1 Week 7: Ruby serial primitives

This part of work is about how to compile Ruby serial primitives to MaxJ code, mainly `pdsr` and `sdpr`.

1.1 `showGate` and Function Device

`showGate` is the core function to generate the MaxJ expression of a Ruby circuit. For instance, `add` primitive will be transformed to "+" by using `showGate`. But things will be different for `pdsr` and `sdpr`, as `showGate` currently could only support infix, eq and some other basic primitives, there's nothing like "function" has been supported, which will be quite useful to definie `pdsr` and `sdpr`.

Briefly, my approach is:

1. Add a condition filter `isFunc` in `showGate`, to check whether the current device is a **function device**, which should be implemented like `func(t1, t2)` in MaxJ.
2. Append the builtin function definition backward in the kernel class definition.

1.2 `pdsr` and `sdpr`

How to define these 2 primitives in MaxJ? The simplest way is using a combination of **counter** and **mutiplexer**, which should be `simpleCounter` and `mux` in MaxJ. A simple definition for `pdsr 2` is:

```
private DFEVar pdsr2(DFEVar t0,DFEVar t1) {
  DFEVar counter = control.count.simpleCounter(
                        MathUtils.bitsToAddress(2));
  return control.mux(counter, t0, t1);
}
```

This snippet of code is auto-generated, and suit for different value of n in `pdsr n` expression.