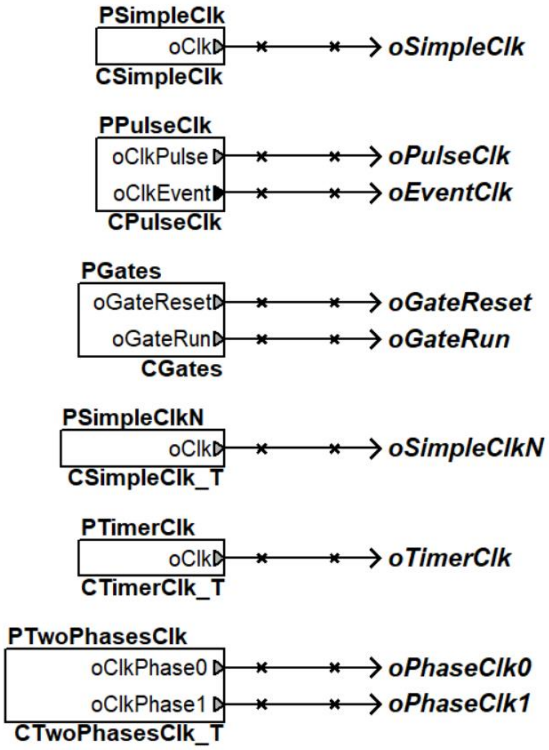
# Library to generate clocks

## Description

Clocks are often required in FPGA applications to synchronize actions with the real time or events, or to enable operations. This library includes 6 examples of clock generators:




## The psC code

The code for each component is self-explanatory, here is a description of each clock generator. The sample code is part of the component code and shows how each clock is generated.

| Component | Description | Sample code |
|---|---|---|
| **CSimpleClk** | Output is FPGA clock divided by 2 | `always()`<br>`{`<br>`  oClk = !oClk;`<br>`}` |
| **CPulseClk** | This core generates a clock pulse and an event. It uses a free running modulo 32 counter: loops from 0 to 31. The pulse is generated when Counter = 31 and the event when Counter = 15. | `always()`<br>`{`<br>`  Counter++;`<br>`  oClkPulse = Counter == 31;`<br>`  if(Counter == 15)`<br>`  {`<br>`  oClkEvent :;`<br>`  }`<br>`}` |

| | | |
|---|---|---|
| **CGates** | This core generates two gate signals. It uses a free running modulo 100 counter: loops from 0 to 99. The oGateReset is one if 5 <= Counter < 10 and the oGateRun if 20 <= Counter < 80 | `always()`<br>`{`<br>`  Counter = (Counter >= 99)`<br>`  ?(0):(Counter + 1);`<br>`  oGateReset = Counter >= 5`<br>`  && Counter < 10;`<br>`  oGateRun = Counter >= 20`<br>`  && Counter < 80;`<br>`}` |
| **CTimerClk_T** | This core template generates a clock using the built-in timer. The template parameter PERIOD sets the frequency. | `timerEnd()`<br>`{`<br>`  oClk = !oClk;`<br>`  startTimer((long)(PERIOD - 1));`<br>`}` |
| **CSimpleClk_T** | This core template generates a clock. The template parameter PERIOD sets the frequency. | `always()`<br>`{`<br>`  if(Counter == cHalfPeriod - 1)`<br>`  {`<br>`  Counter =0;`<br>`  oClk = !oClk;`<br>`  }`<br>`  else`<br>`  {`<br>`  Counter++;`<br>`  }`<br>`}` |
| **CTwoPhaseClk_T** | This core template generates a two-phase clock. The template parameter PERIOD sets the frequency. It uses a free running Counter (0 to 3), phase 0 is one if Counter = 1 and phase 1 is one if Counter = 3 | `switch(State)`<br>`{`<br>`  case 0: oClkPhase0 = 0t;`<br>`  oClkPhase1 = 0t;`<br>`  case 1: oClkPhase0 = 1t;`<br>`  oClkPhase1 = 0t;`<br>`  case 2: oClkPhase0 = 0t;`<br>`  oClkPhase1 = 0t;`<br>`  case 3: oClkPhase0 = 0t;`<br>`  oClkPhase1 = 1t;`<br>`}` |

## Compiling and running the test bench

You can compile and execute the test program, then use the signal viewer to look at the clock signals.

1) Start Novakod Studio with a double-click on the *main.prj*.
2) Select the menu *Run→Run N Steps* to simulate.
3) Double click on *Simulation.evo*.
4) Click on the signal viewer icon.

The various clocks are shown in simulation: