Test Bench for CCounterOprAPI

Core description

The core ***CCounterOprEvent*** is a counter controlled by input ***iOpr***. An event on ***iOpr*** triggers the execution and its value is used to determine the operation according to the following type:

```
enum Opr_t { cOprNone, cOprReset, cOprUp, cOprDown, cOprLoad };
```

Therefore, if ***iOpr*** equals ***cOprUp*** then the counter counts up.

Test bench

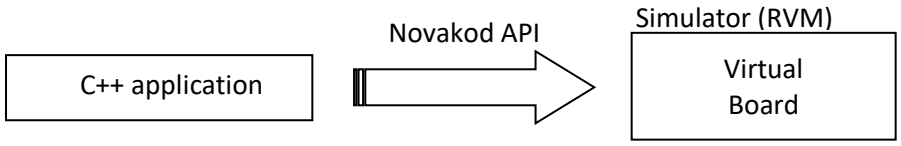
This test bench illustrates how a psC core can be tested with a C++ application.

<< Use of the C++ interface code generator requires a paid license >>

You need to install Visual Studio Community 2019. Or higher

Description

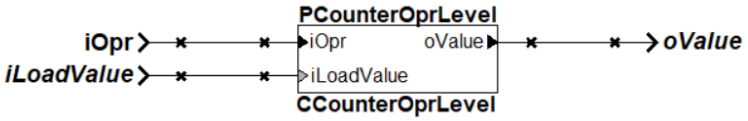
As illustrated on the figure below, the test bench consists of a C++ program communicating with a simulated FPGA, the virtual board, using the ***Novakod API***. The API consists of functions to control input ports and read output ports.



A C++ code generator creates a C++ class Wrapper to encapsulate and simplify using the interface for a specific psC application. This example shows how easy it is to test a psC core using a C++ application.

The psC test program

The test program consists of a single component, the core to be tested, and IO connections. The counter operation is controlled by the value of the input ***iOpr***.



Novakod API C++ class wrapper

The native API allows a C++ application to communicate with the psC program, but it is low level and difficult to use. The C++ interface generator creates a C++ wrapper class ***CCounter*** to encapsulate the native API functions. To use the ***CCounter*** class, the C++ programmer simply creates an object of this class and access the input and output ports by simple function calls. Here are the ***CCounter*** methods for accessing the psC ports:

```
// iOpr
void Set_iOpr(unsigned char val, bool bEvent);  // Set a value w/wo event
void SetEv_iOpr(void);                          // Send an event

// iLoadValue
void Set_iLoadValue(unsigned char val);         // Set a value

// oValue
unsigned char Get_oValue(void);                 // Get a value
bool TestEv_oValue(void);                       // Test for an event
```

The C++ test program

This is the complete code of the C++ test program. The class ***CCounter*** has been automatically generated.

```
#include <windows.h>
#include <iostream>
using namespace std;

#include "api.h"
#include "CCounter.h"

int main(int argc, char* argv[])
{
    int  Value = 0;         // Counter value
    char chKey = ' ';       // Pressed key value
    int  iLoadValue = 0;    // Load value

    // Operation enum, sane as in psC
    enum Opr_t { cOprNone, cOprReset, cOprUp, cOprDown, cOprLoad };

    // Declare the board interface object
    CCounter oFPGA;

    // Initialize and conect to FPGA board (simulated or real)
    if( oFPGA.Initialize() != 0 )
    {
        cout << "Failed to initialize\n";
        return 1;
    };

    // Send a reset, function start() is executed in each component
    oFPGA.pApi->SendStartEvent( );
```

```
    // Infinite loop, until quit
    while(true)
    {
        // Print options
        system("cls");
        cout << "1 - (R)eset\n";
        cout << "2 - (U)p\n";
        cout << "3 - (D)own\n";
        cout << "4 - (L)oad\n";
        cout << "    Load Value: " << iLoadValue << endl << endl;
        cout << "q - (Q)uit\n\n";
        cout << "Counter value " << Value << endl << endl;
        cout << "Select operation: ";

        cin >> chKey;

        // If the user wants to quit
        if((chKey == 'q') || (chKey == 'Q'))
            break;  // Exit the while!

        // Allow reception of future events
        oFPGA.pApi->ReStartEventsReception();

        switch(chKey)
        {
            // Reset counter
            case '1':
                oFPGA.Set_iOpr(cOprReset, true);
                break;

            // Up counter
            case '2':
                oFPGA.Set_iOpr(cOprUp, true);
                break;

            // Down counter
            case '3':
                oFPGA.Set_iOpr(cOprDown, true);
                break;

            // Load value
            case '4':
                cout << "\nEnter the load value 0 - 255 : ";
                cin >> iLoadValue;
                oFPGA.Set_iLoadValue(iLoadValue);
                oFPGA.Set_iOpr(cOprLoad, true);
                break;
        }  // End switch

        oFPGA.pApi->SendEvents();
        oFPGA.pApi->WaitForEvents();

        Value = oFPGA.Get_oValue();
    } // End while

    return 0;
}
```

Compiling and running the application

First, you need to compile the psC program and generate the C++ wrapper class ***CCounter***:

1) Launch Novakod Studio software and open the project.
2) Start the simulation, menu ***Run → Start***. Accept the build confirmation.
3) Select menu: ***Tools-> C++ API Code Generator*** to generate the C++ class.

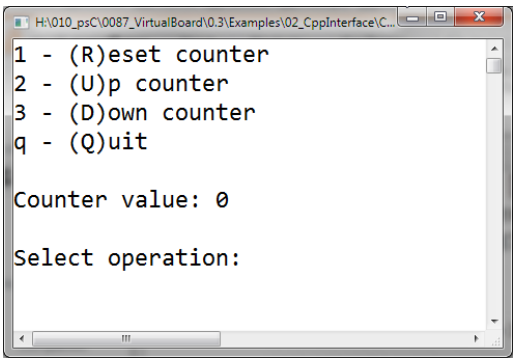
The files ***api.h***, ***CCounter.cpp***, ***CCounter.h***, ***Conversion.cpp*** and ***Conversion.h*** are generated in the ***CppCode*** directory. You are now ready to execute the C++ application to test the core:

4) Go to the ***CppCode*** directory and start the Visual Studio solution (TestProject.sln).

Retarget the solution if necessary.

5) Build the solution, execute the program and follow the instructions.

When the C++ application starts, a C++ console appears. The user can choose between five commands in the console input.



You can use the Novakod Studio ***Watch Window*** while using the C++ program. Simply add the ports to the watch window.