EEE4084F TEST 2 2018

ANSWERS

1.

(a) See textbook pg 151. Basically an offset error is an imperfection of the ADC such that it consistently produces an over – or under – estimate of the true analogue value measured. In particular an ADC could have a Positive Gain Error (PGE) which is above the actual value or it could have a Negative Gain Error (NGE) below the actual value. Indeed an ADC may have both, a NGE or a certain analogue range and PGE for other values.

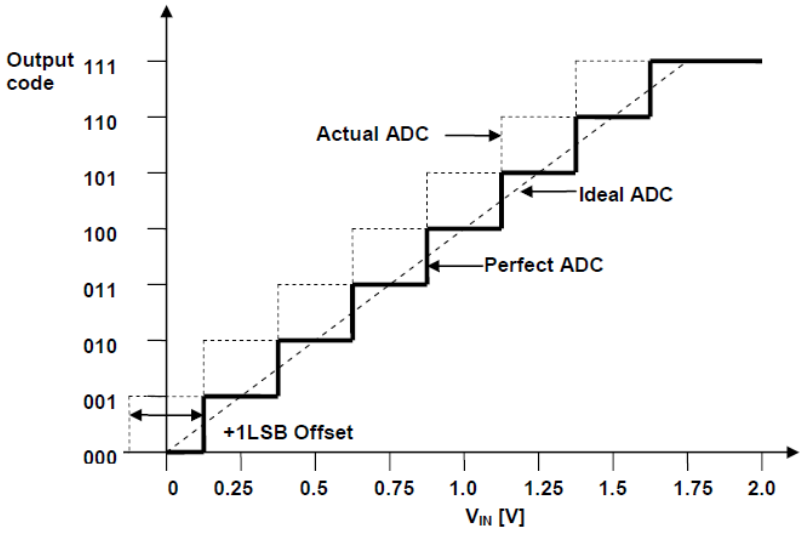


Figure 1: illustration of positive offset errors.

(b) i. it is simply 8-bit resolution ii. 256 codes (0 to 255) *{it was not a trick question!}*

(c) i. ENOB = Effective Number Of Bits.

ii. The ENOB usually decreases as the sampling rate increases, because:

iii. It is more difficult to design systems that will ensure the bits have all settled and reflect the instantaneous voltage properly before the next sample is made. Similarly the SRN tends to be higher for shorter sample periods, the sensed value is less certain, less averaged. See illustration…

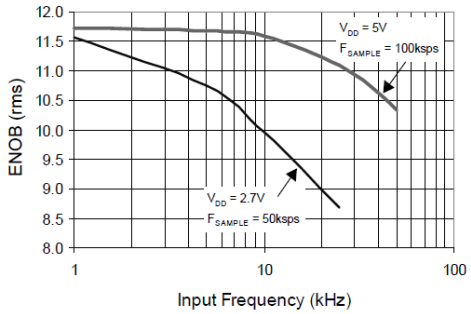


Figure 2: Figure from typical low-cost ADC datasheet showing the reduction in ENOB with sampling frequency source: http://microchipdeveloper.com/()

(d) i. A Flash ADC achieves its high sample rate at the cost of resources, essentially needing one less comparator that the number of codes that the ADC can produce. This means it takes more space and generally more power than the more compact counter-based ADC.
ii. See diagram below…

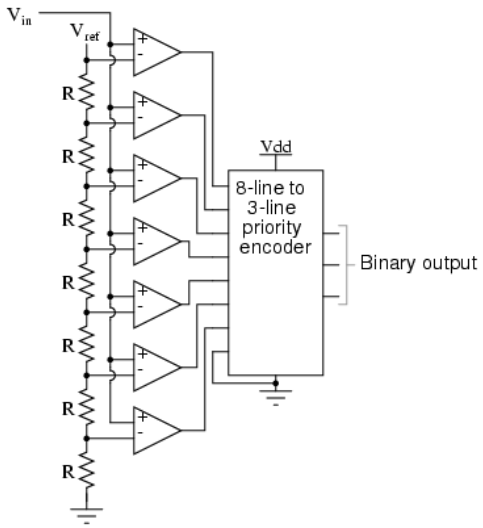


Figure 3: Flash ADC schematic

Marking comments: it is not necessary obviously for the student to draw quite such a detailed sketch, so long as there is an indication of the resistor ladder and comparitors going up the levels, and an encode that should be fine.

2.

(a) OpenCL is (currently) mainly designed around developing small kernel programmes that will do highly parallelized operation. Indeed (as mentioned in the lecture) the OpenCL compiler is generally built into the operating system of the host computer as part of the device driver. The C programs are typically under 1000 lines of code and can be compiled in a few milliseconds (the reasoning being that your OpenCL program would surely take many seconds or longer to run as serial code so that compiling step is negligible). In practice, the critical section code that you want to be parallelized tends to be a very small piece of the overall application – but the learning curve associated with that little piece can be massive. So the OpenCL philosophy is let's give programmers a tool they are comfortable with already (basically C) and add a few features to better support parallelisation and heterogenous architectures – so that basically it's almost just an increment to their existing programming practices instead of a whole new paradigm (which is essentially the case with CUDA and especially with FPGAs). Furthermore OpenCL is being set as a having a standard core that will be compatible with all OpenCL compatible devices, this makes code highly portable. It is something of a compromise, difference devices will allow extensions to the OpenCL standard (which will generally be close to C syntax) that would of course limit portability, but it is a compromise, but the entry point (being pretty much C) is much easier entry (less learning curve) to this approach of programming.

Marking comments: don't need such a long story, but read over it to get a gyst of what is so important about OpenCL and how it can bring developers into the area of heterogeneous programming.

(b) This is an easy one! You should have remembered it is basically Amdahl's law expressed in wallclock units. So it is
observed speedup = (wall clock time of sequential ver) / (wall clock time of parallel ver)

(c) An embarrassingly parallel solution is one that is basically very easy to make a parallel implementation for. Nothing stupid about that! (unless you don't notice the embarrassingly parallel problem in front of you).

(d) I would recommend using the Harvard Architecture. The Harvard architecture is an architecture that physically separate instructions and data. This has multiple advantages, such as eliminating the possibility of writes to data ending up changing the program and leading to a crash or providing a security loophole. It also allows a design simplification, where you could have separate external data for the instructions and data (if needed) which can be easier to manage, e.g. if the instructions needs to be stored in non-volatile memory and the data in separate volatile memory. For simple systems this can be an advantage (e.g. to eliminate the need to have some booting system to copy instructions into main memory so that the processor can run the main application).

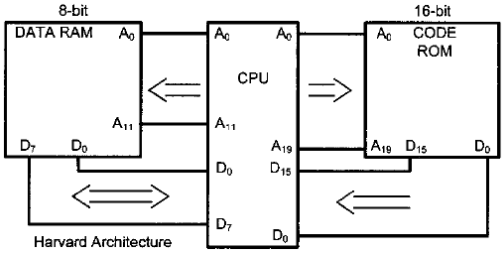


Figure 4: early Harvard Architecture

3.

(a) Kaizen is the philosophy, or rather the dedicated principle, of continuous improvement. It is a Japanese word and the concept (which has been around for 100s of years) comes from Japan.

3.

(b)

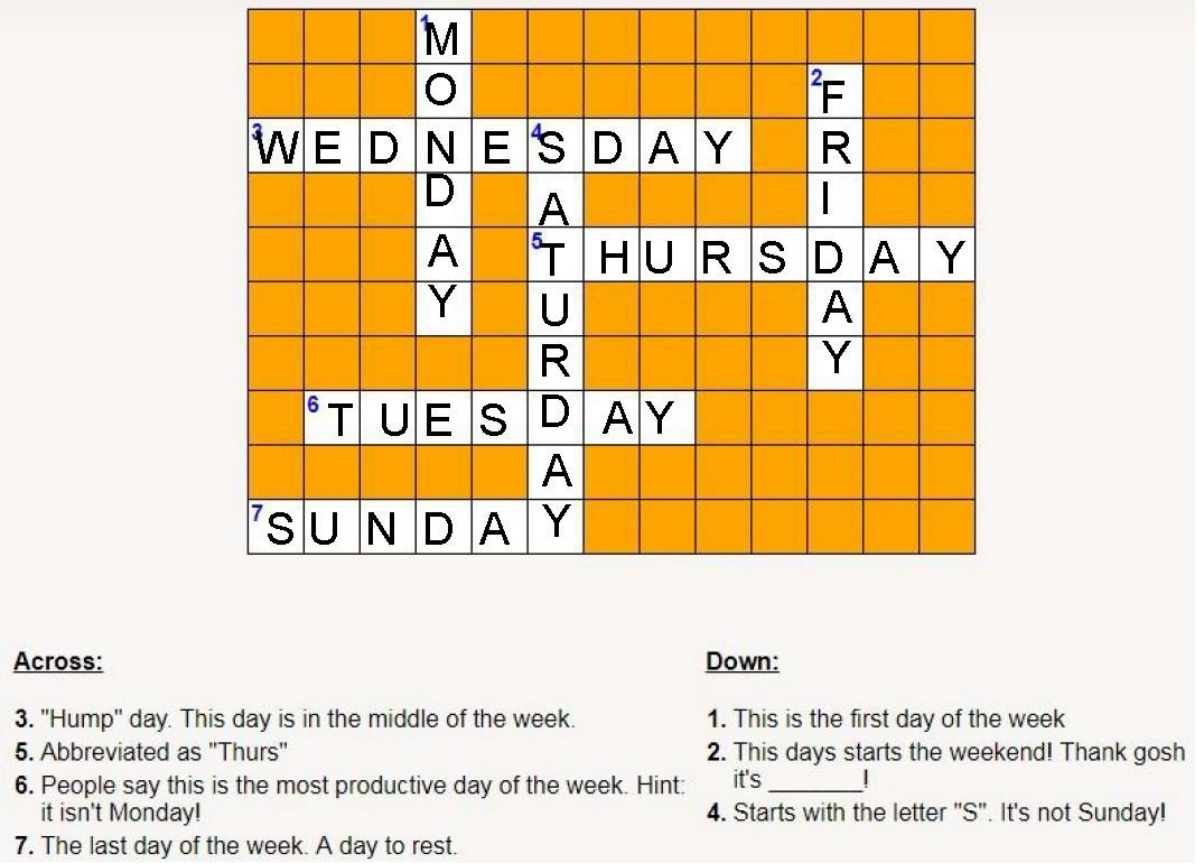
```
module flashingleds    (
   clk  ,   // clock
   reset,   // reset the system
   led      // the LED to flash
);
  //------------Input Ports--------------
  input wire clk, reset;
  //----------Output Ports--------------
  output reg led;
  //------------Internal Variables--------

  reg [19:0] counter;  // a 20-bit value should suffice

  //----------put your code here-----------

  always @(posedge reset) counter = 0;

   always @(posedge clk && (reset == 0))
    begin
      // simply toggle the LED every 100,000 clock cycles,
      //   since the clock is running at 10MHz
      if ( counter >= 20'd100000 )
         begin
          led <= ~led;
          counter = 0;
         end
    end
endmodule
```

(PS: a more thorough reset could trigger on the clock also and check if the reset line is low, this can ensure that the system sees a reset, particularly from startup, sometimes one needs special reset circuitry to make sure that the &*$@ FPGA actually sees the reset, excuse my French.)

Bonus answer:

A2 It's an Altera (fairly) Higher Performance and High Densitry

And finally the solution to the entertainment question ….



Across:

3. "Hump" day. This day is in the middle of the week.
5. Abbreviated as "Thurs"
6. People say this is the most productive day of the week. Hint: it isn't Monday!
7. The last day of the week. A day to rest.

Down:

1. This is the first day of the week
2. This days starts the weekend! Thank gosh it's _____!
4. Starts with the letter "S". It's not Sunday!