Chapter 7

Initial ADSL Line Simulator Design

The initial ADSL line simulator design was based solely on signal manipulation in the frequency domain. At the heart of the design are a FFT, manipulation block and an IFFT. Two implementation paths using Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA) were evaluated in terms of five metrics. In order to give an easily re-configurable environment, parameters describing the access line's response, noise and crosstalk environment must be continuously downloadable during a simulation run, either from a pre-computed data store or generated real time through a PC control interface. A detailed high level design using FPGAs was completed identifying peripheral signal conversion, logic and memory components.

7.1 Line Simulator Block Functionality

Figure 7.1 shows the basic block functionality of the initial simulator design. As previously mentioned, signal manipulation for line response, noise and crosstalk, is performed solely in the frequency domain through frequency filtering and spectral component addition respectively.

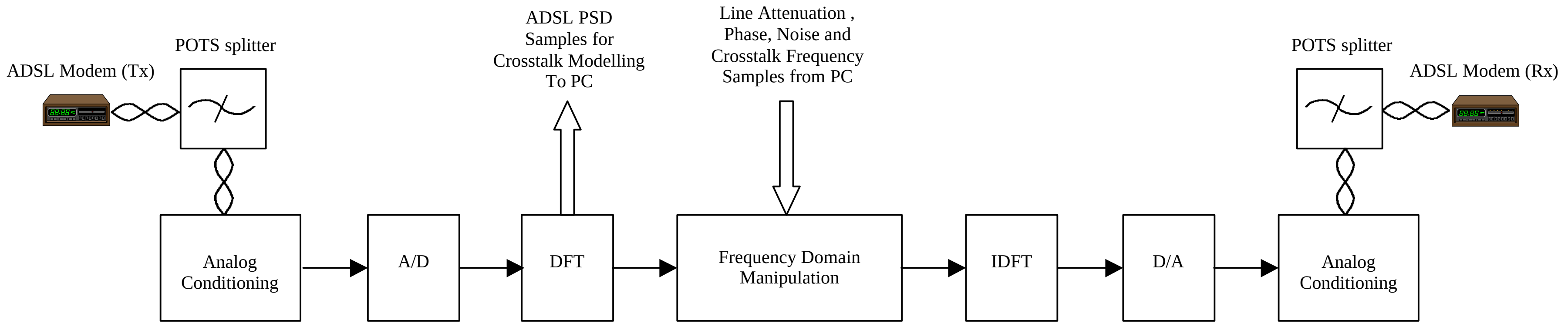

ADSL Modem (Tx)
POTS splitter
ADSL PSD Samples for Crosstalk Modelling To PC
Line Attenuation , Phase, Noise and Crosstalk Frequency Samples from PC
POTS splitter
ADSL Modem (Rx)
Analog Conditioning
A/D
DFT
Frequency Domain Manipulation
IDFT
D/A
Analog Conditioning

Figure 7.1 ADSL line simulator functional block diagram

7.2 Frequency Filter and FFT Implementation

Frequency filtering operates on a block processing principle. Repeatedly over each sampling window, N consecutive time samples are processed to give a block of N computed frequency samples (positive and negative). The frequency samples in each block must be multiplied by N complex discrete filter response samples. The resulting manipulated block of frequency components is then transformed back to the time domain using an IFFT. Clearly, for real-time simulation, each time a block ‘enters’ the FFT, a manipulated block must ‘leave’ the IFFT to prevent a build up of blocks requiring manipulation. If all three processes are carried out within the period of one sampling window, the overall signal will be subject to just one sampling window delay. However, such a scheme is both computationally expensive and unnecessary. The three processes of FFT, manipulation and IFFT may be pipelined so that each takes one time sampling window to compute. The result is a signal delayed by three sampling windows.

The FFT algorithm for an ADSL line simulator must repeatedly compute a 16-bit 1024 point DFT within one sampling window. At 4.416 MSPS, the sampling window, T_w, of a 1024 point DFT is given by:

$$\begin{aligned} T_w &= NT_s \\ &= \frac{N}{f_s} \\ &= \frac{1024}{4.416 \text{ x } 10^6} = 232\text{ms} \end{aligned}$$

Before any peripheral circuitry can be designed, the practical FFT implementation must be chosen. The peripheral circuitry for signal conversion, control logic and PC interfacing will essentially be designed around individual FFT solutions.

Three fundamental hardware approaches to performing the FFT were considered: hardware specific transform chips, DSPs and FPGAs. Most hardware FFTs are designed to operate in the audio spectrum and none were found offering the required performance of a 16-bit 1024 point DFT computed within the 232 µs sampling window.

In order to evaluate the suitability of using a DSP or FPGA to perform the FFT, manipulation and IFFT, the following five metrics were considered:

1. 1024 point 16-bit FFT / IFFT execution time
2. Chip packaging
3. Prototype and production costs
4. Versatility to implement new simulator functionality with minimal hardware redesign
5. Future adaptability to VDSL line simulation

The first metric, speed of execution, is obviously the most important for either a university or industrial based project, but the following four would rate differently in terms of importance within different development teams. For example, chip packaging and prototyping costs are very important to a four month MSc project conducted in a setting with no established development hardware, whereas production cost and future adaptability to VDSL line simulation may be rated more highly in a commercial organisation with on-going DSL modem design programs.

7.2.1 Implementation Using DSPs

DSPs are basically microprocessors with architectures specifically designed for the repeated arithmetic operations commonly encountered in performing signal processing functions such as filtering, correlation and FFTs. As with common microprocessors, the CPU executes machine code instructions sequentially although some degree of parallelism may be incorporated. Complex mathematical functions are written either in machine code directly for specific processors, or in a higher level language such as C which is processor independent, then compiled down to machine code for a specific DSP architecture. Both machine code libraries specific to individual DSP architectures and higher-level software libraries exist for almost every signal processing function imaginable.

7.2.1.1 Speed Metric

From the outset, the decision to use a FFT library function or to write one's own code must be taken. The execution time of an existing function written in C will vary according to the target DSP architecture and compiler used. However, benchmark performance figures are generally available with individual functions on at least one host architecture. Although this will vary on different platforms, the performance figures give a good initial figure for execution time. Machine code library function execution times are fairly easy to determine as these are processor specific and will be published with the number of machine cycles required to run the entire algorithm. With knowledge of the processor clock period, a simple calculation gives the execution time. Although existing functions are not necessarily speed optimised, the prospective improvement through writing speed optimised code is unlikely to be dramatic and without extensive effort may even give reduced performance. Therefore for assessment of the speed metric, published benchmark performance figures are used.

Table 7.1 lists the execution times of 1024 point 16-bit radix 4 FFT algorithms on a representative selection of the latest DSP platforms[1,2,3].

DSP	Software Type	Clock Speed (MHz)	Execution Time (µs)
TMS320C6202	Machine Code	250	53
TMS320C6701	Machine Code	167	108
TigerSHARC	C	250	41
DSP56600	Machine Code	60	287

Table 7.1 FFT execution times on various DSP architectures

The first three entries in table 7.1 represent the latest DSP products, either at the product preview or sample distribution stage. The final entry for the Motorola DSP56600 is a current production device, released four years ago. The difference in processor clock speeds between the latest and established DSPs is quite dramatic and shows DSPs have only recently reached performance levels advanced enough to execute the 1024 point FFT identified for ADSL line simulation within 232 μs on a single device.

Both the Analog Devices TigerSHARC and the Texas Instruments TMS320C6202 DSPs are fast enough to implement both the FFT and IFFT functions within one 232 μs sampling period. Using the TMS320C6202, the 1024 complex (vectored) frequency filter multiplications each require 56 cycles, a total of 229 μs. In contrast, 1024 complex additions require just 2064 cycles or 8 μs. Clearly, to implement the transformations and signal manipulation algorithm would require two DSPs. Alternatively a hybrid solution with a single DSP to perform the transforms and a small FPGA for the multiplications and additions could be used, shown in figure 7.2.

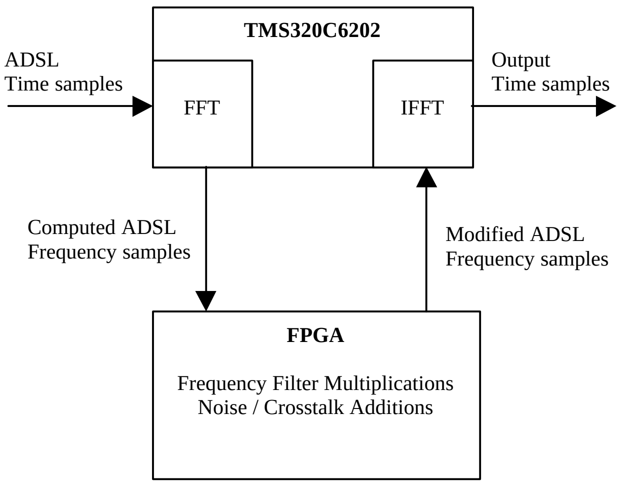


Figure 7.2 Hybrid DSP / FPGA solution

7.2.1.2 Chip Packaging

Table 7.2 shows the packaging options for the four DSPs identified above.

DSP	Package
TMS320C62xx	352 / 348 pin BGA
TMS320C67xx	452 pin BGA
TigerSHARC	400 pin BGA
DSP56600	144 pin QFP / BGA

Table 7.2 DSP packaging options

7.2.1.3 Prototyping and Production Costs

Since DSP solutions are fundamentally software based, some form of software emulation is required to verify code before functionality testing is performed on a development board. Both comprehensive emulation software and development kits cost several thousands of pounds each.

As is seen from table 7.2, the three DSPs capable of meeting the speed requirements are packaged in very large devices and are not really convenient for prototyping in a university environment with extremely limited resources.

Unit costs are difficult to determine as silicon suppliers tend to avoid single quantity supply, preferring instead to offer evaluation samples of devices, then minimum shipments of several tens of units. Samples may possibly be obtained via a large industrial sponsor such as Fujitsu. Currently the TMS320C6202 is quoted at $655 per unit with a minimum shipment of 200 units[4].

7.2.1.4 Versatility

One of the prime advantages of using a DSP to perform the FFT / IFFT functions is the ease with which they can be reprogrammed to give new functionality. Even with the hybrid approach of figure 7.2, on The TMS320C6202 there would be approximately 125 μs of 'free' processor time available each time sample window for added functionality.

7.2.1.5 Future Adaptability to VDSL Line Simulation

A quick calculation of the sampling window for the FFT described in chapter 5 of a 2048 point radix 2 FFT, with time samples at 50 MSPS gives

$$
\begin{aligned}
T_w &= \frac{N}{f_s} \\
&= \frac{2048}{4.416 \text{ x } 10^6} = 41\mu s
\end{aligned}
$$

From the DSP benchmarks, a 2048 point radix 2 FFT would take 183 μs to execute on the TMS320C6202. In terms of the required speed increase to execute the function within the given 41 μs, a quadrupling of processor speed is required. A more efficient 4096 point radix 4 FFT requires 251 μs to perform. With a sampling window of 82 μs (4096 samples instead of 2048), real time processing requires a three-fold increase in computation speed. Considering the increase in clock speeds over the last four years (from the TMS320C32-60 to the TMS320C6202 and from the SHARC to TigerSHARC DSPs) a VDSL line simulator will probably be feasible using one DSP to perform the FFT and another for the IFFT within a few years.

Even if the packaging of new faster DSPs is different to the latest chips from the same manufacturer, the DSP solution does offer a logical proving ground towards a VDSL simulator.

7.2.2 Implementation Using FPGAs

7.2.2.1 Speed Metric

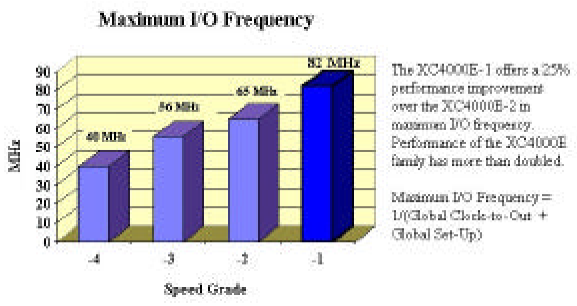
Unlike DSPs, FPGAs don’t operate on a predefined maximum device clock. Instead, performance is determined by logic and path delays within a device. As such, it is impossible to accurately predict the performance of a new design without physically placing and routing it inside a target device. The design of a complex block such as a large FFT would take an experienced designer many man-hours and would be a highly iterative process to minimise logic and routing delays. However, the interest in the possible use of FPGAs to perform FFT functions was spurned by the ‘Core’ program piloted by the largest FPGA supplier, Xilinx.

The Core program provides many pre-designed and verified functional blocks with guaranteed execution times on specific target devices. In complex logic designs, an externally driven clock signal is introduced for synchronous operation so the performance of a particular Core design is given in terms of a maximum clocking frequency and number of clock cycles required. The minimum clock period is determined by the maximum critical net and logic delay within the Core design. Appendix 4 includes a list of available Cores and appendix 5 details three Cores of interest: 1024 point FFTs, parallel multipliers and registered adders.

With reference to the FFT Core data sheets in appendix 5, a 1024 point, 16-bit FFT can be performed in 17408 clock cycles. Further into the data sheet, an approximate period of 60 ns is given for external read access timing for the XC4013E-3 device. From the timing diagrams there are two clock periods for each memory read cycle, giving a clock period of approximately 30 ns (33 MHz) and total FFT execution time of 522 μs. Targeted at this device, the FFT Core isn’t fast enough for real-time processing in the line simulator.

The XC4013E-3 is a fairly old device. Xilinx rate their chips with speed grades because, as previously mentioned, clock speeds only apply to synchronous designs and are different for each design. Two newer, improved versions of the XC4000 FPGA series have since been introduced, the XC4000XL series and in 1998 the XC4000XLA series. The XLA grades are claimed to have significant improvements in maximum clock speeds over the older ‘E’ grades. Figures 7.3 and 7.4 show extracts from Xilinx literature promoting the E and XLA XC4000 series.

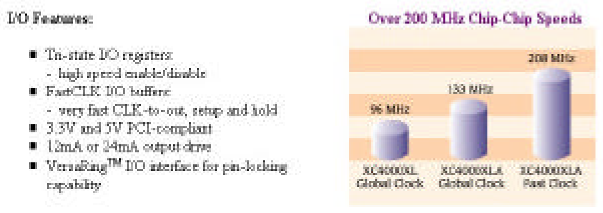
The XC4000XLA speed grade saw the introduction of resources for a special ‘Fast Clock’ in addition to the standard ‘Global Clock’ resources present in the E grades. Comparison of the Global Clock of the E-3 and XLA speed grades show a predicted increase in chip speed of approximately 138% ((133 – 56) / 56). If this improvement is seen in targeting the FFT Core to the fastest XLA device, the execution time should be reduced from 522 μs to just 222 μs, within the target of the time sampling window.



The XC4000E-1 offers a 25% performance improvement over the XC4000E-2 in maximum I/O frequency. Performance of the XC4000E family has more than doubled.

Maximum I/O Frequency = 1/(Global Clock-to-Out + Global Set-Up)

Figure 7.3 XC4000E chip speeds


I/O Features:

- Tri-state I/O registers:
  - high speed enable/disable
- FastCLK I/O buffers:
  - very fast CLK-to-out, setup and hold
- 3.3V and 5V PCI-compliant
- 12mA or 24mA output drive
- VersaRing™ I/O interface for pin-locking capability

Figure 7.4 XC4000XLA chip speeds

Cores also exist for parallel multipliers and scaled adders to implement the signal manipulation. Multiplication using FPGAs is implemented using lookup tables, whereas digital processors use repeated addition. With reference to appendix 5, targeted at XC4000E-1 devices, 16-bit by 16-bit multiplication requires just 5 cycles of the 58 MHz clock, or 86 ns. The multiplication Core deals with scalars, not vectors, therefore separate multiplications are needed for both the real and imaginary parts of each of the computed 1024 ADSL frequency samples. In the XC4000E-1 series, the total of 2048 multiplications would take 176 μs, but targeted at the fastest XC4000XLA speed grade the execution time should fall to about 108 μs (62% speed improvement). The Core multiplies two 16-bit numbers giving a 32-bit result. Obviously since the physical line is always attenuating, the simulator must multiply by numbers less than one. This can be achieved by ignoring the lower 16 bits of the result, shown in figure 7.5.

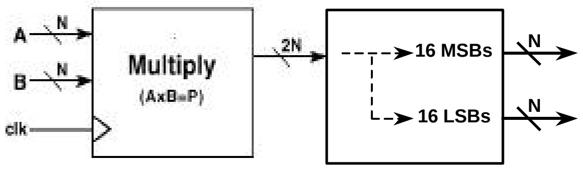


Figure 7.5 Multiplication by coefficients < 1

If the frequency sample to be attenuated is 16-bits wide, then the most significant 16-bits of the result of a multiplication by 2^{16} is equivalent to a unity multiplication. To attenuate by 2 use a multiplying coefficient of $2^{16}/2$ and use the 16 most significant bits of the result only.

The addition Core data sheet in appendix 5 doesn't include performance figures. However, addition should be at least as quick as multiplication when implemented using the CLBs of an FPGA. A ball park figure from the net and logic delay of an N-bit adder[5] of 4.5 + 0.35N ns, gives a processing time of approximately 21 µs for 2048 additions (real and imaginary parts) for the E-3 speed grade.

Combining the processing times for the signal multiplications and additions, using the fastest XLA grade device both functions can be performed within approximately 130 µs, well within one time sampling window.

7.2.2.2 Chip Packaging

Each Core design requires a minimum number of CLBs, listed in the relevant data sheet. The smallest device required for the FFT Core is the XC4013, which is available in 160 pin QFP packaging. For the manipulation block, the 16-bit area optimised multiplier Core requires 213 CLBs, while the 16-bit adder needs just 9. The total of 222 CLBs are available in the smallest XLA device, the XC4013XLA, which contains 576 CLBs.

Commercial, zero insertion force, multiple extraction cycle 160 pin QFP sockets are available with convenient pin layout which would enable prototyping within a university environment.

7.2.2.3 Prototyping and Production Costs

A total of three XC4013XLA devices are required to implement the FFT, IFFT and manipulation blocks using Cores. The XC4013XLA-07PQ160C is quoted at £59.99 from MicroCall. In addition to hardware, the Foundation software design suite is required to generate Cores and finally place and route in target devices. A single university licensed copy of the full package costs £373 (Normally $7995).

Both prototyping and production costs for an FPGA implementation are a fraction of that for a DSP solution.

7.2.2.4 Versatility

Xilinx FPGAs are programmed by a serial bit stream stored in a PROM during the initialisation period immediately after power up. The internal CLB configuration and hence the functionality of the device is determined by the content of this bit stream. Therefore, within the limitations of the physical layout of a FPGA based simulator board, new functionality can be programmed to an existing FPGA device if sufficient CLB resources are available on that device.

The FFT / IFFT Cores each utilise approximately 90% of the two XC4013 devices (532 out of 576 CLBs), with the multiplier and adder using 39% of a similar device, so some scope is present for revised functionality based on a simulator board with three FPGA devices without hardware modification. Combining the spare resources from all three FPGAs after their Cores have been placed and routed, approximately 442 CLBs are free for implementing other control and support logic which may also be required on the simulator board.

7.2.2.5 Future Adaptability to VDSL Line Simulation

A simulator design based on the 1024 point FFT Core is less adaptable to future use for VDSL line simulation because larger 2048 and 4096 point FFT Cores don't exist. In addition to this requiring the in-house development of larger FFT functions, considerably more CLBs would be needed for their implementation. However, the fundamental signal processing approach could be proved using FPGAs for the simulation of ADSL lines then a larger VDSL simulator built.

In terms of speed, the VDSL simulation requirement of a 2048 point FFT computed in just 41 µs implemented using FPGAs requires a projected 10 fold speed increase, considerably more than the three fold increase required of a DSP solution.

7.2.3 Preferred Implementation

Overall, mainly due to ease of prototyping and cost, the FPGA solution to FFT, IFFT and signal manipulation is preferred over the hybrid DSP-FPGA solution.

7.3 Overall Line Simulator Design

Once the decision to use FPGAs to implement the simulator's fundamental functions is taken, an overall design around these blocks can be made. Extensive reference is made to the FFT Core data sheet in appendix 5. Figure 7.6 shows the FFT Core Interface from the data sheet.

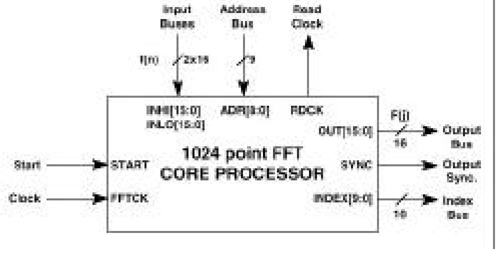


Figure 7.6 1024 point FFT interface pinout

A block schematic diagram of the complete simulator design is shown in figure 7.7.

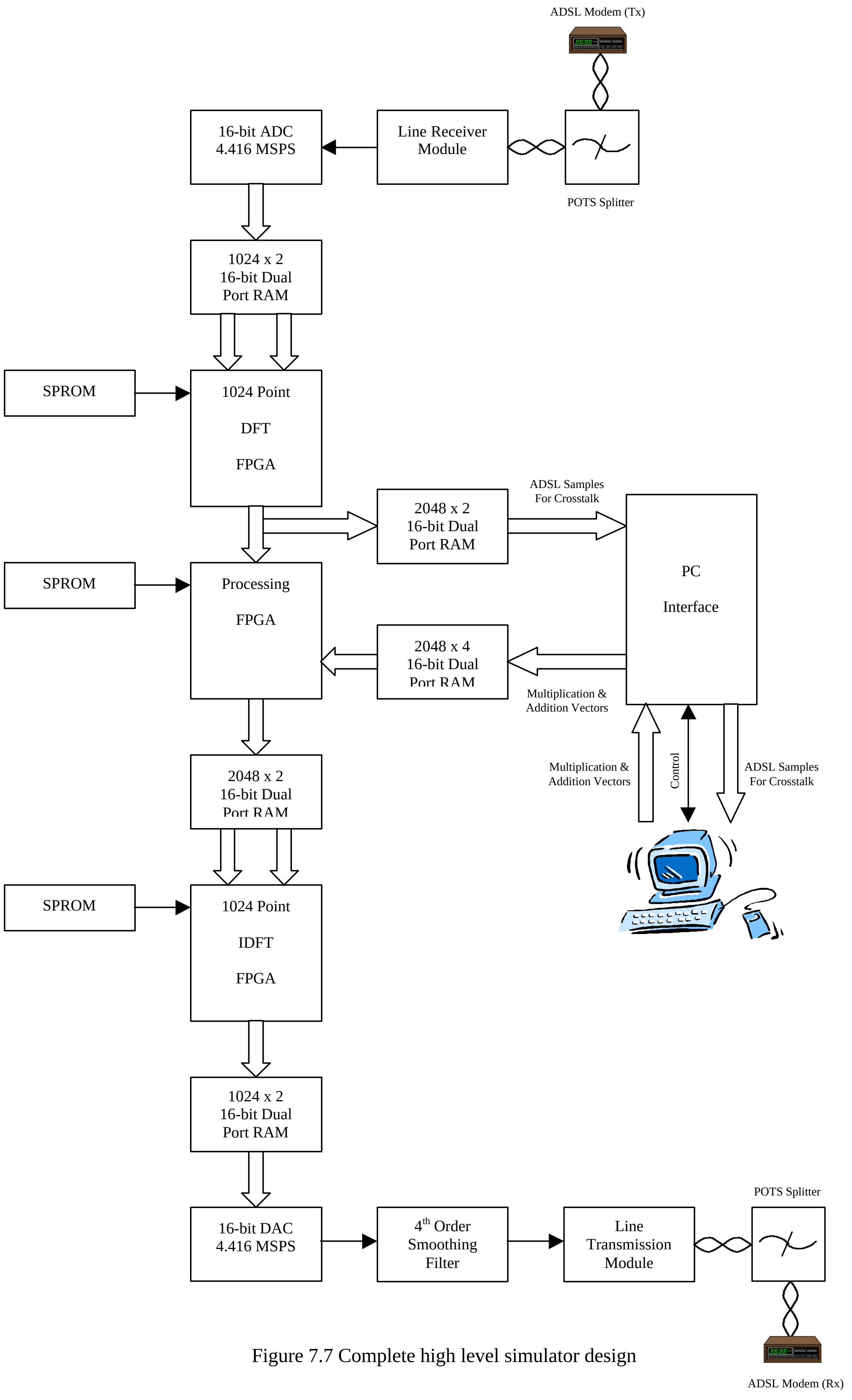


Figure 7.7 Complete high level simulator design

7.3.1 FFT Core Input Data Conditioning

The basic FFT Core is used to perform both the DFT on blocks of 1024 ADSL time samples and the IDFT on blocks of 1024 complex manipulated frequency samples. A new transformation as done each time sample window on a new set of data samples. Since the same FFT Core design is used for both DFT and IDFT functions, they will be referred to as the FFT and (I)FFT Core respectively.

7.3.1.1 ADC – FFT Interface

The FFT Core requires input data split into two blocks. For the DFT, the first block, called the LOBLOCK, consists of time samples t(0) to t(N/2 – 1) and the second, the HIBLOCK, consists of time samples t(N/2) to t(N –1). The FFT Core reads one data word from each block every two FFTCKs. In addition, the core requires un-interrupted access to both memory blocks. Because the ADC produces data at a different rate (once every ADCLK) and in a sequential order, each time windowed group of 1024 time samples from the ADC must be buffered and then read in the order and at the rate required by the FFT Core during the next window period. This can be achieved by loading the time samples from the ADC into two blocks of dual port RAM organised into two pages. Whilst data from the converter is being loaded into one page on one side of the dual port RAM, data from the previous sampling window in the other page will be read by the FFT Core from the opposite side. During the next sampling window, data from the ADC is loaded into the second page whilst the core performs the DFT on the 1024 time samples stored in the first page. This process repeats continuously. Conceptually this arrangement is shown in figure 7.8.

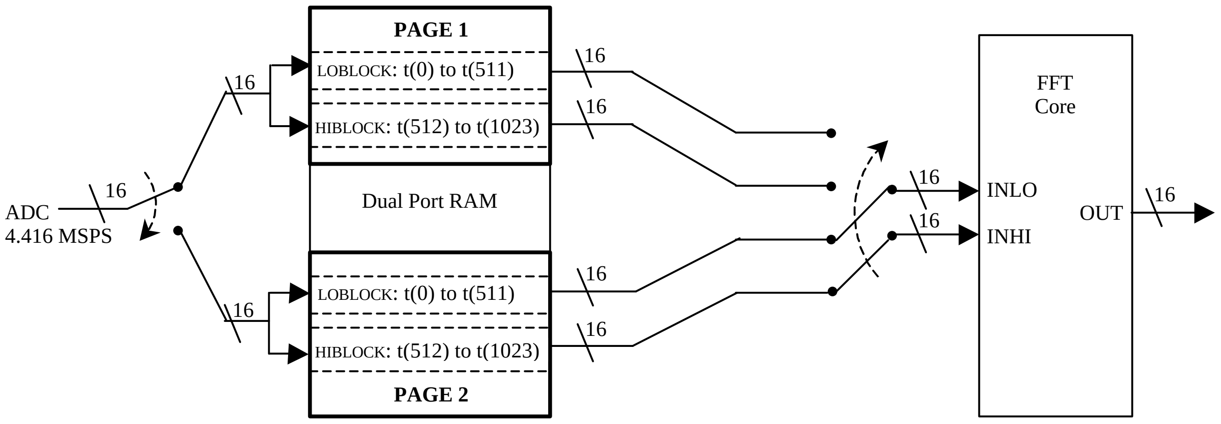


Figure 7.8 Conceptual memory arrangement for time sample input to the FFT

Each block of each page consists of 512 16-bit wide locations. Practically the block structure can be implemented using just two 1 kByte dual port RAM chips, one for each block. Shown in figure 7.9, each block contains 2 pages, page 1 extends from address location 0 to 511 with page 2 from location 512 to 1023.

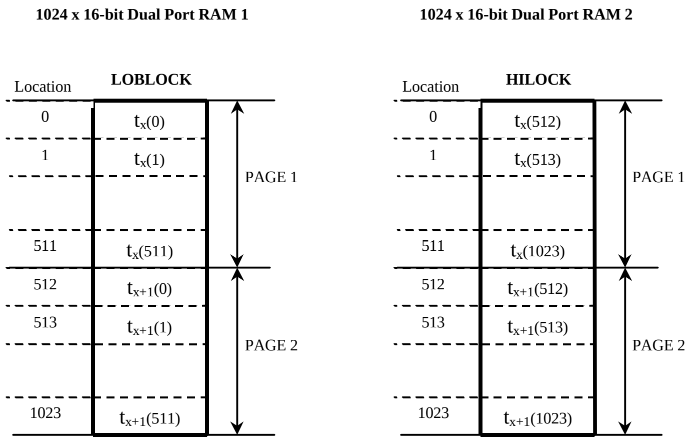


Figure 7.9 Practical memory block and page structure

A single 9-bit parallel output counter can be used to address both pages at the same time with appropriate block and page selection control, shown in figure 7.10 with associated digital timing in figure 7.11. The FFTSTART signal for the FFT Core is generated on the rising edge of the BLOCKSELECT signal which occurs every 1024 ADCLKs (one time sampling window). All the supporting logic shown in figure 7.10 can easily fit into the spare capacity of the FPGAs after the Cores have been placed and routed.

7.3.1.2 IFFT – DAC Interface

The ADC samples at 4.416 MSPS, which is also the rate at which samples must be converted by the DAC. Since the FFT Core reads its input data at a different rate and in a different order to which the ADC produced it, memory buffering of the time samples is required. In the same way, the order and rate of data output from the (I)FFT Core is not sequential and not at 4.416 MSPS, so memory buffering is also required between the (I)FFT and DAC.

After multiplication and addition, the modified 1024 frequency samples must be stored in a similar two block arrangement to that for the DFT shown in figure 7.9 to allow the (I)FFT Core un-interrupted access to its own input data. Here, the LOBLOCK consists of the manipulated complex frequency samples, f(0) to f(N/2 – 1) and the HIBLOCK the samples f(N/2) to f(N – 1). During each window period, data output from the (I)FFT Core is written to one side of the dual port RAM, whilst data written to memory from the (I)FFT Core in the previous window is read out to the DAC from the other side at 4.416 MSPS and in a sequential order. However, whereas the input to the FFT Core is real valued only, its output is complex with both real and imaginary parts. The output from the manipulation block is also complex, therefore the storage space required for the (I)FFT Core input is double that compared with the FFT Core.

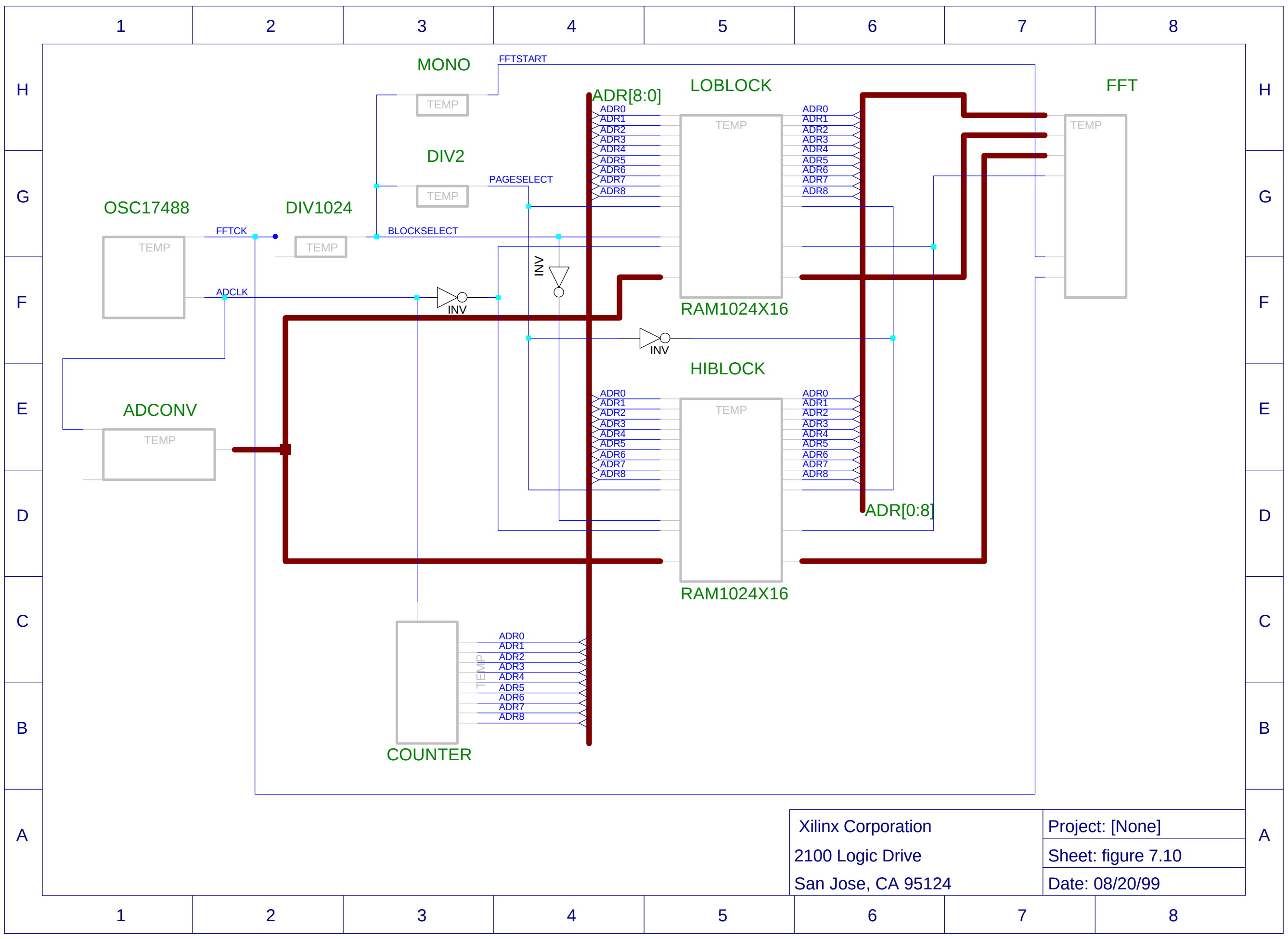


Figure 7.10 Circuit diagram for ADC and FFT Core interface

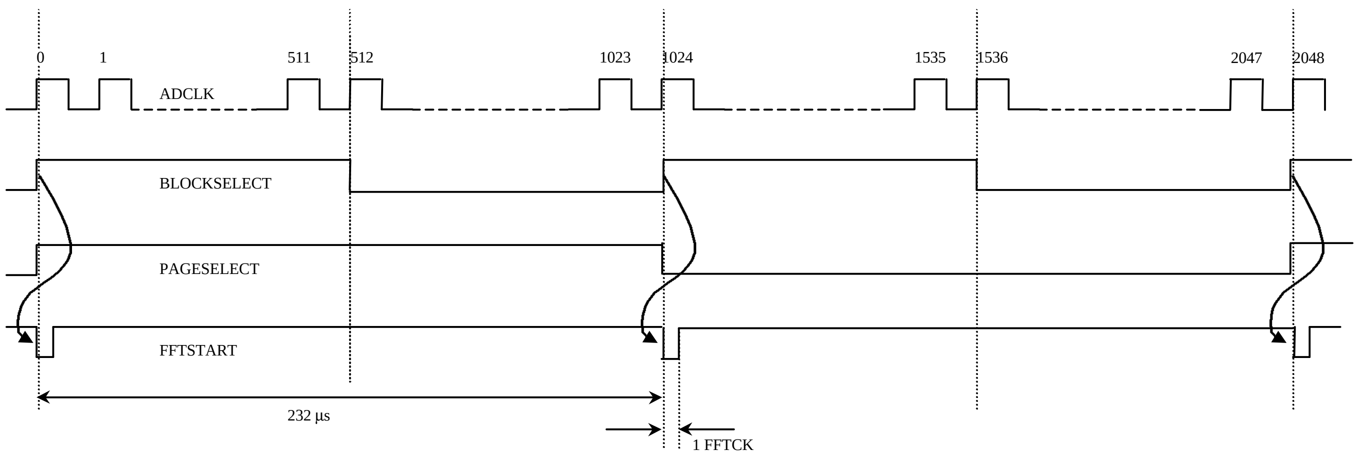


Figure 7.11 Digital timing diagram for ADC and input to FFT Core

7.3.2 FPGA Initialisation

During an initialisation period, serial data is transferred from SPROMs to the FPGAs to configure the devices. Xilinx manufacture specific SPROMs for different sized FPGAs.

7.3.3 PC Interface

Both a complex multiplication vector representing the physical line's frequency response and complex addition vectors for noise and crosstalk simulation must be supplied to the processing FPGA during a simulation run from a controlling PC. The 16-bit 2048 word addition vector (1024 real and 1024 imaginary) must be updated every time sampling window, whereas the same sized multiplication vector requires loading only once at the start of a simulation run. When a new line is to be simulated, a new multiplication vector must be downloaded describing the new frequency response. As previously described, an efficient method to generate self crosstalk is to use a delayed and attenuated copy of the actual ADSL signal's DFT itself. To allow complete control of self crosstalk, the complex 1024 computed ADSL frequency samples from the FFT Core should be uploaded to the PC during each time sampling window, then processed and downloaded as a constituent part of a later addition vector. Ideally a well defined interface should exist between the simulator board and controlling PC.

Shown previously in figure 7.7, dual port RAM can be used as a store and buffer between the PC interface and the DFT and processing FPGAs. The RAM used to store both the copy of the ADSL's DFT and processing addition vector should operate on a split page mode similar to that described previously to allow simultaneous read and write operations for consecutive data blocks from opposite sides of the dual port RAM. The multiplication vector could be stored in single port RAM as only one download per simulation run is required, but since dual port RAM will be used exclusively elsewhere, using an identical device will simplify timing and other circuitry.

Much of the device level logic will be very similar in nature to that shown in figure 7.10. To avoid excessive numbers of circuit level diagrams, only figure 7.7, the high level block diagram of the complete design is included.

7.3.4 Line Receiver and Transmitter Modules

The output from an ADSL modem will be at voltage levels required for twisted pair transmission. The final part of a transmitting modem contains an AFE that matches the line and power driver's impedance. The first part of a receiving modem will also contain an AFE matching its impedance to that of the transmission line's and operate at a suitable sensitivity. The input of the line simulator should be consistent with that normally seen by a transmitting modem and its output consistent with that seen by a receiving modem. This requires matching both impedance and signal levels.

Specific line drivers such as the Analog Devices AD816 differential driver[6], designed for use with ADSL modems, incorporate both transmitter and receiving electronics in a single device and are suitable for use as the simulator's line drivers.

7.3.5 ADC and DAC Converters

Both ADC and DAC must generate and convert parallel 16-bit data vectors. Two Analog Devices converters, the AD9240-EB DAC at £135 and AD768-EB ADC at £104 are both available on fully populated evaluation boards from the supplier SEI Millenium. The use of evaluation boards obviously reduces the design time, as board layout and interface circuitry is already optimised by the manufacturer.

7.3.6 DAC Output Filtering

Between the output from the DAC and line driver, filtering is necessary. From the scant available details of ADSL modems and AFEs, typically these filters are fourth order with cutoff frequencies of approximately 1.2 MHz[7,8]. Because receiving ADSL modems incorporate high order anti-aliasing filters with similar performance, there is no problem in limiting the spectral output from the simulator to 1.2 MHz with smoothing filters as anything above this is effectively removed by the receiving AFE before any A/D conversion takes place in the receiving ADSL modem.

For simplicity, a fourth order analogue Butterworth filler implemented using fast op-amps will be used to filter the DAC's output in the line simulator.

References

[1] Texas Instruments, TMS320C6701 Data Sheet SPRS067, May 1998.
Texas Instruments, TMS320C67x Single Precision Floating Point Assembly Benchmarks, May 1998.
Texas Instruments, TMS320C6202 Data Sheet SPRS072A, January 1998.
Texas Instruments, TMS320C62x Assembly Benchmarks, May 1998.

[2] Analog Devices, "A New Architecture for the Digital Convergence Infrastructure", TigerSHARC DSP Product Preview and Benchmarks, June 1999.

[3] Motorola, DSP56600 DSP Benchmarks, November 1996.

[4] SEI Macro Group Ltd, Component Quotation, 26th March 1999.

[5] "Estimating the Preformance of XC4000E Adders and Counters", Xilinx Application Note XAPP 018, Xilinx, July, 1996.

[6] Analog Devices, AD816 500 mA Differential Driver & Dual Low Noise (VF) Amplifiers, Data Sheet, 1996.

[7] ST Microelectronics, STLC60135 TOSCA ADSL DMT Transceiver, Data Sheet, May 1998.

[8] Fujitsu Microelectronics UK Ltd, MB86626 Keywave ADSL AFE, Data Sheet, December 1998.