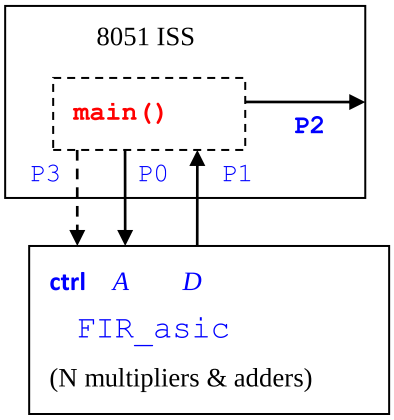
Hardware/Software Codesign: SystemC Exercise Two

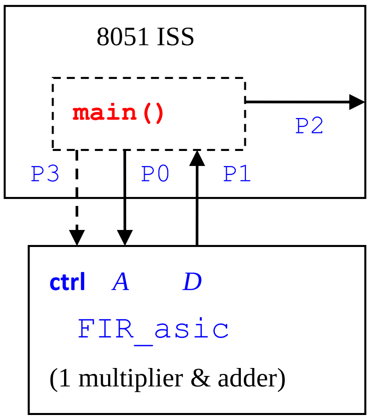
(Due 05/12, 2022)

Try to use SystemC to model a FIR hardware. The hardware `FIR_asic` is assumed to be connected to the 8051 ISS which you can obtain from our course web site. You have to complete this design compliant with 8051 ISS. The input signal *A* is a random value generated by **`main()`** in the 8051 controller, and sent to FIR by `Port0` (`P0`). FIR puts result *D* to 8051 on `Port1` (`P1`). Then, the 8051 controller forwards this result to `Port2` (`P2`). And `Port3` (`P3`) may serve as control signal. The FIR performs the following calculation:

$$D = \sum C_i * A_{t-i}\ , i=0 \sim N-1,\ N \text{ is a pre-defined value}$$

$$C_i \text{ are predefined coefficients, } \sum C_i < 1.$$





All values are 8-bit fixed point within the range of 0~1; for example; a value 31 on P2 represents output *D*=31/255=0.122. You should learn how to use fixed point. Do not use floating point. You should avoid resolution loss during computation.

The implementation of the FIR hardware has the following four versions. Try to compare the cost/performance between each of them.

1. Use N multipliers and adders and the result is generated in one cycle.
2. Use one multiplier and adder, we need N cycles to complete the FIR computation.
3. Use N multipliers and adders, run 8051 at 800MHz and FIR at 100MHz.
4. Use one multiplier and adder, run 8051 at 800MHz and FIR at 100MHz.
   (On most system, peripheral devices are run at PCI-bus clock, which is much slower than CPU clock)

Instructions:

The first step you have to download the 8051 SystemC ISS, and write a hardware SystemC module `FIR_asic`. You may need to write some 8051 C code, to verify your hardware FIR design (you can do this as shown in the following Example Design). According to the given `FIR_asic.h`, design and program the code for your `FIR_asic.c`. You can also modify the code in `FIR_asic.h`, if necessary.

When you submit your program, compress the 8051 ISS code and your own hardware SystemC code together and submit (do not include any object code, please). TA will use a Test-bench machine code to verify the correctness of your design.

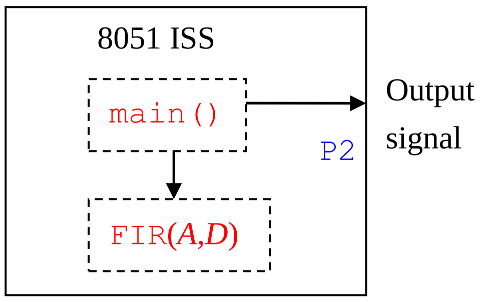
Hard-Copy of SystemC Ex2 should be handed in on 5/12, 2022, and its E-version (using a filename composed with your Registration-number+SystemC-Ex2) sent to our TA: 楊仁傑, r10921098@ntu.edu.tw

Example Design:

Try to use an 8051 Instruction Set Simulator (ISS) to verify a small software FIR program (You can obtain the 8051 ISS and 8051 compiler from our course web-site). You have to complete your program design, then compile it, generate an 8051 object code (machine code), and run this machine code on the 8051 ISS. The input signal A is a random value generated by controller `main()`, and the output D should appear at `Port2` (`P2`), where

$$D = \sum C_i * A_{t-i},\ i=0\sim N-1,\ N \text{ is a pre-defined value}$$

C_i are predefined coefficients, $\sum C_i < 1$.



All values are 8-bit fixed point within the range of 0~1; for example, a value 31 on P2 represents output D=31/255=0.122. You should learn how to use fixed point. Do not use floating point. You should avoid resolution loss during computation.

Instructions for the example design:

1. First, you have to download the 8051 ISS and its C compiler (for example, an IAR compiler from https://www.iar.com/products/architectures/iar-embedded-workbench-for-8051/) or from our course site. To simulate a micro-controller like 8051, we need to provide a segment of machine code to this Micro-Controller core (Here, we use an ISS to emulate the operation of 8051). For example, you may use the following "power down" program to compile and test whether the ISS will operate correctly (stop) or not.

```
#include <io51.h>
main()
{
  P3 = 15;    /* set P3 to 15
  PCON = 2;   /* Set a power down control signal to 8051
}
```

2. Our 8051 ISS accepts only memory dump file. So we have to use a small tool (**`hex2byte.exe`**) for transferring the .hex file to a memory dump file. Take IAR compiler as an example, we first open a project, and use menu `project->file` to add some source C file to the project. Then, we should set an output format, by using
`project->options->Category->XLINK->Format->Others->output format->nec`,

which will change the output format to a `.hex` format. Then you can `make` (compile) the project and get your `.hex` file.

After you get the `.hex` file, use

```
hex2byte -1 -t homework2.hex
```

And you will get a `homework2.t` file, which is a memory dump format that can be loaded into the 8051 ISS. Then, copy `homework2.t` and `null.dmp` to the directory which contains **`8051_iss.exe`** and type the following commend:

```
8051_iss homework2.t null.dmp
```

This comment will start to load the 8051 ISS, load the program machine code (`homework2.t`) and run (emulate) this program on 8051 ISS. The program itself contains a test-bench too, which will show the correctness of your code.

Try to get values from `P0` and `P1`, compute the answer, and store the results to `P2`. If the output values of `P2` are correct, the test-bench will show Correct.

For the environment setting, please use the following setting of Visual Studio 2012, recommended by the senior student, Mr. Guang Huei Lin in my Lab.:

Dear Professor,

Solved. Please download

http://220.134.88.230/filedownload.php?sdir=8051_iss_vs.zip

Students should

1. Rebuild SystemC.lib

   Open 8051_iss_vs\systemc-2.3.3-master\msvc10\SystemC\ SystemC.sln

   Run BUILD/Rebuild Solution

2. Set some compiler parameters

   Open 8051_iss_vs\8051_iss_vs.sln

   On 8051_iss, click right button, choose Properties.

   Under Configuration Properties,

   A. Under VC++ Directories tab

      A1. Include Directories tab

         Add ..\systemc-2.3.3-master\src;

   B. Under C/C++ tab

      B1. Language tab

         Change Enable Run-Time Type Information to Yes (/GR)

      B2. Command Line tab

         In Additional Options, add

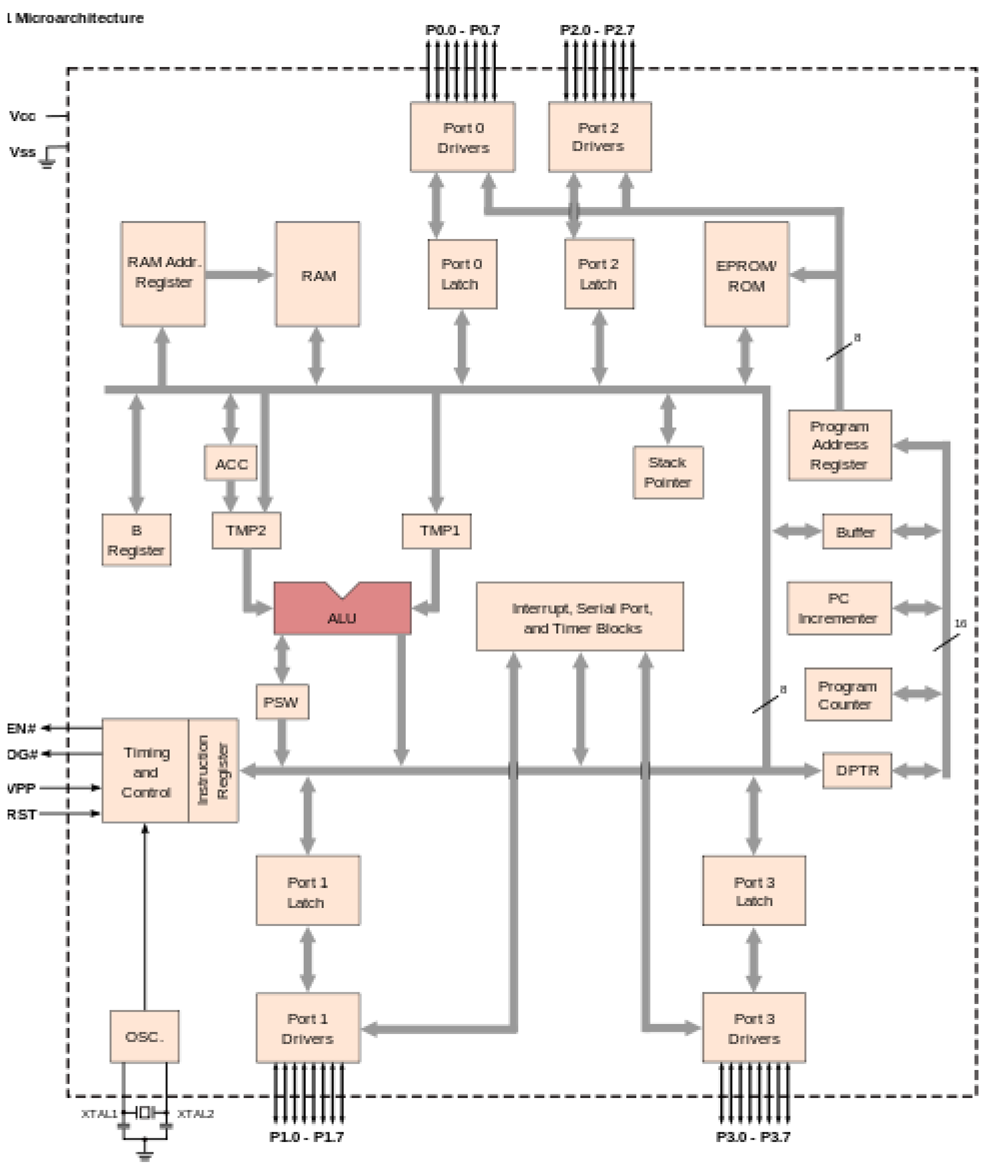
            /vmg

   C. Under Debugging tab

      C.1 Command Arguments

         ..\asm\t8051.t

Guang-Huei Lin



Appendix: Micro-architecture of 8051.