FPGA: DIGITAL DESIGNS

TEAM NAME :DIGITAL DREAMERS

Team Members:-

- Nilay Anurag
- Khagesh Patel
- Krishna Mohan Rai

Team Mentor:- Nikhil Gupta

Aim: - We planned to Construct various digital circuit to enable us to use lcds ,keyboard and other input/output devices and to implement a processor and communicate using the devices

WHAT INSPIRED US:-

Electronics has always fascinated us. When we are mesmerised by the world of technology today we often feel disconnected to them in sense than we cannot make them. we always envision them in some far off lab with geeks with superhuman intelligence working on them. After all when you see you high speed computer you often ponder how is it possible to make by an average engineer.

we decided to bridge that gap. An attempt to show how the basic tools we learnt in electronics fit in this wider pictures of world of technology today.

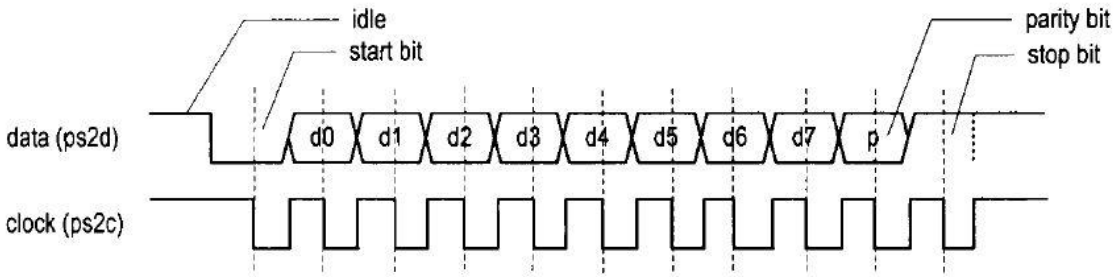
THEORY:-

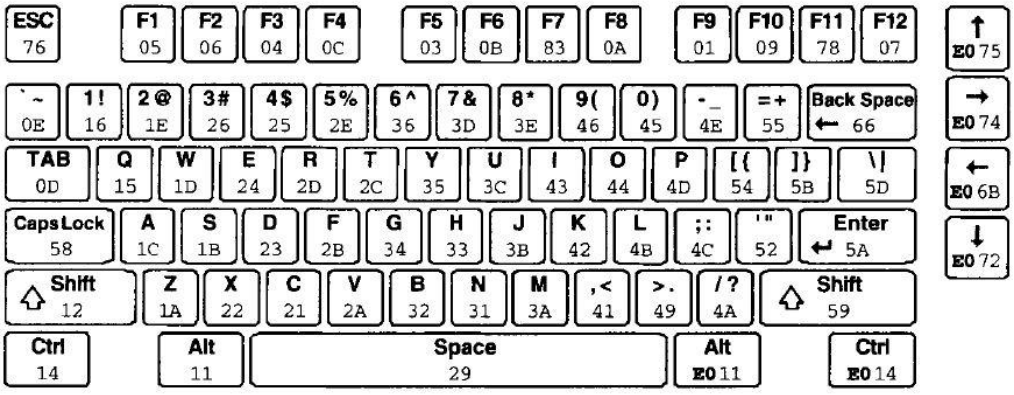
PS2 Protocol KEYBOAD

Protocol is synchronous using two pins one clock pin another data pin. Data is supposed to be sampled at falling edge of clock pin. Data is sent in packet of 11 bits. First start bit “0”,next 8 bits data, next parity then ending bit “1”.

Eight data bits are distinct for each key.

- When you press a key keyboard continuously send data bits corresponding to that key continuously until it is pressed then keyboard sends ending byte ‘f0’.



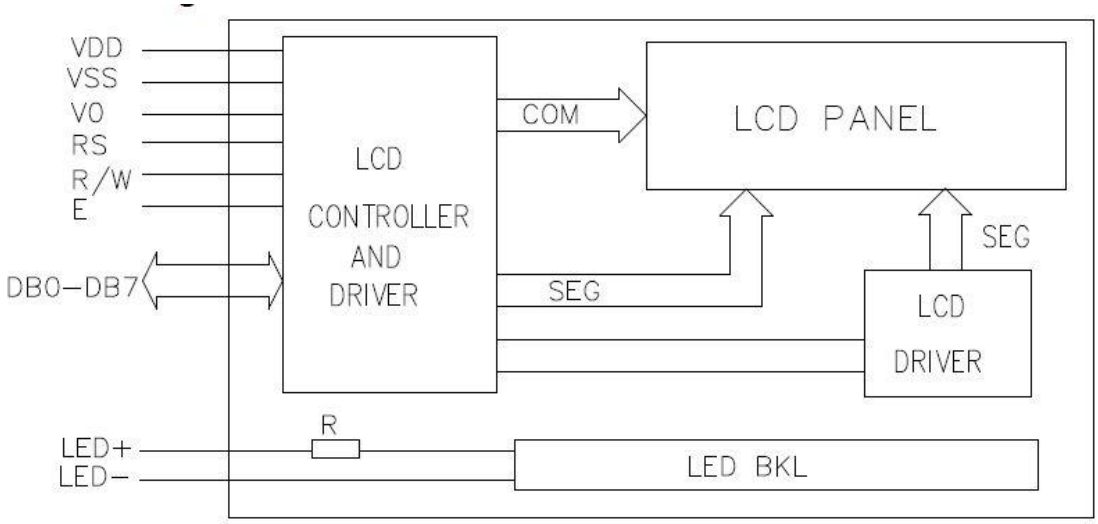


For example if 'a' is pressed then 1c will be transmitted multiple times and bytes received are 1c,1c,1c.....1c,f0.If shift then a is pressed then bytes transmitted will be 12,12...12,1c,....1c,f0,12,.....,12,f0.

LCD PROTOCOL

Basic Pin & Description:

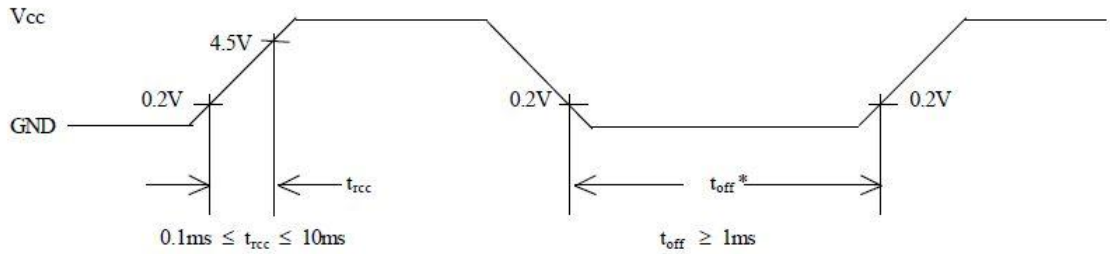
Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL



Initialization Protocol :

1. Initialization using the Internal Reset Circuit

The display can be initialized using the internal reset circuit if the Internal Power Supply Reset timing below is met.



Note: t_{off} represents the time of power off condition for a momentary power supply dip or when cycling power off then on.

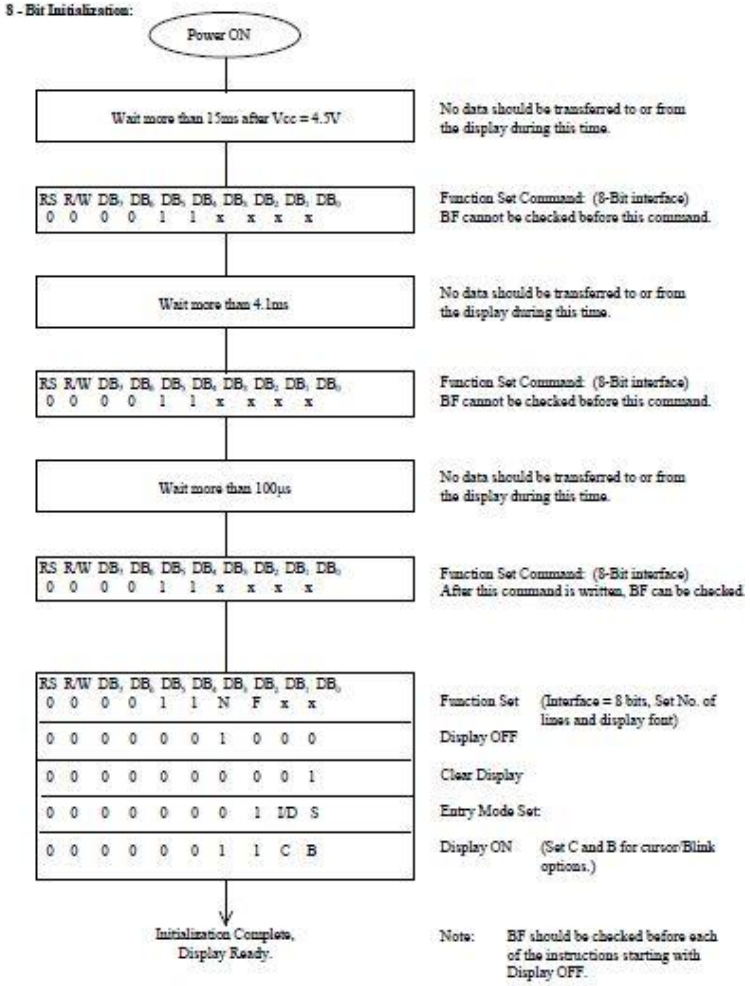
If the internal power supply reset timing cannot be met, the display will not operate normally. In this case, the display can be initialized through software.

Note: Variable power supply may affect timing hence initialization of lcd in that case software initialization is preferred.

2. Software Initialization

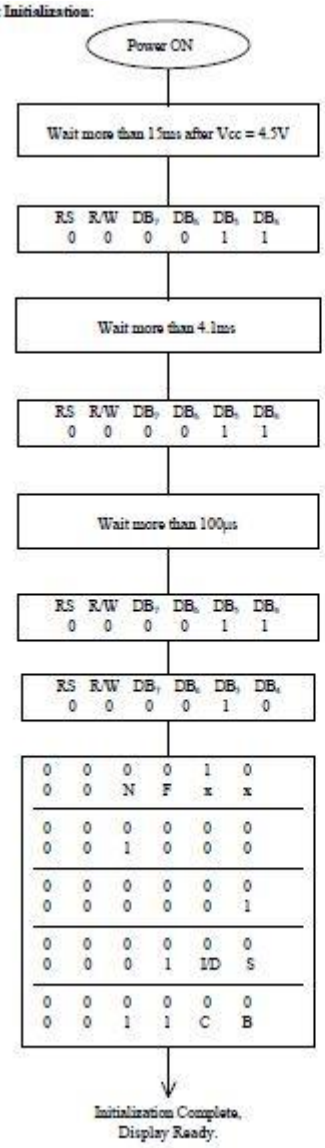
Although software initialization is not mandatory, it is recommended that this procedure always be performed

8 - Bit Initialization:



Note: BF should be checked before each of the instructions starting with Display OFF.

4 - Bit Initialization:



No data should be transferred to or from the display during this time.

Function Set Command: (8-Bit interface)

No data should be transferred to or from the display during this time.

Function Set Command: (8-Bit interface)

No data should be transferred to or from the display during this time.

Function Set Command: (8-Bit interface)
After this command is written, BF can be checked.

Function Set: Sets interface to 4 -bit

Function Set (Interface = 4 -bit, Set N and F for number of lines and character font)

Display OFF

Clear Display

Entry Mode Set:

Display ON (Set C and B for cursor/Blink options.)

Note: BF should be checked before each of the instructions starting with Display OFF.

33

MICROBLAZE

The MicroBlaze is a virtual microprocessor that is built by combining blocks of code called cores inside a Xilinx Field Programmable Gate Array (FPGA). The beauty to this approach is that you only end up with as much microprocessor as you need. You can also tailor the project to your specific needs (i.e.: Flash, UART, General Purpose Input/Output pheriphals and etc.). The MicroBlaze processor is a 32-bit Harvard Reduced Instruction Set Computer (RISC) architecture optimized for implementation in Xilinx FPGAs with separate 32-bit instruction and data buses running at full speed to execute programs and access data from both on-chip and external memory at the same time. The backbone of the architecture is a single-issue, 3-stage pipeline with 32 general-purpose registers (does not have any address registers like the Motorola 68000 Processor), an Arithmetic Logic Unit (ALU), a shift unit,

and two levels of interrupt. This basic design can then be configured with more advanced features to tailor to the exact needs of the target embedded application such as: barrel shifter, divider, multiplier, single precision floating-point unit (FPU), instruction and data caches, exception handling, debug logic, Fast Simplex Link (FSL) interfaces and others. This flexibility allows the user to balance the required performance of the target application against the logic area cost of the soft processor. Figure 1 shows a view of a MicroBlaze system. The items in white are the backbone of the MicroBlaze architecture while the items shaded gray are optional features available depending on the exact needs of the target embedded application. Because MicroBlaze is a soft-core microprocessor, any optional features not used will not be implemented and will not take up any of the FPGAs resources. The MicroBlaze pipeline is a parallel pipeline, divided into three stages: Fetch, Decode, and Execute. In general, each stage takes one clock cycle to complete. Consequently, it takes three clock cycles (ignoring delays or stalls) for the instruction to complete. Each stage is active on each clock cycle so three instructions can be executed simultaneously, one at each of the three pipeline stages. MicroBlaze implements an Instruction Prefetch Buffer that reduces the impact of multi-cycle instruction memory latency. While the pipeline is stalled by a multi-cycle instruction in the execution stage the Instruction Prefetch Buffer continues to load sequential instructions. Once the pipeline resumes execution the fetch stage can load new instructions directly from the Instruction Prefetch Buffer rather than having to wait for the instruction memory access to complete. The Instruction Prefetch Buffer is part of the backbone of the MicroBlaze architecture and is not the same thing as the optional instruction cache.

Writing software to control the MicroBlaze processor must be done in C/C++ language. Using C/C++ is the preferred method by most people and is the format that the Xilinx Embedded Development Kit (EDK) software tools expect. The EDK tools have built in C/C++ compilers to generate the necessary machine code for the MicroBlaze processor. The MicroBlaze processor is useless by itself without some type of peripheral devices to connect to and EDK comes with a large number of commonly used peripherals. Many different kinds of systems can be created with these peripherals, but it is likely that you may have to create your own custom peripheral to implement functionality not available in the EDK peripheral libraries and use it in your processor system.

SOFTWARE

We used ISE development Suite provided by Xilinxs.and used verilog as our HDL

VERILOG

Hardware description languages such as Verilog differ from software programming languages because they include ways of describing the propagation of time and signal dependencies (sensitivity). There are two assignment operators, a blocking assignment (=), and a non-blocking (<=) assignment. The non-blocking assignment allows designers to describe a state-machine update

without needing to declare and use temporary storage variables. Since these concepts are part of Verilog's language semantics, designers could quickly write descriptions of large circuits in a relatively compact and concise form. At the time of Verilog's introduction (1984), Verilog represented a tremendous productivity improvement for circuit designers who were already using graphical schematic capture software and specially written software programs to document and simulate electronic circuits.

The designers of Verilog wanted a language with syntax similar to the C programming language, which was already widely used in engineering software development. Like C, Verilog is case-sensitive and has a basic preprocessor (though less sophisticated than that of ANSI C/C++). Its control flow keywords (if/else, for, while, case, etc.) are equivalent, and its operator precedence is compatible. Syntactic differences include variable declaration (Verilog requires bit-widths on net/reg types[clarification needed]), demarcation of procedural blocks (begin/end instead of curly braces {}), and many other minor differences.

A Verilog design consists of a hierarchy of modules. Modules encapsulate design hierarchy, and communicate with other modules through a set of declared input, output, and bidirectional ports. Internally, a module can contain any combination of the following: net/variable declarations (wire, reg, integer, etc.), concurrent and sequential statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed inside a begin/end block and executed in sequential order within the block. However, the blocks themselves are executed concurrently, making Verilog a dataflow language.

Verilog's concept of 'wire' consists of both signal values (4-state: "1, 0, floating, undefined") and strengths (strong, weak, etc.). This system allows abstract modeling of shared signal lines, where multiple sources drive a common net. When a wire has multiple drivers, the wire's (readable) value is resolved by a function of the source drivers and their strengths.

A subset of statements in the Verilog language are synthesizable. Verilog modules that conform to a synthesizable coding style, known as RTL (register-transfer level), can be physically realized by synthesis software. Synthesis software algorithmically transforms the (abstract) Verilog source into a netlist, a logically equivalent description consisting only of elementary logic primitives (AND, OR, NOT, flip-flops, etc.) that are available in a specific FPGA or VLSI technology. Further manipulations to the netlist ultimately lead to a circuit fabrication blueprint (such as a photo mask set for an ASIC or a bitstream file for an FPGA).

APPLICATIONS

- Exporting a Known C/C++ program and use keyboard and Displays to input and display
- Parallel Processing
- Displaying on Tft/CRT and other modes of I/O

FUTURE IMPLICATIONS:

- Neural network
- Designing our own processor
- Testing systematic
- A central platform for implementation.

VOTE OF THANKS:

We thanks IIT Kanpur and Electronics club for providing us an opportunity.We thank our co-ordinator Nikhil Gupta, Anurag Dwivedi and Rudra Pratap Suman for providing us the option And last but not the least I would like to thank my senior Chirag Sangani for encouraging us and sharing us his experience that led us in pursuit of FPGA.