工學碩士 學位論文

Verilog을 이용한 LCD 제어기 설계

指導敎授 崔 赫 煥

이 論文을 工學碩士 學位論文으로 提出함

2002年 8月

釜慶大學校 大學院

電 子 工 學 科

咸 丁 元

함정원의 공학석사 학위논문을 인준함

2002年 6月 29日

주 심	공학박사	문 광 석	
위 원	공학박사	최 혁 환	
위 원	공학박사	이 원 창	

목 차

A Design of LCD Controller by using Verilog

Jeoung-Won Ham

Department of Electronic Engineering, Graduate School.
Pukyong National University

Abstract

In this thesis, the LCD controller using verilog language is designed in RTL-level and synthesized into logic gates using the Foundation 3.1i of Xilinx software, implemented in Xilinx Virtex FPGA chip library. The LCD controller is composed of DMA(Directd Memory Access) controller, FIFO(First-in First out) block, clock generation block, video block and bank block. The video data is transferred by DMA controller from memory device to FIFO and saved in FIFO. The data get out of LCD panel through MUX ARRAY block and dither logic block.

The designed LCD controller can support four color modes : mono mode, 2-bit gray mode, 4-bit gray mode, 8-bit color mode and display of single scan method, dual scan method. Total equivalent gate count for design is 30,507 gates and Maximum frequency is about 21MHz

Ⅰ. 서 론

오늘날 현대 산업사회가 고도의 정보화 시대로 발전함에 따라 다양한 정보를 전달하기 위한 매체로 전자 디스플레이의 중요성은 나날이 증대되고 있다. 지금까지 사용되고 있는 전자 디스플레이 중 가장 대표적인 것은 TV나 컴퓨터 모니터 등에 사용되고 있는 CRT(cathode ray tube)를 들 수 있다. 그러나 현재 CRT는 그 부피나 중량 때문에 휴대가 곤란하고 소비전력이 높으며 높은 구동 전압으로 인한 사용상의 제약이 많기 때문에 그 한계를 극복하기 위한 다양한 평판 표시소자의 개발이 진행되고 있다.

일반적으로 평판소자에는 LCD(liquid crystal display), FED(field emission display), ELD(electroluminescent display)등이 있으며, 이 중 가장 대표적인 것이 LCD이다. 액정과 반도체 기술이 복합된 기술 집약적 품목인 LCD는 얇고, 가벼우며 소비 전력이 낮은 장점으로 인해 여타 평판소자에 비해 넓은 응용분야와 장점을 갖는데 이런 LCD의 넓은 응용성에 발맞추어 LCD 제어기 또한 기능적인 측면이나 하드웨어적인 측면을 고려하여 다양한 요구에 적합하게 설계하고 있다 [4], [5].

흔히 전자장비에 부착되는 LCD는 문자를 디스플레이 하기 위해 많이 사용되는데 이렇게 제한된 용도로 사용되는 LCD 제어기는 단일 모드 칼라, 혹은 단일주사(single scan) 방식만을 지원하는 경우가 대부분이다. 단순한 문자 디스플레이가 아닌 실시간 화면과 같은 그래픽용 제어기는 8-bit의 256 칼라부터 24-bit의 트루(true) 칼라까지의 다양한 칼라 모드와 주사 방식을 지원한다. 그 외 용도에 따라 DMA(direct memory access) 제어기나 메모리(memory) 제어기와 같은 다른 외부 장치가 LCD 제어기 상에 on-chip화 된다.

본 논문에서 설계된 LCD 제어기는 DMA 제어기를 on-chip화 하였으며 FIFO 블록, 레지스터(register) 블록, 클럭 제너레이션 블록 그리고 비디오 블

록으로 구성하였다. CPU로부터 LCD의 기본 동작을 정의하는 데이터가 레지스터 블록으로 입력되면 DMA 제어기가 메모리 장치의 비디오 데이터를 FIFO 블록에 적재한다. 적재된 데이터는 비디오 블록으로부터 발생한 데이터 로드(data load) 신호에 동기 하여 비디오 블록으로 전송되고 MUX_array와 dither 로직을 거쳐 8-bit의 비디오 데이터를 출력한다 [1], [2].

LCD 제어기에 대한 설계는 HDL 언어인 Verilog을 이용하여 이루어졌으며, Xilinx Foundation 3.1i의 Virtex FPGA 칩 라이브러리로 합성하였다. 전체 블록 시뮬레이션을 위해 single port block memory와 메모리 제어기를 설계하였으며, 이를 LCD 제어기와 연동하여 출력 비디오 데이터 값을 확인함으로써 동작을 검증하였다 [6], [7], [8], [10].

Ⅱ. LCD 제어기의 구조

2.1 LCD 제어기의 개요

LCD 제어기는 CPU로부터 기본적인 제어 신호와 메모리 주소 번지를 입력으로 받아 비디오 데이터, 제어신호, 픽셀(pixel) 클럭, 라인(line) 클럭, 그리고 프레임(frame) 클럭을 생성하는 것이 주요 기능이다 [3], [9].

설계된 LCD 제어기는 32-bit의 데이터를 입력으로 받아 8-bit의 비디오 데이터를 출력하는 구조이다. 이를 위해 데이터의 전송을 담당하는 DMA(direct memory access) 블록, 데이터의 저장을 위한 FIFO(first-in first-out) 블록, LCD 판넬(panel)의 구동을 위한 클럭 제너레이션(clock generation) 블록, LCD 제어기의 기본 동작을 정의하는 레지스터(LCDBANK) 블록 그리고 실제적인 비디오 데이터를 선택 가공하는 비디오 블록으로 구성하였다. 그림 1은 전체 LCD 제어기의 블록도를 나타낸다. 그림에서 보는 바와 같이 CPU로부터 LCD 제어기의 동작 제어를 위한 레지스터 값들이 LCDBANK 블록에 입력되면 DMA 제어기가 메모리 장치로부터 비디오 데이터를 FIFO 블록에 적재한다. 적재된 데이터는 비디오 블록의 data load 신호에 동기 하여 비디오 블록으로 데이터를 전송한다. 비디오 블록은 전송된 데이터를 MUX_array와 color lookup table을 거쳐 최종 비디오 데이터로 가공하고 클럭 제너레이션 블록의 동기 신호에 의해 LCD 판넬로 비디오 데이터를 출력한다 [9].

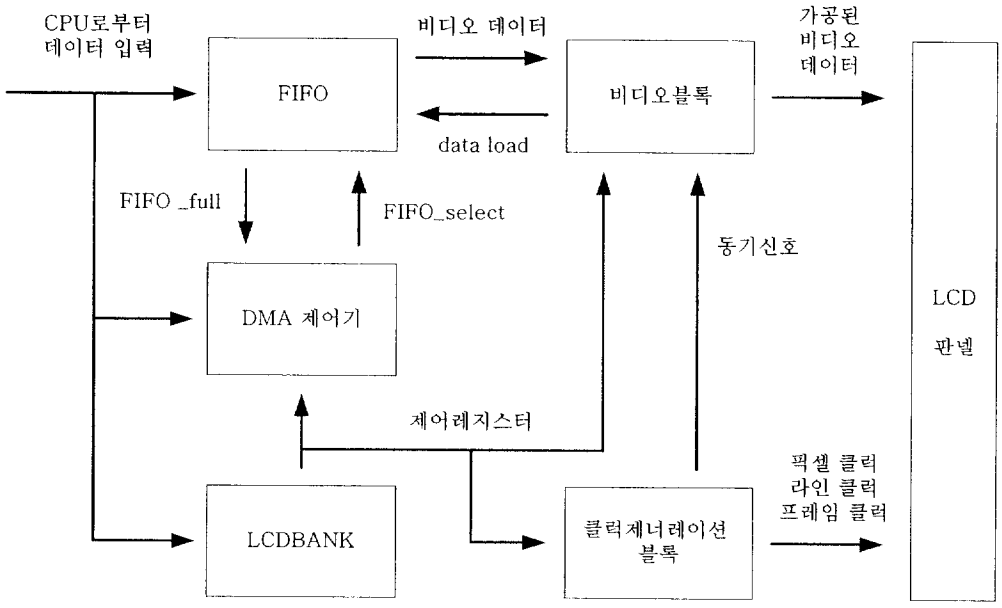


그림 1 LCD 제어기 전체 블록도

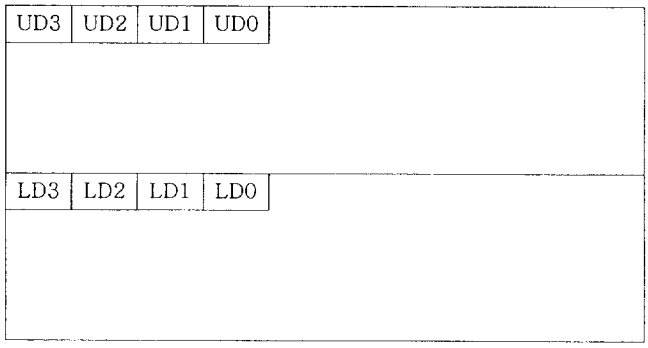
2.2 LCD 제어기의 구조 및 기본 동작

2.2.1 DMA 제어기

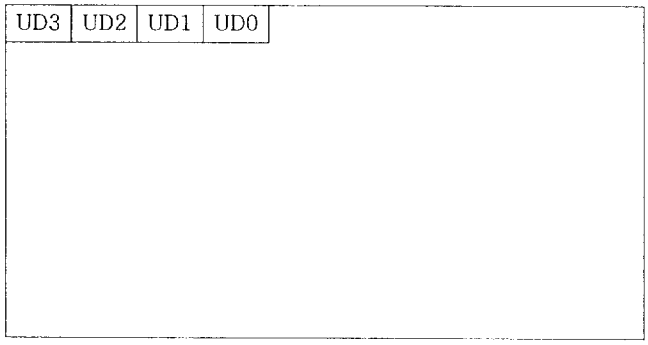
최초의 시스템 리셋(system reset) 신호가 발생한 후 FIFO에는 아무런 데이터가 존재하지 않는다. 이와 같이 FIFO가 비게 되면 FIFO는 data request 신호를 DMA에게 발생한다. DMA는 data request 신호가 활성화되면 CPU에게 버스 제어권을 요청하는 bus request 신호를 발생한다. DMA가 CPU로부터 버스 제어권을 승인 받으면 LCDBANK 블록의 주소 레지스터 값을 유효 주소로 하여 외부 메모리 장치의 비디오 데이터를 FIFO로 전송한다

일반적으로 DMA는 기본적인 동작 제어를 위해 제어로직, 레지스터, 카운터 부분으로 구성되나, 본 논문에서 설계한 DMA는 주사 방식에 따라 FIFO를 결정하는 로직을 추가하였다. 즉, 단일주사(single scan or non-split mode) 방식 혹은 이중주사(dual scan or split mode) 방식에 따라 데이터를 읽어올 FIFO을 결정한다. FIFO 블록은 2개의 작은 블록인 FIFO_H, FIFO_L로 구성하였으며, 각각의 블록에 인에이블(enable)비트를 두어 선택된 블록에서 data request 신호가 발생하게 설계하였다. DMA는 단일주사 방식(single scan mode)의 경우 FIFO_H의 데이터만을 출력하며, 이중주사 방식(dual scan mode)의 경우 FIFO_H, FIFO_L, FIFO_H, FIFO_L, ……, 이와 같이 순차적으로 FIFO를 선택하여 데이터를 출력한다. 이는 한 프레임을 상하로 나누어 상위 프레임은 FIFO_H의 비디오 데이터가, 하위 프레임은 FIFO_L의 비디오 데이터가 출력됨을 의미한다. 그림 2는 주사 방식에 따른 어드레싱 순서를 나타낸다 [1], [4], [9].

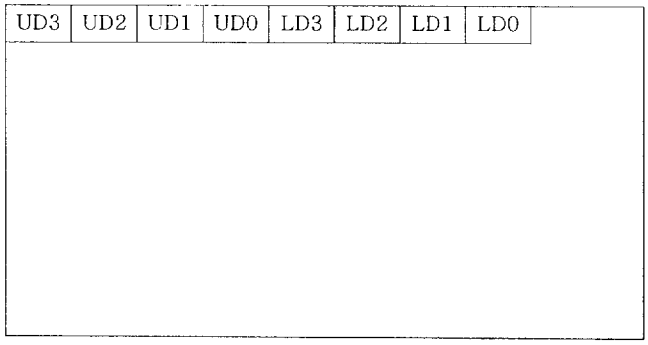
그림 2의 (a)는 한 프레임을 상하로 나누어 상위 어드레스, 하위 어드레스 데이터를 동시에 디스플레이 하는 이중주사(dual scan or split mode) 방식을 나타냈었다. (b)는 상위 어드레스의 비디오 데이터만을 출력하는 단일주사(single scan) 방식 중 4-bit non-split mode을 나타냈었다. (c)는 상위 어드레스, 하위 어드레스 데이터를 순차적으로 출력하는 단일주사(single scan) 방식 중 8-bit non-split mode을 나타냈었다.



(a) 4-bit split display mode



(b) 4-bit non-split display mode



(c) 8-bit non-split display mode

그림 2 주사 방식에 따른 어드레싱 순서

DMA 제어기의 제어로직은 CPU에게 버스 제어권을 요청하는 버스 요청(bus request) 신호, CPU로부터 버스 제어권을 승인 받는 버스 승인(bus grant) 신호, 활성화된 비디오 데이터를 읽어오는 read 신호, 읽어온 비디오 데이터를 FIFO에 저장하는 wirte 신호, 그리고 전송할 데이터가 없음을 알리는 end of transfer 신호등으로 구성한다 [1].

레지스터 부분은 시작주소를 저장하는 32-bit의 DMA 시작주소 레지스터(start address register), 끝주소를 저장하는 32-bit의 DMA 끝주소 레지스터(end address register), 그리고 전송할 데이터 양을 결정하는 32-bit의 전송량 레지스터(page-width register)로 구성한다 [1].

카운터(counter) 부분은 각각 시작주소와 끝주소를 다운(down) 혹은 업(up) 카운팅(counting)하는 주소 카운터(address counter), 데이터 전송량을 다운 카운팅하는 페이지 카운터(page counter)로 구성한다 [1].

그림 3은 DMA의 전체 블록도를 나타낸다. 그림 3에서 도시화된 것처럼 CPU로부터 전송할 데이터의 시작주소, 전송량, 그리고 동작 제어를 위한 제어 레지스터 값을 입력받게 된다. (여기서 VIDVASEHI CNTL REG은 FIFO_H의 제어 신호와 어드레스 값을 가지고 있는 레지스터를 의미하며 VIDBASELO CNTL REG은 FIFO_L의 제어 신호와 어드레스 값을 가지고 있는 레지스터를 의미한다.) 입력된 FIFO_H와 FIFO_L의 어드레스는 각각의 카운터에 로드(load)되어 다운 혹은 업 카운팅한다. 카운팅한 값이 전송량 레지스터의 값과 일치하면 counter_load 신호를 활성화시켜 새로운 주소 값을 입력으로 받아드린다. 단일주사(single scan) 방식의 경우에 DMA는 HI_ADDR(FIFO_H의 주소) 값을 DMA_ADDRESS(실제 DMA 유효주소) 값으로 출력하며, 이중주사(dual scan) 방식의 경우에는 HI/LO_SELECT 신호에 의해 HI_ADDR(FIFO_H의 주소)와 LO_ADDR(FIFO_L의 주소) 값을 번갈아 출력한다 [1], [9].

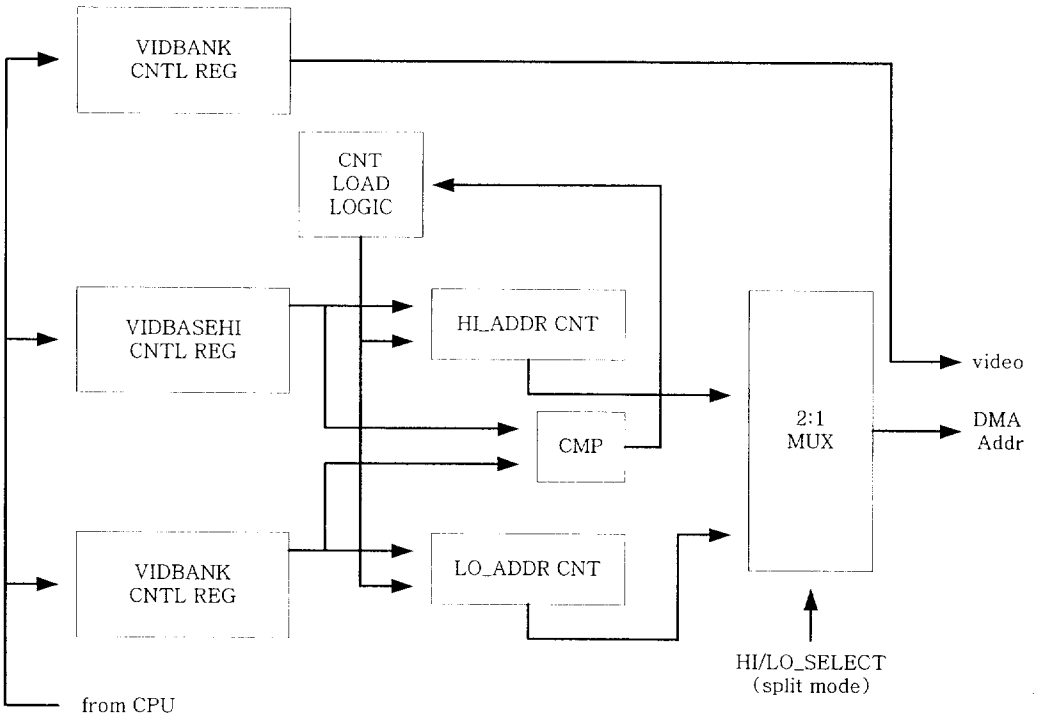


그림 3 DMA 전체 블록도

2.2.2 FIFO 블록

FIFO 블록은 24개의 32-bit 레지스터로 구성하고, 각각 12개의 레지스터로 이루어진 2개의 작은 블록(FIFO_L, FIFO_H)으로 나누었다. 그리고 2개의 블록(FIFO_L, FIFO_H) 또한 각각의 3개의 하위 블록(FA, FB, FC)으로 나누었다. 최초의 data request 신호에 의해 입력된 데이터는 FA 블록에 순차적으로 적재되기 시작한다. FA 블록이 다 차게 되면 FA_full 신호를 활성화시켜 다음 시스템 클럭(system clock)에 동기 하여 FA 블록의 데이터를 FC 블록으

로 적재한다. 최종적으로 FC 블록의 데이터가 FIFO 블록의 출력으로서 나가게 되는데 FC 블록이 데이터를 출력하는 동안 입력되는 데이터는 FA 블록에 적재한다. 마찬가지로 FA 블록이 다 차게되면 FA_full 신호를 활성화시켜 다음 시스템 클럭에 동기 하여 FA 블록의 데이터를 FB 블록으로 적재한다. 만약 FC 블록이 데이터를 출력하고 있는 동안 FA, FB 블록이 다 차게되면 halt 신호를 발생시켜 FC 블록이 빌 때까지 데이터 입력을 중지하도록 DMA에게 요청한다. 그림 4는 FIFO의 데이터 저장 순서를 나타낸다 [9].

FIFO_L, FIFO_H을 각각의 3개의 작은 블록으로 나누어 파이프 라인구조를 이루게 하였는데 이는 연속적인 데이터 read 시 발생하는 지연 시간을 감소시켜 성능 향상을 이루기 위해서이다 [1].

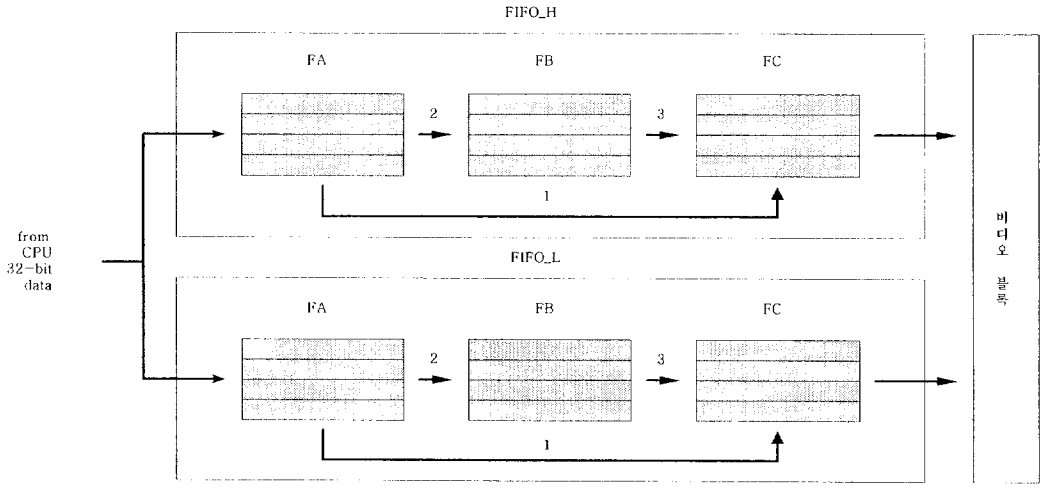


그림 4 FIFO 데이터 적재 순서(1, 2, 3은 적재 순서를 나타낸다)

2.2.3 클럭 제너레이션 블록

클럭 제너레이션(clock generation) 블록은 LCD 판넬(panel)의 동기 클럭을 생성하는 블록이다. LCD 판넬에 비디오 데이터를 디스플레이 하기 위해서는 픽셀 클럭, 라인 클럭, 그리고 프레임 클럭이 필요하다. LCD 판넬 드라이버는 클럭 제너레이션 블록에서 생성된 클럭을 동기 클럭으로 사용하여 LCD 제어기가 출력하는 비디오 데이터를 동기화 시킨다 [3], [9].

클럭 제너레이션 블록에서 생성하는 픽셀 클럭, 라인 클럭, 프레임 클럭은 LCDBANK로부터 입력되는 10-bit의 CLKVAL, 11-bit의 HORZVAL, 11-bit의 HHORZVAL, 10-bit의 LINEVAL, 그리고 11-bit의 LINEBLANK 값이 각각의 클럭 주기를 결정한다.

우선 10-bit의 CLKVAL 값에 의해 픽셀 클럭의 반주기를 결정하는데 최대 (시스템 클럭)/2 클럭에서 최소 (시스템 클럭)/2048 클럭을 생성할 수 있다. 생

성된 픽셀 클럭은 LCD 판넬 상의 1개의 픽셀을 표현할 때 동기 클럭으로 사용한다.

픽셀 클럭의 반주기가 CLKVAL에 의해 결정되면 픽셀 클럭의 rising edge 마다 다운 카운팅하며 HORZVAL 값을 감소시킨다. HORZVAL은 한 라인에 표현될 픽셀 수를 의미하는데, 감소된 HORZVAL 값이 '0' 이 되었을 때 1-클럭의 라인 클럭이 발생한다. 만약, LCDBANK에서 입력된 HHORZVAL값이 '0' 이 아닌 값이 존재하면 라인 클럭은 (HORZVAL-HHORZVAL)값만큼 픽셀 클럭의 수가 감소된 후 라인 클럭이 발생한다. LINEBLANK 값은 라인 클럭을 지연시키는 역할을 하는데 지연기간은 LINEBLANK의 값이 결정한다. 즉, 비디오 블록에서 출력된 데이터가 판넬에 디스플레이 될 때 LINEBLANK 값에 의해 공백으로 표현되어 라인 클럭이 지연된 것처럼 보이게 한다.

프레임 클럭은 LINEVAL 값이 주기를 결정하는데 라인 클럭의 rising Edge 마다 LINEVAL 값을 다운 카운팅하여 LINEVAL 값이 '0' 이 되었을 때 1-클럭의 프레임 클럭이 발생한다.

2.2.4 LCDBANK 블록

LCDBANK 블록은 4개의 컨트롤 레지스터, 2개의 프레임 버퍼 레지스터 그리고 3개의 R, G, B LookUp Table 레지스터로 구성하였다 [9]. 그 각각의 정의는 다음과 같다.

4개의 제어 레지스터(LCD control register 1, 2, 3 4)는 다음과 같이 정의하였다. 그림 5, 6, 7, 9는 제어 레지스터를 나타낸다.

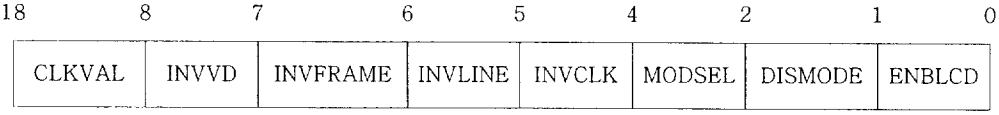


그림 5 LCD Control Register 1

ENBLCD : LCD 제어기를 enable 시키는 비트이다.

⇒ 0 = disable lcd controller

1 = enable lcd controller

DISMODE : LCD 제어기의 주사방식(display mode)을 선택한다. 즉, 이중주사(dual scan)방식 또는 단일주사(single scan)방식을 선택하는 비트이다.

⇒ 00 = 4-bit dual scan display mode

01 = 4-bit single scan display mode

10 = 8-bit single scan display mode

11 = reserved(사용하지 않는다)

MODSEL : LCD의 칼라를 선택하는 비트이다. 모노, 2-bit 그레이, 4-bit 그레이, 8-bit 칼라 4가지 칼라를 지원한다.

⇒ 00 = 모노 모드

01 = 2-bit 그레이 모드

10 = 4-bit 그레이 모드

11 = 8-bit 칼라 모드

INVCLK : 픽셀 클럭의 극성을 선택하는 비트이다. 1인 경우 픽셀 클럭을 반전하여 출력한다.

⇒ 0 = 정상 상태 출력

1 = 반전 상태 출력

INVLINE : 라인 클럭의 극성을 선택하는 비트이다. 1인 경우 라인 클럭을 반전하여 출력한다.

⇒ 0 = 정상 상태 출력

1 = 반전 상태 출력

INVFRAME : 프레임 클럭의 극성을 선택하는 비트이다. 1인 경우 프렘임 클럭을 반전하여 출력한다.

⇒ 0 = 정상 상태 출력

1 = 반전 상태 출력

INVVD : LCD 판넬로 출력되는 비디오 데이터 값의 극성을 선택하는 비트이다. 1인 경우 비디오 데이터 값을 반전하여 출력한다.

⇒ 0 = 정상 상태 출력

1 = 반전 상태 출력

CLKVAL : 11-bit로서 픽셀 클럭의 반주기를 결정하는 비트이다. 최대 (시스템 클럭)/2 클럭에서 최소 (시스템 클럭)/2048 클럭을 발생한다.

그림 6 LCD Control Register 2

LINEBLANK : LCD 판넬에 비디오 데이터가 표현될 때 한 라인에 공백으로 표현될 픽셀의 수를 결정한다.

HORZVAL : LCD 판넬에 표현될 한 라인의 픽셀 수를 결정한다. 최대 2048수의 픽셀이 표현 가능하다.

LINEVAL : LCD 판넬에 표현될 라인 수를 결정한다. 최대 1024수의 라인이 표현 가능하다.

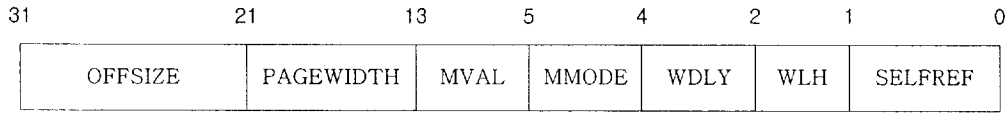


그림 7 LCD Control Register 3

SELFREF : LCD self refresh mode enable bit 이다. enable 될 경우 LCD 판넬로 입력되는 픽셀 클럭이 발생하지 않는다. 즉, LCD 판넬 구동을 중지하게 한다.

⇒ 0 = LCD self refresh mode disable

1 = LCD self refresh mode enable

WLH : 시스템 클럭을 기준으로 라인 클럭의 high level 기간을 결정하는 비트이다. 즉, CLKVAL 값에 의해 결정된 픽셀 클럭이 HORZVAL의 값만큼 발생하면 라인 클럭이 발생하는데 이 라인 클럭의 high level 유지 기간을 결정하는 비트이다.

⇒ 00 = 4 클럭

01 = 8 클럭

10 = 12 클럭

11 = 16 클럭

WDLY : 시스템 클럭을 기준으로 픽셀 클럭과 라인 클럭 사이의 지연 기간을 결정하는 비트이다. 일반적으로 픽셀 클럭이 HORZVAL의 값만큼 발생하면 다음 시스템 클럭을 기준으로 라인 클럭이 발생하는데 이 2-bit가 활성화되면 해당하는 시스템 클럭 수만큼의 지연이 발생한다. 그림 8은 WLH와 WDLY에 의해 발생하는 라인 클럭의 high level 유지 기간과 지연 관계를 나타낸다.

⇒ 00 = 4 클럭

01 = 8 클럭

10 = 12 클럭

11 = 16 클럭

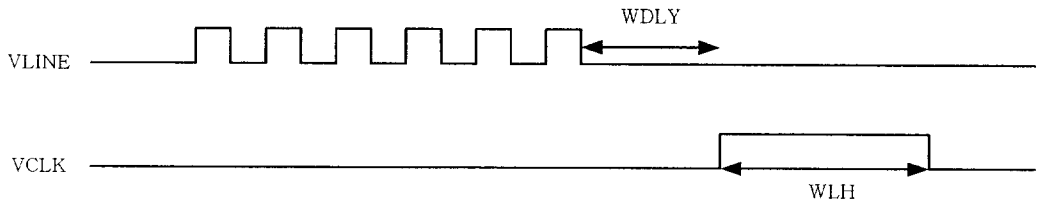


그림 8 라인 클럭의 high level 유지 기간 및 지연 관계

PAGEWIDTH : DMA 데이터 전송 시 전송할 데이터 양을 결정하는 비트이다. 최대 512 프레임까지 데이터 전송이 가능하다.

그림 9 LCD Control Register 4

HHORZVAL : hidden horizontal size 비트이다. (HORZVAL-HHORZVAL) 값만큼 픽셀 클럭의 수가 감소한다.

HACTIVE : enable hidden horizontal 비트이다.

2개의 프레임 버퍼 레지스터의 정의는 다음과 같다. 그림 10, 11는 프레임 버퍼를 나타낸다.

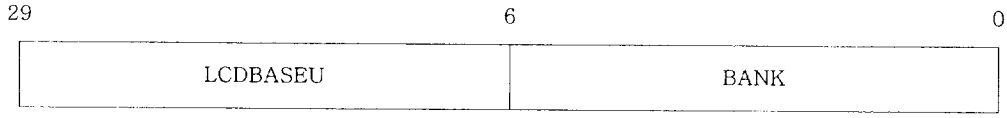


그림 10 Frame Buffer Register 1

BANK : 메모리 장치로부터 비디오 데이터를 FIFO로 적재하기 위해서는 DMA가 메모리 주소 어드레싱(addressing)을 해야한다. 이 때, DMA가 메모리의 유효주소 값을 얻기 위해 사용하는 유효주소의 상위 7-bit에 해당한다. 이는 4GB(32-bit 주소 어드레싱을 함으로)에 해당되는 메모리 영역을 128블록으로 나누어 특정 영역을 비디오 메모리로 할당 할 수 있게 한다.

LCDBASEU : DMA의 32-bit 유효주소비트 중 하위 23비트의 값으로써 비디오 데이터 전송이 이루어질 시작주소를 나타낸다. BANK의 7비트 값을 상위 값으로 하여 32비트 유효주소를 생성한다.

그림 11 Frame Buffer Register 2

LCDBASEL : DMA의 32-bit 유효주소비트 중 하위 23비트의 값으로써 전송될 비디오 데이터의 끝주소를 나타낸다. 마찬가지로 BANK의 7비트 값을 상위 값으로 하여 32비트 유효주소를 생성한다.

3개 R, G, B LUT(Lookup Table)은 원하는 색상을 표현하기 위해 프로그램어(programmer)의해 수정될 수 있다. 초기 값은 '0'으로 하였다. 그림 12는 R, G, B LUT 레지스터를 나타낸다.

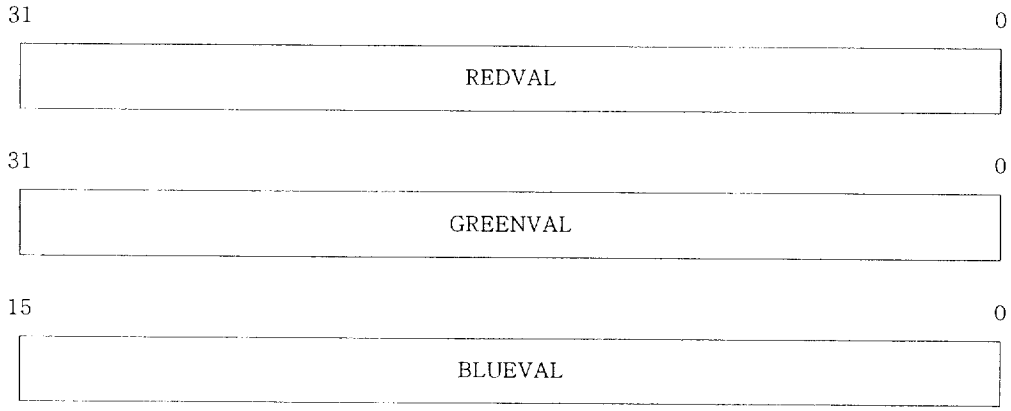


그림 12 R, G, B LUT Register

2.2.5 비디오 블록

비디오 블록은 칼라 모드에 따라 입력된 32비트의 데이터 중 특정 비트를 선택, 쉬프트, 재배열하는 MUX_array 블록, 쉬프트된 비트 값을 선택비트로 사용하여 pre-dither 값을 얻는 LUT(LookUp Table) 블록, pre-dither 값에 의해 dither 레지스터의 28-bit 중 1-bit를 선택하는 dither 블록으로 구성한다. 칼라 모드에 따른 데이터 처리과정은 다음과 같다. 모노 모드의 경우 1-bit가 1 픽셀을 표현함으로 FIFO 블록으로부터 입력된 비디오 데이터 32-bit 값을 MUX_array 블록을 통해 1-bit 쉬프트 시켜 LCD 판넬로 출력한다. 즉, 모노 모드의 경우는 메모리 장치로부터 입력된 데이터 자체가 실제적인 비디오 데이터 값을 의미한다. 2-bit 그레이 모드의 경우 FIFO 블록으로부터 입력된 비

디오 데이터 32-bit 값을 MUX_array 블록을 통해 2-bit 쉬프트 시킨다. 쉬프트된 2-bit 값은 2-bit gray scale LUT 거쳐 16개의 dither 레지스터 중 1개의 dither 레지스터를 선택하는 4-bit의 인덱스(index) 값을 얻게 된다. 인덱스 비트에 의해 선택한 dither 레지스터는 dither line counter 값을 선택비트로 하여 28-bit의 dither 레지스터 중 1-bit를 LCD 판넬로 출력한다. 8-bit 칼라 모드는 FIFO 블록에서 입력된 32-bit 비디오 데이터 값을 MUX_array 블록을 통해 각각 3-bit(R), 3-bit(G), 2-bit(B)씩 쉬프트 시킨다. 쉬프트된 비트들은 각각의 LUT을 거쳐 dither 블록으로 입력된다. 이후의 과정은 동일하다 [2], [9]. 그림 13은 비디오 블록을 나타낸다.

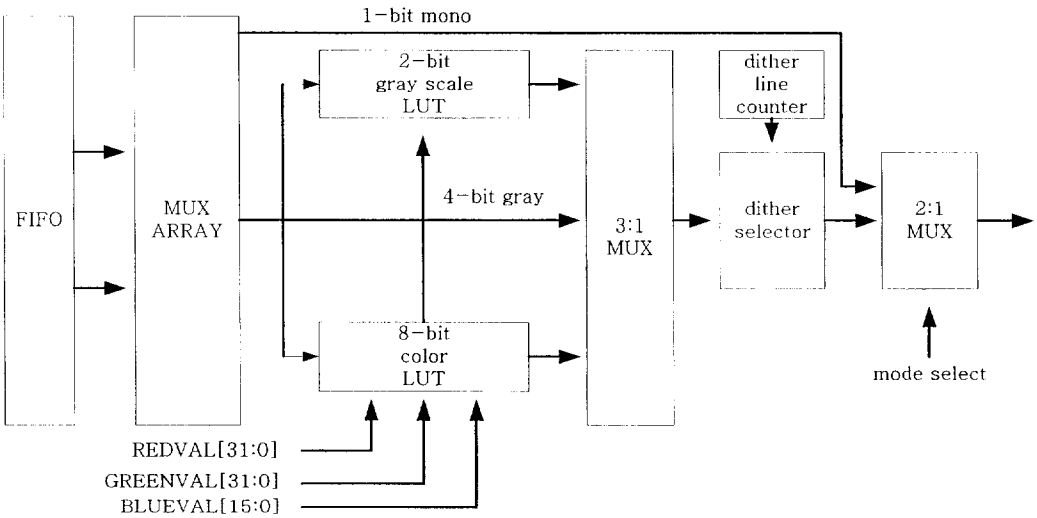


그림 13 비디오 블록의 구조

2.2.6 Lookup Table

32-bit의 데이터가 메모리 장치로부터 FIFO에 적재된 후 활성화된 data_load 신호에 의해 비디오 블록으로 출력되면 비디오 블록의 MUX_array 블록을 통해 칼라 모드에 따라 각각의 LUT에 의해 인덱스 비트로 확장한다. 그림 14는 2-bit 그레이 모드의 LUT을 나타낸다. MUX_array 블록을 지나 2-bit 쉬프트된 데이터가 Bule LUT을 통해 4-bit의 인덱스 비트로 확장한다. 그림 15는 8-bit 칼라 모드의 color lookup table을 나타낸다. MUX_arry 블록을 지나 쉬프트된 각각의 R(3-bit), G(3-bit), B(2-bit) 값들이 R, G, B LUT를 통해 각각 4-bit의 인덱스 비트로 확장한다 [9].

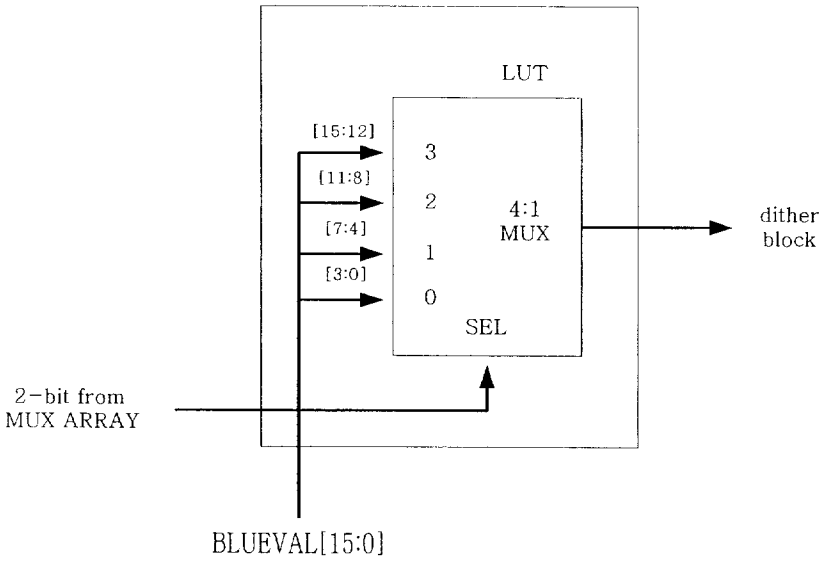


그림 14 2-bit 그레이 모드의 lookup table

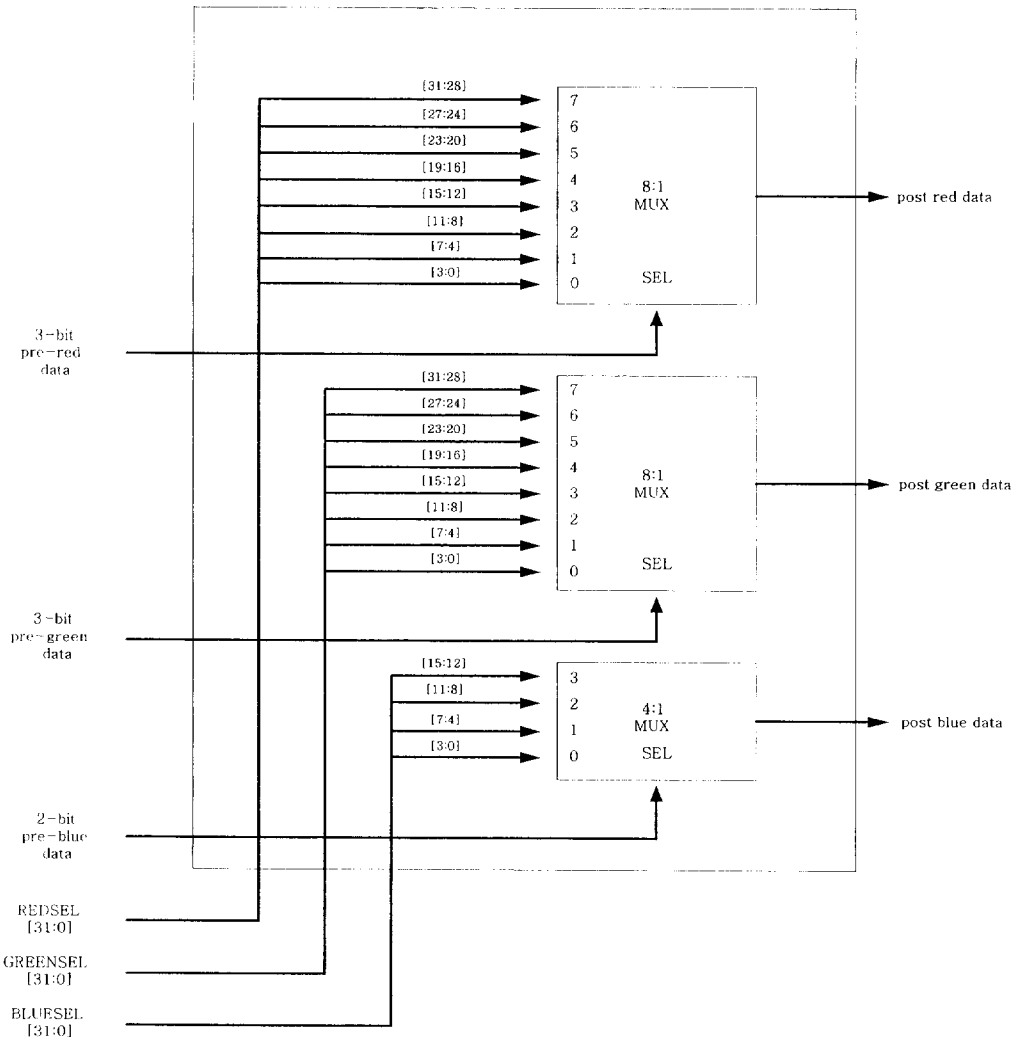


그림 15 8-bit 칼라 모드의 color lookup table

2.2.7 Dithering Logic 블록

LUT를 통해서 확장된 4-bit의 인덱스 비트들은 dithering logic 블록으로 입력된다. 입력된 인덱스 비트들은 16개의 dithering 레지스터 중 하나를 선택

하는 선택비트로써 사용하며, 선택된 28-bit의 dithering 레지스터는 dither line counter에 의해 1-bit의 최종 비디오 데이터를 출력한다. 여기서 사용된 16개의 dithering 레지스터의 값은 TOSHIBA TMPR3911 RISC(Reduced Instruction Set Computer) 프로세서에서 추천한 값을 사용하였다. 그림 16은 dithering logic 블록의 구조를 나타낸다. Table 1은 TOSHIBA TMPR3911에서 추천한 dithering 레지스터의 값을 나타낸다 [9].

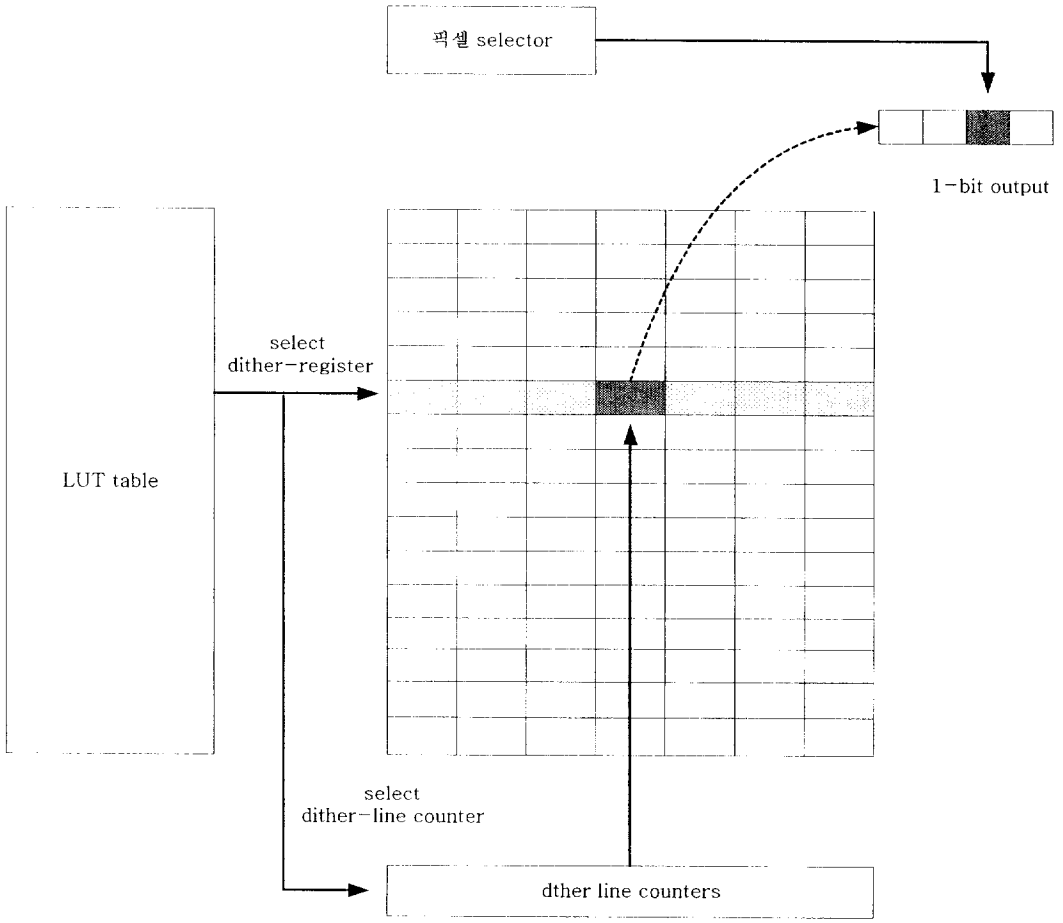


그림 16 Dithering Logic의 구조

Table 1 TOSHIBA TMPR3911 추천 Dithering Register값

Ditherin Pattern Register	
DP15	1111 1111 1111 1111 1111 1111 1111
DP14	0111 1111 1101 1111 1011 1111 1110
DP13	0111 1110 1011 1101 1111 0000 0000
DP12	0111 1101 1011 1110 0000 0000 0000
DP11	1101 0110 1011 0000 0000 0000 0000
DP10	0101 1010 0101 1011 1110 0000 0000
DP9	1011 0101 1010 0101 1010 0101 1110
DP8	1010 0101 1010 0101 0000 0000 0000
DP7	0100 1010 0101 1010 0101 1010 0001
DP6	1010 0101 1010 0100 0001 0000 0000
DP5	0010 1001 0100 0000 0000 0000 0000
DP4	1000 0100 0001 1010 0010 0111 0001
DP3	1000 0010 0100 0001 0000 0000 0000
DP2	1000 0001 0100 0010 0000 0000 0000
DP1	1000 0000 0010 0000 0100 0000 0001
DP0	0000 0000 0000 0000 0000 0000 0000

Ⅲ. 시뮬레이션 및 합성

3.1 DMA 제어기 블록 시뮬레이션

그림 17과 18은 DMA 제어기 블록의 시뮬레이션 파형을 나타낸 것이다. 그림에서 보는 바와 같이 AddrH와 AddrL은 각각 FIFO_H와 FIFO_L에 저장될 데이터의 유효주소 중 하위 23-bit를 나타내고, AddrHL은 주사 방식(single scan, dual scan)에 따라 출력이 되는 DMA 유효주소 중 하위 23-bit 주소를 나타낸다. 단일주사 방식(single scan mode)의 경우 AddrH(FIFO_H)의 유효주소가 AddrHL로 출력이 되고 이중주사 방식(dual scan mode)의 경우 AddrH(FIFO_H), AddrL(FIFO_L), AddrH(FIFO_H), AddrL(FIFO_L), ……, 순으로 번갈아 유효주소을 출력하는 것을 볼 수 있다. 최종 출력으로 나갈 32-bit 유효주소는 LCDBANK에서 입력된 7-bit와 합쳐져서 32-bit의 유효주소를 만든다.

그림 17은 단일주사 방식(single scan mode)을 나타낸다. AddrHL을 통해 FIFO_H에 저장될 데이터의 유효주소을 출력한다.

그림 18은 이중주사 방식(dual scan mode)을 나타낸다. AddrHL을 통해 FIFO_H와 FIFO_L에 저장될 데이터의 유효주소을 번갈아 출력한다.

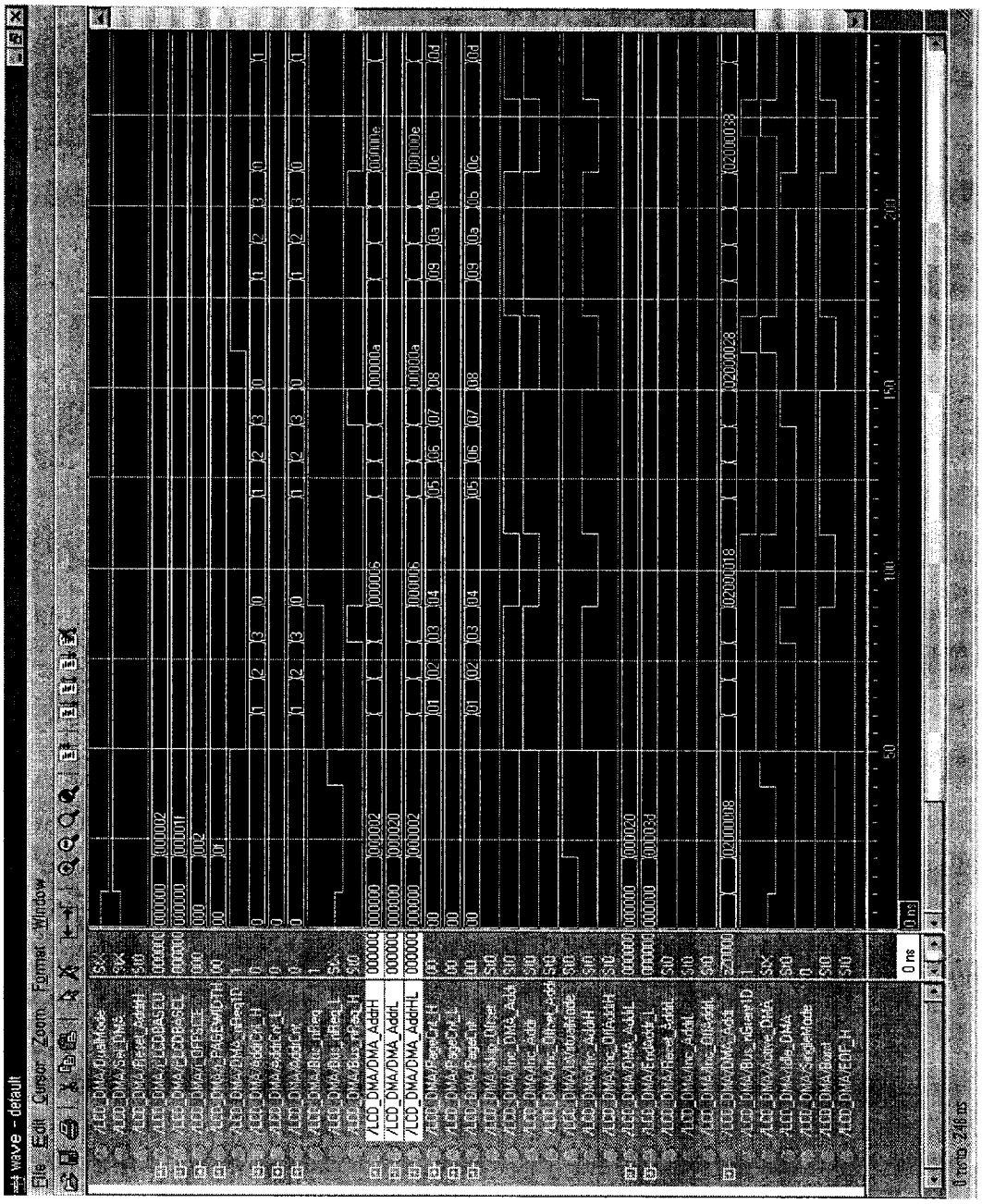


그림 17 DMA 블록 시뮬레이션(single scan)

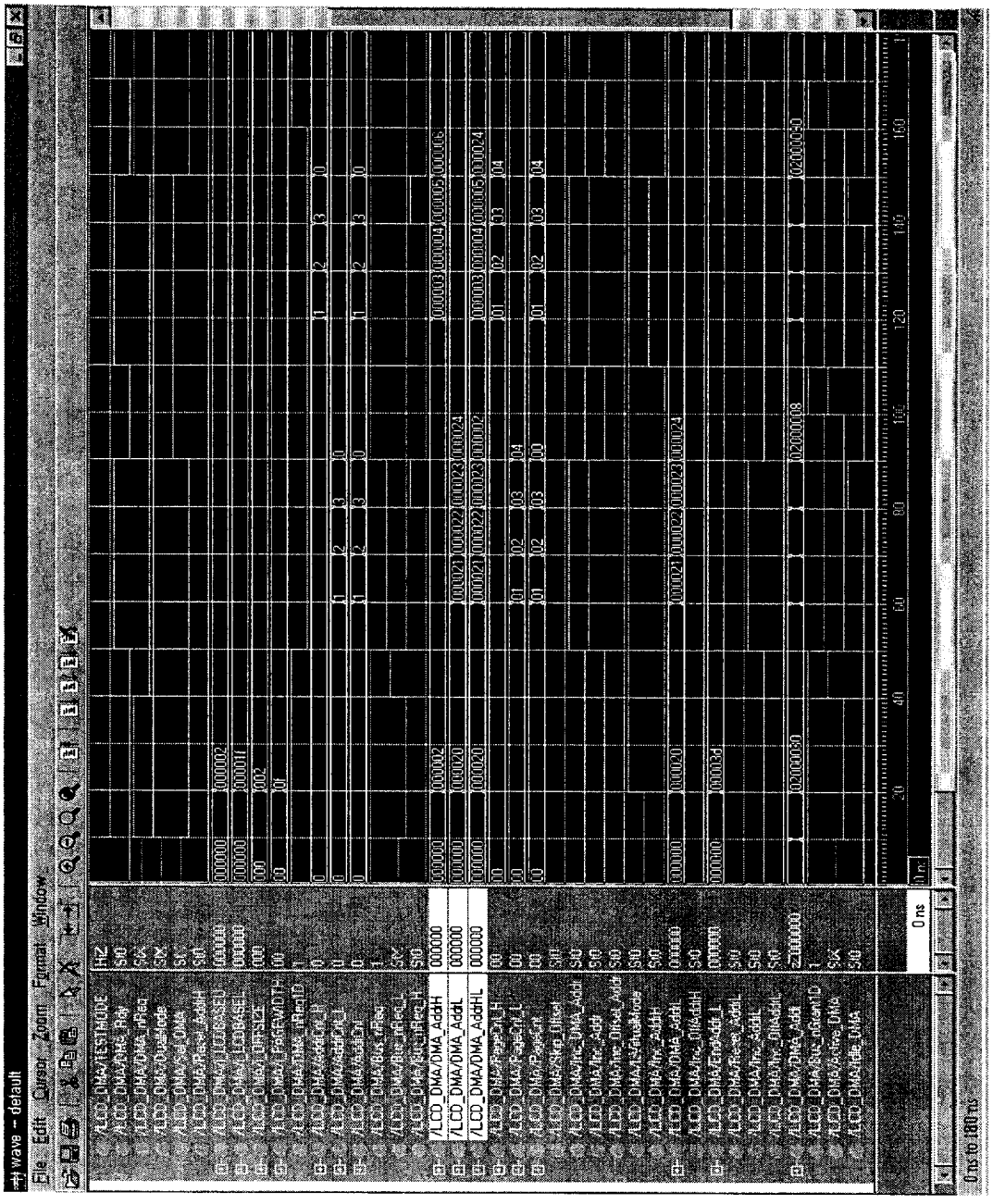


그림 18 DMA 블록 시뮬레이션(dual scan)

3.2 FIFO 블록 시뮬레이션

그림 19는 FIFO 블록을 시뮬레이션한 파형이다. 초기 활성화된 시스템 리셋(nReset)에 의해 FIFO가 빈 상태가 되면 D_REQ(data request) 신호가 발생하여 DMA에게 데이터를 요구한다. 시뮬레이션 파형에서 보는 바와 같이 FA 블록이 차면 FC 블록으로 적재하여 데이터를 출력한다. 마찬가지로 FC 블록이 출력하는 동안 입력되는 데이터가 FA 블록을 채우면 FB 블록으로 적재한다.

3.3 클럭 제너레이션 블록 시뮬레이션

그림 20과 21은 클럭 제너레이션 블록의 시뮬레이션 파형이다. 그림 20은 HHORZVAL, LINEBLANK 값이 존재하는 않는 경우의 픽셀 클럭(VCLK), 라인 클럭(VLINE) 그리고 프레임 클럭(VFRAME)을 나타낸다. 그림에서 보는 바와 같이 HORZVAL 값만큼 픽셀 클럭이 발생한 후 다음 시스템 클럭에 동기 하여 라인 클럭이 발생한다.

그림 21은 HHORZVAL, LINEBLANK 값이 존재하는 경우의 클럭 제너레이션 블록의 시뮬레이션 파형을 나타낸다. 픽셀 클럭은 (HORZVAL-HHORZVAL) 값만큼 감소하여 발생하고 라인 클럭은 LINEBLANK 값만큼 지연되어 발생한다.

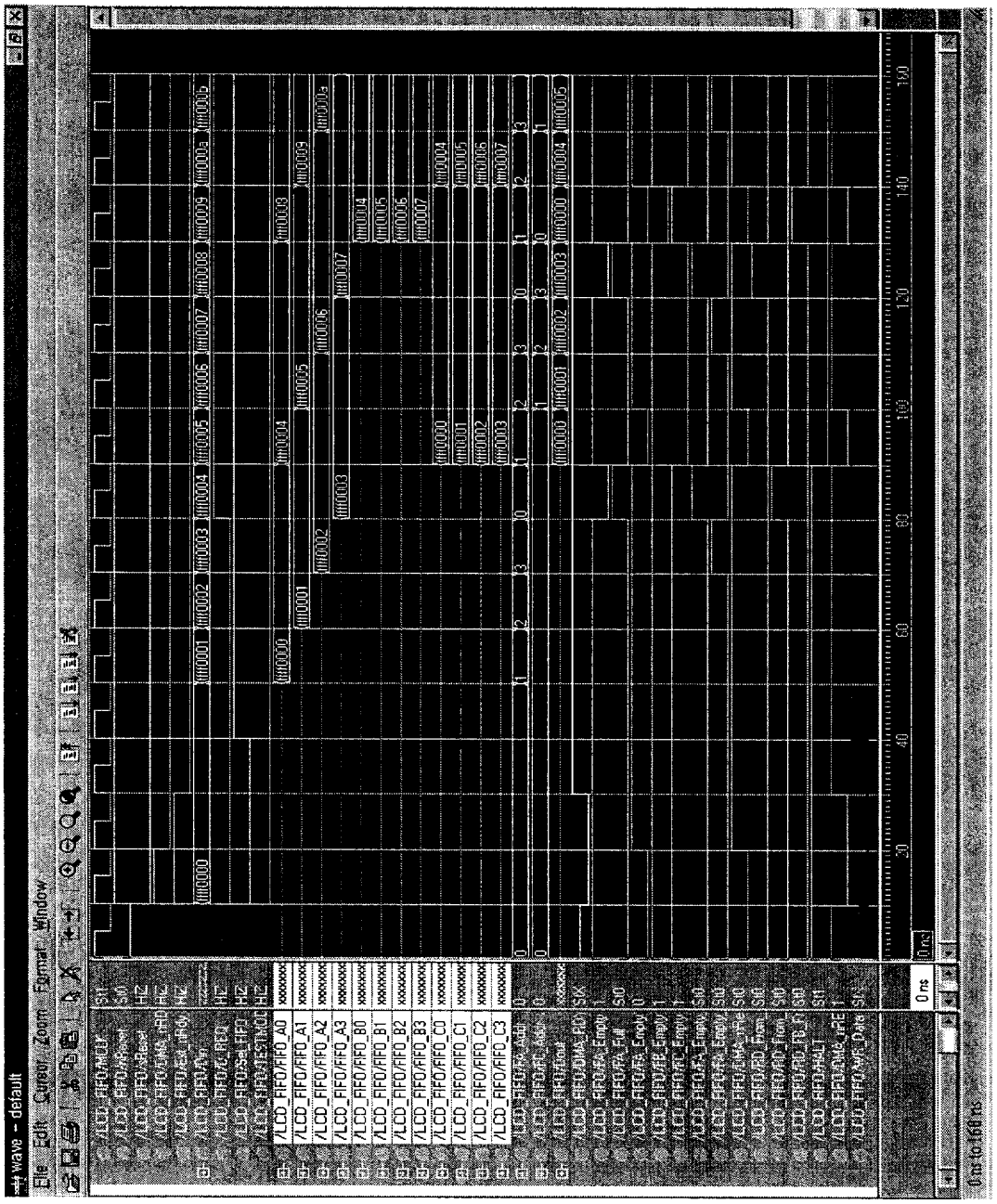


그림 19 FIFO 블록 시뮬레이션

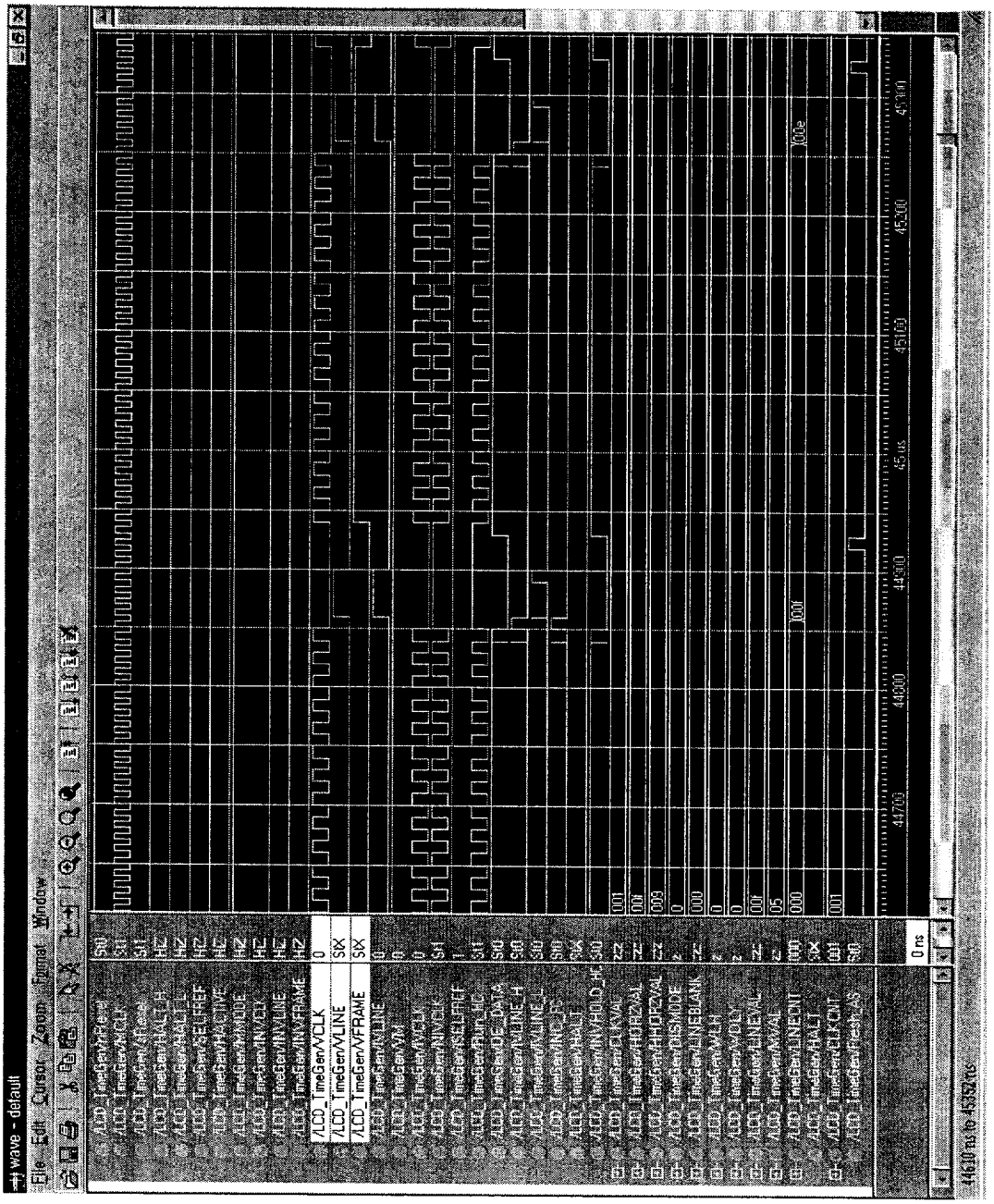


그림 20 클럭 제너레이션 블록 시뮬레이션

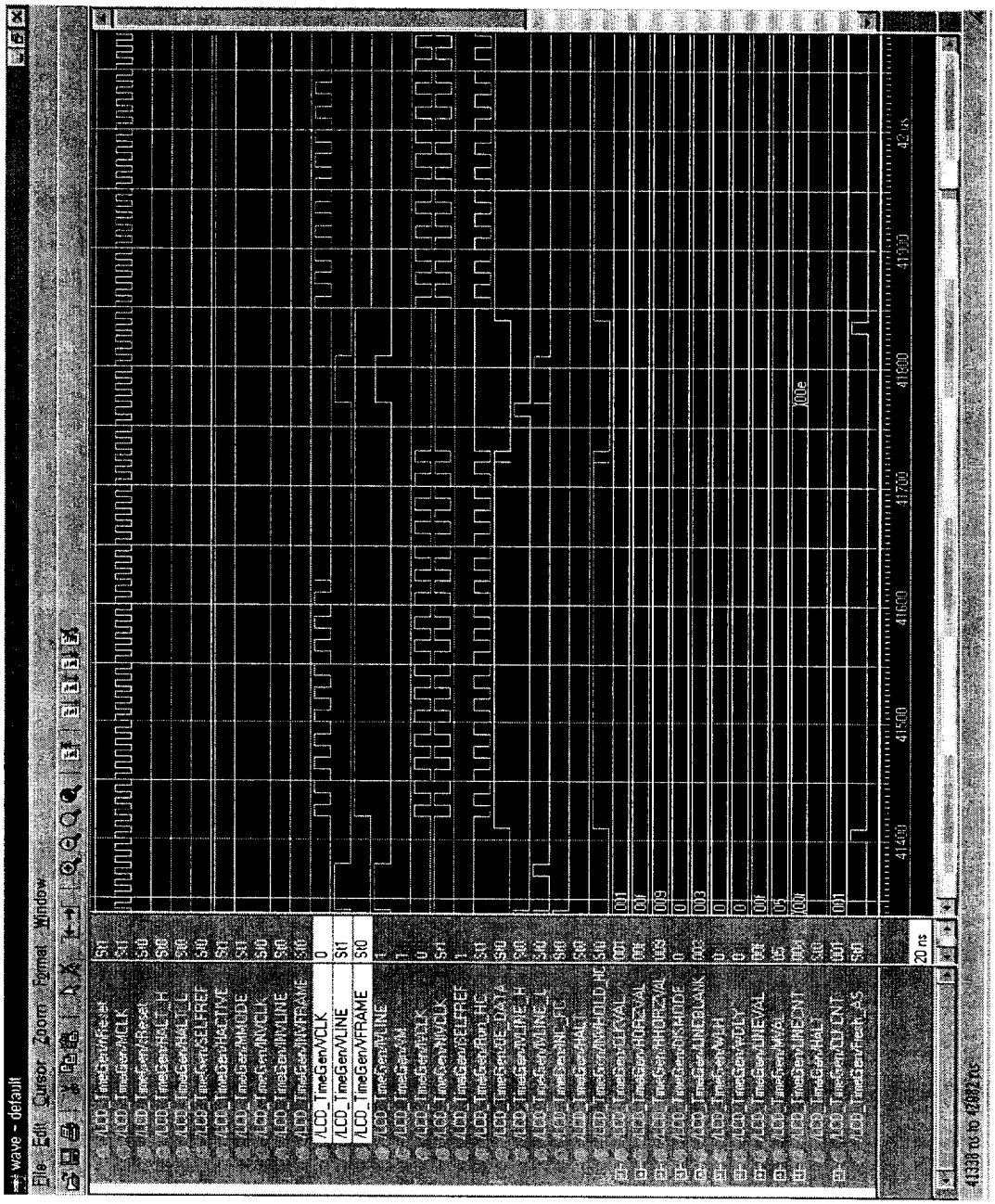


그림 21 클럭 제너레이션 블록 시뮬레이션

3.4 비디오 블록 시뮬레이션

그림 22는 모노 모드의 경우 시뮬레이션 파형이다. 모노 모드는 1-bit가 1 픽셀을 표현함으로 LUT 블록이나 dithering logic을 거치지 않고 LCD 판넬로 출력한다. 시뮬레이션 조건은 각각 FIFO_H와 FIFO_L에 16진수 데이터 f320001, f1200002 값을 입력하고 단일주사 방식 중 8-bit scan mode 일 때를 시뮬레이션 하였다. 여기서 VD가 최종 출력 비디오 데이터이다. 시뮬레이션 파형에서 보는 바와 같이 FIFO_H의 상위 8-bit의 데이터을 순차적으로 출력한다.

그림 23은 2-bit 그레이 모드의 경우 시뮬레이션 파형이다. FIFO로부터 입력된 32-bit 데이터 값이 MUX_array, LUT, dither 블록을 거쳐 8-bit의 비디오 데이터가 출력된다. 시뮬레이션 조건은 모노 모드와 같은 조건으로 하였다. GRAY2_H, GRAY2_L은 쉬프트된 2-bit 그레이 값이다. VD의 출력은 dither 블록을 거쳐 출력된 데이터이다.

그림 24는 4-bit 그레이 모드의 경우 시뮬레이션 파형이다. FIFO로부터 입력된 32-bit 값이 MUX_array 블록을 거쳐 4-bit 쉬프트한다. 그러나, 4-bit 그레이 모드의 경우 쉬프트한 값 자체가 dithering 레지스터의 인덱스 값을 나타낸다. 그러므로 LUT을 거치지 않고 dither 블록으로 입력된다. 시뮬레이션 조건은 동일하며 GRAY4_H, GRAY4_L은 쉬프트된 4-bit 값이다.

그림 25는 8-bit 칼라 모드의 경우 시뮬레이션 파형이다. R값 3-bit, G값 3-bit, B값 2-bit가 각각 쉬프트한 뒤 LUT을 거쳐 16개의 dither 레지스터 중 하나를 선택하여 1-bit 데이터를 출력한다. 시뮬레이션 조건은 동일하다.

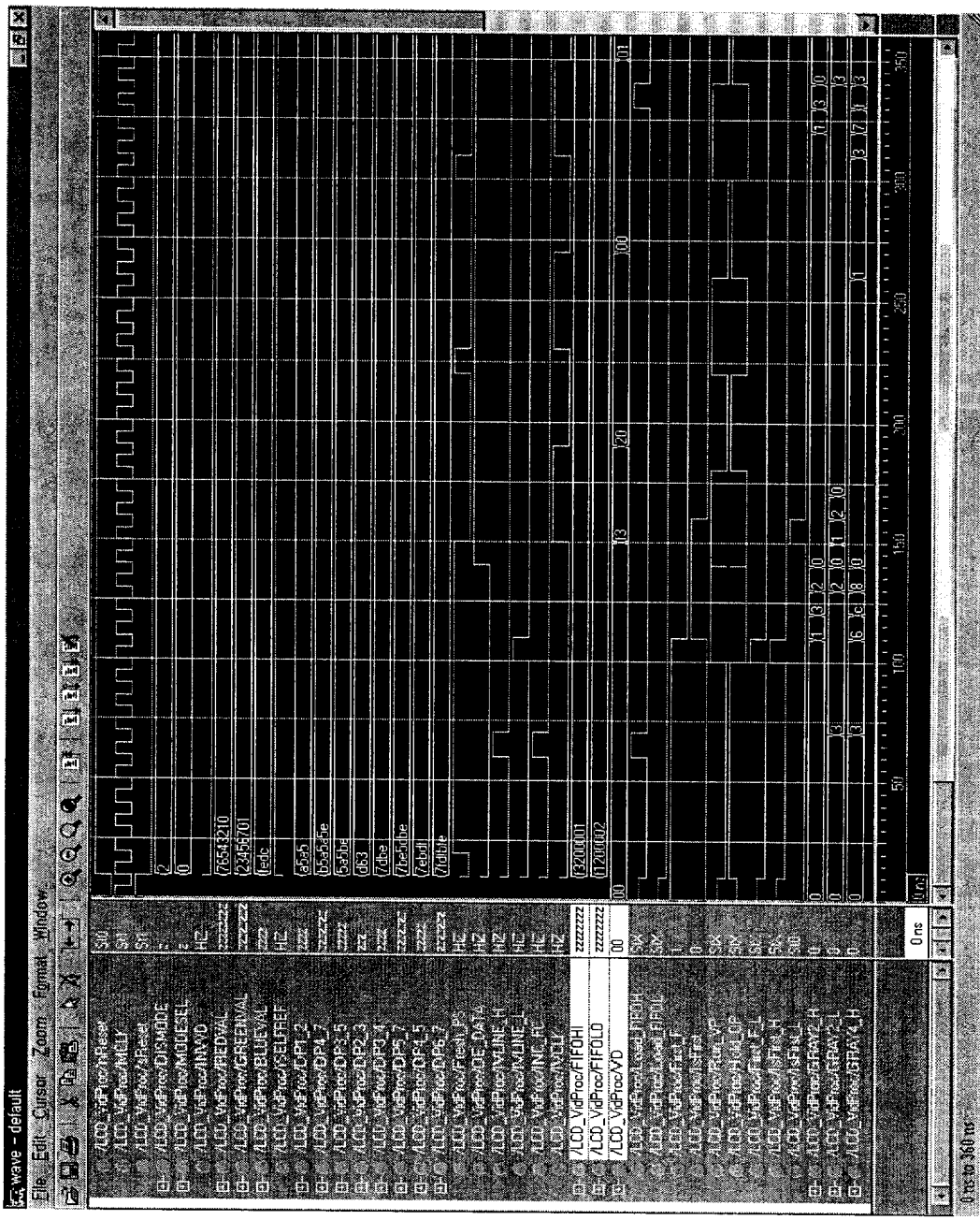


그림 22 비디오 블로 시뮬레이션(모노 모드)

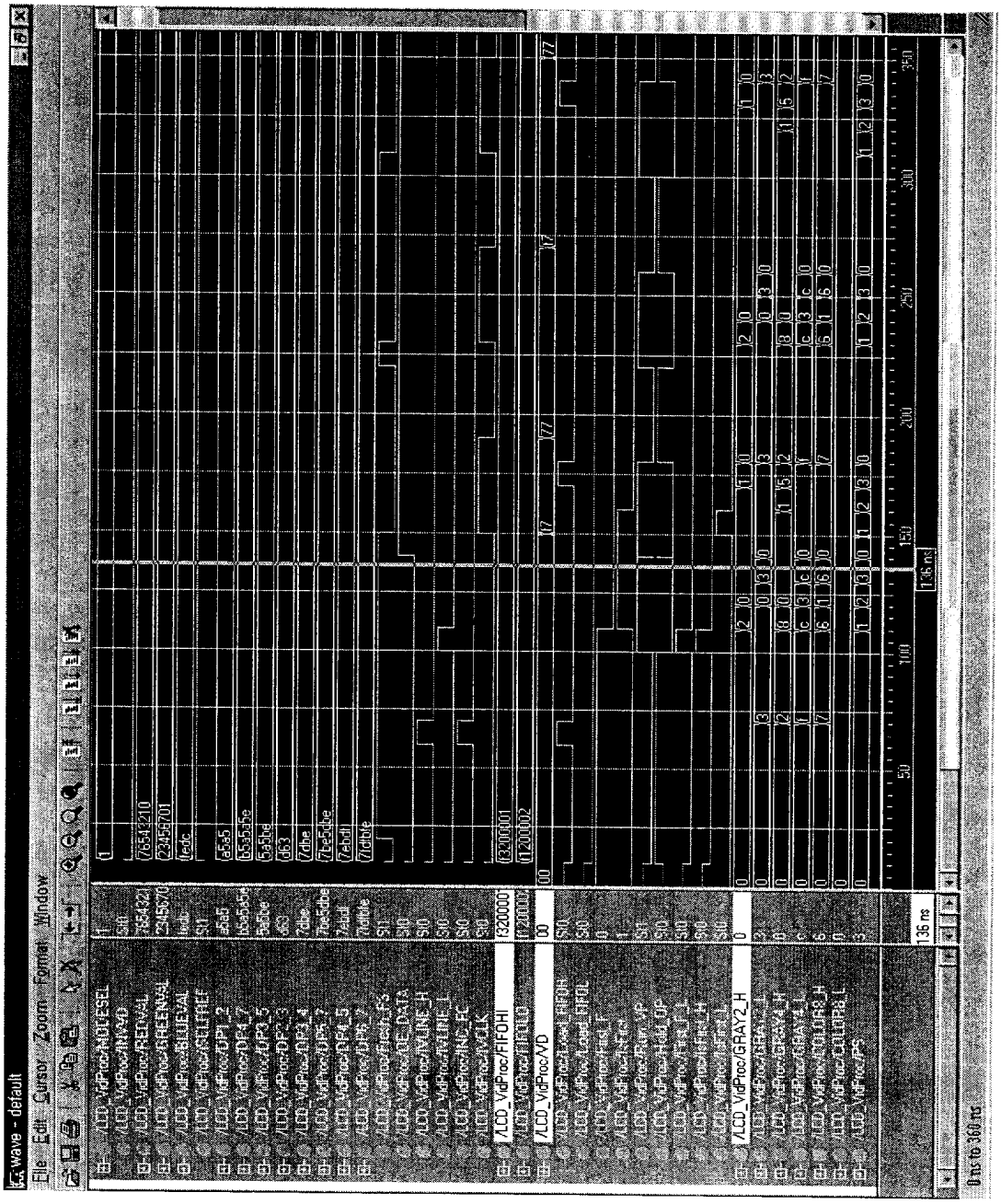


그림 23 비디오 블록 시뮬레이션(2-bit 그레이 모드)

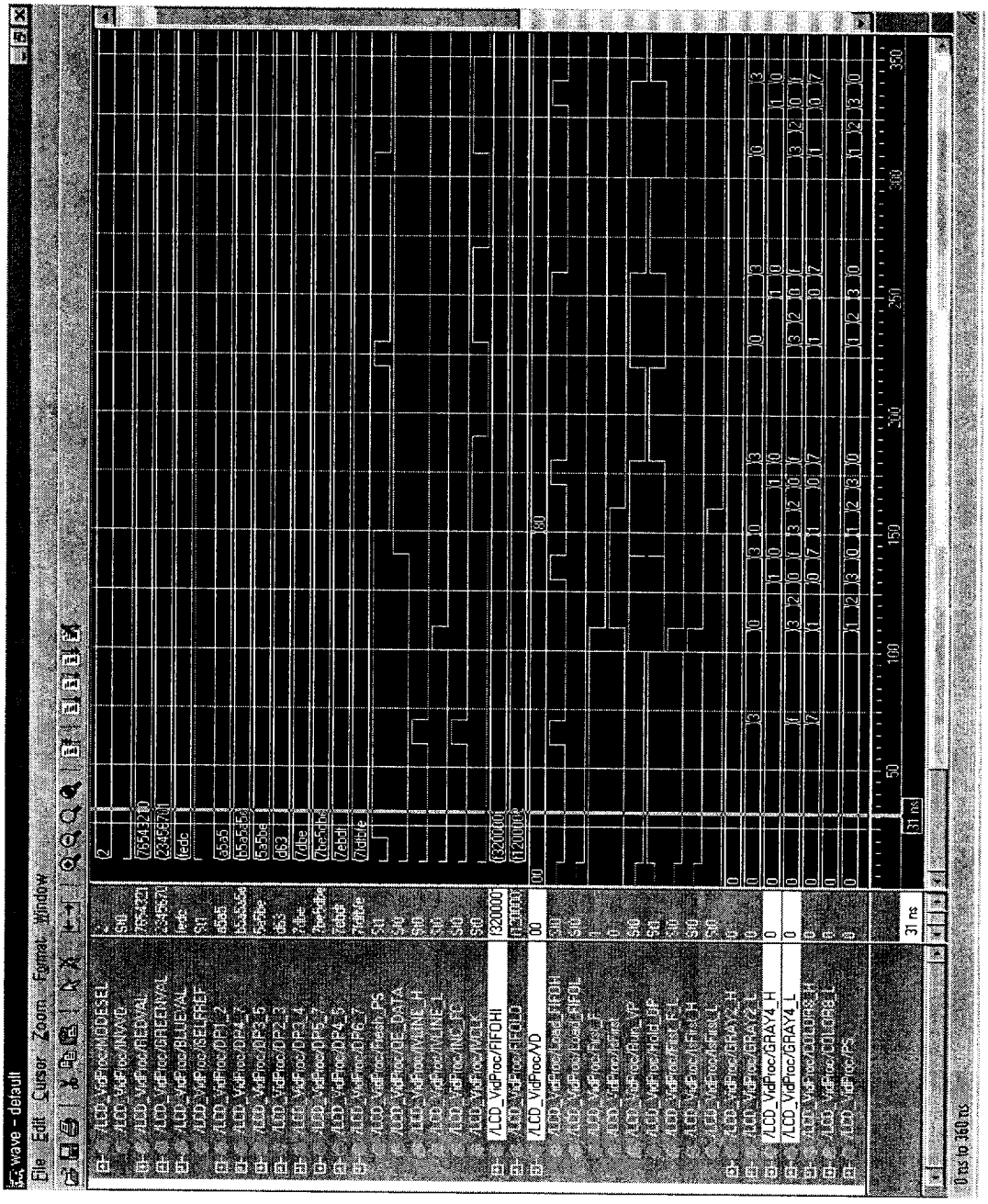


그림 24 비디오 블록 시뮬레이션(4-bit 그레이 모드)

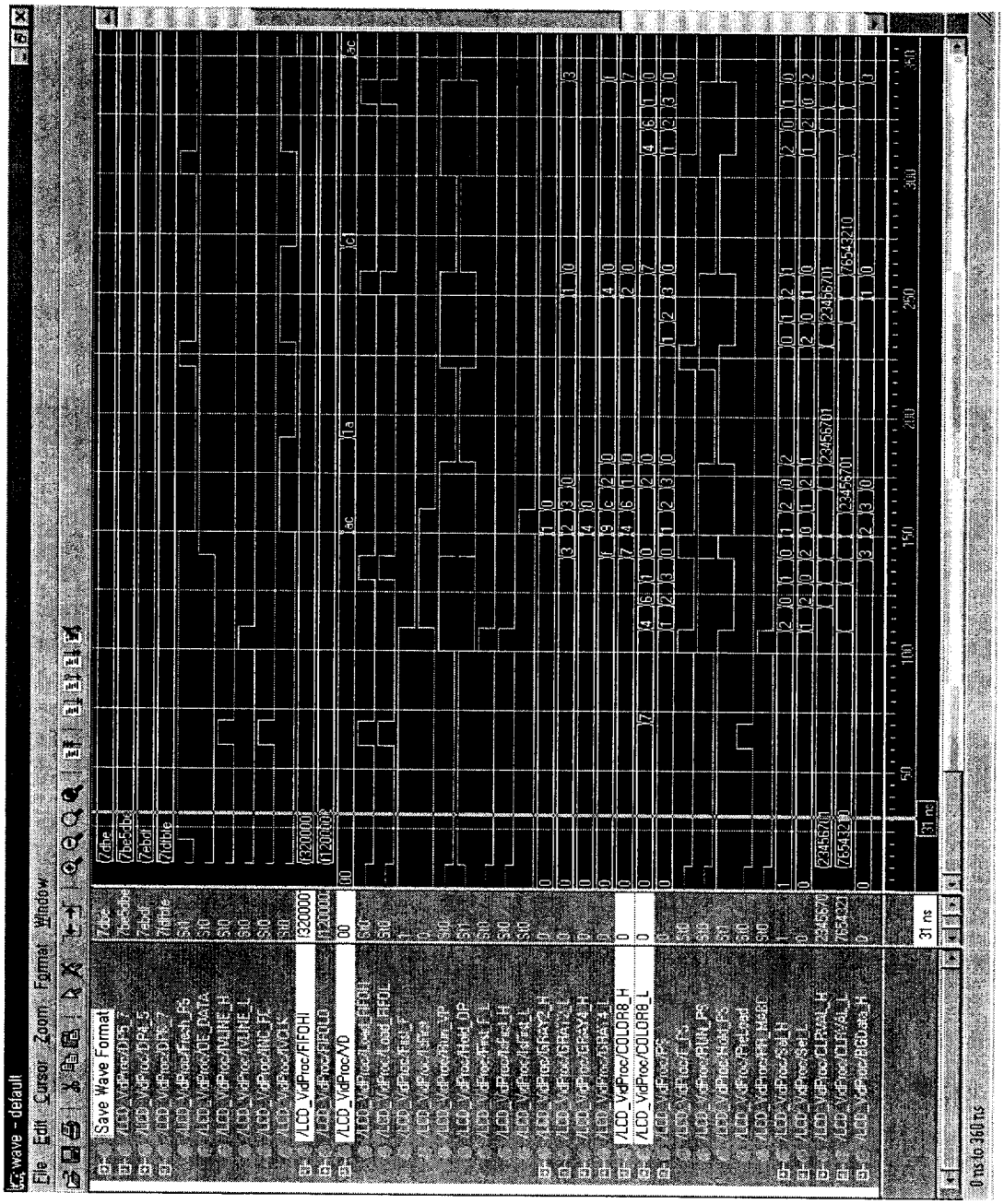


그림 25 비디오 블록 시뮬레이션(8-bit 칼라 모드)

3.5 LCD 제어기 전체 시뮬레이션 및 합성

본 장에서 설계한 LCD 제어기의 각 블록은 Verilog 코드로 설계하였다. 시뮬레이션은 Model Sim 프로그램을 사용하였고, 합성은 Xilinx Foundation 3.1i의 Virtex FPGA 칩 라이브러리로 하였다. 전체 시뮬레이션을 위해 single port block memory와 메모리 제어기를 설계하여 single port block memory에 저장되어 있는 비디오 데이터를 메모리 제어기가 LCD 제어기와 인터페이스 하여 데이터를 전송하도록 설계하였다. single port block memory은 Xilinx의 core generator을 사용하여 32-Kbyte 용량의 롬(rom)과 같이 초기 값을 가지게 core file을 작성하여 설계하였다. 또한 LCD 제어기는 32-bit 범용 CPU를 기반으로 하여 설계하였기 때문에 그에 필요한 신호 및 메모리와의 인터페이스를 위해 메모리 제어기를 같이 설계하였다. single port block memory의 초기 값은 회상도 200*150의 "VLSI"라는 문자의 16진수 코드 값을 사용하였다. 총 게이트 카운트 수는 30,507 게이트 카운트가 소요되었으며, 블록별로 DMA 제어기가 4,103 게이트, FIFO가 4,760 게이트, LCDBANK가 4,584 게이트, 클럭 제너레이션 블록이 2,598 게이트가 소요되었다. 최대 동작 클럭은 21-㎒ 였다 [6], [7], [8], [10]. 그림 26은 LCD 제어기의 전체 시뮬레이션 파형을 나타낸다.

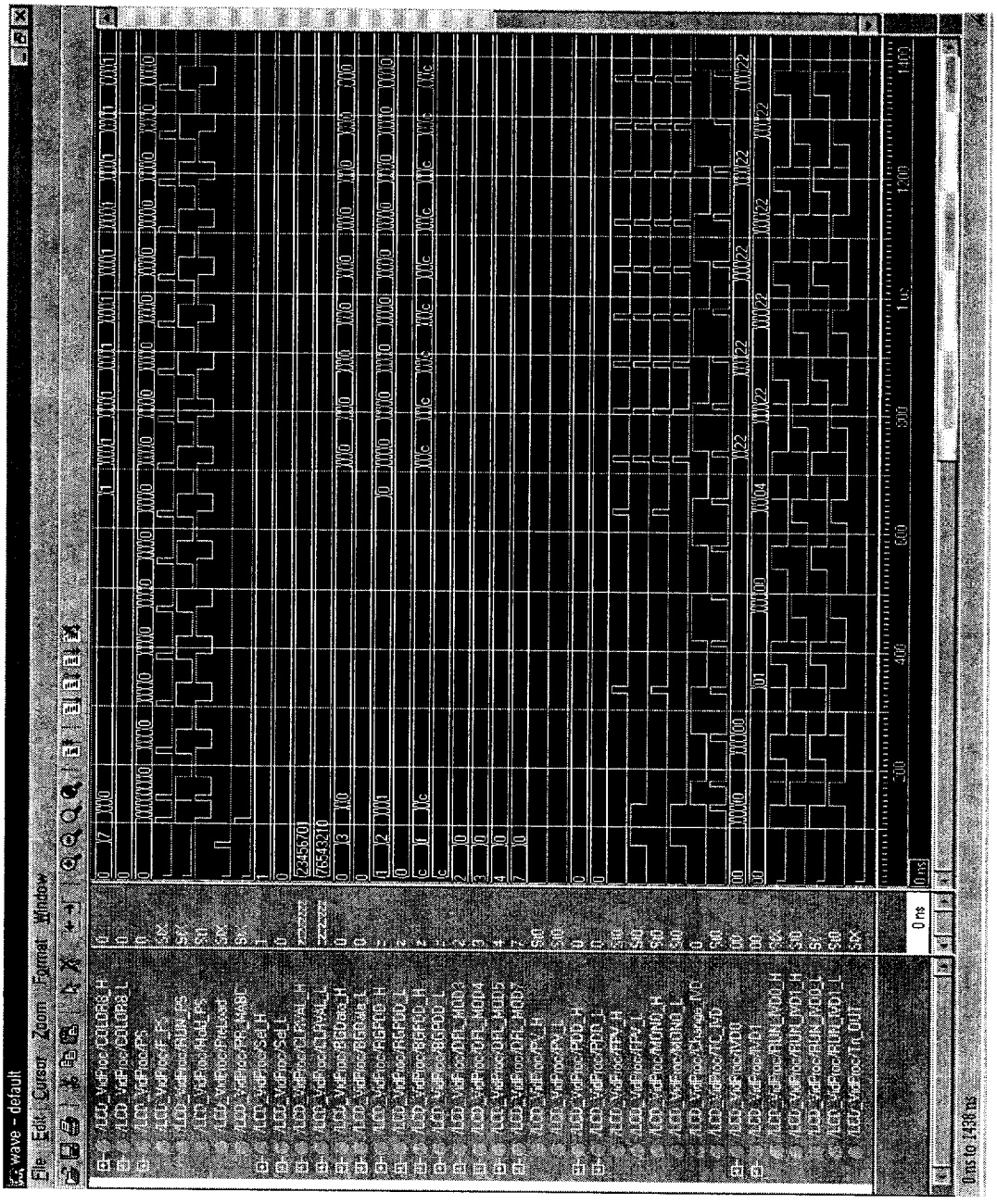


그림 26 LCD 제어기 전체 시뮬레이션

Ⅳ. 결 론

본 논문은 4가지 칼라 모드(모노 모드, 2-bit 그레이 모드, 4-bit 그레이 모드, 8-bit 칼라모드)와 2가지 주사(single scan, dual scan) 방식을 지원하는 LCD 제어기를 설계하였다.

전체 시스템의 구조는 DMA 제어기가 비디오 데이터를 메모리 장치로부터 FIFO로 저장하고 주사방식에 따라 선택된 FIFO의 데이터가 비디오 블록으로 전송되어 LUT 블록과 dither 블록을 거쳐 8-bit의 비디오 데이터를 출력하는 구조로 설계하였다. 실제 설계는 Verilog로 작성하였으며, Xilinx Foundation 3.1i의 Virtex FPGA 칩 라이브러리로 합성하여 전체 시뮬레이션을 검증하였다. 총 게이트 카운트는 30,507 게이트 카운트가 소요되었고, 최대 21-㎒의 시스템 클럭으로 동작하였다. HDL언어로 작성되어 있어 특수한 제어를 목적으로 다른 외부 디바이스가 추가 또는 수정이 용이하며 쉽게 응용이 가능하리라 본다. 실제 FPGA 칩으로 합성한 결과 래치 회로로 인해 많은 게이트 카운터가 소요되었다. 이를 수정 보안하여 래치 회로의 발생을 최소화시킨다면 하드웨어 면적은 감소하고 동작 속도는 증가될 것으로 생각한다.

참고 문헌

[1] M.Morris Mano, Computer System Architecture 3nd Ed, 희중당, 1994

[2] George Sutty, Steve Blair, 고급 프로그래머를 위한 EGA/VGA, 가남사, 1992

[3] LG, LP104V2 LCD판넬 스펙

[4] TOSHIHISA TSUKADA, TFT & LCD, 북스힐, 2000

[5] 노봉규 외, LCD ENGINEERING, 성안당, 2000

[6] Samir Palnitkar, Verilog HDL, 영한출판사, 2001

[7] Bob Zeilman, Verilog Designer's Library, Prentice Hall PTR, 1999

[8] Michael D. Ciletti, Modeling Synthesis and Rapid Prototying with The Velilog HDL, Prentice Hall, 1999

[9] Toshiba, TMPR3911 매뉴얼

[10] Douglas J. Smith, HDL Chip Design, Doone Publications, 1996