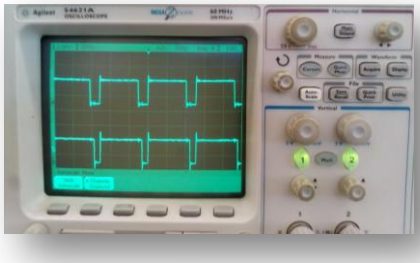
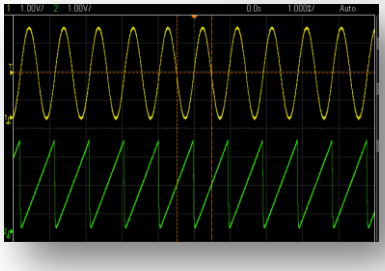
Australian National University

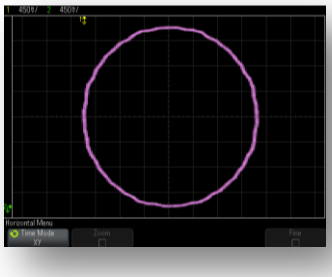
Assignment 01: digital-to-analog conversion via filtered PWM signals and XY image composition based on FPGA

ENGN3213/6213
Digital Systems and Microprocessors

Semester 1, 2015







ANU College of Engineering and Computer Science

TABLE OF CONTENTS

1. PROJECT STATEMENT

You are required to create a ***digital design to turn your FPGA development board into a simple user interface which allows a user (after having read this manual) to generate a variety of pulse-width-modulation (PWM) signals, which are then fed into an analogue filter (provided) to obtain waveforms of varying shape and amplitude***. Further information about PWM signals is provided in the Tips section.

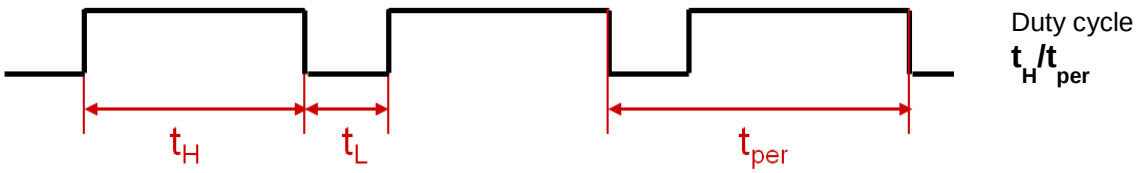
2. PROJECT SPECIFICATIONS

The overall aim of the project is to create a machine capable of four functions of increasing complexity. The functions are listed below:

2.1 Function 1: PWM generation and steady analogue output

The system should be able to generate PWM signals with 6-bit quantization (64 levels), that is, square waves with variable duty cycle between 0% and 100% with increments of 1.56% approximately. The frequency of the PWM signal will be 781250Hz (1/64 of the onboard 50MHz clock).

The design will run in positive logic (HIGH signals = value “1” = logic level “TRUE”), so the duty cycle is defined here as the time spent HIGH by the PWM signal over 1 signal period.



A combination of the 4 slide switches determines the duty cycle of the PWM signal, spanning the full duty cycle range. Note that you have 4 switches and 64 duty cycle levels so you will not be able to show every possible duty cycle ratio. The mapping of the switch combinations to the PWM ratios shown can be of your choosing, but this must be documented. The wave should be output through FPGA pin A6. The corresponding analogue signal observed at the output of the filter will be a constant voltage (save any small residual interferences) ranging from 0V for 0% duty cycle to 3.3V for 100%.

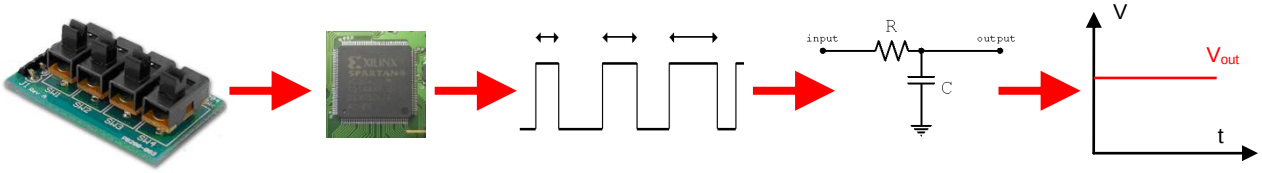


Figure 1: From slide switches to a PWM signal to filtering into an analogue signal

2.2 Function 2: waveform generation

As the next feature, your system should be able to modulate the PWM signal over time so that analogue waves of specific shape and magnitude can be generated through the filter. The following waves will be required:

- Square wave (50% duty cycle, high value 3.3V, low value 0V)

- Sawtooth wave (a 0V to 3.3V upward ramp terminated by a step returning to 0V, repeated)
- Triangle wave (signal ramps up 0V to 3.3V then back down to 0V, repeated)
- Sinusoid (sinusoidal wave, amplitude 3.3V)

The waves are controlled (turned on/off) by the slide switches. There are 4 switches, one for each wave type. Multiple waves can be enabled by using multiple switches. When multiple waves are enabled, the output signal is given by the sum of the enabled waves. To avoid signal trimming, the system will be required to automatically apply scaling to the waves by a factor of 2 or 4, as appropriate, to ensure values above 3.3V are not clipped.

Sample interval to describe waves in the time dimension (i.e., how many distinct digital values are processed in order to render one period of a wave) is required to be 1/64 of the wave period.

User interface requirements for function 2:

- *Slide switches* turn on/off the various waves.
- *Buttons East/West* can be used to increase/decrease the frequency of the output wave between 300Hz and 1 kHz in 100Hz increments

2.3 Function 3: XY signals

This function requires the system to generate two separate PWM signals simultaneously and output them from FPGA outputs A6 and B6. Both PWM signals are filtered through the provided filter. The purpose of the resulting waves is to drive the oscilloscope in XY mode (see Tips section for more details) in order to display a desired geometric shape on the screen. The frequencies of the X and Y waves can be freely determined by the designer.
The required shapes are:

- A circle
- A figure 8
- A square
- A solid square (filled)

User interface requirements for function 3:

- *Buttons East/West* are used to switch to the previous/next geometric shape output. The switching will be continuous, i.e., pressing *West* when the last shape is active will cause the first shape to be shown again.
- *Button North* turns on/off automatic switching between shapes. When automatic switching is active, the system shows the next shape after a human-appropriate time delay and continues to do so until the mode is turned off.

2.4 Function 4: Custom shape drawing

The most difficult function of them all!
This function allows the user to draw a custom shape on the oscilloscope's XY screen. Upon initialisation, the output of this function will be a blank screen with a single dot (corresponding to constant signals at 0V,0V). Based on button presses by the user, the system must generate and output appropriate periodic PWM signals so that:

- the dot displayed on screen follows the trajectory specified by the user, going either up (North button), down (South button), left (West button) or right (East button)
- when one of the slide switches is turned on (designer choice of which one to use), the locations touched by the dot while the switch is on will be memorised in the system and will remain lit up even after the dot has moved on. This allows for a shape to be drawn.

The frequencies of the X and Y signals can be freely determined by the designer.

User interface requirements for function 4:

- *Buttons North/South/East/West* move the dot's current position
- *One of the slide switches* enables/disables memorisation of dot's location
- *Consider reducing the need for multiple button presses for repeat steps in the same direction,* whether you associate prolonged button holding to multiple movements or recruit a switch for this purpose, or choose another method is up to you, just ensure it is user-friendly and well documented.
- *A reset user action* (e.g., through pressing simultaneous buttons or flicking a switch – designer choice) resets the screen to the initial value

2.5 All functions working together ("Function 5")

The four functions can be implemented as separate designs*, but you **will get extra marks** for creating a single state machine which controls the switching between the different functions*. Since your four buttons are taken up by the four functions, you can consider using the big centre button to switch between modes, although this is not a strict requirement. I also recommend using your LED array in an informative LED output to make the user aware of what function they are using.

3. RECOMMENDED WORKFLOW

As you will have noticed, the assignment is defined in terms of levels of increasing difficulty. The following recommendations *are not prescriptive* but they might help you find your way through to the end of the project:

1. Find a partner and register your group on Wattle; select time slots for lab work on doodle (read about how that works in the lab-scheduling section)
2. Read the tips section towards the end of this document for some practical suggestions to get you started
3. Complete function 1. You can probably do most of this design in simulations before you move on to the hardware.
    - Simulations might be easier at submodule level as this allows for simpler test benches. Once you are certain that your submodules run as intended, assembling them in a top module should be much easier to do (and troubleshoot).
4. Once you have tested and confirmed that you can deliver a suitable PWM signal in the lab, develop a solution for function 2.
5. Consider the requirements of function 5 and create a state machine which can toggle between the first two functions and operate them independently.
6. Test all the work done so far in hardware.
    - *If it all works, and if you clearly document your results, this constitutes an above pass result for the assignment*
7. Then, develop and hardware-test function 3.
8. Integrate function 3 in your existing machine.
9. Depending on your level of confidence in your skills and/or on how much time you have got you can at this stage stop working or start to work on function 4.
    - *if all work done this far operates very well and the document is well written, this is a distinction outcome for the assignment*
10. Then, develop and hardware-test function 4.
11. Integrate all functions into one overall state machine.
12. Test, test again just to make sure, document, don't make any last-minute changes, submit.

These are just suggestions. You may decide to work differently.

4. DELIVERABLES AND ASSESSMENT

4.1 Assessment process

For this project **students will work in groups of 2**. I have recommended that you work with your lab partner but you can ultimately choose any student in the course. The work will be assessed as group work, with the same mark awarded to both members. There are two exceptions to this:

- where one of the group members has not contributed to the project in a fair manner. Given that there is no group presentation, the quality of team work will be tested through an honesty system based on individual declarations (more on this later);
- for *Masters students* there will be an additional question which will be **submitted individually** (and individually marked).

The assessment of the work will be done by testing the design implementation (testing conducted by the tutors) in hardware and using the report as documentation to ensure that all design choices have been well thought out and justified. Testing will probably be conducted during weeks 9 and 10.

Written project document

Your delivered project document should give a *clear and concise but complete presentation of your work*. You should imagine that your document is to serve as a technical manual to be read by a technically-knowledgeable user (the lecturer/tutors know a lot about electronics but nothing about your design, unless you tell them). At a minimum your document should include:

- a clear description of the functions' user interface (how can a user operate your design: what inputs need to be asserted to perform operations and what outputs are expected)
- a detailed technical description of the structure and operation of your system, i.e., how the inner workings of your design allow it to achieve the required behaviour. At a minimum this should include:
    - a block diagram at submodule level, where by submodule I mean a functional subunit of your design. You should not show every logic gate or flip-flop in your design, but I also do not want to see too many "magic" black boxes in your circuit diagram whose operation is mysterious (e.g., a "clockdivider" block clearly identifies a common digital design and does not require much more to be said, while a block called "PWM generation" would require a much more detailed explanation of what lies inside).
    - where you have used state machines, a state transition diagram (and tables where they provide additional information) is essential, along with an explanation of your choice of states. Explicit descriptions of the next state or output logic equations may or may not be particularly significant depending on your design choices.
    - motivation of your design choices (nothing too verbose: it should be a commentary on your block diagrams and, where applicable, state representations. For example: *function xx is enacted through the use of a down-counter enabled by signal YY*. *This solution was chosen as it avoids duplication of a bank of flipflops and saves hardware resources.*)

Whatever you do, *do not write an essay*. Your reference document for the style of your "paper" deliverables should be a technical data sheet. We will discuss documentation standards during one of the lectures. I do not expect the final document to exceed 7-8 pages. (there is no hard limit but overlong documents may be penalised on clarity grounds).

You can organise your document by separate functions if it helps, but remember that what is expected is the **technical description of one complex machine with several functions, not four separate machines.** It is most likely, for example, that some submodules in the design will be shared

amongst the various functions and this should be commented upon in your manual as part of your description of the one overarching machine.

Code

Your code will not be marked for programming style, but it *will* be looked at, especially if the report is not very clear or the project does not work hence requiring us to dig into the program to see what partial marks can be awarded. Use comments for your benefit and the markers'.

You are required to submit a functioning, compilable version of the Verilog code corresponding to your final implementation as an attachment to your report. This version will be compiled for deployment on the day of the test.

Group work declaration

Regardless of the overall joint submission, every student will submit a one-page group work report. This will be a form where you will describe your own contribution and your partner's. The content of the form will not be disclosed to the other group member. The markers may compare the two and where significant discrepancies are observed, a penalty may be imposed. Failure to submit this component will result in a *penalty of 10%* on the overall mark.

4.2 Due dates

The project submissions will be done through Wattle. Submissions are due **before the end of the day (11.55pm) on Thursday 30th of April** (mid-week of week 9).

Extensions are unlikely to be granted because in Week 9 C/HLAB activities resume and you will be receiving the briefing for Assignment 2 so there is very little room for flexible arrangements. The maximum I can envisage is a day or two *for people with exceptional circumstances*.

4.3 Marking Criteria

The project will be marked as follows:

ITEM	WEIGHT
FUNCTION 1	23%
FUNCTION 2	20%
FUNCTION 3	17%
FUNCTION 4	20%
FUNCTION 5	10% if two functions operate under the same state machine 15% if three functions operate under the same state machine 20% if four functions operate under the same state machine
EXTRA QUESTION **Master students only**	11% (Master students are marked out of 111% and then scaled to 100%)

For each component of the design project, the following table shows how partial mark is made up:

ASSESSMENT CRITERION		WEIGHT
Clarity of written documentation	Have the students provided documents which clearly show their design approach, the design structure and the features of their signal generating machine? Does the document allow the markers to gain a clear, comprehensive understanding of the design without having to refer to the Verilog code?	30%
Quality of design	Is the design structured in a clear and logical manner? (e.g., are sequential and combinational logic sections clearly separated, are submodules used appropriately to isolate specific functions, have states been assigned in a logical fashion?) Is the design synchronous? Is the likelihood of malfunction due to timing hazards low? Have the designers considered (where applicable) potential issues with dealing with asynchronous, potentially noisy external inputs? Have the designers created a design which is neat in the sense of avoiding the use of excessive hardware resources? Have the designers paid attention, in formulating their design, to incorrect inputs which may be asserted accidentally? Does the design respond to these concerns in a reasonable manner? (i.e. where accidental inputs are not considered, why has it been deemed safe to do so? When unwanted inputs may create ambiguity or other operation issues, does the design deal with these appropriately?)	30%
Correct operation	Does the system operate as required?	30%
Teamwork	Does the group appear to have worked well together as a team?	10%

Late delivery of the project work will incur a penalty of 10% per day (I realise that this is harsh but it is the School of Engineering policy, so *make sure you are on time*). This project is worth 18% of your total course marks.

Plagiarism will not be tolerated. The markers will make a deliberate effort to identify non-genuine work, including the use of software tools to find similarities between submitted designs. Several offenders have been caught in past years, so please make sure you are the author of your own work as ANU penalties are very severe. *You must acknowledge any external sources* (if any) you may have referred to in creating your design and specify the extent of the impact of said external resources on your work. If you are in doubt, ask questions before making your submission.

5. OPEN-LAB TIMES

You will have the following times to work in the R103 lab.

- **Friday 27 March** (morning: 9am-1pm)
- **week 7** (9am-5pm every day *except lecture times*)
- **week 8** (9am-5pm every day *except lecture times*)
- **Friday 17 April and Saturday 18 April** (9am-5pm. This is during the teaching break and it includes a Saturday for students who have difficulties getting to ANU at other times)
- **Monday, Tuesday, Wednesday of week 9** (mornings only, *except lecture times*)

NOTE: during the teaching break supervision will not be available so the lab will not open on days other than the days specified.

The lab will have minimum supervision so please be careful and considerate with the equipment. Also please do not bring any food/drink into the lab and make sure you wear enclosed footwear.

Open Lab scheduling is based on a public calendar developed using the web platform Doodle. This is done to ensure that every student can get a fare share of lab time. This is how it will work:

1. after you register as a member of an Assignment 1 group in Wattle, you will be able to see a link to the Doodle page for the lab time slot reservations
2. follow the link and choose what time slots work best for your group. Please read the times carefully as slot times may vary on different days. There are 54 time slots in total, with a maximum of 14 groups per time slot, each group can select up to **10** time slots. Enter your name as GroupXX where XX is your group number. NOT your student name or ID
3. You can then turn up on the day/time you selected and have access to a workstation for the duration of the time slot.
4. If you make a mistake you can edit your selection at a later time, or email the lecturer for changes if that does not work
5. If you cannot attend a time you scheduled yourself for, please remove that entry from the calendar so other students know they can use the lab. Clearly, you are not allowed to de-register from a time slot once that time has passed.
6. If a group is found to have allocated themselves more than 10 time slots, the system will regularly perform a check and delete *all* of their selections and the group will then be forced to re-register. Please follow the rules or you will risk losing your favourite slots.

A note about overcrowding

We have limited facilities in R103 with 14 computers, *development boards should not be removed from the lab under any circumstances*. With 70+ groups, overcrowding is a potential concern.
A lot of open lab time has been allowed (20 hours of board time per group scheduled time). Still, in the interest of this working smoothly, please see the following recommendations:

- **don't leave your project to the last minute,** the project is long and access to resources is likely to become tighter as the deadline approaches
- **conduct design and simulation work outside the lab**, e.g., on your laptops and/or on other ANU computers.
- **strictly keep to the times you have registered your group for**
- **at no time should a group take up more than one workstation in the lab**
- **if a time slot has vacant workstations people may join on a first-come first-served basis, but people registered should always have precedence (i.e., if a registered group arrives late and finds someone has taken their workstation, they may ask the unauthorised group to leave)**

- **if there is a complaint about a group impacting unfairly on other groups' time slots, this may result in a *penalty on the final mark of up to 5% per complaint, at the discretion of the lecturer*. The tutors, lecturer and lab assistants will monitor the situation and ensure times are adhered to.**

For people developing on their own board and requiring access to a filter/oscilloscope, arrangements may be possible. Please inform the lecturer if you need this service and we will try to organise something through our lab technicians.

6. TIPS AND USEFUL INFORMATION

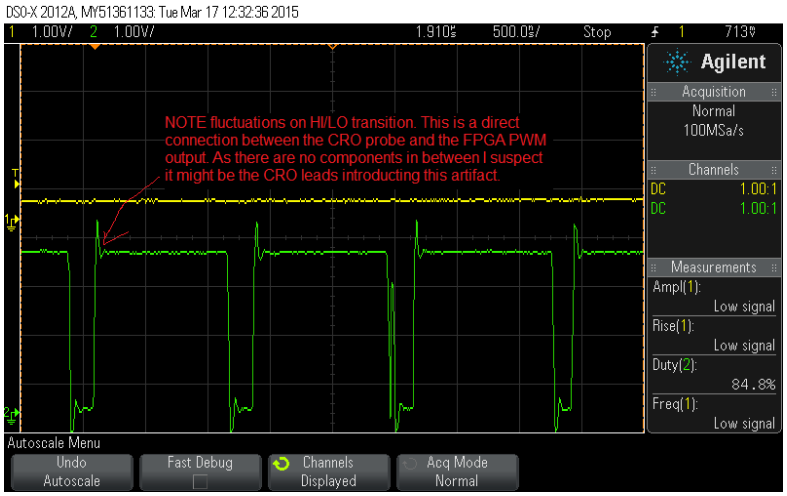
Function 1:

For you to successfully address this function, it is essential that you understand clearly what a PWM signal is, i.e., a square wave where the duty cycle can be varied. This page has some broader information about uses and characteristics of PWM signals.

You can verify that your signals are correct:

- through simulations, by measuring the ON/OFF times of your signal
- in the lab, by connecting the oscilloscope to the output port of the development board to view the signal before and after the analog stage. Knowledge of how to use an oscilloscope is a pre-requisite for the course but if you do not know how to use one please look up some resources and/or check out this tutorial.

A PRACTICAL NOTE: If you use the oscilloscope and inspect the fast-switching PWM signal straight out of the FPGA, you may see some overshoot/undershoot artefacts similar to those shown in the picture below.

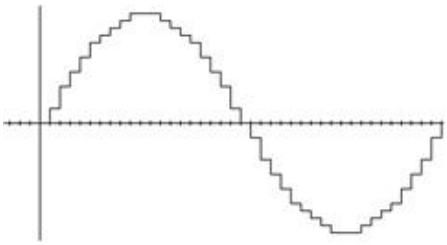


These are just artefacts and you should not be concerned about whether the PWM square wave you have created is having issues. Trust us, it is just fine; however, the oscilloscope probes do struggle to deal with the ultra-fast signal transitions from 0V to 3.3V and vice-versa generated by the FPGA. The only suggestions to reduce this behaviour are to use the SLEW=SLOW and DRIVE=2 as additional properties in your UCF file when creating the line that relates to your output pin A6/B6, although that may not resolve it completely. Don't waste time troubleshooting these artefacts as the output of the analogue filter is going to be nice and accurate regardless.

Function 2:

Devising resource-efficient ways to vary PWM over time so that the right waves are generated is the greatest challenge in this section. Once each individual wave works in its own right, dealing with adding and scaling multiple waves and frequency changes can pose a few additional hurdles – there are many possible solutions, some will be neater than others. Try to create submodules that are versatile... don't be tempted to design a specific waveform generator for every wave type/amplitude/frequency or you'll run out of space!

CONCEPTUAL NOTE: it is important that you conceptually understand that in this digital-to-analog approach you will be approximating a target analogue wave with a "staircase" wave (see below for a staircase sine). The more steps are used, the better the resolution will be and the more accurate the final approximation of the ideal wave.



This staircase is determined by the height and width of each step. The height resolution is written in the specifications as $1/64^{th}$ of full output (3.3V), while the width of each step (interval length in the time dimension) is specified as $1/64^{th}$ of the period of the wave, but do note that the period of the wave changes as you are required to change the frequency.

Function 3:

This function requires you first to figure out how to set and use the oscilloscope in XY mode. In XY mode, the oscilloscope will transform the magnitudes of the two input channels into the X and Y coordinates of the trace shown on the screen.

You will need to analyse the required shapes to figure out what two waves generated in parallel can provide the right sequence of XY coordinates to achieve your shape. You will also need to address the problem of generating two waves simultaneously as your functions 1 and 2 only require a single wave.

A state machine may assist with shape switching...

Function 4:

This section has a focus on user interactions. Always make sure you debounce controls where appropriate. The challenges here are posed by user-dictated events and by the need for appropriate structures to store the image as it is modified in real-time by the user. Have fun with the mapping of the screen!

(less hints here as this is one for the top students)

Function 5:

If you think of your design in a coherent manner, you should be able to have multiple functions exploit the same submodules. This might save you a lot of hardware resources and coding.

Could the wrong button pressed at the wrong time cause your system to malfunction?

General suggestions:
- Is your system initialised/reset to a known state when needed?
- Have you avoided inferred latches (incomplete conditional statements in combinational logic)?
- Are all of your flip flops running from the same clock? (synchronous machine)
- Are you dealing with any asynchronous signals appropriately, including external inputs?
- Are you using the correct pullup/pulldown settings for your buttons/switches?
- Have you included the system clock in your UCF file?
- With the exception of Function 5, you do not have to use state machines if you do not want to. But there may be other areas where state machines could simplify your approach. Some designs can be conceived as master machines which selectively enable/disable/control other parts of your design (a bit like the exercise you did in HLAB3).

One final note. The assignment is designed to be challenging and **it is anticipated that not all students will be able to complete all of the work**. That is why it is designed in stages. Completing some sections well is likely to give you a better outcome than submitting all parts half-done. Be ambitious but realistic.
Good luck!

7. ADDITIONAL QUESTION FOR MASTER STUDENTS ONLY

Consider a variation of Function 2 where the system does not select between different waveform options but instead outputs a wave according to data received from another device (e.g., a PC) via serial connection (could be USB, RS-232 or other serial communication, the details are not crucial) and stored in a memory implemented in the FPGA.

In a short exposition (500 words or less), discuss:
- what memory you would implement (type/size) to store the waveform information, assuming it uses the same quantisation/resolution used in the assignment
- potential challenges associated with the serial data transfer, which may include
    - clock domains,
    - ensuring data completeness and/or integrity, assuming one-directional communication only (i.e., the FPGA cannot communicate back to the source device)
    - choice of data receive registers

Remain at a high level in your answer and don't enter into the fine details of submodules (however, you do need to show a conceptual understanding of any solutions you describe). The items above don't need to be discussed one at a time, you can present an organic answer. You may provide a simple sketched diagram but only if it assists your explanation.