asserted at the begining of each data block to be transfered. A data block is usually a single sector except when declared otherwise via the Set Multiple Command. An exception occurs on Format Track, Write Sector(s), Write Buffer and Write Long commands and INTRQ will not be asserted at the begining of the first data block to be transferred.

IORDY : is negated to extend the host transfer cycle of any host register read/write access when the drive is not ready to respond to a data transfer request. When not negated, it is in a high impedance state.

다음의 IDE LSB/MSB data register는 16bit inout되는 DATA를 저장하는 regiser이다. read strobe후의 다음의 2개의 8bit regiser에 있는 data를 읽어 오면 원하는 16bit data가 얻어진다.

■ IDE LSB DATA REGISTER : DEh (Byte access SFR) reset=> 00h

7(I/O)	6(I/O)	5(I/O)	4(I/O)	3(I/O)	2(I/O)	1(I/O)	0(I/O)
DATA7	IDATA6	DATA5	DATA4	DATA2	DATA2	DATA1	DATA0

■ IDE MSB DATA REGISTER : DFh (Byte access SFR) reset=> 00h

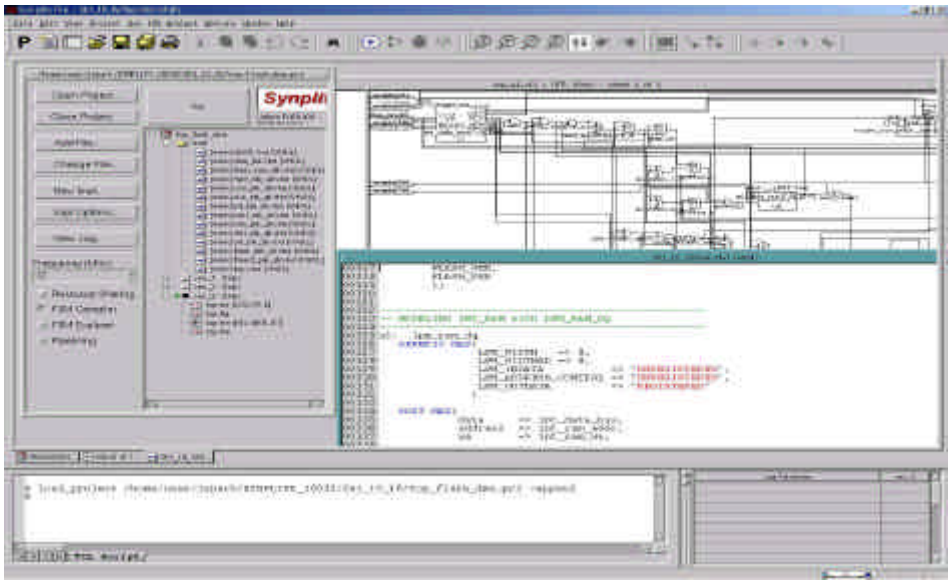
7(I/O)	6(I/O)	5(I/O)	4(I/O)	3(I/O)	2(I/O)	1(I/O)	0(I/O)
DATA15	IDATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8

더 자세한 IDE interface register의 자세한 설명은 IDE(ATA-1) standard rev 1.0의 문서를 참고하면 된다.

Ⅴ. 시뮬레이션을 통한 검증

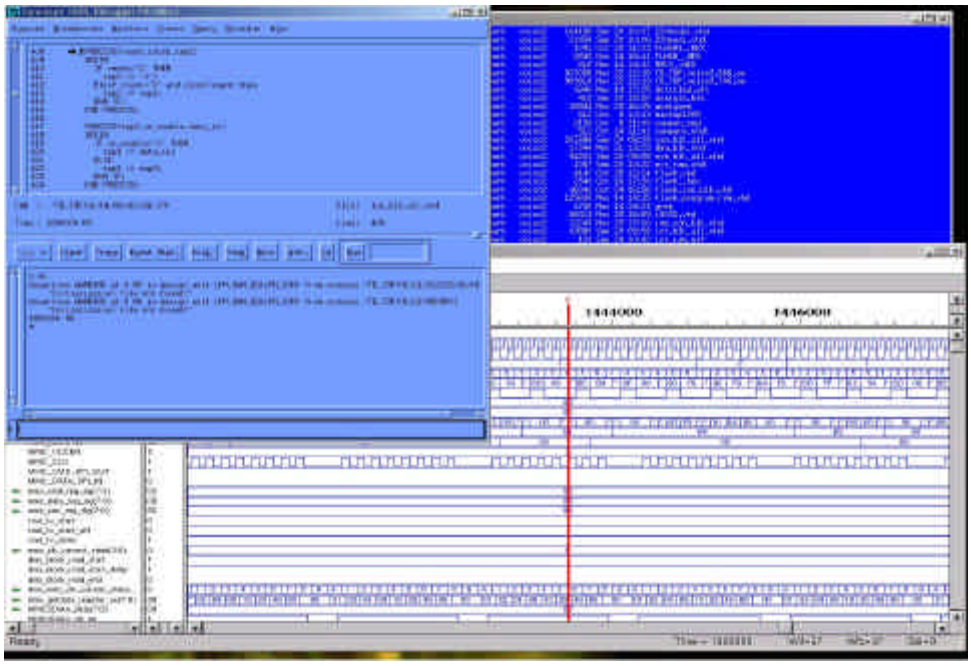
1. Synplify / Synopsys tool을 이용한 simulation환경

설계된 인터페이스용 마이크로프로세서의 VHDL code검증은 Synplify와 Synopsys Design compiler 및 FPGA express에서 이루어 졌다. 기본적인 VHDL구문 check 및 compile은 synplify에서 한번 수행한후 simulaiton은 Synopsys VSS에서 수행하였다.



[그림5-1] Synplify의 HDL 개발환경툴

Synplify의 compiler를 통해 syntex check 및 async동작의 latch를 없애고 모두 flipflop으로 작업바꾸었으며 if 문이나 when문의 don't care처리를 모두 해주었다. synplify에서 수정된 VHDL code를 Synopsys VSS를 이용하여 simulaiton하였다. 다음은 Synopsys VSS의 기본화면을 나타내었다.



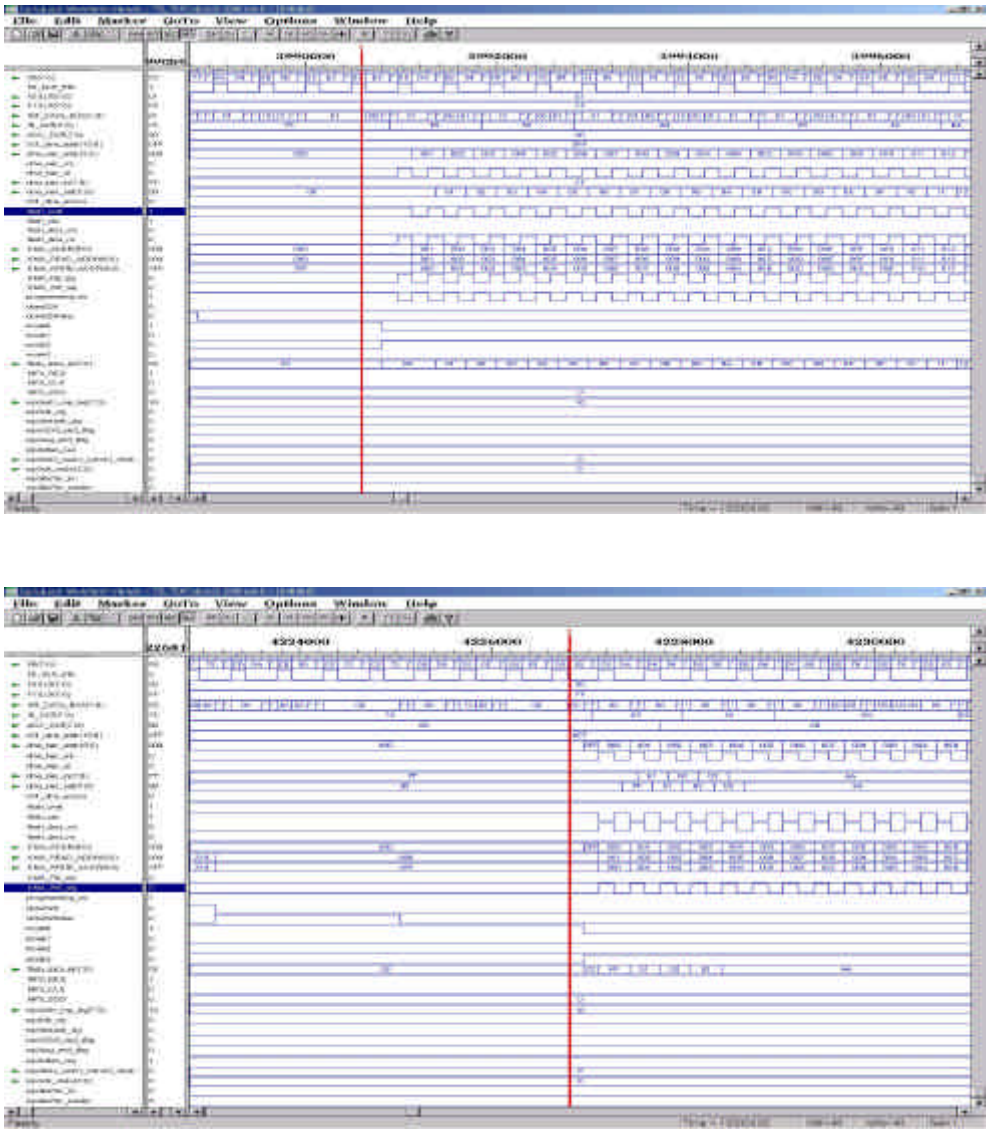
[그림5-2] Synopsys VSS 개발환경툴

2. 설계된 Interface용 마이크로프로세서 기능 블럭의 simulation

2.1 DMA BLOCK

DMA 블럭에서의 simulation의 핵심은 address와 read/write control신호에 따라 data가 제대로 RAM-ARRAY에 read/write되는 것을 보는 것이다. 각 모드별 그리고 각 기능 블럭별로 DMA가 사용될 때 read/write strobe나 address에 따라 data가 read/write되는 것을 보면 된다. [그림4-2] DMA buffer의 read/write timing에 나타난 것처럼 사용된 FPGA의 timing이 고려 되어 simulation을 수행하였다. 주로 각 인터페이스 블록에서 DMA block read명령을 통해 512 byte의 DMA에 data가 read/write되는 timing을 simulation 하였

다. 다음은 FLASH memory를 읽어와서 DMA에 write 그리고 그 반대의 동작 모습을 simulation한 것이다.



[그림5-3] FLASH의 DMA buffer의 read / write simulation

DMA의 simulation에 있어서 embedded RAM array의 parameter를 바꿈으로서 timing의 변동이 왔으나 4장 1절에서 나온 lpm_ram_dq의 generic map에서 3가지 parameter(LMP_INDATA, LPM_OUTDATA,LPM_ADDR_CON)를 UNREGISTER, REGISTER, REGISTER mode로 고정시킨후 [그림4-2]의 timing대로 설계하였다. 그 외 다른 mode에서는 timing이 다르므로 고려하지 않았다.

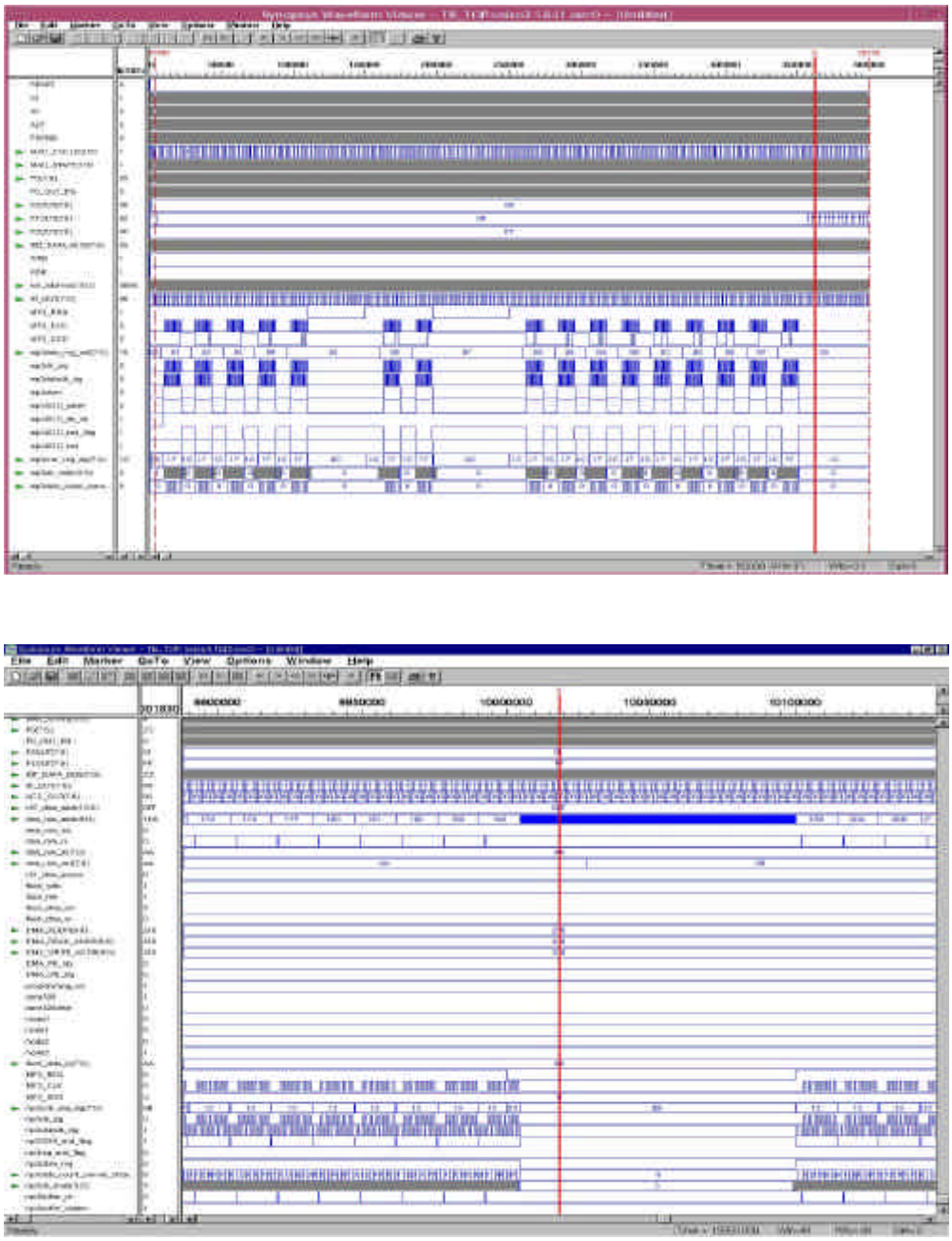
2.2 MP3_CON BLOCK

MP3 control block의 simulation에 있어서 check해야 할 가장 중요한 사항은 DMA를 통한 MP3 stream data가 외부 MP3 decoder chip의 REQEST 신호에 따라 SDO로 출력되는 것을 보는 것이다. simulation은 주로 DMA에 data를 write한후 MP3 control register의 setting을 통해 MP3 stream을 출력하는 것으로 하였다.

실제 MP3 REQ의 timing은 어떻게 input될지 모르기 때문에 MP3REQ신호의 assert여부에 따라 동작하게 된다. 실제 MP3REQ가 negated하게 되었을때의 현재 출력되고 있는 SDO에 대한 8bit 값을 다음 MP3REQ가 assert되었을때 다시 출력시키게 되면 MP3 decoder(MAS3507)에서 한 sample이 빠진 것으로 인식되는 것을 나중에 FPGA 실시간 검증때 밝혀졌다.

따라서 한 byte의 MP3 stream data가 SDO로 출력되고 있을 때 7번째에서 0번째 bit사이에 P3REQ가 negated 되더라도 SDO출력은 계속 유지시켜주며 그 위치가 어디라도 상관없다. 따라서 다음 MP3REQ가 asserted되었을 때 그 다음의 MP3 stream byte를 출력시켜주면 된다는 것을 명심하자.

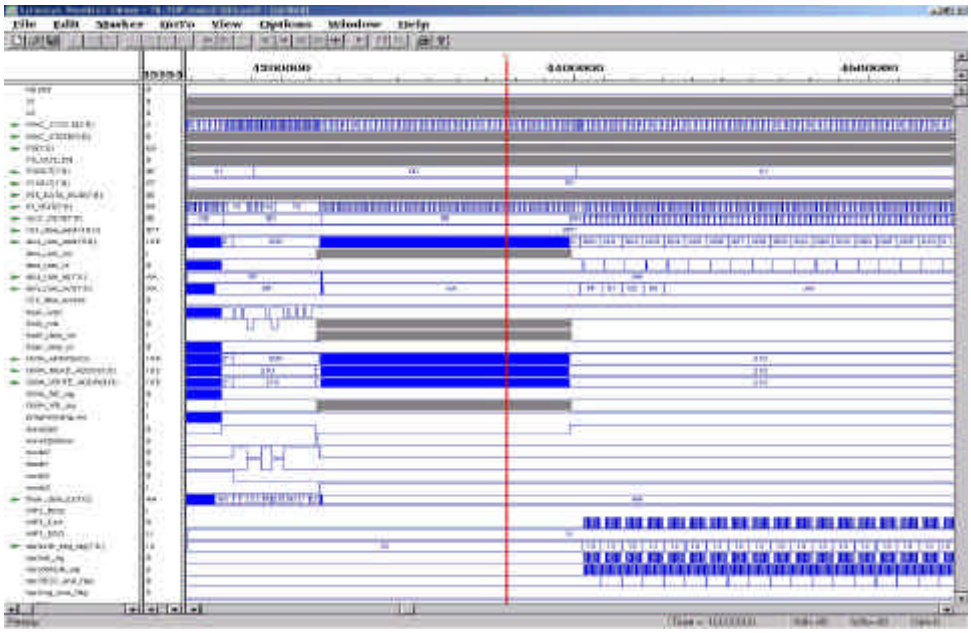
다시 말하면 외부 MP3 decoder는 MP3REQ가 negated되는 그때의 byte까지 buffer에 담아두고 처리한다는 것이다.



[그림5-4] MP3REQ에 따른 MP3 stream SDO출력

2.3 FLASH_SMC_CON BLOCK

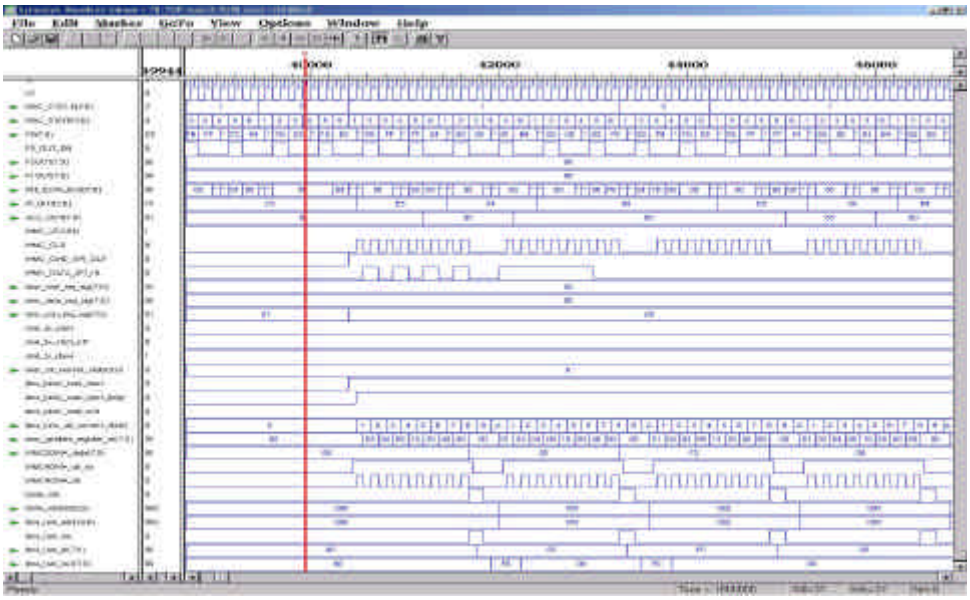
각 mode별 FLASH control simulation은 [그림5-5]에 전체 FLASH simulation을 나타내어 보았다. FLASH simulation에 있어서 check해야 할 사항은 mode별 FLASH/SMC selection에 따라 각 control signal pin의 출력이 올바로 나오는 것을 보면 된다. 위 3장에서 설명한대로 FLASH/SMC의 각 기능별, reading, programing, erasing에 따라 기본신호들이 timing diagram을 따르는가만 보면 되겠다. 여기서 주의해야 할점은 각 FLASH size별로 control address가 하나씩 더 늘어간다는 것에 유의하자. 아래는 [그림5-5]는 FLASH에 DATA를 write한다음 read하여 MP3로 출력시키는 과정을 simulation한 것이다.



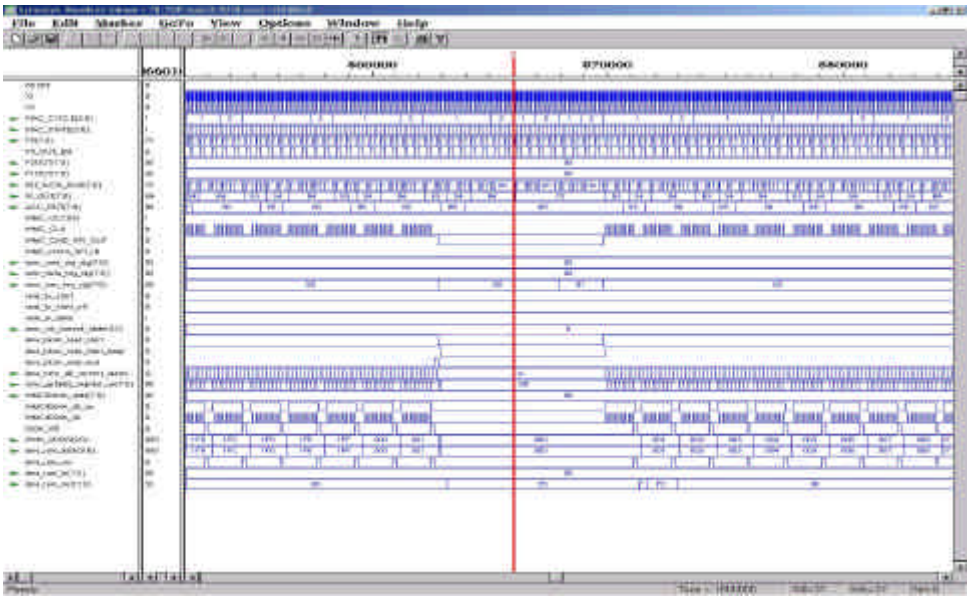
[그림5-5] FLASH와 DMA를 거친 512byte의 MP3 출력 simulation

2.4 MMC_CON BLOCK

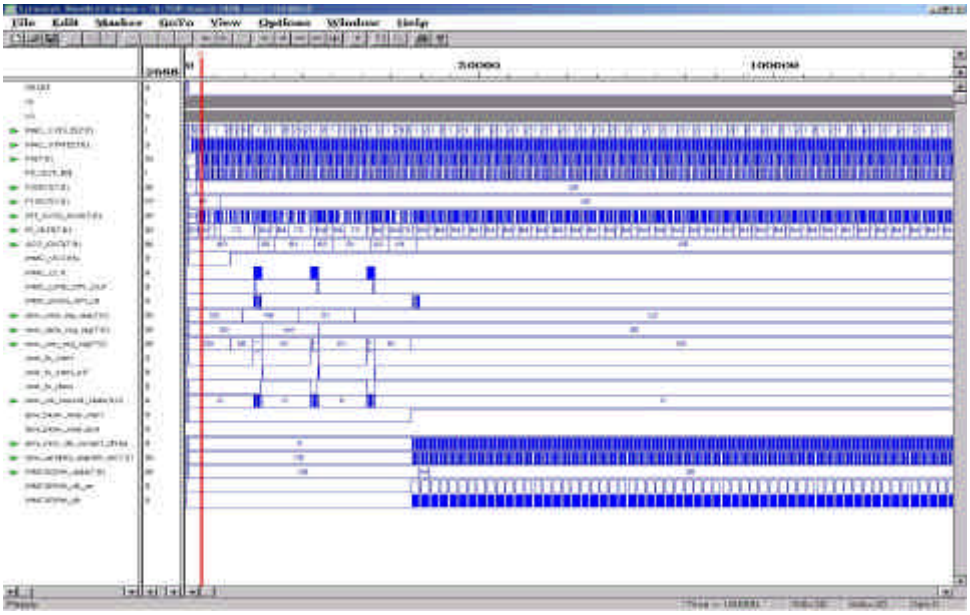
MMC의 simulation에서 제일 먼저 이루어진 것은 원하는 command를 register에 넣어서 SPI_OUT으로 전송시키는 일이었다. 입력인 SPI_IN은 임의의 데이터를 넣어 수행하였으며 MMC의 response는 올바른 것으로 인식해서 skip되도록 처리해서 simulation 하였다. MMC clock은 최대 20Mhz로 정해져 있으며 실제 설계한 프로세서에는 6Mhz를 사용하였다. 다소 속도가 느린감이 있기는 하나 1초에 750Kbps를 출력 시킬수 있기 때문에 일반적인 MP3 128Kbps보다 충분히 크므로 문제는 없다. 그러나 다른 application에서 사용시 MMC clock을 control할 수 있는 추가적인 register가 필요할 것이다. 다음 [그림5-6,7,8]은 MMC의 simulation으로 얻어진 파형들이다.



[그림5-6] 기본 MMC data read/write simulation



[그림5-7] MMC->DMA로의 data writing simulation



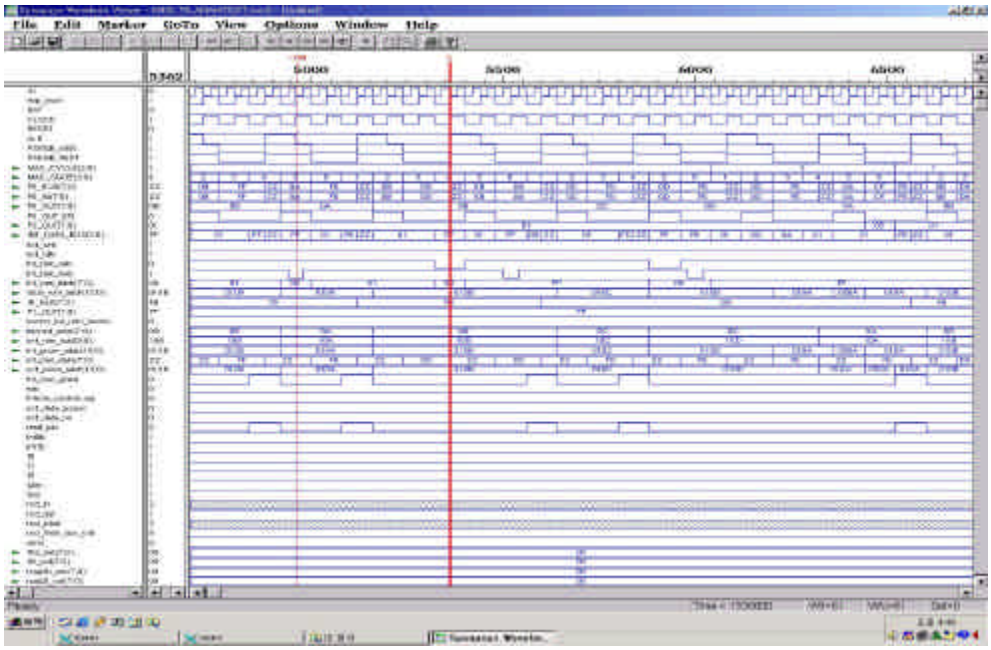
[그림5-8] MMC의 Block reading simulation

2.5 IDE_CON BLOCK

IDE control block에 대한 simulation은 HDD에 대한 interface에서의 response 지식의 부재로 단순 control register의 read/write만 했으며 설계후 바로 FPGA로 검증했다. 해당 register에 data가 제대로 write되며 input pin을 통해 IDE interface response signal들리 register로 제대로 write 되는지만 check하였다.

2.6 전체 controller 검증

전체 모든 block을 interface 한후의 simulation은 위의 기능블럭에 대한 control register의 setting 후 위에서 검증한 simulation wave가 제대로 나오는지에 대한 검증으로 수행했으며 i80c32에서 사용한 256 byte의 internal RAM을 위한 FPGA embed RAM array의 simulation에 더 치중했다.



[그림5-9] I80c32 core의 FPGA Internal RAM array의 simulation

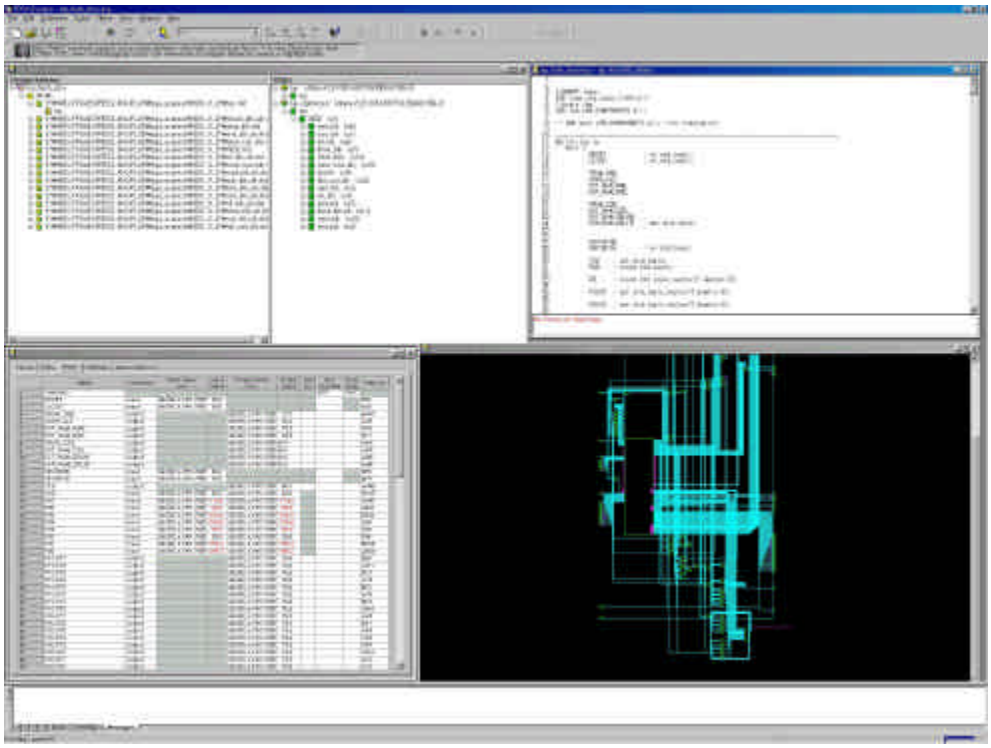
Ⅵ. FPAG를 이용한 실시간 검증

1. FPGA compile과정

본 논문에서 검증용으로 사용한 FPGA는 Altera의 FLEX10k250A-3로 25만 gate급의 적당히 큰 FPGA이다. 설계한 VHDL code를 compile하여 FPGA로 설계한 회로를 다운로드 하는 과정은 다음과 같다.

Step #1 : FPGA express v3.4 for Altera

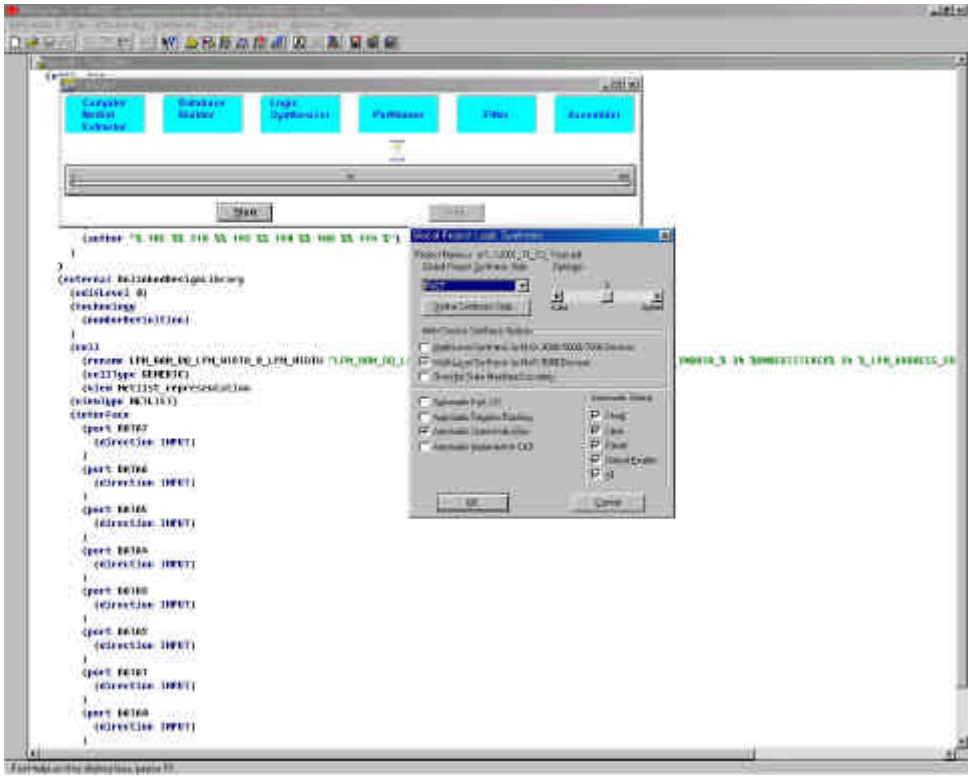
사용환경 : P-III 933Mhz, RAM: 521Mbyte => compile시간 :20분



[그림6-1] FPGA express for Altera의 화면 capture

Step #2 : Maxplus2 v9.3 for Altera

사용환경 : P-III 933Mhz, RAM: 521Mbyte => compile시간 :15분



[그림6-2] Altera MAXplus2의 화면 capture

※ 참고로 timing SNF는 generation하지 않았다. compile시간을 빨리하기 위해서 이며 timing operation을 ON했을 경우 예기치 않은 compile오류가 많이 발생하여 acf화일을 강제로 수정하여 timing SNF 화일을 만들게 하지 않았다.

Step #3 : FPGA board에 JTAG mode로 sof file downloading....

최종 작업한 data => EPC2 FLASH에 2개의 pof화일을 program