



A|RT Builder v2.1

Modeling a multirate design

Version 1.4

Application Note

Copyright © Frontier Design 2000. All rights reserved.

This document contains information that is proprietary to Frontier Design and may be duplicated in whole or in part by the original recipient for internal business purposes only, provided that this entire notice appears in all copies. In accepting this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use of this information.

Frontier Design, Modeling a multirate design v1.4

The document is for informational and instructional purposes. Frontier Design reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Frontier Design to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Frontier Design products are set forth in the written contracts between Frontier Design and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Frontier Design whatsoever.

FRONTIER DESIGN MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

FRONTIER DESIGN SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF FRONTIER DESIGN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the Government is subject to restrictions as set forth in the subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

A|RT is a registered trademark of Frontier Design Inc.

FRONTIER DESIGN
9000 Crow Canyon Road, Suite S-221

Danville, California CA 94506

United States of America

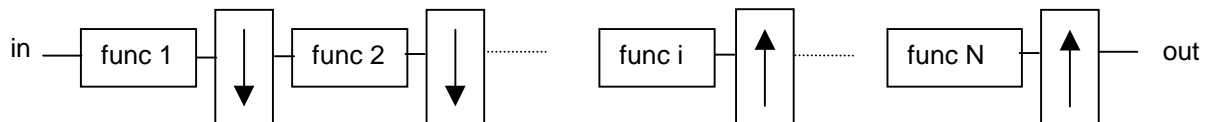
This is an unpublished work of Frontier Design.

1. Introduction

Multirate designs are characterized by decimation (or downsampling) and interpolation (or upsampling) of the data samples. This application note gives a possibility of how to model this behavior in C/C++, by making use of a finite state machine (fsm).

2. General case

The general case has N function-blocks, which are separated by a decimation -or an interpolation stage.



A fsm with N boolean outputs is required to enable or disable each function block at the appropriate cycle during the execution of the top function. One output from the fsm is input for one function block. This function is executed when the input coming from the fsm is set to 1. The number of bits for the counter in the fsm is based on the max. decimation factor that is reached, e.g. if the block diagram consists of *input - function1 – decimation by 2 – function2 – decimation by 2 - output* then the max. decimation factor is 4. In this case a counter with 2 bits is sufficient.

The following C code shows how A|RT can be used to model this behavior :

```

#include <fxp.h>

#define W xxx    // width of the counter is determined by the decimation factor

void fsm(
    bool& Phase_0,
    bool& Phase_1,
    ...
    bool& Phase_i,
    ...
    bool& Phase_N
)
{
    static Uint<W> counter=0;

    Phase_0 = // boolean expression based on the counter register
    Phase_1 = // boolean expression based on the counter register
    ...
    Phase_i = //boolean expression based on the counter register
    ...
    Phase_N = //boolean expression based on the counter register

    ++counter;
}

void func_0(
    bool run,
    DATA in,
    DATA& out)
{
    #pragma OUT out

    if (run)
        // function 0
  
```

```

}

void func_1(
    bool run,
    DATA in,
    DATA& out)
{
#pragma OUT out

    if (run)
        // function 1
}

void func_i(
    bool run,
    DATA in,
    DATA& out)
{
#pragma OUT out

    if (run)
        // function i
}

void func_N(
    bool run,
    DATA in,
    DATA& out)
{
#pragma OUT out

    if (run)
        // function N
}

void multirate(
    DATA in,
    DATA& out)
{
    #pragma OUT out

    DATA tmp_1, tmp_2, ..., tmp_i, tmp_i+1, ..., tmp_N;
    bool phase_0, phase_1, phase_2, ... phase_i, ... phase_N;

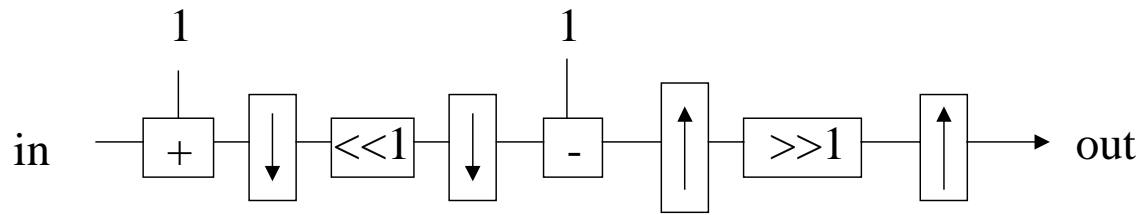
    fsm(phase_0, phase_1, ..., phase_i, ..., phase_N);

    func_0(phase_0, in, tmp_1);
    func_1(phase_1, tmp_1, tmp_2);
    ...
    func_i(phase_i, tmp_i, tmp_i+1);
    ...
    func_N(phase_N, tmp_N, out);
}

```

3. Example

This design example has the same input and output sample rate. However, the input is followed by two downsamplings of a factor 2 and this is followed by two upsamplings of a factor 2 before the output is produced.



The following C code shows how A|RT can be used to model this design:

```

#include <fxp.h>

typedef Uint<4> DATA;

void fsm(
    bool& Phase1,
    bool& Phase2,
    bool& Phase3,
    bool& Phase4
)
{
    static Uint<2> counter=0;

    Phase1 = (Uint<1>(counter)==0);
    Phase2 = (counter==0);
    Phase3 = (counter!=2);
    Phase4 = (Uint<1>(counter)==0);
    ++counter;
}

void func0(
    DATA in,
    DATA& out)
{
    #pragma OUT out

    out=in+1;
}

void func1(
    bool run,
    DATA in,
    DATA& out)
{
    #pragma OUT out

    if (run)
        out=in << DATA(1);
    else
        out="dontcare";
}

void func2(
    bool run,

```

```

    DATA in,
    DATA& out)
{
#pragma OUT out

    if (run)
        out=in -1;
    else
        out="dontcare";
}

void func3(
    bool run,
    DATA in,
    DATA& out)
{
#pragma OUT out

    if (run)
        out=in;
    else
        out=0;
}

void func4(
    bool run,
    DATA in,
    DATA& out)
{
#pragma OUT out

    if (run)
        out=in >> DATA(1);
    else
        out=0;
}

void multirate(
    DATA in,
    DATA& out)
{
    #pragma OUT out
    DATA tmp1,tmp2,tmp3,tmp4;
    bool phase1, phase2, phase3, phase4;

    fsm(phase1, phase2, phase3, phase4);

    func0(in,tmp1);
    func1(phase1,tmp1,tmp2);
    func2(phase2, tmp2, tmp3);
    func3(phase3, tmp3, tmp4);
    func4(phase4, tmp4, out);
}

```

For this design, A|RT Builder will generate one top component consisting of 7 subcomponents. The top component will run at a clock-frequency corresponding with the input and output sample-rate. The fsm makes the subcomponents process the samples at a different rate.