sub-space, there are \( n \) scan bits (scan for \( n \) times). For simplicity of discussion, denote \( H_n \) as the grayscale weight of the \( k \)th scan bit, \( E_n \) as the sum of weight of each scan bit, and \( T_n \) as the total scan time which is the sum of \( E_n \) and the blanking interval. As the plane is divided into \( 2^{n-1} \) sub-spaces, the optimal scan structure makes use of the interval in between two scans of the same pixel (in the same column) to scan the pixels in other sub-spaces. The new scan scheme, according to the fractal characteristics of the structure, takes full advantage of the time interval and achieves 100% scan utilization. Figure 1 shows the fractal scan topology for 16-bit grayscale (\( M_4 = 8 \), \( E_4 = 32 \) and \( T_4 = 40 \)). The black solid dot represents the bit being scanned. All the dots from \( t_0 \) to \( t_9 \) make a set of “Z” patterns. Each of the eight sub-spaces is scanned 5 times in total at the time (vertical) axis. The scan distance for the first four scans is among 1 (\( H_1 = 1 \)), 3 (\( H_2 = 3 \)), 8 (\( H_3 = 8 \)) and 20 (\( H_4 = 20 \)), which corresponds to the grayscale weight. At each time unit, only one sub-space is scanned. The last eight scans (\( t_2 \) to \( t_9 \)) deal with blanking.

It is easy to see in Figure 1 that each sub-space code (see column C) at each time unit is just the encoding assigned to that sub-space. In addition, there is a one-to-one correspondence...
between the grayscale weight and the bit code (see column D): for a weight of 20, the bit code is 3; for a weight of 10, the bit code is 2. Table 1 lists different weight sequences and bit code for weights of 20, 10, and 5. The formula for the sequence is determined by the gray levels and the scan scheme. Since there are $2^n$ sub-spaces for n-bit grayscale, $NB = \log_2 n$ and NS = n-1. At the positive edge of the clock when GetNext is asserted (GetNext = 1), the module outputs current bit code Bit[NB-1:0], sub-space code Segment[NB-1:0], and the Hidden signal to control row/column addresses and gray value.

To avoid circuit hazards, the eight sub-spaces are encoded with the Gray code which has the property that when advancing from one index to the next adjacent index, only a single bit changes value. In actual hardware design, quasi-Gray code is adopted.

<table>
<thead>
<tr>
<th>Sub-space encoding</th>
<th>Weight sequences</th>
<th>Bit code sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20 8 3 1</td>
<td>3 2 1 0</td>
</tr>
<tr>
<td>1</td>
<td>20 8 1 3</td>
<td>3 2 0 1</td>
</tr>
<tr>
<td>2</td>
<td>20 3 1 8</td>
<td>3 1 0 2</td>
</tr>
<tr>
<td>3</td>
<td>20 1 3 8</td>
<td>2 1 0 3</td>
</tr>
<tr>
<td>4</td>
<td>8 3 1 20</td>
<td>1 0 2 3</td>
</tr>
<tr>
<td>5</td>
<td>8 1 3 20</td>
<td>0 1 2 3</td>
</tr>
<tr>
<td>6</td>
<td>3 1 8 20</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1 3 8 20</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Weight and Bit Sequences (n = 4)

3. DEDUCTION AND DESCRIPTION OF THE FRACTAL SCAN ALGORITHM

It is appropriate to briefly go through some basic terminology associated with sequences. Sequences of math are a string of associated with sequences. Sequences of math are a string of rules, that determines the next number in the sequence. Formally, the string of numbers is referred to as terms of a sequence. The formula for the nth (general) term is always used to represent a given sequence.

To represent different forms of sequences, the following naming conventions are used:

- The braces, $\{ \}$, are used to enclose all the terms of a sequence. Hence, $\{ A \}$ means the sequence A.
- $\{ A, B \}$: the union of the sequence A and B.
- A : k : the kth term of the sequence A
- !A : the complement of the sequence A
- $\{ A \}^n$ : Repeat the sequence A for n times
- $A + d$ : Add the numeral d to each term in the sequence A
- $|A|$ : the number of terms in the sequence A

Deduction of the Sub-space Sequence and Bit Code Sequence

According to the notations above, all the relevant sequences are defined as follows for n-bit (n+0) gray-scale:

$Z_n$: a sequence containing all the sub-space codes from 0 to $T_n$
\begin{align*}
B_n: & \text{ a sequence containing all the bit codes from 0 to } T_{En-1} \\
X_n: & \text{ a sequence containing the first } |B_n| \text{ terms of } Z_n \\
P_n: & \text{ n-bit Gray code sequence} \\
P_{n:1}: & \text{ a sequence containing the first half terms of } P_n \\
P_{n:2}: & \text{ a sequence containing the second half terms of } P_n \\
\end{align*}

For \( n = 1 \), there is only one sub-space and two scan time units for scan and blanking respectively. Thus, we have

\[
Z_1 = \{0, 0\}, \ B_1 = \{0\}, \ X_1 = \{0\} \quad \text{Eq.(1)}
\]

Similarly, for \( n = 2 \), we have

\[
Z_2 = \{0, 1, 1, 0, 0, 1, 0, 0, 1\}, \\
B_2 = \{1, 0, 1, 0, 1, 0, 1, 0\}, \\
X_2 = \{0, 1, 1, 0, 0, 1, 0, 1\}.
\]

Obviously, the complexity of the sequences \((n \geq 2)\) requires a general method to deduce each sequence in a systematic way. In the following paragraphs, we describe how to formulate the sequences \(Z_n\), \(B_n\), and \(X_n\).

**Characteristics of the Bit Code Sequence**

Since the general formula for each sequence is deduced, it is necessary to analyze the characteristics of the sequences as a basis for logic implementation. From Figure 1 and Eq.(2), we can obtain the following characteristics of \(B_n\) when \( n = 4 \).

1) The term with the number 3 \((k = 3)\), of \(B_n\) occurs consecutively for \(2^{n-1}\) times in the first and third phases, and the term with the number 0 only occurs for one time.

2) When entering the second phase from the first one, if the last term (from the perspective of the first phase) is non-zero, then the first term in the second phase is minus one from the last term. In Figure 1, we see that three minus one is two. The same relationship holds for the two adjacent terms in the third and fourth phases.

3.1) In the second and fourth phases, if \(k = (k = 1, 2)\) occurs consecutively for odd times prior to a term of zero, then the term after the said term also equals to \(k\).

3.2) In the second phase, if \(k = (k = 1, 2)\) occurs consecutively for even times prior to a term of zero, then the next phase will be the third phase.

3.3) In the fourth phase, if \(k = (k = 1, 2, 3)\) occurs consecutively for even times prior to a term of zero, then the next phase will be the fifth phase.

**Characteristics of the Sub-space Code Sequence**

From Eq.(1), Eq.(2) and Eq.(3), we can deduce that for a \(k\) sequence terms, the first and the last terms of the sequence \(X_n\) \((n = k)\) equal zero, and the last term of the sequence \(B_n\) \((n = k)\) is also zero.

Through analysis of the \(Z_n\) sequence terms in each of the five phases (since \(Z_n\) contains all the terms of \(X_n\), we only focus on \(Z_n\)), the following results can be found:

In the first phase of \(Z_n\), the first term is zero and the last term is \(2^{n-1}\). In the second phase, both the first term and the last term are \((2^{n-2} + 2^{n-3})\). In the third phase, the first term is \((2^{n-2} + 2^{n-3})\) and the last term is \(2^{n-3}\). In the fourth phase, both the first and the last term are zeroes. In the fifth phase, the first term is zero and the last term is \(2^{n-2}\).

Then consider the adjacent terms of the two consecutive phases. The first term in the second phase is the next Gray code bits; the last term in the second phase and the first one in the third phase are the same; the last one in the third phase is \(2^{n-3}\) and the first one in the fourth phase is 0; the two terms in the fifth and fourth phases are zeroes.

By collecting these findings, we can reach the following conclusions:

1) The last term in the second (fourth) phase and the first term of the third (fifth) phase are identical.

2) The first term in the fourth phase derives from the last term in the third phase XORed with \(2^{n-2}\).

3) In other cases, the next terms of the sequence \(X_n\) are either those of the sequence \(X_{n:1}\) or the next Gray codes.

Since the characteristics of \(B_n\) can be made use of to determine the boundary of the consecutive phases and there is a
one-to-one correspondence between the terms of $B_n$ and $X_n$, the characteristics of $X_n$ can be formulated as follows:

1) If the current term of $B_n$ is zero, then the next term of $X_n$ is the same as the current term of $X_n$. Namely,
\[ X_n: (k+1) = X_n: k \quad \text{Eq.(5)} \]

2) If the current term of $B_n$ is the last term prior to a set of consecutive terms which occur for $i$ times ($i$ is an even number), then the next term of $X_n$ is the current term of $X_n$ XORed with $2^{i-1}$. Namely,
\[ X_n: (k+1) = X_n: k \oplus 2^{i-1} \quad \text{Eq.(6)} \]

4. LOGIC IMPLEMENTATION OF THE ALGORITHM

Implementation of the Bit Code Sequence

From section three, we see that the $B_n$ sequence also plays a vital role in deducing a general formula of $X_n$. So it is reasonable to implement $B_n$ first. Without the general formula getting in the way or cluttering up the picture, a state transition diagram for 16-bit grayscale to generate the $B_n$ sequence is shown in Figure 4. Since there are four unique numerals (i.e., 3, 2, 1, 0) in $B_n$, four states B3, B2, B1, B0 are needed. In addition, state Bend indicates blanking interval in the fifth phase. Suppose array $K$ has three elements $K[3]$, $K[2]$ and $K[1]$, and each of which indicates either the odd or even number of consecutive occurrences of the corresponding numeral. For example, if $K[2] = 1$, then the numeral 2 occurs consecutively for odd times. Then the state transitions to B2 and $K[2]$ is cleared to indicate an even number of transitions to B2.

If $K[3]$ alone is set, then the next state is B3 for the output of the numeral 2 and $K[1]$ is cleared to indicate an even number of transitions to B3. If $K = 000$, the next state is Bend. In Bend, the state outputs Gray code in sequence, sets $K[3]$ and transitions to B3 for the next turn of the same entire operation above. Through careful observation of the output numerals in each state, the state encodings themselves can be made use of to generate the bit code sequence and the Hidden signal. With five states we encode the states so that they are identical to the outputs. An assignment of 011, 010, 001, 000 and 100 for the five states we encode the states so that they are identical to the outputs.

The state diagram to generate the $X_n$ sequence is also shown in Figure 4. Similar to the function of array $K$, array $D$ specifies the next term of the $X_n$ sequence. According to (5) and (6), if $B0$ is the current state, then sel = 1 and $D = 0$, indicating that the next term of $X_n$ is the same as the current one. For example, in Figure 1, the terms of $X_n$ at $t_1$ and $t_2$ are identical (both equal 4); if B1(B2) is the current state, then $D[0]=!K[1]$ ($D[1]=!K[2]$), indicating that the next term of $X_n$ is the current term XORed with the numeral 1. So the general conclusion can be reached that if $B_i$ is the current state, then $D[i-1]=!K[i]$. The next term of $X_n$ is the current one XORed with $2^{i-1}$. The input signal Sel, from the current state module, controls the multiplexer (MUX) to select between the output of the Gray locator (Sel = 0) and the array $D$ (Sel = 1).

5. DESIGN AND SIMULATION OF THE FRACTAL SCAN IP CORE

To meet the requirement of different grayscale scans, we design a parameterized module which covers from a minimum of 4 to a maximum of 65536 grayscale levels.
**Design of the Fractal Scan IP core**
The block diagram of the core is illustrated in Figure 5. In this figure, Gray Location Code and Parity Flag generate the next gray code for the current sub-space code Segment. BST is the current state encoding in Figure 4. The control modules, Count and Finish, record the current counting state. Bit Code Parity Count K records the number of occurrences for each state.

Next state code, NextBST, is the code generated for the next clock cycle. Note that the current state is the combination of BST, current count state and K to generate, for the next sub-space, bit code select signal Sel, transition code D as well as the bit code (Bit), blanking code (Hidden) and cycle done code (Last). The XOR logic and the multiplexer output the next sub-space code NextSeg.

**Simulation & Results**
For a parameterized IP core, we can obtain simulation waveforms for various grayscales. Figure 7 shows the waveform for 16-bit grayscale \((n=4, \text{NB}=2)\) in the Active HDL™ software environment. At the 10MHz clock frequency, the simulation waveform displays the code sequences Segment, Bit and Hidden as expected in Figure 1.

### 6. CONCLUDING REMARKS
In this paper, the deduction and logic implementation of the fractal scan algorithm are described in details. The fractal scanning IP core, embedded in the FPGA hardware frame for scan controller, has successfully increased image quality. In addition, replacing the traditional serial scanning method with the technique of parallel column decoding, we have significantly raised the system's frame frequency. Real application verifies that high resolution image display can be achieved in our system. What is more, FPD driver circuit cost can be reduced without high-speed circuits and a new approach to grayscale scan control has been established.

![Figure 7: Simulation Waveform for 16-bit Grayscale](image)

### 7. REFERENCES


### 8. ACKNOWLEDGMENTS
The authors would like to acknowledge the National Natural Science Foundation of China for providing financial support for this work under Grant No. 60773081 and Grant No. 60777018, and also to acknowledge the financial support by the Shanghai Municipal Committee under Grant No.AZ028 and Grant No.06DZ22013

Zhangjin Chen is an associate professor of the Computer Center at Shanghai University. His research interests include digital IC design, high resolution flat panel display technology and system integration. He has finished more than 30 research projects and published about 30 technical papers. He holds the BS and the MS in computer science from Shanghai University.