Operating System for Switched Analog Mixed-Signal Circuits

Bakr Younis & Joanne DeGroat

The Ohio State University

Department of Electrical and Computer Engineering

Introduction

- Operating Systems for Analog-MS (general introduction)
- OS for AMS Implementation
- MSP430
  - Example (multi-channel ADC)
  - SystemC, Verilog, Verilog-A
- Reconfigurable modules: AMS library
- Conclusion

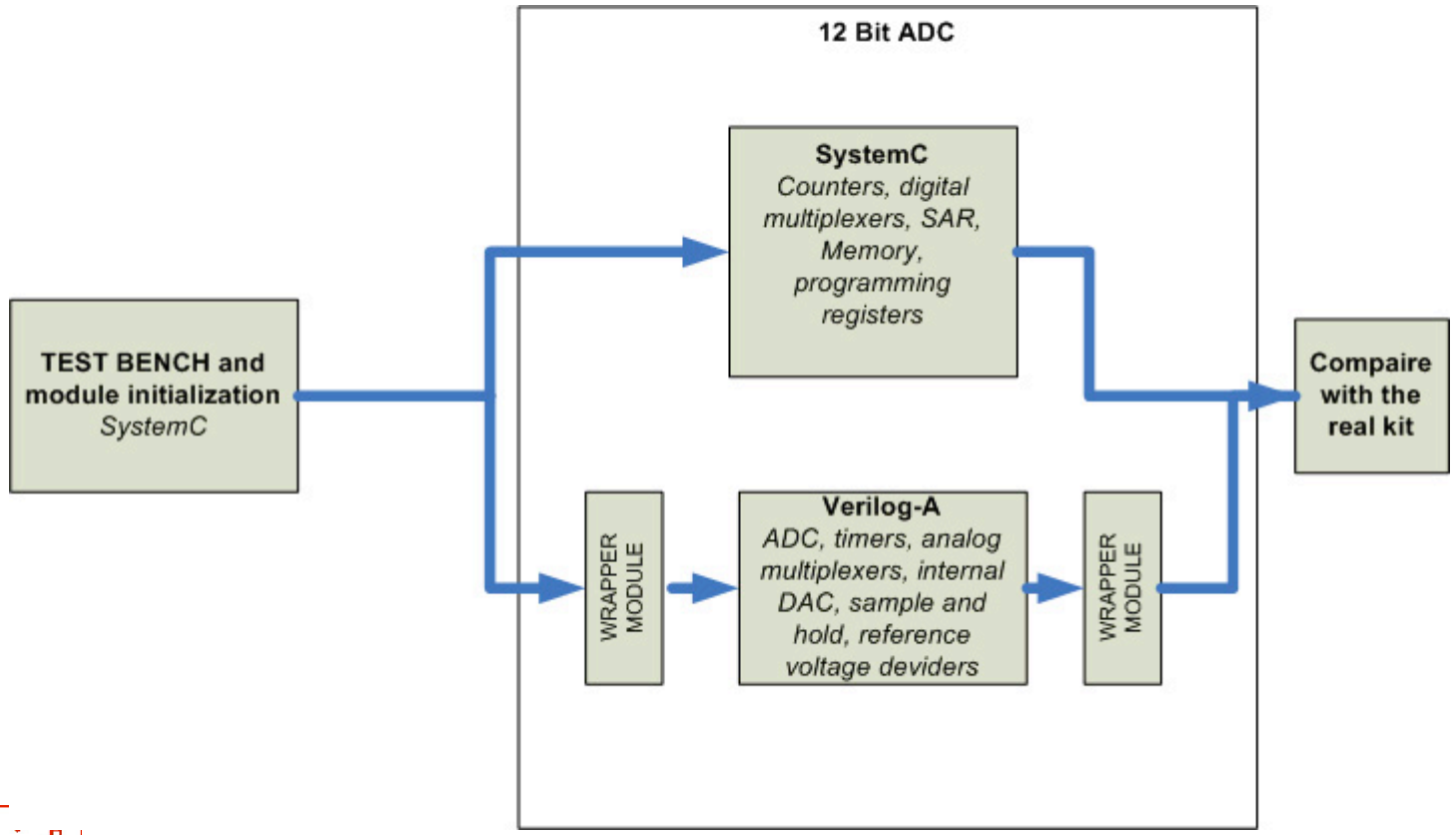
Operating System for AMS Design

- Faster application development
- Ease of reconfiguration or new capabilities
- Application designers need less chip details
- Ease of programming with a well defined OS
- OS defined loops and tasks which are scalable
- Stability and timing considerations are transparent to application engineers

OS for AMS (Implementation)

- Using the MSP430 features as a base for OS design exploration and OS applications
- Macro pseudo-OS provides a higher abstraction of function call groups
- “while loop” approach allows for a simple OS without any resource optimization or process scheduling
- Plans to explore priority and interrupt based OS

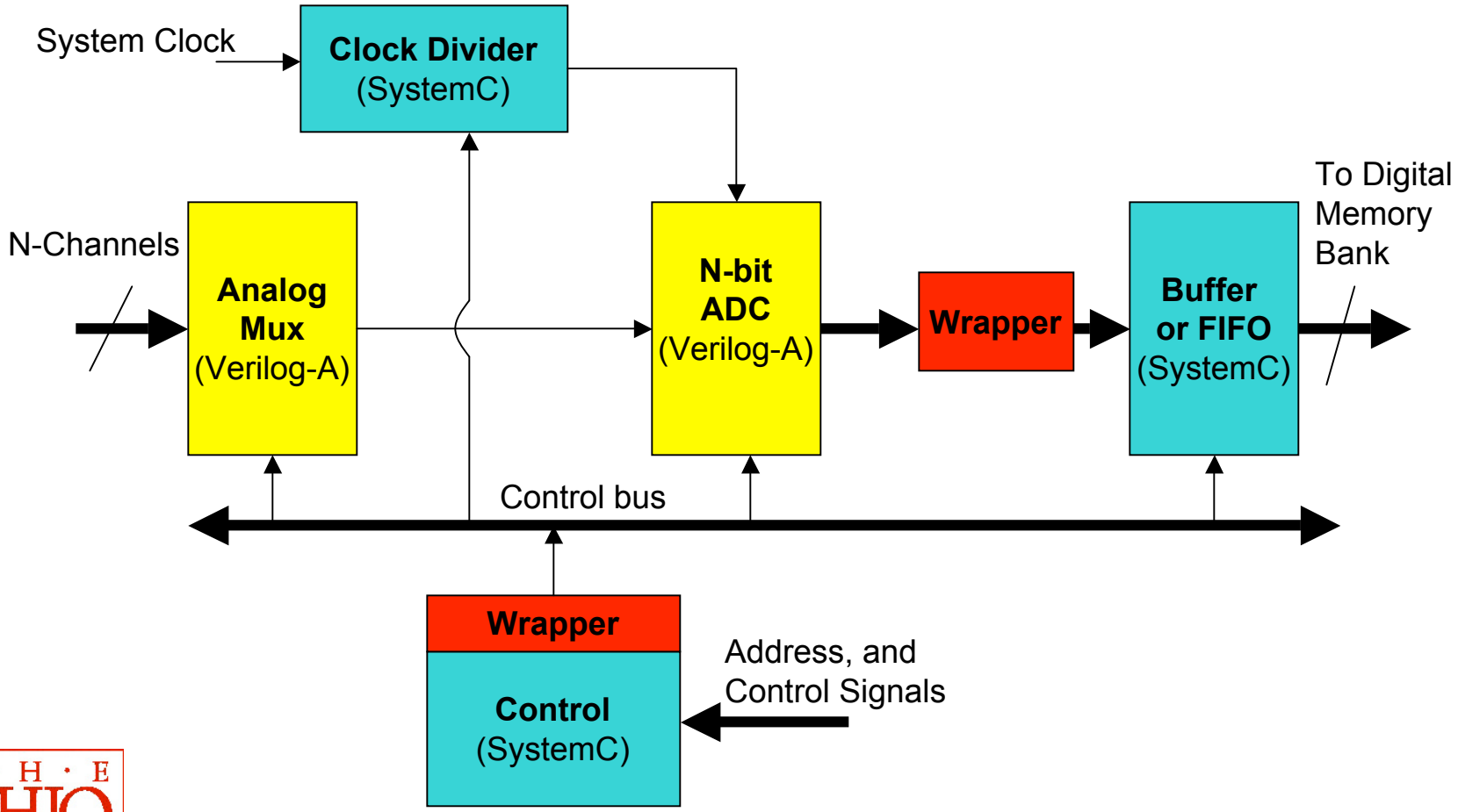
System Level Diagram



Wrapper Module features

- Hierarchical channel
- Interface definitions
- Data type conversion
- Decoding higher level to lower level
- Mixed mode signal conversion
- Control data masking techniques

SystemC and Wrapper Example



Example Design Files

Verilog

```
`define DC_FIF
reset
//`define DC_FIFO_ASYN
reset

module fifo(rd_clk, wr

parameter dw=8;
parameter aw=8;
parameter n=32;
parameter max_size = 1

input
input    [dw-1:0]
input
output   [dw-1:0]
input
output
output
output
output
output   [1:0]
```

Verilog-A

```
analog begin

    @ ( initial_step ) begin
        halfref = vref / 2;
    end

    @ (cross(V(vclk) - vtrans_clk, 1))
        unconverted = V(vin);
        for (i = (`NUM_ADC_BITS-1); i >
            vd[i] = 0;
            if (unconverted > halfref) b
                vd[i] = vlogic_high;
                unconverted = unconverted
            end else begin
                vd[i] = vlogic_low;
            end
            unconverted = unconverted *
        end
    end

    //
    // assign the outputs
    //
    V(vd7) <+ transition( vd[7], tdel,
    V(vd6) <+ transition( vd[6], tdel,
    V(vd5) <+ transition( vd[5], tdel,
```

SystemC

```
#include "test

#ifdef SYSTEMC_ORIG

int sc_main(int argc, char* argv[])
{
    test_ringbuf test("test");
    sc_start(500000, SC_NS);
    return 0;
}

#else

SC_MODULE(top)
{
        test_ringbuf test;

        SC_CTOR(top)
                : test("test")
        {}
};

SC_MODULE_EXPORT(top);

#endif
```

Analog Behavioral Description

- Many design and performance parameters
- Scalability issues
- Design reuse
- Design for specific technology libraries
- Synthesis

Towards a SystemC-AMS and SystemC-OS standard

- Design development combined with on-going effort to make SystemC-OS and SystemC-AMS a successful design methodology
- Ability to describe more complex systems and verify at a higher level of abstraction
- Attempting to deliver accurate high level abstraction analog descriptions

Analog-MS library for standard modules

- New AMS SystemC library including common analog modules (ADC, PLL, etc.)
- Reconfigurable and scalable library components based on accurate simulations at RTL level
- Attempt to standardize refinement process for analog mixed-signal designs

Conclusion

- Simple project for advanced AMS and OS description
- System Portioning at an early stage of design
- Testbench reuse for AMS
- Ease of Interface refinement enables smoother module swap