RS232

Serial Communications Interface (SCI) is an old communication protocol originally used to connect input/output terminals to mainframe computers. The common name for this protocol is Universal Asychronous Receiver Transmitter or UART.
Serial transmission involves sending one bit a time, where the data is spread out over time. Engineers have found that one can send data farther, faster, less expensively, and more reliably using serial versus parallel channels. This is because it is easier to control capacitive loading and added noise within a serial cable. The total number of bits transmitted per second is called the baud rate. Most of the Freescale embedded microcomputers supports at least one Serial Communications Interface or SCI. Before discussing the detailed operation of particular devices, we will begin with general features common to all devices. Each SCI module has a baud rate control register, which we use to select the transmission rate. There is a mode bit, M, which selects 8-bit (M 0) or 9-bit (M 1) data frames. Each device is capable of creating its own serial port clock with a period that is an integer multiple of the E clock period. The programmer will select the baud rate by specifying the integer divide-by used to convert the E clock into the serial port clock. A frame is the smallest complete unit of serial transmission. The difficulty with serial transmission is synchronizing the receiver with the transmitter. With the SCI protocol, a start bit that always has a 1 to 0 transition is sent by the transmitter signifying the start of the frame. Figure 8.1 plots the signal versus time on a serial port, showing a single frame, which includes a start bit (0), 8 bits of data (least significant bit first) and a stop bit (1). The stop bit is also required so that when a start bit occurs there will be a 1 to 0 transition. This protocol is used for both transmitting and receiving. The information rate, or bandwidth, is defined as the amount of data or usable information transmitted per second. From Figure 8.1, we see that 10 bits are sent for every byte of usable data. Therefore, the bandwidth of the serial channel (in bytes/second) is the baud rate (in bits/sec) divided by 10.

INTERCONNECTION FPGA

A field-programmable gate array (FPGA) is a programmable logic device that supports implementation of relatively large logic circuits.
FPGAs are quite different from SPLDs and CPLDs because FPGAs do not contain AND or OR planes. Instead, FPGAs provide configurable logic blocks (CLBs) for implementation of the required functions (which is a completely different architecture from the one provided by SPLDs and CPLDs).
It contains three main types of resources:
-(CLBs) configurable logic blocks (containing the functional logic and sequential elements like FFs)
-I/O blocks for connecting to the pins of the package (they usually have some additional sequential element in block)
-interconnection wires and switches (there are horizontal and vertical channels use for interconnecting the CLBs. Channel intersections have, in general, a switch matrix)

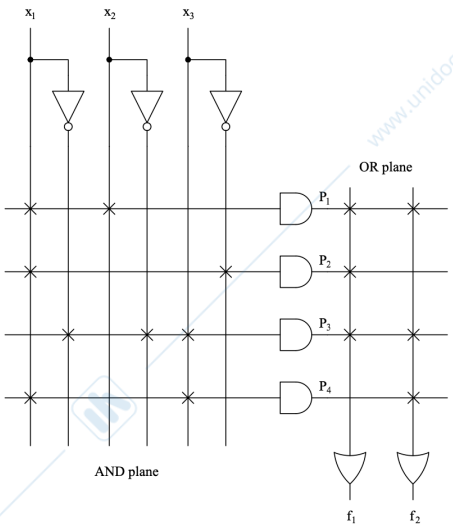
Each logic block in an FPGA typically has a relatively small number of inputs and outputs. Nowadays, a variety of FPGA products are on the market, featuring different types of logic blocks. The most commonly used logic block consists mainly in a lookup table (LUT), which contains storage cells that are used to implement a small logic function.
Each cell is capable of holding a single logic value, either 0 or 1. The stored value is produced as the output of the storage cell. LUTs of various sizes may be created, where the size is defined by the number of inputs.
It has two inputs, x1 and x2, and one output, f . The parameters A,B,C and D are bits and as such can assume the value 1 or 0.

After having introduced the general structure of a field programmable gate array, we now analyze a real structure of a real FPGA.
We will focus, in particular, on the Intel Altera Cyclone V Soc architecture.

PAL AND DIFFERENCE WITH PLA

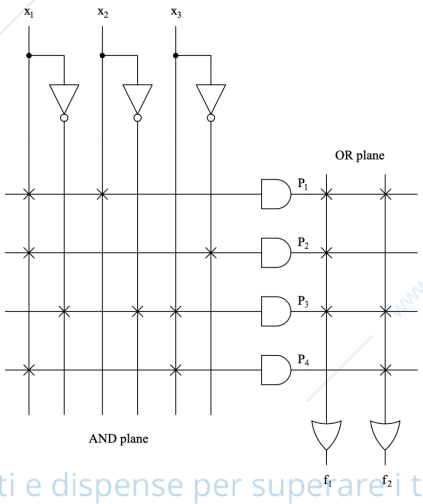
a PLD is a general-purpose chip for implementing logic circuitry.
It contains a collection of logic circuit elements that can be customized in different ways. A PLD can be viewed as a “black box” that contains logic gates and programmable switches
Due to the fact that each Boolean function can be expressed as a sum of products, we can develop a device consisting on the combination of two levels of logic, namely an AND gates array and an OR gates array.
There are three basic organizations depending on which of the AND/OR logic arrays are made programmable:
-PLA(ProgrammableLogicArray): both the OR and the AND arrays are programmable
-PAL(ProgrammableArrayLogic): the AND array is programmable where as the OR array is fixed
-PROM (Programmable Read Only Memory): the AND array is fixed whereas the OR array is programmable

A **PLA** is a programmable logic device that has both programmable AND array and programmable OR array.



Each AND gate is depicted as a single horizontal line attached to an AND-gate symbol. The possible inputs to the AND gate are drawn as vertical lines that cross the horizontal line.
At any crossing of a vertical and horizontal line, a programmable connection, indicated by an X, can be made (in our example, the proper connections in order to implement the described functions are made).
Each OR gate is drawn in a similar manner, with a vertical line attached to an OR-gate symbol. The AND-gate outputs cross these lines, and corresponding programmable connections can be formed.

he development of a similar device in which the AND plane is programmable, but the OR plane is fixed. Such a chip is known as a programmable array logic (**PAL**) device. Because they are simpler to manufacture, and thus less expensive than PLAs, and offer better performance, PALs have become popular in several practical applications.



So far we have assumed that the OR gates in a PAL, as in a PLA, connect directly to the output pins of the chip.
In many PALs extra circuitry is added at the output of each OR gate to provide additional flexibility. It is customary to use the term macrocell to refer to the OR gate combined with the extra circuitry.

BASIS FOR COMPARISON	PLA	PAL
Stands for	programmable logic array	programmable array logic
Construction	programmable array of AND and OR gates	programmable array of AND gates and fixed array of OR gates
Availability	less prolific	more readily available
Flexibility	provides more programming flexibility	offers less flexibility, but more likely used
Cost	expensive	intermediate cost
Number of functions	large number of functions can be implemented	provides the limited number of functions.
Speed	slow	high

COUNTER IN GRAY CODE AND MOST IMPORTANT COUNTERS

-Modulo-N counter: counts from decimal 0 to N– 1 (and then starts back from 0). (for example, a modulo-5 counter sequence in decimal is 0, 1, 2, 3, and 4).

-Binary coded decimal (BCD) counter: just like a modulo-N counter, except that N is fixed at 10. Thus, the sequence is always from 0 to 9.

-N-bit binary counter: similar to modulo-N counter, but the range is from 0 to 2n–1 and back to 0, where n is the number of bits used in the counter. (for example, a 3-bit binary counter sequence in decimal is 0, 1, 2, 3,4, 5, 6, and 7).

-Gray-code counter: the sequence is coded so that any two consecutive values must differ in only one bit. (for example, one possible 3-bit gray-code counter sequence is 000, 001, 011, 010, 110, 111, 101, and 100).

-Ring counter: The sequence starts with a string of 0 bits followed by one 1 bit, as in 0001.Thiscounter simply rotates the bits to the left on each count. (for example, a 4-bit ring counter sequence is 0001, 0010, 0100,1000, and back to 0001).

COUNTER WITH INC/DECREMENT AND PARALLEL LOAD

PARALLEL LOAD
Often it is necessary to start counting with the initial count being equal to 0. This state can be achieved by using the capability to clear the flip-flops as previously mentioned.
However, sometimes it is desirable to start with a different count. To allow this mode of operation, a counter circuit must have some inputs through which the initial count can be loaded.
A two-input multiplexer is inserted before each D input. One input to the multiplexer is used to provide the
normal counting operation. The other input is a data bit that can be loaded directly into the flip-flop. A control input, Load, is used to choose the mode of operation. The circuit counts when Load = 0.
A new initial value, D3 D2 D2 D0, is loaded into the counter when Load = 1.

UP COUNTER
In general, for an N-bit up-counter, a given flip-flop changes its state only when all the preceding flip-flops are in the state Q=1.

DOWN COUNTER
In general, for an N-bit down-counter, a given flip-flop changes its state only when all the preceding flip-flops are in the state Q=0.

INTERRUPT EXECUTION

Polling vs interrupt

As previously mentioned, peripherals are very important in embedded systems since they allow to interface the computing system with the outside world.
In particular, in embedded systems, the CPU serves and interacts with several peripherals.
The way for the microprocessor to receive information from the hardware peripheral can be implemented with two different approaches:
-polling approach
-interrupt approach
Polling and Interrupt let CPU stop what it is currently doing and respond to the more important task.

The time between the generation of the interrupt request and the entry into the ISR is called the "interrupt latency," and faster (lower latency) is always better.

When an interrupt request is received, the CPU:
-ends the execution of the current instruction (in the best case)
-stores the location of the next instruction it was going to execute (by storing that instruction address in a register or memory location)
-(often) saves the content of the CPU status register
-jumps to the code designated by the programmer for that particular interrupt
-executes some commands (i.e the interrupt service routine)
-once the servicing is completed, the processor would resume exactly where it left off

A microcontroller CPU will be designed to respond to a number of different interrupt sources and each source can have specific user-written code which executes when that interrupt triggers. The code that exe- cutes for an interrupt is called the interrupt service routine or ISR.
Interrupt service routines are triggered by specific hardware interrupt requests or specific software conditions

Normal program can be interrupted in three ways:
1. by external signals
2. by special instruction in the program
3. by occurrence of some condition

To handle interrupts from multiple devices, the processor has to perform the following tasks:
-it has to recognize the device requesting the interrupt
-it has to obtain the starting address of the interrupt service routine corresponding to the interrupt request. Notice that in general each interrupt request has its own interrupt service routine
-it has to manage the situation in which it receives an interrupt from a device while another interrupt is being serviced
-it has to take decisions on which interrupt should be serviced first when there are simultaneous interrupt requests from two different devices

TIMERS IN STM32

Another peripheral which is avaiable in most embedded system is the timer.
Timers are simple but very useful peripherals which basically consist in programmable counters and can be used to measure time events.

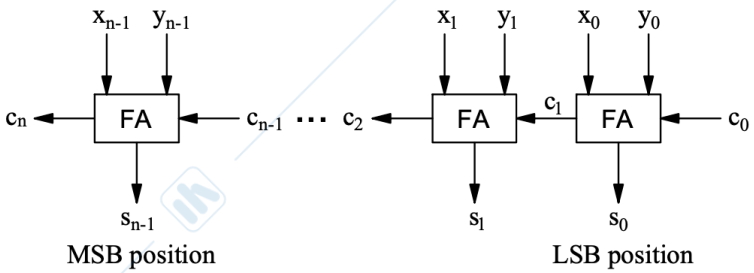
As for all peripherals, the timer unit can be programmed by configuring some special registers. For example, we can configure the prescaler for the timer, or the mode of operation and many other properties.
Timers count from 0 to a certain upper level which can be programmed. The count is also compared with a threshold value in order to understand when it is reached.
When this happens, either a flag (stored inside a register) is lifted or an interrupt request is generated.

using a timer allows to exploit the CPU only when the timer generates an interrupt request.

The most common applications are related to:
- periodic event generation
- PWM (pulse width modulation)
- counting external clock signals

STRUCTURE AND FUNCTION OF A 8-BIT RCA

To perform addition by hand, we start from the least-significant digit and add pairs of digits, progressing to the most-significant digit.
If a carry is produced in position i , then this carry is added to the operands in position i + 1.
For each bit position we can use a full-adder circuit



Each full-adder introduces a certain delay before its si and ci +1 outputs are valid. Let this delay be denoted as Δt .
Thus the carry-out from the first stage, c1, arrives at the second stage Δt after the application of the x0 and y0 inputs. The carry-out from the second stage, c2, arrives at the third stage with a 2Δt delay, and so on. The signal cn−1 is valid after a delay of (N −1)Δt, which means that the complete sum is available after a delay of NΔt.
Because of the way the carry signals “ripple” through the full-adder stages is called a ripple-carry adder.
Due to the fact that the carry has to "ripple" all the way up to the last stage, RCA (ripple carry adders) are not very fast.

BEHAVIOURAL SHIFT REGISTERS

A flip-flop is a 1 bit memory cell which can be used to store digital data. To increase the storage capacity in terms of number of bits, we have to use groups of flip-flop. Such groups of flip-flops are known as a registers.
The n-bit register will consist of n flip-flops and is capable of storing an n-bit word.
Binary data in a register can be moved within the register from one flip-flop to another. The registers that allow such data transfers are called as shift registers.

There are four mode of operations of a shift register:
-Serial Input Serial Output (SISO)
-Serial Input Parallel Output (SIPO)
-Parallel Input Serial Output (PISO)
-Parallel Input Parallel Output (PIPO)
Among these SISO, SIPO and PISO registers are shift registers, while PIPO registers are not.

INTERNAL OF A PROCESSOR CORE

The processor technology is the architecture of the computation engine chosen to implement a given functionality.
The term processor is often confused with the general purpose processor. A processor is any architecture that implements a given function, it must not necessarily be programmable or general-purpose.

There are 3 main possibilities when it comes to choosing a processor architecture:
1. General-purpose processors
2. Single-purpose processors (ASIC: Application Specific Integrated Circuit)
3. Application-specific processors (ASIP: Application Specific Instruction set Processor)

STRUTTURA DI UN CLB IN FPGA

The **configurable logic blocks** are arranged in a two-dimensional array, and the interconnection wires are organized as horizontal and vertical routing channels between rows and columns of logic blocks.
The routing channels contain wires and programmable switches that allow the logic blocks to be interconnected in many ways.
Programmable connections also exist between the I/O blocks and the interconnection wires.
The actual number of programmable switches and wires in an FPGA varies in commercially available chips.

CPLD

Complex programmable logic device. PLAs and PALs are useful for implementing a wide variety of small digital circuits. Each device can be used to implement circuits that do not require more than the number of inputs, product terms, and outputs that are provided in the particular chip. These chips are limited to fairly modest sizes, typically supporting a combined number of inputs plus outputs of not more than 32.
For implementation of circuits that require more inputs and outputs, either multiple PLAs or PALs can be employed or else a more sophisticated type of chip, called a complex programmable logic device (CPLD), can be used.
A CPLD comprises multiple circuit blocks on a single chip, with internal wiring resources to connect the circuit blocks. Each circuit block is similar to a PLA or a PAL; we will refer to the circuit blocks as PAL/PLA-like blocks.

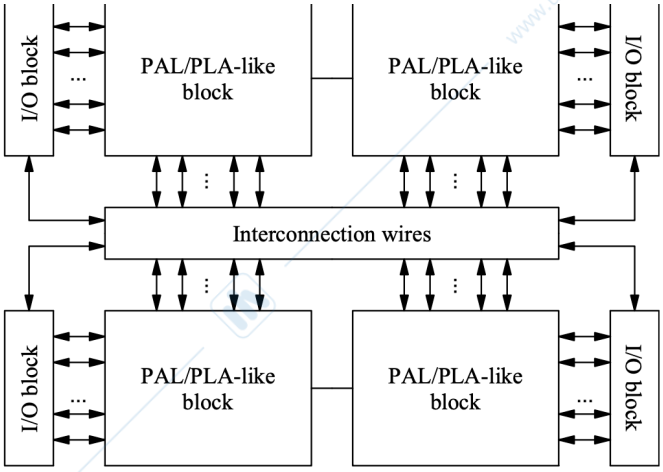


Figure 5.14: CPLD schematic

In general the set of interconnection wires is made up by a set of horizontal and vertical wires. Each of the horizontal wires can be connected to some of the vertical wires that it crosses, but not to all of them. Extensive research has been done to decide how many switches should be provided for connections between the wires. The number of switches is chosen to provide sufficient flexibility for typical circuits without wasting many switches in practice (we have already

mentioned that the higher the number of switches, the higher the cost, the higher the complexity of the PLD, the slower the performance).
Each PAL/PLA-like block is also connected to a sub-circuit labeled I/O block, which is attached to a number of the chip's input and output pins.

ONE HOT ENCODING FOR FSM (AND HOW TO CREATE COMBINATIONAL CC1 WITHOUT KARNOUGH MAP)

In the one-hot state coding approach, given N states, we can represent them using N bits.
To each state in the finite state machine we associate a single bit in the binary word using an invertible mapping.
Then, the binary code for a state S is the binary word made of all zeros but the bit associated to state S, whose value is instead one.

state	one-hot code
state 0	**1**0000000
state 1	0**1**000000
state 2	00**1**00000
state 3	000**1**0000
state 4	0000**1**000
state 5	00000**1**00
state 6	000000**1**0
state 7	0000000**1**

ble 13.3: One-hot coding for eight states

It is therefore evident that each state code in a one-hot encoding approach must contain exactly one bit equal to one (no more, no less).
The advantage of this approach is that, if we use a register to hold the current state code, the i-th flip flop in the register (that is the i-th bit in the word) will tell us whether the corresponding state is the current one or not.

In order to implement the finite state graph, for each transition on the diagram we must make sure that the circuit:
1. checks if the transition is going out from the current state (and is therefore a viable transition)
2. checks if the transition condition is satisfied
3. if both condition(1) and (2) are satisfied for a transition, the circuit must make the destination of the transition is set as the new current state

Notice now that with a one-hot encoding:
-in order to check if a state is the current one it is enough to check the flip-flop (that is the bit) corresponding to that specific state
-in order to check the input values it is enough to use a boolean function in the sum-of-products form. If to each transition is associated only one binary input pattern (that is only one possible combination of input values) then a single AND gate is enough to check if the condition for the transition is satisfied.
-To set the new state as the current one it is enough to write a logical '1' into the flip-flop (i.e. bit) corresponding to the new current state) and reset to logical '0' the old current state

We also have some general (not only for the simplified case) advantages from employing a one-hot coding approach:
-parallel branching is allowed. Parallel branches are two branches of the finite state graph that start from the same state and end in the same state

However, also some disadvantages can be identified:
-no flexibility is offered by the one-hot coding approach: the circuit must be redesigned when a change is performed
-circuits can become extremely complex if the number of states is greater than 20

www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari
www.unidocs.it - Appunti e dispense per superare i tuoi esami universitari