Week 1 Intro Slides

Monday, January 3, 2022 1:44 PM

- Logistics:
  - in person: 12:30-3:20 pm EE137B
  - groups of 2
  - 3 labs, each 3 weeks
  - lab kit: will get DSP in lab
  - download CCS w/ C2000 support } available on PC 01-10
  - download PLECS
  - prey EE447, EE452
  - assume some programing experence
- Agenda:
  - Intro to RT simulation

- Intro to C and CCS

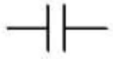
- PLECS Review

Background of real time simulation

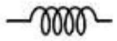
- In this section, we will learn about real time simulation
- We will use a basic buck converter as an example
- We expect you to know the following
  - circuit theory
  - fundamentals of PWM converters

Real time simulation

In real life, we have physical components like capacitors, inductors, devices relating voltage through them and current across them by some mathematical equations.



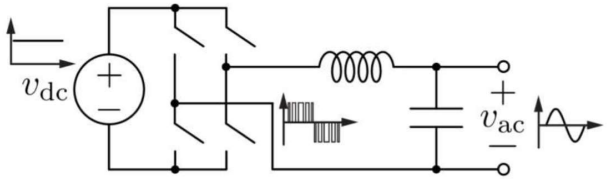
$$i = C\frac{dv}{dt}$$



$$v = L\frac{di}{dt}$$

$$v = iR$$

In simulation environment, the algebraic and differential equations are solved numerically.



A power converter is made up of many such equations

$$\dot{x} = f(x,t)$$
$$g(x,t) = 0$$

The idea of time step

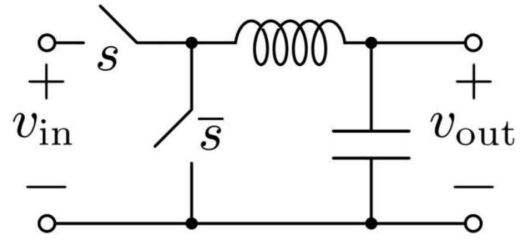
Let us say a buck converter is switched

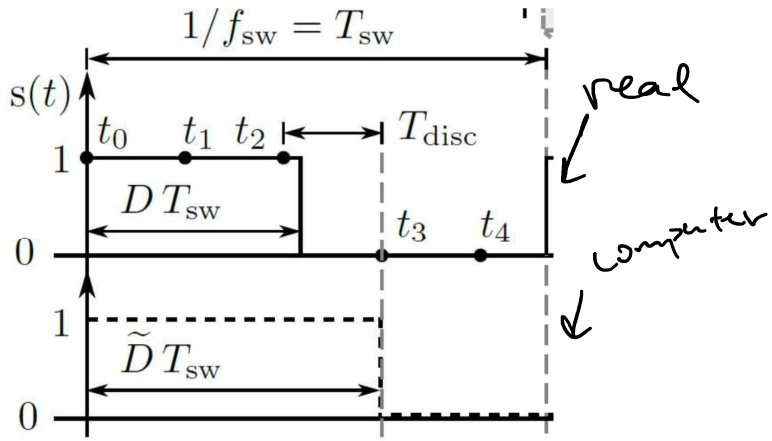
The simulation calculates the value of variables like voltage and currents of different active and passive components at **certain fixed intervals.** This is called the discrete time step.

From the figure, we can see the finer the resolution, the lesser the error between the actual switching in real world, vs what the simulation predicts.

Error in resolution : $|D - \widetilde{D}|$

From fig, we can say : $|D - \widetilde{D}| < \frac{T_{disc}}{T_{sw}}$





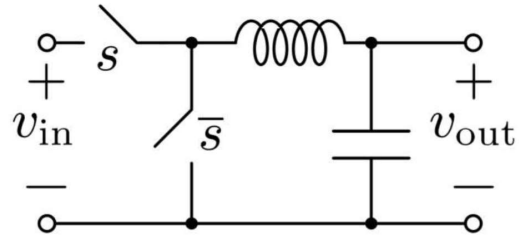
$f_{sw} \uparrow$, $T_{sw} \downarrow$, error $\uparrow$

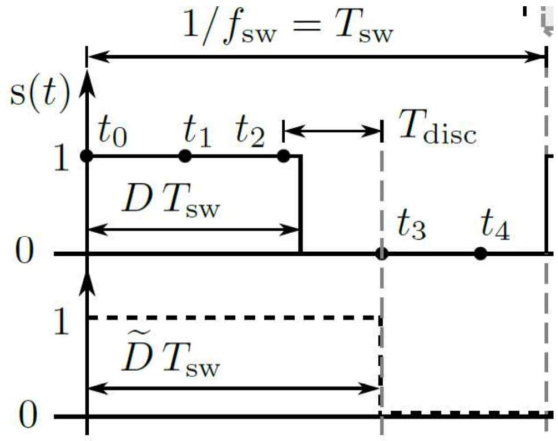
How small can the time step be?

In simulation?
Make time step as small as possible, say 1ns, it still works.

What does it do?
Lets say, converter switching frequency is 10 kHz. So, $T_{sw} = 100\mu s$.
Now, In each switching cycle, we have to solve, all the algebraic and differential equations $\frac{100\mu s}{1ns} = 100000$ times, with each step solving from where the last step left off.

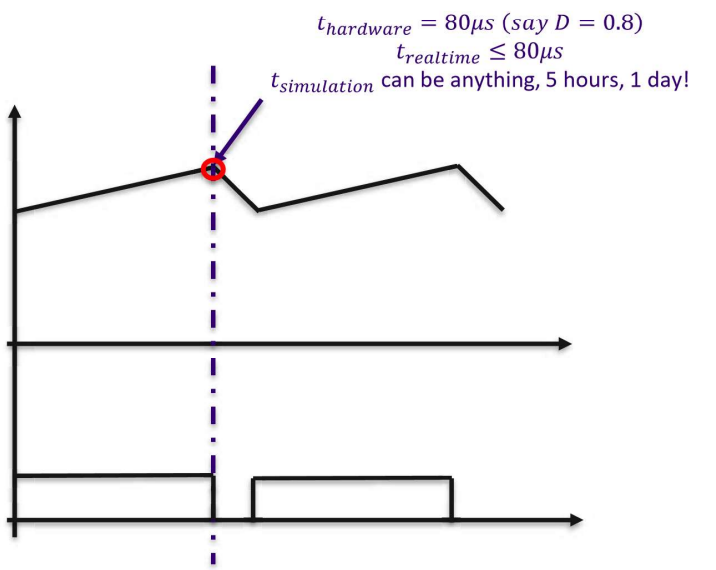
In PLECS simulation, this might take about an hour, to get a whole 1s simulation to complete

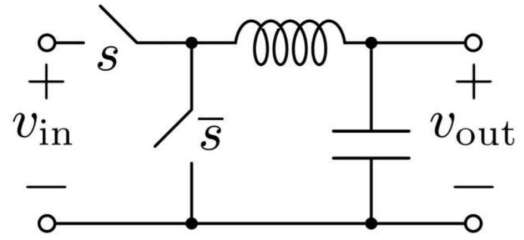


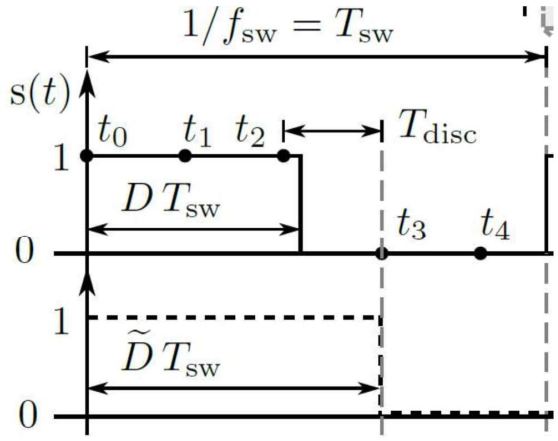


Idea of real time

In this same simulation, what we want is, the following: Same waveforms at same time!

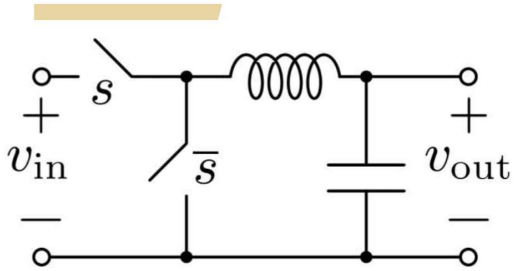


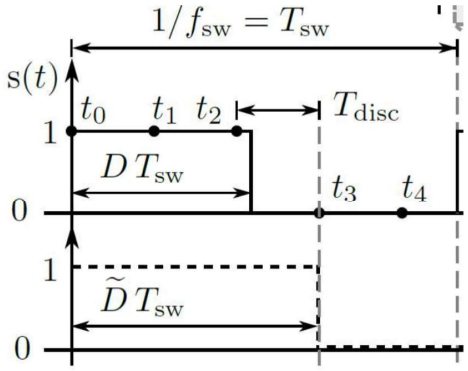


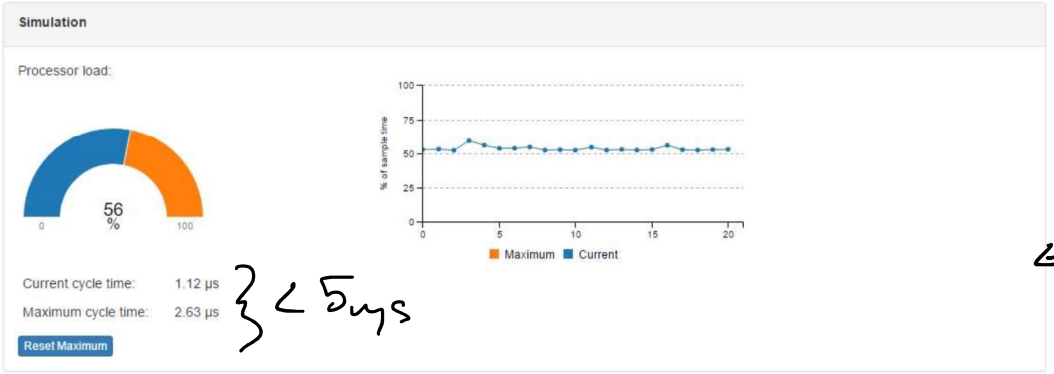


- The value of the inductor current should be say, 3.4 A in the hardware at the end of the S on duration. The simulation should exactly give that value, if not, then it would give an error message to reduce the time step. In the real time, it should take lesser time than real time and approximate the solution as close as possible

PLECS Realtime Simulation







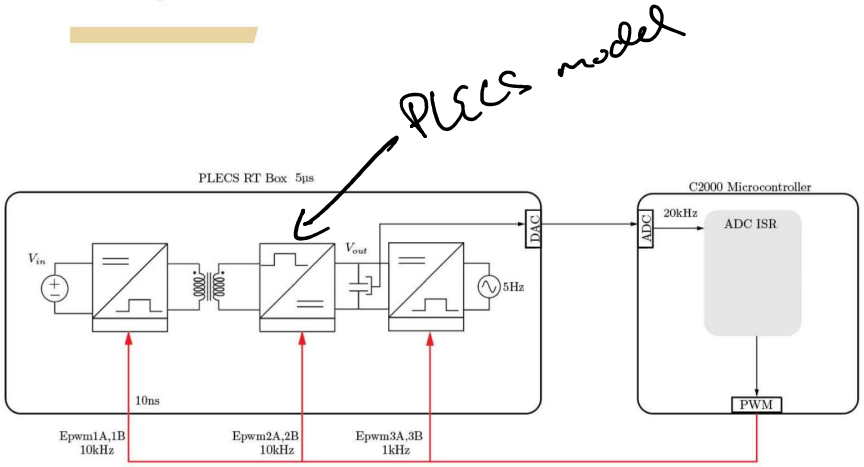
$< 5\mu s$

$T_{sw} = 100\mu s$
$T_{disc} = 5\mu s$

← RT Box Web interface

In this implementation, it shows that, all calculations in each step takes less than the step duration of 5us, so, all the predicted values of current and voltages are correct, and we *are doing* real time simulation

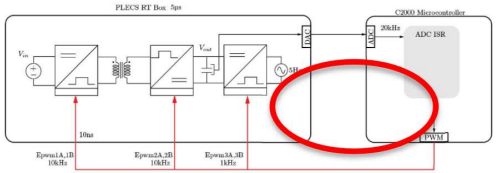
Why do we need real time simulation: Hardware-in-loop



We want to study the controller performance. So imagine from the real world, physical hardware, we keep the controller intact and just replace the actual hardware with the box in picture.
The RTBox now ensures that it is an exact time-synced replica of the hardware. So new control algorithms can now be written on the microcontroller and we will be able to predict, from the behavior of real time, how the actual hardware will behave.

Hardware-in-loop

Here the HIL is the controller. It is like a closed loop.



The advantages of HIL are :

- No associated hardware risks, or cost involved
- Large number of uncertainties (signal corruption through power supplies, faulted gate drivers etc) that contribute to erroneous results cannot be isolated from control performance. RT Box (HIL) ensures we decouple control performance from all such uncertainties.
- Extremely quick execution from design to verification

Getting started with CCS

- In this section, we will learn how to start with CCS and embedded C coding.
- Microcontroller used : TI 280049C (Launchpad version)
- We expect you to read the following **before starting Experiment 1** (see Canvas Files/Lab/Reference Materials directory).
  - ADC and DAC submodules (F280049C_ADC.pdf)
  - EPWM submodule (F280049C_EPWM.pdf, understand Figs 2.7 through 2.10)
  - Background reading of this specific microcontroller (F280049C_Overview.pdf)

read through to start familiarizing, don't need to understand everything.